Exploring D3 Implementation Challenges on Stack Overflow

Leilani Battle[*] University of Washington
Danni Feng[†] University of Maryland
Kelli Webber[‡] University of Maryland

Abstract

Visualization languages help to standardize the process of designing effective visualizations, one of the most prominent being D3. However, few researchers have analyzed at scale how users incorporate these languages into existing visualization programming processes, i.e., *implementation workflows*. In this paper, we present an analysis of the experiences of D3 users as observed through Stack Overflow, summarizing common D3 implementation workflows and challenges discussed online. Our results show how the visualization community may be limiting its understanding of users' visualization implementation challenges by ignoring the larger context in which languages such as D3 are used. Based on our findings, we suggest new research directions to enhance the user experience with visualization languages. All our data and code are available at: `https://osf.io/fup48/`.

Keywords: Web mining, visualization language evaluation

Index Terms: Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

Visualization languages provide great flexibility in programming and reusing visualization designs [13,31]. Browser-based languages in particular have made it easier for a wide range of people to experiment with programming visualizations online [7,13,43], one of the most influential being D3 [13]. On the one hand, D3 is wildly successful: it has been starred over 101 thousand times on GitHub and has been identified on many thousands of webpages, including those of highly regarded media outlets [7]. On the other hand, D3 also has steep learning curve, and can be challenging for analysts and data enthusiasts to adopt [31,42].

One approach to addressing this problem is to investigate the challenges D3 users face with existing visualization programming processes and toolsets, i.e., *implementation workflows*. This knowledge could be used to strengthen existing infrastructure for implementing D3 visualizations (e.g., documentation, examples, tutorials), and even lead to targeted support tools that integrate directly with these workflows. Furthermore, D3 is an important baseline for comparison with other visualization languages [44], and existing languages adopt a similar support structure (e.g., Vega [44] and Vega-Lite [43]). By investigating how to address implementation challenges for D3, similar improvements could be propagated to other languages.

However, we observe a dearth of academic corpora regarding how users interact with visualization languages such as D3. Given the thousands of Stack Overflow posts made about D3, online forums could be a rich resource regarding how D3 users implement new visualizations, and the challenges they run into along the way. Furthermore, these forums could enable us to study D3 users on a scale not yet seen in existing evaluations [28].

[*]e-mail: leibatt@cs.washington.edu
[†]e-mail: fengdn@terpmail.umd.edu
[‡]e-mail: kwebber1@terpmail.umd.edu

Dearth: a scarcity or lack of something

In this paper, we make a first step towards filling this gap by mining and analyzing 37,815 posts about D3 collected from Stack Overflow. To ground our analysis, we evaluate how D3's original design goals have been realized among these users to improve compatibility, debugging, and performance [13]. Through this work, we make the following contributions:

1. We present a mixed methods analysis of the experiences of D3 users as observed on Stack Overflow. To the best of our knowledge, this paper is the *first* to center on online forums, which is known to be a critical resource for D3 users.
2. Through our analysis, we provide: a collection of common implementation workflows discussed by D3 users on Stack Overflow, common challenges when integrating D3 into these workflows, and common debugging patterns for D3 programs.
3. We contribute empirical support for several "common knowledge" assumptions made about D3 users that have yet to be rigorously evaluated by the visualization community, such as the frequent incorporation of non-visualization tools into D3 implementation workflows and persistent barriers to finding relevant D3 examples online.
4. Based on our findings, we identify new areas for future visualization research. As one example, more research is needed on how to strategically design example galleries to boost user creativity and innovation, which would benefit not only D3 users but users of all visualization languages.

2 Related Work

Analyzing Online Forums. A number of projects investigate the practices [30,49,52], attitudes [14,26] and impacts [24,38] of engagement in various online Q&A communities. Others consider what impact online forums can have on individual users [4,5,17,32,48]. We utilize prior work to explore the intersection between analysis of online Q&A communities and visualization implementation workflows and behaviors. We observe similar phenomena as those reported in prior work, such as notable proportion of unanswered questions [48], strong associations between certain tools and libraries [16], prevalent code re-use [3], and an emphasis on web development [6]. However, given our focus on visualization, we can speak to impacts on D3 users specifically, and adapt existing guidance to the direct interests and needs of the visualization community. Furthermore, we believe this paper is the *first* to analyze users' D3 implementation workflows by mining posts from online forums.

Visualization Mining, Modeling, and Analysis. Several projects highlight the value in analyzing existing designs from the web, such as website designs [27], visualization designs [1,7,19,20,23,41], and even code structure [21]. Online platforms such as bl.ocks.org [12] and Observable [34] also help communities engage with visualizations online. A number of works analyze visualizations as images to automatically extract information about the author's implementation decisions (e.g., [7,19,20,25,37,45]). We extend prior work with new ways of automating the process of analyzing users' challenges in implementing visualizations, rather than just analyzing the visualizations themselves. Our findings could guide the design of automated techniques, e.g., by extending systems such as VizML [23] and Draco [33], to identify and fill specific blind spots in a user's visualization knowledge in a way that aligns with existing user implementation workflows.

https://dc-js.github.io/dc.js/

https://nvd3.org/

https://d3fc.io/

3 Analysis Overview

Goal

Our primary objective in this work is to identify opportunities to enhance the D3 visualization implementation process. To this end, we use the original D3 design goals to drive our analysis (compatibility, debugging, performance) [13]. For sake of space, we focus on two of these goals in this paper: *compatibility* and *debugging*[1]. We summarize each goal as a concrete question for our analysis:

1. **Compatibility**: How is D3 used in conjunction with other tools and environments?
2. **Debugging**: How do users explore, interpret, and augment the behavior of D3 code?

To answer these questions, we downloaded relevant Stack Overflow posts, using the Selenium [2] and Beautiful Soup [39] Python libraries. We used ~~using~~ Stack Overflow's tag search to limit the corpus to posts that include the "d3.js" tag. Only posts that were still accessible on Stack Overflow at the time of analysis were included. With our approach, we are able to analyze **37,815 total StackOverflow posts** from 17,591 unique D3 users, showing the power of scale afforded by our techniques.

We apply a mixed methods analysis approach:

1. **Explore** a randomized sample of posts to identify patterns of potential interest as qualitative codes;
2. **Filter** the full 37,815 corpus using specific keyphrases derived from our qualitative codes and associated quotes;
3. **Count** observations of keyphrases on the full corpus to produce quantitative measures;
4. **Compare** our results to follow-up analyses of Stack Overflow posts, the D3 documentation, or relevant GitHub issues and release notes (if applicable), to provide additional context.

For example, to analyze external tools and libraries using these steps, we: (1) **explore** our randomized sample and qualitatively code mentions of various tools, e.g., mentions of React.js are coded as "react" and Angular.js as "angular"; (2) **filter** relevant Stack Overflow posts and GitHub issues programmatically, using keywords derived from our qualitative findings, e.g., "reactjs," "react.js," "angular," etc.; (3) **count** the observations of each tool/library from the posts that pass our filters; and (4) **compare** our results between our coded sample and the distribution of tools detected in the full Stack Overflow corpus.[2]

In line with prior qualitative studies (e.g., [24, 30]), we created a representative sample of Stack Overflow posts for the **explore** phase of our analysis pipeline. This sample contains 817 posts randomly sampled through the year 2020. Three authors manually coded all 817 Stack Overflow posts using *descriptive* deductive and inductive codes. Deductive codes were used for known categories in visualization[3], such as visualization types (e.g., [7]) and interaction types (e.g., [15]). Inductive codes labeled self-reported visualization and/or user characteristics, such as labeling when bar charts, Adobe Illustrator, or SQL were mentioned. Codes were not exclusive, and could be applied in parallel. The resulting codes and their frequencies are reported in Table 1.

We used a multi-phase coding process. In the first phase, all three coders independently coded the same 20 posts, discussed disagreements, and refined problematic codes. Then, this process was repeated two more times, each time with 20 new posts (60 posts total). In the second phase, the remaining posts were coded by the three coders over a 13 week period. All three coders still regularly reviewed and discussed the codebook[4]; new potential codes were carefully evaluated and discussed, and any agreed upon codes were added to the codebook.

Note that for most of our qualitative findings from the **explore** phase, we verify our results quantitatively using the full corpus (in Sect. 4.1, Sect. 5.1, and Sect. 5.2).

4 Compatibility: Integrating D3 With Other Tools

In this section, we analyze mentions of external APIs and tools in our random sample, and summarize the types of issues that Stack Overflow users discuss online.

4.1 Analyzing Users' Visualization Toolsets

23.4% of all posts we qualitatively analyzed referenced at least one other external library or tool. We found 55 different languages, tools, and libraries mentioned in conjunction with D3.js. To see if these frequencies suggest a larger pattern, we compared with observations of the same tools in the full Stack Overflow corpus. We found that the random sample and full corpus have very similar distributions, pointing to broader patterns in how D3 is being used in conjunction with other tools (please see our technical report for more details). We highlight a few interesting observations here.

D3 is Often Integrated Into Larger Web Applications. For example, D3 users often use React, Angular, or Vue to manage the front-end, and Python, Node, Ruby on Rails, Electron, or Java Spring to run the back-end. 34 of 55 (or 61.8%) of the tools we observed are JavaScript Libraries, and React.js, Angular, dc.js, jQuery, and NVD3 are mentioned the most. However, each tool covers a small fraction of posts. For example, React is mentioned in 5% of coded posts. Thus there are a few popular application structures, but Stack Overflow users vary widely in the libraries/tools they use with D3.

D3 is Paired With Specialized Visualization Libraries. Though D3 provides support for geographic maps and graphs, we find some users attempting to integrate D3 with specialized visualization libraries, such as leaflet, simplify, datamaps, and openweathermap for maps, and web cola for graphs. Greensock and Three.js, specialized libraries for 2D and 3D animation, were also mentioned.

D3 is Used Outside of JavaScript. D3 is not used solely with JavaScript. For example, some Stack Overflow users seek help using D3 in R and Jupyter Notebooks. Users mentioned three packages in particular for R: shiny, r2d3, and radialnetworkr. Users also mention other computational environments, such as PostgreSQL and SparQL. However, the overwhelming majority of Stack Overflow posts that we analyzed focused on a JavaScript programming context.

Graphics Editors and Spreadsheets are Used for Prototyping. We also find some interesting implementation variants, such as using Microsoft Excel or Adobe Illustrator for brainstorming prior to implementation (e.g., posts D-73 and A-296 from our dataset files, respectively). Three graphics editors were mentioned on Stack Overflow (Illustrator, CorelDraw, and InkScape), suggesting that a number of users brainstorm visualizations in non-code environments as part of their implementation workflows, and prior to using D3.

4.2 Common Assumptions Clash With User Workflows

When investigating one of the more popular application structures (React components), we find two common integration challenges. First, React is known to have a steep learning curve, and Stack Overflow users often encounter challenges simply in getting React to work properly. For example, we find that many posters have more trouble understanding React than D3 itself (e.g., post B-502). The second theme we observe is a clash in functionality between React and D3. D3 was originally designed to manipulate the DOM directly [13], with known issues for integrating D3 with other libraries that also modify the DOM such as React [46]. We observed this integration challenge in our qualitative analysis; for example, one poster's solution to their D3 integration problem was to use React-focused utilities designed to integrate with D3, rather than use D3 directly (post B_C-63). We also observed Stack Overflow users mentioning libraries that replace D3's DOM manipulation operations

[1] Our analysis of *performance* is shared on OSF at the following wiki page: `https://osf.io/fup48/wiki/performance-analysis/`.

[2] A spreadsheet of this analysis is provided in our supplemental materials, available here: `https://osf.io/t8s37/`.

[3] See the appendix for examples of deductive codes used for visualization types (Appendix A) and interaction types (Appendix B).

[4] The complete codebook is available in our supplemental materials `https://osf.io/8r79d/`, which includes code definitions and examples.

Deductive coding: a top-down approach where you start with a set of predetermined codes and then find excerpts that fit those codes

Inductive coding: a bottom-up approach where you start with no codes and develop codes as you analyze the dataset

https://delvetool.com/blog/deductiveinductive

Table 1: The resulting codes from our qualitative analysis of 817 sampled Stack Overflow posts (total observations in parentheses). The specific codes applied to each post are shared in our supplemental materials, available here: https://osf.io/3m97y/.

Category	Codes
Learning D3	existing-public-examples (179), new-to-d3 (89), add-text-labels (51), search-difficulty (46), previous-post (34), best-practices (31), d3-documentation (28), novice-programmer (12), why-d3 (7), guide (2)
Errors & Behaviors	observed-unexpected-behavior (337), desired-output (222), observed-error (87), procedure (73), issue-not-d3 (44), error-hypothesis (43), experimentation (12), verifying-correctness (9)
APIs	other-apis-tools (191), d3-method-[name] (86), d3-version-issues (50), import-d3 (25), d3-plugin-[name] (11)
Visualization Details.	vis-[type] (398), interaction-[type] (181), data-[type] (154), animation (37), output-[type] (44), multidimensional (16), real-time (10), 3d-model (4)

with those of another library, such as how ngx-charts uses Angular instead of D3 for rendering purposes (e.g., post D-38). Some answers even suggested removing D3 entirely (e.g., post B-149).

We observe in our analysis that when incorporating a JavaScript library that manipulates the DOM, a Stack Overflow user tends to anchor their workflow on the use of this library, since the DOM is the physical structure of the webpage itself. By choosing to manipulate the DOM directly, earlier versions of D3 tried to act as the anchoring library. Thus in a way, the design of D3 assumed that D3 was the focal point of an implementation workflow, even though D3 is scoped primarily for data interaction and visualization, which represent only a fraction of a user's overall interface and webpage design. However, in reviewing D3's release notes, we found that D3's structure has shifted significantly over time from being an anchoring language to a modular collection of data manipulation and visualization libraries, likely due in part to these challenges. Consider this excerpt from the D3v4 release notes [9]:

> *D3 is now modular, composed of many small libraries that you can also use independently. Each library has its own repo and release cycle for faster development. The modular approach also improves the process for custom bundles and plugins.*
>
> *There are a lot of improvements in 4.0: there were about as many commits in 4.0 as in all prior versions of D3. Some changes make D3 easier to learn and use, such as immutable selections. But there are lots of new features, too!*

Modularity

D3's evolution in response to user (and developer) implementation challenges provides critical context for how visualization languages and tools can be designed in the future. For example, the visualization community often develops and evaluates new tools without considering how they will be incorporated into existing user workflows [35, 47]. This lack of broader awareness could lead to less functional tools, and ultimately lower adoption and impact for visualization work. Integration into existing workflows is important

4.3 Takeaways.

We see a wide range of libraries and tools used as part of larger visualization implementation workflows, from specialized visualization libraries to usage with non-JavaScript languages such as Python and R, and even graphics editors and spreadsheets. We explore how D3 evolved in response to users' implementation challenges, such as D3 competing with other libraries to manipulate the DOM. These challenges may speak to broader limitations in how visualization research is conducted, in particular the perspective that visualization tools are the focal point of a user's visualization implementation workflow. Our findings suggest that the visualization community may benefit from taking the goal of compatibility even further, for example by treating visualizations as just one component of a larger application that a user wants to create, rather than the main focus. This shift in perspective necessitates a change in how visualization tools are designed and evaluated. Long term case studies could be a useful starting point [47], as they enable researchers to observe how people use visualization tools in real world environments over time.

5 Debugging: Interpreting & Applying D3 Concepts

In this section, we qualitatively analyze the kinds of bugs that Stack Overflow users often run into with D3, and the different strategies they use to articulate and fix their D3 implementation bugs.

"(...) often post about unexpected behaviors rather than explicit errors."

5.1 Analyzing Implementation & Debugging Methods

Stack Overflow Users Wrestle With Odd D3 Behavior More Often than Explicit Errors. Users mention explicit compilation or runtime errors only 10.6% of the time (87 out of 817 posts). Instead, Stack Overflow users' bugs tend to involve runnable code that exhibits unexpected or unwanted behaviors (337 out of 817 posts, or 41.2%), especially unexpected rendering effects or unexpected interaction behaviors in the visualization output.

Stack Overflow users often rely on existing examples to find solutions to their bugs. We find that Stack Overflow users often reference existing D3 examples that are publicly available online when discussing their bugs (179 out of 817, or 21.9% of posts). This suggests that Stack Overflow users often rely on existing examples when debugging their D3 code. To better understand the influence of the D3 documentation on example usage, we counted the number of Stack Overflow posts containing a link to examples from the visual index of the D3 visualization gallery, bl.ocks.org, and Observable [34]. We find that 13.6% of all posts directly reference examples from just these three sources, representing a significant fraction of all referenced examples in our qualitative dataset.

To see if these findings point to a larger pattern, we compare them with the full Stack Overflow corpus. Across all 37,815 posts analyzed, we find that 14% include references to bl.ocks.org, Observable, or the D3 Gallery visual index, which is consistent with our coded data. Thus D3 examples and documentation seem to be key components of users' D3 visualization implementation workflows.

5.2 Challenges in Using Relevant Examples to Fix Bugs

Adapting existing D3 examples seems to be an important but also complicated part of the D3 implementation and debugging process, which we explore further in this section.

Some Users Struggle to Find the Most Relevant Examples. When searching for relevant examples, D3 users may struggle to match their own terminology for different visualization and interaction types to the terminology of others. For example, we find that 46 of 817 (or 5.6%) of posts mention difficulties in finding *any* relevant D3 examples or Stack Overflow posts. If a D3 user is unaware of the appropriate D3 methods for implementing certain functionality, then a typical keyword search in Google or Observable may fail due to mismatches in terminology. Advanced visualization search engines may also be of limited use, due to their reliance on language-specific terminology, such as [22] and `bl.ocksplorer.org`. However, once relevant examples are identified, D3 users still seem to struggle with separating relevant code from irrelevant code, and with distinguishing D3 components from those of other libraries.

Complicated, Irrelevant Functionality Makes Examples Hard to Reuse. The utility of a D3 example seems to depend on not only the ratio of relevant to irrelevant functionality, but also the complexity of this functionality. Unfortunately, existing examples routinely contain more functionality than the user wants or is familiar with, which can easily confuse the user, and lead to unnecessary bugs.

https://observablehq.com/@d3/gallery

https://github.com/d3/d3/wiki/Gallery

Border Mesh example: https://bl.ocks.org/mbostock/8ba407a7a53d80be62a8d54774663c2f

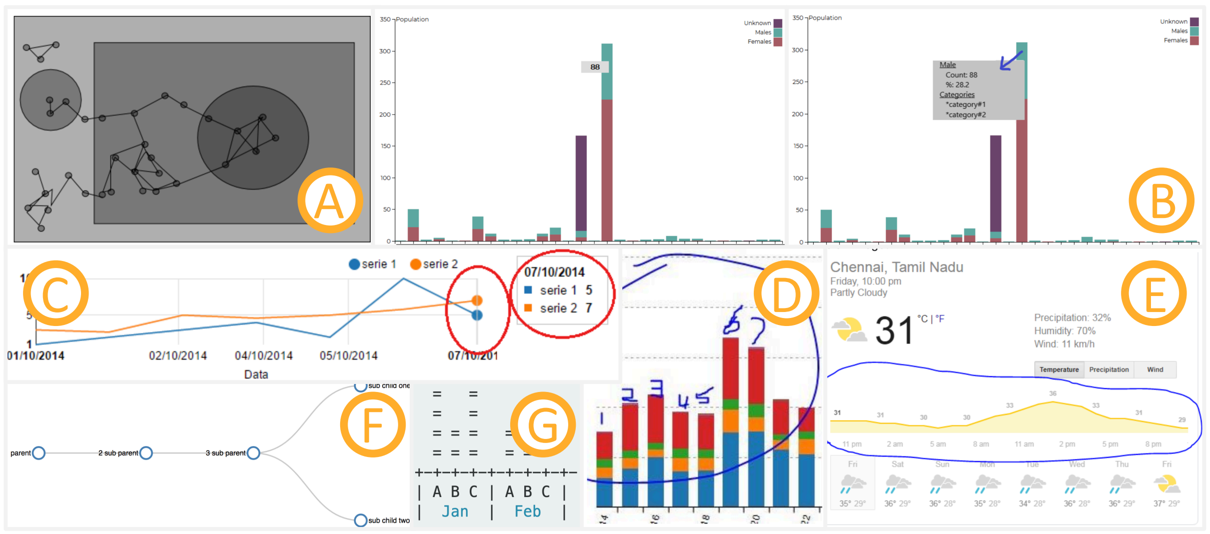


Figure 1: Example images shared online to convey desired visualizations (A, E, F, & G), interactions (B), and modifications (C & D).

Consider the following description from Stack Overflow, where a user is struggling to reuse the zoom functionality from a geographic map example that uses meshes:

> "*I am trying to implement a 'zoom to bound' feature on my D3 map. This is the Block that I have in mind... My issue is that it looks like the implementation requires a topoJSON mesh.*" (post A-587)

From follow-up comments on the post, we learn that meshes are not needed, revealing unnecessary functionality:

> "*You don't need the mesh, that's just for the states' strokes.*" (post A-587)

The commentor also suggests a more relevant example "*Have a look at this bl.ocks without mesh...*" (post A-587), pointing to a separate problem that we observed: some users struggle to find the most relevant examples in the first place.

Users Struggle to Connect D3 Code Components With Their Corresponding Visual Outputs. We find that many Stack Overflow users struggle to pinpoint the root causes of their D3 bugs. One challenge for Stack Overflow users may be that the way in which users intuitively reason about visualization outputs may not match how these outputs are generally specified using the D3 language. Rather than using the parlance of the D3 language to describe what they want to create, many Stack Overflow users illustrate their desired outputs by embedding one or more images within a post, often with annotations added. For example, 21.1% of the posts we analyzed included linked or embedded images, which also held true for our full Stack Overflow corpus (19.2%).

The annotations generally appeared to be hand-drawn, or applied using image editing or presentation software. Fig. 1 provides examples from five different Stack Overflow posts. For example in Fig. 1-E, the user circled a specific visualization within an existing image (online weather information) that they want to copy. In Fig. 1-G, the user has made a diagram in ASCII showing how they want to order and group the bars of a bar chart, reminiscent of ASCII-based diagramming seen in other debugging contexts (e.g., database debugging [8]). Some users would even go so far as to draw the specific changes they expect to make. In Fig. 1-D, the user has manually illustrated their desired results by annotating a stacked bar chart with the totals drawn above each stack. Users also illustrate desired interactions by sharing videos (e.g., post E-110), .gifs (e.g., post D-38), and sequences of annotated images (e.g., Fig. 1-A and B).

5.3 Takeaways.

We find that Stack Overflow users often post about unexpected behaviors rather than explicit errors. We observe that one common implementation and debugging strategy among Stack Overflow users is to compare their code with relevant examples that are publicly available online. However, Stack Overflow users often struggle to find the most relevant D3 examples to inform their implementation and debugging process. When they do find relevant examples, they may still struggle to extract the most relevant functionality. This is due in part to how many D3 examples lack a modular structure, and often contain functionality that other users do not need, which can confuse these users and lead to unnecessary bugs.

If users could search for desired functionality without domain-specific keywords, they could find relevant D3 examples with less time and effort. Furthermore, some users have found their own way of communicating desired visualization outputs through annotated images, videos, and gifs. If Stack Overflow users could demonstrate their desired changes by *directly manipulating the visualization output* (e.g., [29, 40, 53]), then they could debug their code using a search-by-example process, rather than a language or keyword-based search. Then, visualization tools could compute the corresponding implementation deltas and update the code automatically. Debugging via direct manipulation has been proposed in other contexts, such as for updating Jupyter notebooks [50] and debugging SQL queries [18], and could be adapted for visualization languages.

6 Discussion: Implications for Future Research

D3 has made incredible contributions to the visualization community. In this paper, we investigate opportunities to further enhance the experience of D3 users and of visualization language users in general. We present an analysis of 37,815 posts made by D3 users on Stack Overflow. We evaluate D3 from two perspectives: compatibility and debugging. Our findings show that when we focus on developing visualization languages but not how and where people actually use them, we may struggle to fully understand the user experience, and may thus fail to fully identify and address the needs of these users. By being mindful of how users interact with visualization languages and relay their implementation challenges, we can develop innovative strategies to enhance users' implementation processes, increase users' information access, and empower users to explore a wider range of effective visualization designs. In this section, we highlight opportunities for future work based on our findings.

6.1 Emphasize Integration With Non-Visualization Tools

Keyword: implementation workflow

When visualizations become the sole focus of visualization tool development, developers and researchers may build tools that conflict with other critical needs within users' implementation workflows, potentially hindering adoption. As demonstrated through the evolution

Relay: receive and pass on (information or a message)

Parlance: a particular way of speaking or using words, especially a way common to those with a particular job or interest

Dampen: make less strong or intense

of D3, our community needs to shift its mindset towards building modular visualization *components* that can integrate smoothly with other tools. Furthermore, we encourage more formal evaluations of how users integrate new tools and languages (or not) into their implementation workflows over time [35,47]. For example, we encourage our community to conduct more large scale, quantitative studies of how visualization languages are used in popular development environments, e.g., Jupyter, Observable, and R Studio.

6.2 Better Support Infrastructure Could Boost Adoption

We find that users on Stack Overflow often rely on examples to implement and debug new D3 visualizations, but struggle to find relevant examples and reuse them correctly. Our findings point to two challenges in supporting current debugging workflows. First, we lack *modularized building blocks* for implementing new visualizations in D3. This issue may stem in part from D3's mixing of declarative specification of encodings with imperative specification of interactions, which is addressed in later languages such as Vega-Lite [43]. Second, the problem may not only be with code structure but also *insufficient infrastructure* for helping users understand the flow of the code, i.e., how results propagate through the various parts of a D3 visualization. This issue has led to recent developments such as the Observable notebook environment [34], but Observable still expects users to manually segment their own code. Similar environments such as Jupyter notebooks suffer from the same problems. Both challenges highlight how D3 users struggle to break their implementation challenges down into modular, solvable pieces.

Based on our findings, we argue that both challenges could be addressed effectively by improving the *support infrastructure* around D3, rather than by modifying D3 itself. For example, we could develop more intuitive search interfaces that support search by example or search by demonstration, such as searching existing D3 examples for specific visual outputs or intended interaction behaviors. Furthermore, these issues could be addressed in a data-driven way by mining solutions directly from the thousands of existing D3 examples we observed. For example, new tools could leverage this data to help users identify separate D3 components within existing examples, and extract only the components that are needed. Development environments could also be augmented to automatically show relevant documentation, or recommend relevant code blocks, based on inferred user goals, experience levels, and potential biases.

6.3 Make Effective Example Gallery Design Active Research

Our findings in Sect. 5 show that a basic web search is just not good enough to help users find relevant D3 examples. Asking users to search for solutions using specialized D3 or visualization keywords are sub-optimal alternatives. Even Stack Overflow is insufficient; translating a specific D3 bug into a self-contained Stack Overflow question requires significant time and effort [17], and 37% of the 37,815 posts analyzed were left unanswered by other users. We believe these issues persist because our community views them as engineering rather than research problems.

In an effort to shift this perspective, we suggest some interesting research opportunities to expand existing design galleries. We could synthesize current best practices in visualization design as a diverse set of modularized visualization examples that all visualization language developers aim to provide. Rather than expecting developers to create design galleries meeting these requirements, we could also find ways to automate the process of design gallery generation itself. This solution could involve a mixture of automation and the crowd, where automated processes are developed not only to detect gaps in existing galleries, but also to encourage users to fill these gaps with new examples. This approach could also be used to detect redundant or low quality examples and replace them automatically. Developing design galleries and documentation takes time, whether for open-source languages such as Vega-Lite [43], or commercial APIs such as Plotly [36]. Automating the documentation process could speed up the learning process and dampen learning curves for users of *all* visualization languages.

Imperative programming: "telling (...) how to do something, and as a result what you want to happen will happen."

Declarative programming: "telling (...) what you would like to happen, and let the computer figure out how to do it."

http://latentflip.com/imperative-vs-declarative

6.4 Takeaways Summary

Here we list three major takeaways derived from our research:

TL;DR

- Develop and test visualization languages as part of larger implementation workflows involving multiple tools.
- Provide support infrastructure for helping users find relevant examples, extract meaningful code components, and integrate these components into their workflows.
- Automate the example gallery generation process, to ease the difficulty of designing effective examples and make the process more consistent across languages/tools.

6.5 Limitations and Future Work

One limitation is that we only focus on D3 users who post on Stack Overflow, a subset of all D3 users. However, we are still able to study 17,591 total D3 users, showing the scale afforded by our approach. Given that posters do not have to share personal information on Stack Overflow, user characteristics are not consistently available in our dataset; thus we exclude them from our analysis. An interesting direction for future research is to conduct follow up interviews with D3 users to better understand their backgrounds, motivations, and experiences, providing additional context for our findings. However, we believe that our ability to analyze thousands of D3 users helps to balance this limitation out. Certain D3 functionality may not be well represented in our dataset, e.g., animations. It would be interesting to introduce filters to our Stack Overflow crawler to extract specific posts for more targeted analyses in the future, e.g., downloading animation-focused posts for further analysis. In general, we hope that by sharing our materials, we can empower the community to explore visualization languages in new ways. For example, a promising avenue of future work could be to analyze iteration on visualization languages and user reasoning in tandem over time.

Acknowledgments

This research was funded in part by NSF Award #1850115. We thank Dominik Moritz, Zhicheng Liu, Niklas Elmqvist, the UMD Human-Computer Interaction Lab, the BAD Lab, the UW Interactive Data Lab, and our (many) reviewers for their invaluable feedback. We also thank Arjun Nair and Rishik Narayana for their help with data collection.

References

[1] End-users Publishing Structured Information on the Web: An Observational Study of What, Why, and How. CHI '14. doi: 10.1145/2556288.2557036

[2] SeleniumHQ Browser Automation, 2021.

[3] R. Abdalkareem, E. Shihab, and J. Rilling. On code reuse from stackoverflow: An exploratory study on android apps. *Information and Software Technology*, 88:148–158, 2017.

[4] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow. KDD '12, pp. 850–858. ACM, 2012. doi: 10.1145/2339530.2339665

[5] A. Ardichvili, M. Maurer, W. Li, T. Wentling, and R. Stuedemann. Cultural influences on knowledge sharing through online communities of practice. *Journal of Knowledge Management*, 10(1):94–107, Jan. 2006. doi: 10.1108/13673270610650139

[6] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, 2014.

[7] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. CHI '18, pp. 594:1–594:8. ACM, 2018. doi: 10.1145/3173574.3174168

[8] L. Battle, D. Fisher, R. DeLine, M. Barnett, B. Chandramouli, and J. Goldstein. Making Sense of Temporal Queries with Interactive

Visualization. CHI '16, pp. 5433–5443. ACM, 2016. doi: 10.1145/2858036.2858408

[9] M. Bostock. Release v4.0.0 - d3/d3, 2016.

[10] M. Bostock. D3 api reference, 2020.

[11] M. Bostock. Home - d3/d3 wiki, 2020.

[12] M. Bostock. Popular blocks - bl.ocks.org, 2021.

[13] M. Bostock, V. Ogievetsky, and J. Heer. D3 Data-Driven Documents. *IEEE TVCG*, 17(12):2301–2309, Dec. 2011. doi: 10.1109/TVCG.2011.185

[14] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building Reputation in StackOverflow: An Empirical Investigation. MSR '13, pp. 89–92. IEEE Press, 2013.

[15] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE TVCG*, 19(12):2376–2385, 2013. doi: 10.1109/TVCG.2013.124

[16] C. Chen and Z. Xing. Mining technology landscape from stack overflow. ESEM '16. ACM, 2016. doi: 10.1145/2961111.2962588

[17] D. Ford, J. Smith, P. J. Guo, and C. Parnin. Paradise Unplugged: Identifying Barriers for Female Participation on Stack Overflow. FSE 2016, pp. 846–857. ACM, 2016. doi: 10.1145/2950290.2950331

[18] S. Gathani, P. Lim, and L. Battle. Debugging database queries: A survey of tools, techniques, and users. CHI '20, p. 1–16. ACM, New York, NY, USA, 2020.

[19] J. Harper and M. Agrawala. Deconstructing and Restyling D3 Visualizations. UIST '14, pp. 253–262. ACM, 2014. doi: 10.1145/2642918.2647411

[20] J. Harper and M. Agrawala. Converting Basic D3 Charts into Reusable Style Templates. *IEEE TVCG*, 24(3):1274–1286, Mar. 2018. doi: 10.1109/TVCG.2017.2659744

[21] A. Head, E. L. Glassman, B. Hartmann, and M. A. Hearst. Interactive Extraction of Examples from Existing Code. CHI '18, pp. 85:1–85:12. ACM, 2018. doi: 10.1145/3173574.3173659

[22] E. Hoque and M. Agrawala. Searching the visual style and structure of d3 visualizations. *IEEE TVCG*, 26(1):1236–1245, Jan 2020. doi: 10.1109/TVCG.2019.2934431

[23] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo. VizML: A Machine Learning Approach to Visualization Recommendation. CHI '19, pp. 128:1–128:12. ACM, 2019. doi: 10.1145/3290605.3300358

[24] R. Jones, L. Colusso, K. Reinecke, and G. Hsieh. R/science: Challenges and opportunities in online science communication. CHI 2019, p. 1–14. ACM, New York, NY, USA, 2019.

[25] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo. ChartSense: Interactive Data Extraction from Chart Images. CHI '17, pp. 6706–6717. ACM, 2017. doi: 10.1145/3025453.3025957

[26] T. Kauer, M. Dörk, A. L. Ridley, and B. Bach. The public life of data: Investigating reactions to visualizations on reddit. CHI '21. ACM, New York, NY, USA, 2021.

[27] R. Kumar, A. Satyanarayan, C. Torres, M. Lim, S. Ahmad, S. R. Klemmer, and J. O. Talton. Webzeitgeist: Design Mining the Web. CHI '13, pp. 3083–3092. ACM, 2013. doi: 10.1145/2470654.2466420

[28] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE TVCG*, 18(9):1520–1536, 2012. doi: 10.1109/TVCG.2011.279

[29] B. Lee, G. Smith, N. H. Riche, A. Karlson, and S. Carpendale. Sketchinsight: Natural data exploration on interactive whiteboards leveraging pen and touch interaction. PacificVis '15, pp. 199–206, 2015. doi: 10.1109/PACIFICVIS.2015.7156378

[30] K. Mack, J. Lee, K. Chang, K. Karahalios, and A. Parameswaran. Characterizing scalability issues in spreadsheet software using online forums. CHI EA '18, p. 1–9. ACM, New York, NY, USA, 2018. doi: 10.1145/3170427.3174359

[31] H. Mei, Y. Ma, Y. Wei, and W. Chen. The design space of construction tools for information visualization: A survey. *Journal of Visual Languages & Computing*, 44:120–132, Feb. 2018. doi: 10.1016/j.jvlc.2017.10.001

[32] A. Merchant, D. Shah, G. S. Bhatia, A. Ghosh, and P. Kumaraguru. Signals Matter: Understanding Popularity and Impact of Users on Stack Overflow. WWW '19, pp. 3086–3092. ACM, 2019. doi: 10.1145/3308558.3313583

[33] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE TVCG*, 25(1):438–448, 2019. doi: 10.1109/TVCG.2018.2865240

[34] Observable, Inc. Observable - make sense of the world with data, together / observable, 2021.

[35] C. Plaisant. The challenge of information visualization evaluation. AVI '04, p. 109–116. ACM, 2004. doi: 10.1145/989863.989880

[36] Plotly. Plotly: The front end for ml and data science models, 2021.

[37] J. Poco and J. Heer. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Comput. Graph. Forum*, 36(3):353–363, June 2017. doi: 10.1111/cgf.13193

[38] S. Ravi, B. Pang, V. Rastogi, and R. Kumar. Great question! question quality in community q&a. 8(1):426–435, May 2014.

[39] L. Richardson. Beautiful soup: We called him tortoise because he taught us., 2021.

[40] B. Saket, A. Srinivasan, E. D. Ragan, and A. Endert. Evaluating Interactive Graphical Encodings for Data Visualization. *IEEE TVCG*, 24(3):1316–1330, Mar. 2018. doi: 10.1109/TVCG.2017.2680452

[41] B. Saleh, M. Dontcheva, A. Hertzmann, and Z. Liu. Learning Style Similarity for Searching Infographics. GI '15, pp. 59–64. Canadian Information Processing Society, 2015.

[42] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. Stasko, J. Thompson, M. Brehmer, and Z. Liu. Critical reflections on visualization authoring systems. *IEEE TVCG*, 26(1):461–471, 2020. doi: 10.1109/TVCG.2019.2934281

[43] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE TVCG*, 23(1):341–350, Jan. 2017. doi: 10.1109/TVCG.2016.2599030

[44] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE TVCG*, 22(1):659–668, Jan. 2016. doi: 10.1109/TVCG.2015.2467091

[45] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. ReVision: Automated Classification, Analysis and Redesign of Chart Images. UIST '11, pp. 393–402. ACM, 2011. doi: 10.1145/2047196.2047247

[46] D. Scanlon. How (and why) to use d3 with react — hacker noon, 2017.

[47] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies. BELIV '06, p. 1–7. ACM, 2006. doi: 10.1145/1168149.1168158

[48] R. Slag, M. de Waard, and A. Bacchelli. One-day flies on stackoverflow - why the vast majority of stackoverflow users only posts once. MSR '15, pp. 458–461. IEEE, 2015. doi: 10.1109/MSR.2015.63

[49] C. Treude, O. Barzilay, and M.-A. Storey. How Do Programmers Ask and Answer Questions on the Web? (NIER Track). ICSE '11, pp. 804–807. ACM, 2011. doi: 10.1145/1985793.1985907

[50] Y. Wu, J. M. Hellerstein, and A. Satyanarayan. *B2: Bridging Code and Interactive Visualization in Computational Notebooks*, p. 152–165. UIST 2020. ACM, 2020.

[51] J. S. Yi, Y. a. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE TVCG*, 13(6):1224–1231, 2007. doi: 10.1109/TVCG.2007.70515

[52] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim. Are code examples on an online q&a forum reliable? a study of api misuse on stack overflow. ICSE '18, p. 886–896. ACM, 2018. doi: 10.1145/3180155.3180260

[53] J. Zong, D. Barnwal, R. Neogy, and A. Satyanarayan. Lyra 2: Designing interactive visualizations by demonstration. *IEEE TVCG*, 27(2):304–314, 2021. doi: 10.1109/TVCG.2020.3030367

A Observed Visualization Types

In this appendix, we report on the distribution of visualization types observed in our sample of 817 Stack Overflow posts, and explore potential reasons for why we see this particular distribution of visualization types in our sample.

Coding Method. We use the visualization taxonomy proposed by Battle et al. [7] as a starting point for establishing deductive codes for visualization types, since this taxonomy is based on visualizations shared by users of D3, as well as other tools such as

Hive plot:

- http://www.hiveplot.com/
- https://bost.ocks.org/mike/hive/

Chord diagram:

- https://www.data-to-viz.com/graph/chord.html
- https://en.wikipedia.org/wiki/Chord_diagram_(information_visualization)

OHLC: Open-high-low-close chart

Plotly and Fusion Charts. Battle et al. observed 24 visualization types: `area`, `bar`, `box`, `bubble`, `chord`, `contour`, `donut`, `filled-line geographic map`, `graph/tree`, `heatmap`, `hexabin`, `line`, `radial/radar`, `pie`, `sankey`, `scatter`, `treemap`, `voronoi`, `waffle`, `word cloud`, `sunburst`, `stream graph`, and `parallel coordinates`. We extended this taxonomy with nine more visualization types observed on Stack Overflow (`bullet`, `funnel`, `hive`, `marimekko`, `OHLC`, `polygon`, `table`, `gantt`, and `waterfall`), and three more types from the D3 gallery visual index (`dial`, Gauge `icicle`, and `horizon`). The final taxonomy includes $24+8+4=36$ visualization types.

We labeled each Stack Overflow post with any mentioned visualization types from our extended taxonomy. For example, when a user mentions bar charts: "I'm working on creating a stacked bar chart..." (post A-207) , we label the post with "vis-bar". Then, we tabulate the visualization types observed across the 817-post sample. We repeated this same coding process for issues discussed on the D3 github page, as well as the public gallery of visualizations shared on the D3 website. The top 16 visualization types observed for each corpus are reported in Fig. 2.

Stack Overflow Users Favor A Handful of Visualization Types. We find that most posts focus on a specific visualization type. We find that 48.7% of our coded posts mention explicit visualization types; the top 16 are shown in Fig. 2 (center). 29 of the 36 visualization types from our extended taxonomy were observed (or 80.6%), however 12 appear only once (or 33.3%). One might assume that the most common visualization types (e.g., `bar` charts and `line` charts) are least likely to cause problems for Stack Overflow users, given their pervasiveness and simplicity. However, the top three visualizations we observed were `bar` charts, `line` charts, and geographic `maps`, which are also three of the most common visualization types observed by Battle et al. [7]. More complex visualizations, such as `parallel coordinates`, are rarely mentioned in comparison. These results suggest that observed visualization types on Stack Overflow are indicative of user visualization preferences rather than complexity or challenge in implementing specific visualization types.

Observed Visualizations on Stack Overflow Match the D3 Gallery. We compared the occurrence of visualization types in our coded dataset to those in the D3 example gallery, shown in Fig. 2. 30 of 36 visualization types were observed in the D3 gallery visual index (or 83%). `graphs` (and `trees`) are the most popular visualization type in D3 gallery visual index, along with `maps`, `bar` charts, and `line` charts. Furthermore, we see that the top eight visualization types mentioned in our coded sample closely match the top eight visualization types observed in the D3 gallery. In contrast, the top visualization types mentioned on github only match roughly half of our coded sample. https://github.com/d3/d3/wiki/Gallery

To gauge the importance of the D3 example gallery within the documentation, we analyzed how often the gallery page was updated on GitHub compared to other pages, e.g., the home [11] or API reference [10] pages. At the time of our analysis, the D3 Gallery page had been updated 1,295 times, three times as often as any other main page in the D3 documentation. We also found 29 GitHub issues that cited the D3 gallery visual index, supporting our findings.

Takeaways. When we combine our findings across GitHub issues, the D3 gallery, and in previous work [7], our results paint a more holistic picture of users' design choices. We see that `line` and `bar` visualizations may dominate because they are universally popular across visualization tools, as observed in prior work [7]. `graph` and `tree` visualizations may be of interest to Stack Overflow users because they appear frequently in the D3 gallery, documentation, and GitHub issues. These results may suggest that D3 users rely heavily on existing D3 examples and documentation when implementing new visualizations. Thus the D3 gallery likely represents, and perhaps even influences, the range of visualizations created by Stack Overflow users. As a result, these findings may also suggest that if the D3 documentation is skewed to prioritize specific visualization types, it could potentially introduce bias in to users' implementation workflows. No need or no good starters?

Given the breadth of visualization types observed in the academic literature, one interesting question for the future is why Stack Overflow users do not seem to experiment with these other visualization types. On the one hand, these findings might suggest that the full design breadth of D3 may be interesting but also overkill for most Stack Overflow users. On the other hand, given that Stack Overflow users frequently reference existing D3 visualizations as part of their implementation workflows, the issue could be that there is insufficient scaffolding for users to feel confident in creating these other types of visualizations. The quantity and variety of gallery examples may be just as important as the quality of examples in helping users expand their design thinking.

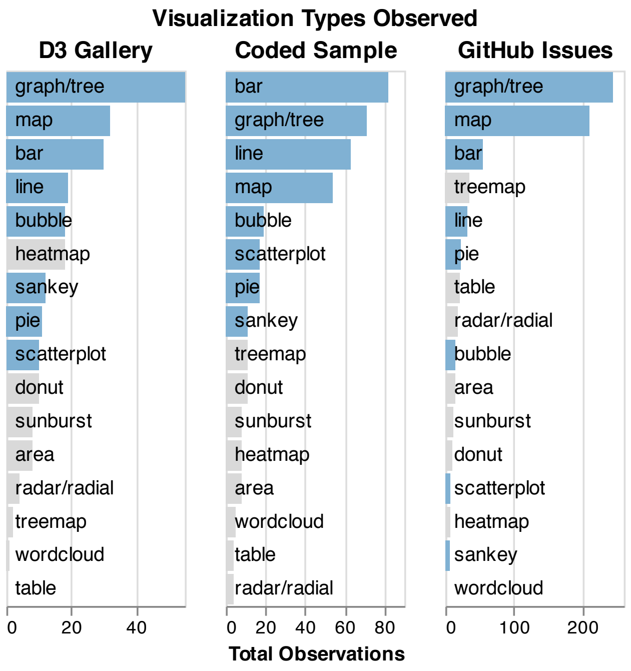


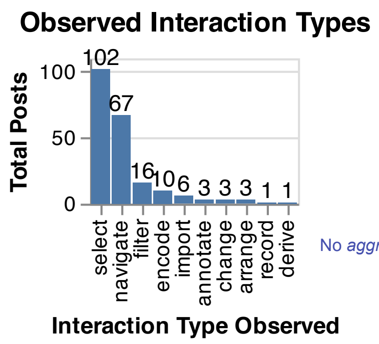
Figure 2: Total observations for the top 16 visualization types observed in (a) the D3 gallery, (b) our randomized sample of qualitatively coded Stack Overflow posts, and (c) GitHub issues for D3. The top eight visualizations from the coded sample are colored blue.

B Observed Interaction Types

In this appendix, we report on the distribution of interaction types observed in our sample of 817 Stack Overflow posts, and compare our findings with the corresponding descriptions of these interactions in the D3 documentation.

Coding Method. We use the typology proposed by Brehmer and Munzner [15] to establish deductive codes for interaction types. We focus on the "manipulate" interactions in the typology, which summarize and align closely with other interaction taxonomies for information visualization (e.g., [51]). Brehmer and Munzner define six "manipulate" interactions and a separate "encode" interaction, which we use in our analysis:

- `encode`: change the encodings
- `select`: hover, click, lasso, or otherwise highlight marks
- `navigate`: pan, zoom, rotate
- `arrange`: reorder axes, change spatial layout
- `change`: alter/format the visualization (not the encodings)



No *aggregate*

Figure 3: Total observations (y axis) for all interaction types from our taxonomy (x axis) in the qualitatively coded Stack Overflow data.

- `filter`: include/exclude data records
- `aggregate`: group, adjust granularity.

We stress however that this is only a subset of the "how" level of the Brehmer and Munzner typology, and expert D3 implementationers likely implement the full range of interactions, which include not only "encode" and "manipulate" interactions but also "introduce" interactions (`annotate`, `import`, `derive`, `record`). We focus on "encode" and "manipulate" interactions from the typology because they are likely to be most familiar to Stack Overflow users, but we also mention coverage of "introduce" interactions in our analysis.

Mostly *select* or *navigate*

Stack Overflow Users Favor a Few Interaction Types. Our results are shown in Fig. 3. We observed "encode" interactions, five of six "manipulate" interactions, and all four "introduce" interactions. "Manipulate" interactions (20.4% of posts) were far more prevalent than "encode" (1.2%) and "introduce" interactions (1.1%). Given our results for visualization types, "manipulate" interactions are probably more popular and thus discussed more online. However, just two of the 11 interaction types (`select` and `navigate`) represent over 83% of our observations.

We see some Stack Overflow users combining interactions in their interfaces. Here is one example, where a user aims to display specific statistics on hover `selection`s, while also supporting `filter`ing:

> "*... I want my charts to have the total Overall Packages number (Object) available so I can display that in the tooltip, etc. This object will not change during the filter, I just need that total.*" (post A-528)

However, Stack Overflow users tend to focus on one key interaction type in their code, often `select` or `navigate`.

The Interactions in the D3 Documentation Match Observed Interactions on Stack Overflow. To understand how the D3 documentation reflects the prevalence of certain interactions, we analyzed the ~~the~~ D3 API Reference. Three "manipulate" interactions seem to be well-represented in the documentation (with quoted topics in parentheses): `select` ("Search", "Brushes", "Selections"), `filter` ("Brushes", "selection.filter"), and `navigate` ("Dragging", "Zooming"). Other interactions such as `change` have relevant topics (e.g., "Time Formats"), but the documentation focuses on how to implement their functionality as immutable visualization components, rather than manipulable interaction widgets. These results suggest that the D3 documentation emphasizes a small subset of "manipulate" interactions, which correlates to a narrower range of interactions being discussed among Stack Overflow users.

Takeaways. Not all users describe their challenges on Stack Overflow with the same quality and specificity [38]. However, we do see some general trends. For example, Stack Overflow users seem to favor a narrow subset of interaction types. We observe only five of seven "manipulate" interactions and one out of four "introduce" interactions from the Brehmer and Munzner typology. Subsequent analyses of the D3 documentation suggest that these interaction types are not described equally (if at all), suggesting that some interaction types are not prioritized among Stack Overflow users. Taking these ideas a step further, the D3 documentation could potentially be a *driver* of this skew in user preferences, leading to visualization design bias among Stack Overflow users. However correlation does not equal causation; we leave quantitative evaluation of these hypotheses for future work.