Exploring Visualization Implementation Challenges Faced by D3 Users Online

LEILANI BATTLE, University of Washington
DANNI FENG, University of Maryland
KELLI WEBBER, University of Maryland

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 a new method for evaluating visualization languages. Our method emphasizes the experiences of users as observed through the online communities that have sprouted to facilitate public discussion and support around visualization languages. We demonstrate our method by analyzing D3 implementation workflows and challenges discussed on Stack Overflow. 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/?view_only=2cdecfd00f1948a98064b381bde27c18.

CCS Concepts: • Human-centered computing → Empirical studies in visualization; Visualization design and evaluation methods.

ACM Reference Format:
Leilani Battle, Danni Feng, and Kelli Webber. 2021. Exploring Visualization Implementation Challenges Faced by D3 Users Online. 1, 1 (September 2021), 21 pages. https://doi.org/10.1145/nmnnnn.nn

1 INTRODUCTION
Visualization languages provide great flexibility in programming and reusing visualization designs [13, 57], and are gaining in popularity. Browser-based languages in particular have made it easier for a wide range of people to experiment with programming visualizations online [6, 13, 53], one of the most influential being D3 [13]. On the one hand, D3 is wildly successful: it has been starred over 98 thousand times on GitHub and has been identified on many thousands of webpages, including those of highly regarded media outlets [6]. On the other hand, D3 is also known for its steep learning curve, and can be challenging for fledgling analysts and data enthusiasts to adopt [41, 52]. D3 has made incredible contributions to the visualization community and the world, and like the developers of D3, we simply seek to answer the question: what more can we do to provide these users with the best visualization experience possible?

Many systems (e.g., [37, 40, 68]), recommendation engines (e.g., [19, 27, 43]), and languages (e.g., [53, 64]) have been developed to ease the programming burden of visualization users. Though powerful, they require the user to change or learn entirely new implementation processes, which may not match the user’s goals. For example, if a user is focused on improving her skills with a popular and expressive language like D3, then these other tools will be of limited use.

Authors’ addresses: Leilani Battle, leibatt@cs.washington.edu, University of Washington; Danni Feng, fengdn@terpmail.umd.edu, University of Maryland; Kelli Webber, kwebber1@terpmail.umd.edu, University of Maryland.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.

Manuscript submitted to ACM
Instead, we could investigate the challenges 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 [54], and existing languages adopt a similar support structure (e.g., Vega [54] and Vega-Lite [53]). By investigating how to address implementation challenges for D3, similar improvements could be propagated to other languages.

To understand the challenges that D3 users face, we first need to collect relevant data on users’ implementation workflows. However, we observe a dearth of academic corpora regarding how users interact with visualization languages such as D3. Given the thousands of GitHub issues and 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 [35]. In this work, we focus on identifying high level challenges that users consistently face regardless of D3 version, because a persistent issue in a mature language like D3 likely points to exciting directions for future visualization research.

In this paper, we make a first step towards filling this gap by presenting a new approach for evaluating visualization languages in a scalable and data-driven way. Our approach focuses on mining data from online forums to learn: (1) common visualization implementation workflows, (2) high level challenges faced when integrating languages like D3 into these workflows, and (3) opportunities to address these challenges with new features that could integrate with existing user workflows. We apply our approach to evaluate D3 using 37,815 posts collected from Stack Overflow. To ground our analysis, we evaluate how D3’s original design goals have been realized among users to improve compatibility, debugging, and performance [13]. We find that D3 seems to be used often for compatibility reasons, is straightforward to debug on an individual operators level, and is sufficiently fast for most Stack Overflow users. However, the way in which D3 is presented online could be hindering its accessibility for new users, which we observe through recurring challenges with reusing existing D3 examples, as well as through the relatively low fraction of D3’s design space that is regularly used and discussed online. In summary, this paper makes the following contributions:

1. We present a new approach to evaluating visualization languages, where we qualitatively and quantitatively analyze the experiences of D3 users observed in online forums such as Stack Overflow and GitHub. To the best of our knowledge, this approach to evaluating visualization languages is the first to center on online forums, which are widely known to be a critical resource for users of D3 (and visualization languages in general).

2. Through our analysis, we provide: a collection of common implementation workflows discussed by D3 users on Stack Overflow, common challenges users face when integrating D3 into these workflows, and common debugging patterns for D3 programs.

3. We contribute empirical support for many “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, how the D3 documentation and example galleries influence the way users implement new D3 visualizations, and persistent barriers to finding relevant D3 examples online.

4. Based on our findings, we identify new areas for future research in visualization languages (see subsection 7.4). 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
2.1 Analyzing Online Forums.
A number of projects investigate the behaviors, attitudes and impacts of engagement in various online Q&A communities. However, we look specifically at the intersection between analysis of online Q&A communities and visualization implementation workflows and behaviors. Bosu et al. quantitatively measure how Stack Overflow users build their reputations through question structure and quality [14]. Ravi et al. find that higher question quality leads to greater popularity and higher impact questions [47]. Zhang et al. evaluate the quality of code posted on Stack Overflow [67], and find that one third of code examples may contain API usage errors. Jones et al. qualitatively analyze a sample of Reddit posts and use their findings to execute a larger quantitative analysis of how science is communicated to and discussed among the general public in online forums [30]. Kauer et al. qualitatively analyze a sample of the dataisbeautiful subreddit to understand how the general public reacts to shared visualizations [32]. Mack et al. analyze Reddit posts to identify challenges in scaling up data analyses in Microsoft Excel [39]. These findings provide motivation for our qualitative and quantitative analysis methods, which can help to detect and mitigate the effects of variance in question quality across posts. However, we also provide novel results regarding how D3 users communicate visualization implementation challenges that have not yet been observed in existing work. For example, Treude et al. [61] utilize a hybrid analysis approach to investigate how questions and answers are structured, and identify common question archetypes. However, they consider implementation challenges to be mutually exclusive, whereas our findings suggest that the implementation challenges of visualization users are often overlapping and interconnected in important ways.

Others consider what impact online Q&A communities and discussion forums can have on individual users [3, 4, 20, 42, 59]. We observe similar phenomena as those reported in prior work, such as notable proportion of unanswered questions [59], supporting the reliability of our approach and findings. However, given our focus on visualization, we can translate the significance of results from these broader studies in terms of impact on D3 users specifically, and potentially provide valuable insights for visualization language users in online communities as a whole.

2.2 Modeling the Visualization Implementation Process.
Existing models of the visualization implementation process, such as those proposed by Grammel et al. [22] and Isenberg et al. [29], improve our understanding of users’ mental models when implementing visualizations, and how barriers in the process can hinder users’ abilities to achieve their goals. We present a new web mining and analysis approach for investigating the D3 implementation workflows of Stack Overflow users, and challenges encountered by these users when integrating D3 into existing workflows. Our new analysis approach could help the visualization community to construct analogous models of user behavior for visualization languages like D3 in the future.

2.3 Visualization Mining, Modeling, and Analysis.
Several projects highlight the value in extracting and analyzing existing designs from the web, such as website designs [34, 49], visualization designs [6, 8, 23, 24, 27, 51], and even code structure [25]. Online platforms such as bl.ocks.org [12] and Many Eyes [63] also make it easier for user communities to engage with visualizations online. A number of works analyze visualizations as images to automatically extract information about the author’s implementation decisions [6, 23, 24, 28, 31, 46, 55]. 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.

A number of projects focus on developing visualization recommendation systems [18, 33, 40, 62, 65]. Given an input dataset, these systems generate a range of visualizations using various heuristics, such as maximizing attributes explored...
or finding significant deviations within particular sub-populations [62]. Other techniques apply machine learning techniques to users’ visualization specifications to learn how to generate new visualizations [16, 19, 27, 38, 43]. These visualization mining (e.g., Beagle [6]) and automated design (e.g., VizML [27], Draco [43]) approaches focus on generating recommendations, which often emphasize visualization strategies that may not necessarily match existing D3 development and implementation processes. In contrast, our approach can shed light on what StackOverflow users find difficult about using languages like D3, identifying usage and adoption barriers in a data-driven way. As a result, our findings could guide the design of automated techniques to identify and fill specific blind spots in a user’s visualization knowledge in a way that aligns with existing user implementation workflows. To the best of our knowledge, this paper is the first to programmatically analyze users’ D3 implementation workflows by mining posts from online forums.

2.4 Bl.ocks.org and Observable
Bl.ocks.org [12] was the definitive repository for existing D3 examples until 2019; it has since been deprecated and replaced by Observable [44], an online service that also provides an interactive notebook environment for developing D3 (and JavaScript) programs. Both bl.ocks.org and Observable were conceived by Mike Bostock, the principal creator of D3, and are widely known for being invaluable resources for locating relevant D3 examples. That being said, bl.ocks.org and Observable are not panaceas. The main limitation of both services is that their search features emphasize a programming rather than visualization design perspective [26]; they either assume the user wants to search by low-level D3 function calls (bl.ocks.org) or a basic text search (Observable), neither of which can fully capture the semantics behind a user’s visualization design and implementation decisions. Furthermore, there is no guarantee that the user will actually know which code snippets within a particular D3 example are relevant and which are not, an issue particularly challenging for D3, since multiple libraries are often used alongside it. Thus there are still interesting opportunities to enhance the way D3 users search for and reuse existing visualization examples, which we elucidate through our analyses.

3 ANALYSIS OVERVIEW
In this section, we present the motivation and steps behind our new analysis approach for evaluating visualization languages such as D3 using posts from online communities.

3.1 Analysis Goals
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 of compatibility and debugging to drive our analysis [13]. However, there is a difference between D3’s capabilities in theory, and how D3 supports the average user in practice. To assess this difference, we also incorporate a third goal of analyzing the breadth of visualizations observed on Stack Overflow. We summarize each goal as a concrete question to address through our analysis:

1. **Compatibility**: How is D3 used in conjunction with other tools and environments?
2. **Debugging**: How do users explore and interpret the behaviors of D3 code?
3. **Design Breadth**: How does the breadth of users’ visualizations compare to the theoretical breadth of D3?

3.2 Data Collection
To answer these questions, we mined the web for D3 users’ comments and questions. Previous work finds that searching the entire web for visualization content is highly labor intensive and yields few useful results [6]; instead, the proposed alternative is to mine content from information-rich “islands.” We discuss two analogous islands, which are well-known
resources for seeking help with implementation challenges: StackOverflow and GitHub. In our supplemental materials, we show how a third (Reddit) can also be evaluated with our approach. We downloaded all relevant Stack Overflow posts as of January 2021 using the Selenium [2] and Beautiful Soup [48] Python libraries. We used Stack Overflow’s tag search to limit the corpus to posts that include the “d3.js” tag. 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.

3.3 Summary of Analysis Steps
The major steps of our analysis process are depicted in Figure 1. We employ a hybrid analysis strategy, where we use manual coding to capture important qualitative features as codes, and programmed scripts to systematically measure quantitative features derived from our qualitative findings.

To answer each research question, we apply the same high-level 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 the observed results.

For example, to analyze design breadth using these steps, we: (1) **explore** our randomized sample and qualitatively code mentions of visualization types, e.g., mentions of trees are coded as “tree” (see Figure 1); (2) **filter** relevant Stack Overflow posts and GitHub issues programmatically, using keywords derived from our qualitative findings, e.g., “tree,” “scatterplot,” “voronoi,” etc.; (3) **count** the observations of each visualization type from the posts that pass our filters; and (4) **compare** our results with the distribution of visualization types in the official D3 example gallery to understand the influence of existing examples on visualization design choices.

Note that for the majority of our qualitative findings from the **explore** phase, we verify our results quantitatively using the full corpus. For the few exceptions, these analyses could not be automated, such as manually labeling posts by the interaction types discussed. In these cases, we verify our results against GitHub data or the D3 documentation.

3.4 Qualitative Analysis Details
Given the manually-intensive nature of qualitative analyses, they are often performed on a relatively small number of data points [35, 60]. In line with prior qualitative studies in visualization and HCI (e.g., [17, 30, 39]), 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. We manually coded all 817 Stack Overflow posts along the 32 dimensions in Table 1, which include both deductive and inductive codes. Deductive codes were used for known categories in visualization, such as visualization types (e.g., [6]) and interaction types (e.g. [15]). Inductive codes where developed through review of the data. We used a multi-phase coding process. In the first phase, all three coders independently
Table 1. Codes for our qualitative analysis of Stack Overflow posts (total observations in parentheses).

| 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) |

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; new potential codes were carefully evaluated and discussed, and any agreed upon codes were added to the codebook.

3.5 Paper Structure
We cluster our analyses by the D3 implementation goals listed in subsection 3.1, and discuss our results in the following sections. Each analysis section is organized as follows: first we review our research questions, then we discuss a series of analyses executed to answer these questions, using the steps described above, and finally we summarize our findings and highlight major takeaways. To contextualize our findings, we also share direct quotes from Stack Overflow posts; these are indexed by the corresponding file and row of our dataset.

4 COMPATIBILITY: CHALLENGES IN INTEGRATING D3 WITH OTHER TOOLS
We explore the compatibility of D3 through three research questions:
- **Q1**: How are Stack Overflow users integrating D3 into their existing visualization implementation workflows?
- **Q2**: What compatibility issues do Stack Overflow users run into, and what may be the cause(s) of these issues?

To answer these questions, we analyze mentions of external APIs and tools in our random sample, and characterize the types of issues that Stack Overflow users seem to encounter in related posts. Then, we compare our observations with mentions of external tools in the D3 GitHub issues and targeted analyses of our full Stack Overflow corpus.

4.1 Q1: Analyzing Users’ Visualization Toolsets
As we qualitatively analyzed our Stack Overflow sample, we noted mentions of other tools used alongside D3. We found 55 different languages, tools, and libraries mentioned in conjunction with D3.js, a sample of which are shown in Figure 2. 23.4% of all posts we analyzed referenced at least one other external library or tool.

Comparing with the full Corpus and GitHub. To see if these frequencies suggest a larger pattern, we compare with observations of the same tools in the full Stack Overflow corpus. Figure 2 shows the percentage of posts that mention the 11 most frequently discussed libraries and tools. Other than a small difference in the percentage of posts for React (2% versus 5%), the random sample and full corpus have very similar distributions. We also compared our findings with open and closed GitHub issues for D3, and see similar relative frequencies across the top tools. For example, we see that Angular and jQuery are mentioned more frequently than crossfilter and leaflet. Thus, our findings seem to point to broader patterns in how D3 is being used in conjunction with other tools.

In the remainder of this section, we highlight a few interesting observations from our qualitative analysis.

D3 is Often Part of Larger Web Applications. 34 of the 55 (or 61.8%) the tools we observed on Stack Overflow are JavaScript Libraries. We find that React.js is mentioned the most, followed by Angular, dc.js, jQuery, and NVD3. Our findings suggest that D3 is often integrated into larger applications, often using React, Angular, or Vue to manage front-end components, and Python, Node, Ruby on Rails, Electron, or Java Spring to run the full application. However,
Exporing Visualization Implementation Challenges Faced by D3 Users Online

Fig. 2. The most common external tools mentioned in D3 posts. The x axis and color encodings represent the dataset analyzed: the “coded” Stack Overflow sample, “full” Stack Overflow dataset, or “github” issues for D3 that mention the given tool.

React is only mentioned in 5% of our coded posts. The wide range of observed tools but universally low percentages of Stack Overflow posts per tool suggest that Stack Overflow users have many different contexts within which they use D3. In other words, though there are a few popular application structures, Stack Overflow users vary significantly in which libraries and tools they use in conjunction with D3.

* D3 is Paired With Specialized Visualization Libraries. Though D3 provides basic support for geographic maps and graphs, we find some users attempting to integrate D3 with libraries specializing in these visualization types, 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 With Other Languages. 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.

* Some Users Prefer Simpler Libraries Over D3. Of the 34 JavaScript libraries we observed, we found that 10 of them (or 29.4%) are separate visualization libraries built upon D3. The most popular examples were dc.js, NVD3, C3, and crossfilter. Crossfilter and dc were developed to improve the performance of dynamic queries in D3. But others such as C3 seem to have been explicitly developed to help people avoid writing D3 code. For example the C3 website says: “C3 makes it easy to generate D3-based charts by wrapping the code required to construct the entire chart. We don’t need to write D3 code any more.” [1]

Our findings suggest that some Stack Overflow users want the benefits of D3 without the complexities of writing D3 code, highlighting potential frictions in integrating D3 into their implementation workflows.

* 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 in D3 (e.g., posts D-73 and A-296 from our dataset files, respectively). Three different 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 Q2: Assuming Visualization is the User’s Focus May Clash With Existing Implementation 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 [56]. 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...
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 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 (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 more of a modular collection of data manipulation and visualization libraries [9], likely in response to these challenges.

D3’s evolution in response to user implementation challenges provides critical context for how visualization languages and tools should 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, 58]. This lack of broader awareness could lead to less functional tools, and ultimately lower adoption and impact for visualization work.

4.3 Takeaways.

In response to Q1, we see a wide range of libraries and tools used as part of larger visualization implementation workflows, from specialized visualization libraries, to direct competitors to D3, to other languages such as Python and R, and even graphics editors and spreadsheets. In response to Q2, 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 user’s 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 for this design shift [58], since they enable researchers to observe how people use visualization tools in conjunction with other tools over time.

5 DEBUGGING: CHALLENGES IN INTERPRETING & APPLYING D3 CONCEPTS

We explore how Stack Overflow users implement D3 visualizations and debug D3 code through two questions:

- Q3: How do Stack Overflow users reason about, implement and debug D3 visualizations?
- Q4: What challenges do Stack Overflow users encounter as they debug their D3 visualizations?

5.1 Q3: Analyzing Implementation & Debugging Methods

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.

5.1.1 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.
5.1.2 Scales, axes, and Data Updates May Still Confuse Some Stack Overflow Users. Stack Overflow users often post about specific D3 methods that are causing bugs, which we labeled in our coded dataset. We find that 19 of our observed operator class codes appear more than once. The top eight operator classes we observe are: select (13), scale (10), exit (5), axis (4), enter (4), update (4), nest (3), stack (3). Given that selection is required to perform most other operations in D3, we ignore it in our analysis. Considering the remaining operators, these preliminary results suggest that creating and formatting scales and axes, as well as creating and updating existing data bindings, may still be confusing for some users and thus key contributors to D3 bugs.

Comparing with the Full Corpus. We verify these results with the full Stack Overflow corpus, where the top ten D3 calls are: select/selectAll, scale.linear/scaleLinear, max, svg.axis, csv, axisBottom/axisLeft, scale.ordinal/scaleOrdinal, extent, range, and scaleBand. Except for missing enter-update-exit calls which are normally chained to earlier D3 calls, these results are consistent with our qualitative findings. We also see clusters of keywords associated with our original operator classes. For example, svg.axis, axisBottom, and axisLeft align with axis, and scale.linear, scaleLinear, scale.ordinal, scaleOrdinal, and scaleBand align with scale. Furthermore, max, extent, and range are often used when creating scales and axes in D3.

Comparing with GitHub. We also analyze the relative frequency of these terms on GitHub. We find that scale is mentioned most frequently (in 17% of GitHub issues), followed by update (14%), format (10.5%), axis (8.4%), enter (5.8%), and exit (2.5%), which are consistent with our Stack Overflow results. For example, we find that many GitHub issues seem to be primarily about formatting axes and axis labels.

5.1.3 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). These findings suggest 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 [44]. 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.

Comparing with the Full Corpus. To see if these findings point to a larger pattern, we also 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 the D3 examples and documentation seem to be key components of users’ visualization implementation workflows when using D3.
5.2 Stack Overflow Users Struggle to Find and Use Relevant Examples to Fix their Bugs

Adapting existing D3 examples seems to be an important but also complicated part of the D3 implementation and debugging process. To provide additional context for users’ debugging challenges, we qualitatively analyze what Stack Overflow users struggle with when trying to use existing D3 examples.

5.2.1 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 [26] 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.

5.2.2 Complicated, Irrelevant Functionality Makes Examples Hard to Reuse. The closer an example visualization is to the user’s target visualization, the easier it will be to reuse this example. For this reason, the utility of a D3 example depends 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. 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 commenter 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.

5.2.3 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. Figure 3 provides examples from five different Stack Overflow posts. For example in Figure 3-E, the user circled a specific visualization within an existing image (online weather information) that they want to copy. In Figure 3-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 [7]). Some users would even go so far as to draw the specific changes they expect to make. In Figure 3-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., Figure 3-A and B).

Manuscript submitted to ACM
5.3 Takeaways.
In response to Q3, we find that Stack Overflow users often post about unexpected behaviors from runnable D3 code, rather than explicit errors. At the individual operator level, these bugs often involve issues in formatting scales and axes, as well as in processing data and updating data bindings. 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.

In response to Q4, we find that Stack Overflow users struggle to find the most relevant D3 examples to inform their implementation and debugging process. When users do find relevant examples, they may still struggle to extract the most relevant functionality. 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. We also find that Stack Overflow users also struggle to pinpoint the root causes of observed errors or undesired behaviors within the corresponding D3 code.

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 intuitively communicating their desired visualization outputs by sharing annotated images, videos, and .gifs. If Stack Overflow users could just demonstrate their desired changes by directly manipulating the visualization output (e.g., [36, 50, 68]), then they could debug their code using a search-by-example process, rather than a language or keyword-based search. Visualization tools could then compute the corresponding implementation deltas and update the code automatically, rather than making users update the code directly. Debugging via direct manipulation has been proposed in other contexts, such as for debugging SQL queries [21], and could be adapted for visualization languages.

6 DESIGN BREADTH: CHALLENGES IN HARNESSING THE EXPRESSIVENESS OF D3
Visualization languages are frequently evaluated by their expressiveness, in other words by the breadth and depth of the visualization and interaction design spaces that they represent [13, 41, 53]. However, theoretical expressiveness rarely matches what the average user does in practice. In this section, we seek to understand the scope of this gap between theory and practice for D3, and how we might work towards closing it. For example, to understand whether the expressiveness of D3 has translated to many novel visualizations from Stack Overflow users, we can compare the frequency and breadth of visualization types observed for D3 on Stack Overflow and in the academic literature. If we observe a significant gap between what D3 enables and what users actually do, we can investigate potential causes by analyzing D3’s support infrastructure, such as by analyzing how these different visualization types are supported by existing examples and the D3 documentation. We can perform a similar analysis for interaction types. We focus on two questions in this analysis regarding visualization and interaction design breadth:

- Q5: What are D3 users’ visualization preferences and how do they relate to implementation challenges in D3?
- Q6: What are D3 users’ interaction preferences and how do they relate to implementation challenges in D3?

6.1 Analyzing Observed Visualization Breadth
6.1.1 Analysis Method. We focus our analysis of visualization breadth on visualization types. Our goal is to understand how existing taxonomies compare with the breadth of visualizations discussed on Stack Overflow. We define visualization breadth as the total distinct visualization types observed, which we base on existing evaluations of visualization and interaction design coverage (e.g., [53]). We use the visualization taxonomy observed by Battle et al. [6] as our starting point, because it is based on visualizations shared by users of D3, as well as other tools such as 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
Fig. 4. 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.

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, icicle, and horizon). The final taxonomy includes $24 + 8 + 4 = 36$ visualization types.

To analyze visualization breadth, we labeled Stack Overflow posts 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 and interaction types observed for Stack Overflow, existing taxonomies, the D3 gallery and documentation, and GitHub issues reported for D3. Finally, we compare the visualization and interaction types observed, and discuss the significance of observed differences.

6.1.2 Stack Overflow Users Favor A Handful of Visualization Types. 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 Figure 4 (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. [6]. 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.

6.2 Q5: Stack Overflow Users May be Exhibiting Design Bias in Preferences for Visualization Types

6.2.1 Complexity Does not Drive User Interest on Stack Overflow. However, choosing a visualization type is one of many decisions in the visualization implementation process. We also investigated whether the mention of simpler visualization types may be due to complexity elsewhere in the visualization, such as from implementing interactions, e.g., “... when I change the zoom the SVG also change the location” (post A-374). Here, we focus on bar charts as a case
study. We found that only 20.7% of bar chart-related posts discuss interactions, and only 17.0% of these posts were made by self-professed D3 newbies (34.1% total). Thus, we cannot fully explain the skew towards basic visualization types on Stack Overflow simply by pointing to novices, nor to more advanced functionality elsewhere in users’ visualizations. Stack Overflow users seem to prefer simpler visualization types, regardless of their programming and D3 expertise.

Despite their complexity, trees and graphs combined are the second most frequently mentioned visualization type on Stack Overflow. This result contrasts with prior work, where graphs/trees do not even appear in the top five visualization types observed for D3 or other common visualization tools [6]. Thus complexity does not seem to be a driver for which visualization types are discussed on Stack Overflow.

6.2.2 Stack Overflow and GitHub Users Focus on Somewhat Different Visualization Types. We compare our findings with reported GitHub issues for D3 to observe how they translate across user groups. Using the search bar on the D3 GitHub issues page, we searched for mentions of each visualization type in our extended taxonomy. The results are shown in Figure 4 (right). 31 out of 36 visualization types were observed in the GitHub issues (or 86%). Some of the top visualization types observed in our coded dataset are also observed in the GitHub issues. A couple of visualization types are consistent between Stack Overflow and GitHub users. For example, the popularity of graphs/trees is similar to our results for our coded dataset, and the prevalence of maps is similar to the findings of Battle et al [6]. However, in general observations on GitHub do not match our coded dataset. For example, bubble and scatterplot visualizations are the ninth and thirteenth most common visualization types on GitHub, rather than the fifth and sixth most common as seen on Stack Overflow. Furthermore, more specialized types such as chord visualizations are mentioned more frequently on GitHub. Thus, the experiences of Stack Overflow users are similar to, but may not necessarily match the implementation challenges encountered by users posting GitHub issues for D3.

6.2.3 Observed Visualizations on Stack Overflow Match the D3 Gallery. We then compared the occurrence of visualization types in our coded dataset to those in the D3 example gallery. We labeled each visualization observed in the D3 gallery visual index using our extended taxonomy; the results are shown in Figure 4 (left). 30 of 36 visualization types were observed in the D3 gallery visual index (or 83%). Similar to our findings on GitHub, graphs (and trees) are the most popular visualization type in D3 gallery visual index, along with maps, bar charts, and line charts.

When we combine our findings across GitHub issues, the D3 gallery, and in previous work [6], our results paint a more holistic picture of users’ design choices. We see that line and bar visualizations dominate because they are universally popular across visualization tools, as shown in the work of Battle et al. [6], 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 further support the idea that D3 users seem to 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. There seems to be an interesting cycle of dependence established between D3 users and developers through the documentation and through GitHub in general.

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.

6.3 Takeaways for Visualization Breadth Analysis
In response to Q5, Stack Overflow users seem to favor a few simple visualization types, but existing observation-based taxonomies cannot fully explain our results [6]. For example graph and tree visualizations are discussed more on
Stack Overflow than they are seen shared online [6], but their popularity on Stack Overflow is consistent with the prevalence of graph and tree visualizations in the D3 gallery. These findings suggest a strong connection between Stack Overflow users and the D3 developers, captured through the D3 documentation. However, these findings suggest that if the D3 documentation is skewed to prioritize specific visualizations, it could potentially introduce bias into users’ implementation workflows.

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. We explore these ideas further in the next section.

6.4 Analyzing Observed Interaction Breadth

6.4.1 Analysis Methods. In this section, we dive deeper into users’ design preferences, specifically for interaction types. We define interaction breadth as the total distinct interaction types observed, such as selection, filtering, or navigation. We use the typology proposed by Brehmer and Munzner [15] for our analysis. We focus on the “manipulate” interactions in the typology, which summarize and align closely with other interaction taxonomies for information visualization (e.g., [66]). 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)
- 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 implementers 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.
6.4.2 Stack Overflow Users Favor a Few Interaction Types. Our interaction breadth results are shown in Figure 5. 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 while also supporting filtering:

"... 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.

6.5 Q6: Existing Support Structures for D3 May Mirror Rather than Enrich Users’ Design Thinking

Given that Stack Overflow users seem to rely heavily on the D3 documentation, we analyze how the documentation relates to users’ understanding of interactions, specifically through terminology.

6.5.1 Certain Terms are Associated with Specific Interaction Types. To better understand what terms are often used to describe interactions, we calculated the top five term roots that appear in Stack Overflow posts labeled with an interaction type. For example, we see in Figure 6 that “zoom”, “drag”, and “pan” are common descriptors for navigation interactions. “Tooltips” seem to be an important topic of discussion for select interactions. In contrast, few terms seem to be associated with filter interactions, except for “filter” itself.

6.5.2 Stack Overflow Users Describe Interaction Types With Different Levels of Specificity. We can also infer the granularity at which Stack Overflow users reason about specific interaction types by analyzing the specificity of the terms associated with each interaction type. When we analyze how interaction types are described, we see interesting differences in specificity. We observe fine-grained and detailed descriptions for the select interaction that showcase a variety of selection strategies. For example, “hover,” “click,” and “drag” are all used to describe the behavior of select. Consider this description of a user’s expected results for hover functionality across two maps:

"... i have two maps (based on topojson-data, visualised through d3), and on mouseover over certain parts of map1, the corresponding parts of map2 should light up." (post A-280)

The user is describing specific JavaScript events relevant to the hover interaction (mouseover), and the expected result.

In contrast, filter interactions seem to be described primarily at a high level (e.g., using terms such as “select” or “filter”), and are not often associated with particular low-level manipulations. Consider the following quote from a user implementing a filter interaction:

Fig. 6. The top five term roots (x axis) counted across all coded posts (y axis) that mention an interaction type (color). Term root refers to the root of a word that appears in one or more posts.
“Currently I have a script that is pulling data and filtering objects in a JSON file. I am trying to filter (show/hide) the restaurants by a star rating of 1-5.” (post C-251)

Here, “filter” describes both the interaction and the intended behavior. Though the desired result is mentioned (i.e., showing or hiding restaurants), the manipulation to execute the filter (clicking on a checkbox) is not clearly specified. However, an embedded image does hint at checkboxes and restaurant rating stars being involved.

6.5.3 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.

Stack Overflow users describe different interactions with varying levels of specificity, and the variations seem to be correlated with how these same interactions are described in the D3 documentation. These results suggest that Stack Overflow users may not understand these interaction types equally well. On the one hand, the documentation could simply reflect the aspects of interactions that Stack Overflow users find confusing. On the other hand, these findings could indicate that the D3 documentation may cause users to understand some interactions, or even parts of interactions, better than others. In either case, we believe these findings point to an interesting echo chamber between Stack Overflow and the D3 documentation.

6.6 Takeaways for Interaction Breadth Analysis

Not all users describe their challenges on Stack Overflow with the same quality and specificity [47]. However, we do see some general trends. In response to Q6, Stack Overflow users seem to favor a narrow subset of interaction types. Specifically, we observe only five of seven “manipulate” interactions and one out of four “introduce” interactions from our taxonomy. Subsequent analyses of the D3 documentation reveal that these interaction types are not described equally well (if at all), suggesting that some interaction types are not prioritized among Stack Overflow users. Given that observations on Stack Overflow seem to correlate with the popularity of visualizations rather than confusion around particular visualizations (see subsection 6.2), we hypothesize that these gaps reflect a lack of user awareness or interest in implementing certain visualization and interaction types. Our results suggest that the non-uniform coverage of interaction types in the D3 documentation, both in terms of occurrence and specificity, may reflect users’ understanding of different interaction types, and thus how clearly Stack Overflow users can articulate challenges encountered when implementing these interactions. Taking this idea one 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 new hypotheses for future work.

7 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 utilized a mixed methods approach, using our 817-post qualitative analysis to direct a more targeted quantitative analysis of the full corpus, and compare our findings with the D3 documentation and issues posted on GitHub. We evaluate D3 from three different perspectives: compatibility, debugging, and design breadth. Our findings show that when we focus on developing visualization languages but not how and where people actually use them, we may fail to fully understand the user experience [5], 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 visualizations. In this section, we highlight opportunities for future work based on our findings.

7.1 Compatibility: Design Visualization Languages to Integrate With Other Tools in Users’ Workflows
By making visualizations the sole focus of visualization language development, we may be encouraging developers to create languages that conflict with other critical tools within users’ implementation workflows, potentially hindering user adoption. As demonstrated through the evolution 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 [58]. 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.

7.2 Debugging: Improve the Language and its Support Infrastructure to Boost User Adoption
We find that even though users on Stack Overflow rely on existing examples to implement and debug new D3 visualizations, they often struggle to find relevant examples and reuse them correctly. Our findings point to two challenges in supporting current debugging workflows. First, we lack easily distinguishable 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 [53]. Second, the problem may not only be D3’s code structure but also insufficient infrastructure for helping users understand the flow of D3 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 [44], 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 do not just support simple text search or search by function name, but also 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, given a corpus of D3 visualizations (e.g. [6]), we could programmatically extract D3 templates for different visualization and interaction types, and automatically segment these templates into interchangeable building blocks. New tools could leverage this metadata to help users delineate separate D3 components within existing examples, and extract only the components that users actually need. We could automate the debugging process further by using crowdsourced data to derive new models for automatically detecting a user’s specific background and goals, as well as potential blind spots and visualization biases. Development environments could then be augmented to automatically show relevant documentation, or recommend extensions to the user’s code, based on inferred user goals, experience levels, and potential biases.

7.3 Design Breadth: Effective Documentation and Example Gallery Design Should be Active Research
Our findings in section 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 poor alternatives. Even Stack
Overflow is insufficient; translating a specific D3 bug into a self-contained Stack Overflow question requires significant time and effort [20], and 37% of the 37,815 posts analyzed were left unanswered by other users. Our results in section 6 also show that the ad-hoc and manual process used to develop D3 examples and documentation has clear blind spots, which could potentially be preserving or even driving user design bias. We believe these issues have persisted for a decade now because the visualization 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. As a first step, we could synthesize current best practices in visualization design as a diverse set of modularized visualization examples that all visualization language developers should aim to provide. Furthermore, rather than expecting the developers to create design galleries meeting these requirements, we could instead find ways to automate the process of design gallery generation itself. This solution could also be 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 themselves with new examples. This approach could also be used to score example quality, so redundant or low quality examples can be detected and replaced automatically. These solutions would help not only D3 users, but visualization language users as a whole. Developing design galleries and documentation takes time, whether for newer open-source languages such as Vega-Lite [53], or commercial APIs such as Plotly [45]. Automating the documentation process could speed up the learning process and dampen learning curves for users of all visualization languages.

7.4 Takeaways Summary
Here we list five major takeaways for the visualization community derived from our research:

- Conduct quantitative studies of how visualization languages are used in various development environments.
- Develop and test visualization languages as part of larger implementation workflows involving multiple tools.
- Treat the development of language documentation, example galleries, and support tools as active research.
- Design example galleries strategically based on analysis of user data, e.g., from the studies recommended above.
- Automate the example gallery generation process, to make it more consistent across visualization languages.

7.5 Limitations and Future Work
One limitation is that we only focus on D3 users who post on Stack Overflow and GitHub, 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. As a result, we have little knowledge of who these users are, their exact motivations, and their past experiences with programming, D3, and visualization design. An interesting direction for future research is to conduct follow up interviews with D3 users from Stack Overflow 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. Given our focus on communication among Stack Overflow users, analysis of the D3 language itself was outside of our analysis scope. However, interesting future work could be to compare the timestamps of Stack Overflow posts with updates to the documentation, as well as to changes made in the design of D3 itself through GitHub commits. 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.
