2017

Seeing Library Data: A Prototype Data Visualization Application for Librarians

Mark E. Eaton
CUNY Kingsborough Community College

Recommended Citation

Eaton, Mark. "Seeing Library Data: A Prototype Data Visualization Application for Librarians." Journal of Web Librarianship 11, no. 1 (2017): 69-78.

Seeing Library Data: A Prototype Data Visualization Application for Librarians

Mark Eaton

Assistant Professor / Reader Services Librarian

Kingsborough Community College, City University of New York

mark.eaton@kbcc.cuny.edu

Abstract

Visualizations can add value to raw library data. Tools that programmatically make such visualizations interactive can further increase the value of this data, by giving librarians visual tools to analyze their data sets. To demonstrate these benefits, Kingsborough Community College Library built SeeCollections (http://b7jl.org/seecollections), a web application that visualizes our libraries' collections of books and ebooks.

This article discusses the how SeeCollections gathers data from Kingsborough's discovery layer API (Application Programming Interface), and transforms this data to create visualizations for the web. SeeCollections is a lightweight, proof-of-concept data application based on a vendor API. API data is often accessible in academic libraries, but is frequently underutilized. SeeCollections was built to demonstrate the immediate value of these API data sources to Kingsborough librarians, by providing a visual interface to discovery layer data. Keeping in mind the broader literature on data visualization, this article will explore how SeeCollections was built.

Keywords

visualization, academic libraries, data, api (application programming interface)

Introduction

Data visualization can bring a great deal of value to librarians' work. Data often have compelling practical uses, and data visualization can provide librarians with important insights. As Stephen Few puts it:

> When we represent quantitative information in visual form, our ability to think about it is dramatically enhanced. Visual representations … make patterns, trends and exceptions in numbers visible and understandable (quoted in Murphy 2015, 482).

Visualization can increase the power of data, by showing the "patterns, trends and exceptions" that Few describes. According to Eric Phetteplace, this "data visualization has a clear purpose: to aid in our understanding of data" (2012, 93). Moreover, as Cole Nussbaumer Knaflic argues, the effective use of data can help drive decision-making and make strong business cases for initiatives (2015a). Librarians can benefit when they visually leverage data in support of library projects.

Beyond producing actionable data, the process of building visualization tools can be useful to librarians in other ways. Nathan Yau suggests that exploratory learning is a significant benefit of data visualization initiatives (2013). We can learn about our libraries by tinkering with data. In addition, handling data can also challenge librarians to improve their technical skills. Visualization projects allow librarians to not only learn about their libraries, but to also learn programming and data science skills.

Academic libraries are well positioned to reap these benefits, because there is usually a great deal of useful data on hand (Chapman and Lown 2010; Chapman and Woodbury 2012). To demonstrate the value of this opportunity, Kingsborough Community College Library has created a web application called SeeCollections that interactively visualizes library holdings of books and e-books. While considering the existing literature of library visualization, this paper will document how and why we built the SeeCollections application. It will explore the technical considerations that went into building SeeCollections, and will conclude by suggesting that it is a useful prototype for further, more elaborative library visualization projects.

Reading the Visualization Literature

Many librarians have already been doing in-depth visualization work and have documented their achievements in scholarly articles (Braun 2015; Morton-Owens and Hanson 2012; Murphy 2013; Murphy 2015). These investigations are primarily case studies that offer interesting visual solutions to data problems, and they have greatly informed this project. However, these articles do not offer much theoretical treatment of data visualization in libraries. Therefore, for theoretical context, fairly rich, recent, technical literature on data visualization practice by non-librarians was examined (Cairo 2013; Knaflic 2015a; Tufte 1990; Wong 2010; Yau 2013). These latter authors offer high-level perspectives on the goals of data visualization, and suggest methodologies that best achieve these goals. I will discuss a few of these non-librarian perspectives on visualization below. These theories of visualization will help frame the SeeCollections project.

The classic voice on data visualization theory is Edward Tufte. In *Envisioning Information*, Tufte unequivocally advocates for multi-dimensionality in visualizations. He praises some incredibly complex paper-based visualizations (1990). However, writing in 1990, Tufte was largely unaware of the coming ubiquity of interactive, digital visualizations. As a result, *Envisioning Information* understandably overlooks programmatic interactivity as a means of representing complexity. Today's technology makes it possible to create interactive web visualizations that offer rich, multidimensional information in ways that were not foreseen by Tufte's model.

Yet not all visualization experts value multi-dimensionality as much as Tufte does. Wong (2010) and Knaflic (2015a) advocate for very sparse visualization aesthetics. Knaflic is more strategic and practical than Wong, arguing that data visualizations should have actionable agendas. Knaflic maintains that visualizations should be expedient, and should focus on advancing organizational objectives. While Wong argues that visualizations should be simple to avoid deception, Knaflic's minimalism is for tactical rather than philosophic reasons. Their minimalist approaches will be useful to librarians working on applied, practical projects.

Yau (2013) and Cairo (2013) both provide useful syntheses between Tuftian complexity and Wong and Knaflic's visually minimalist counterpoints. Yau argues that sometimes data is complex, requiring complex representation. Like Tufte, he suggests that complex visuals need not be immediately understood to be effective. In this respect, he moves away from a strict visual minimalism; however, he moderates his enthusiasm for complexity by insisting that such complexity must be contextual. Cairo likewise balances conflicting approaches, strongly seconding Yau's position.

This discussion suggests that the principles of data visualization are strongly contested. Although Yau's even-handed approach and Cairo's willingness to find common ground are laudable, their positions are not authoritative or the only approach to data visualization. On her blog, Knaflic states bluntly: "Many of the [data visualization] studies that come out demonstrating one thing are opposed via counter-studies that show the opposite" (2015b). The literature on visualization remains fractious.

Yet despite this discord, the authors mentioned above all more or less maintain that visualization can drive insights. While there is often disagreement about the methods and techniques of visualization, there is usually common ground on the value of visualization (Cairo 2013; Knaflic 2015a; Tufte 1983; Wong 2010; Yau 2013). I attempted to build on this shared enthusiasm for visualization by creating SeeCollections, a prototype data visualization tool for libraries.

While this study does not attempt to draw theoretical conclusions about library data visualization, I hope to suggest a practical way forward. This study reaffirms that visualized data can drive insights. Our project's unique contribution to the literature is therefore a technical approach that demonstrates the value of data visualization in libraries. As shown above, this project was informed by the visualization literature from beyond librarianship. This case study seeks to provide an example of how data visualization techniques can be used productively at an academic library. Before looking at the technical aspects of SeeCollections in detail, I will give an overview of how it can be used to manipulate and display library data.

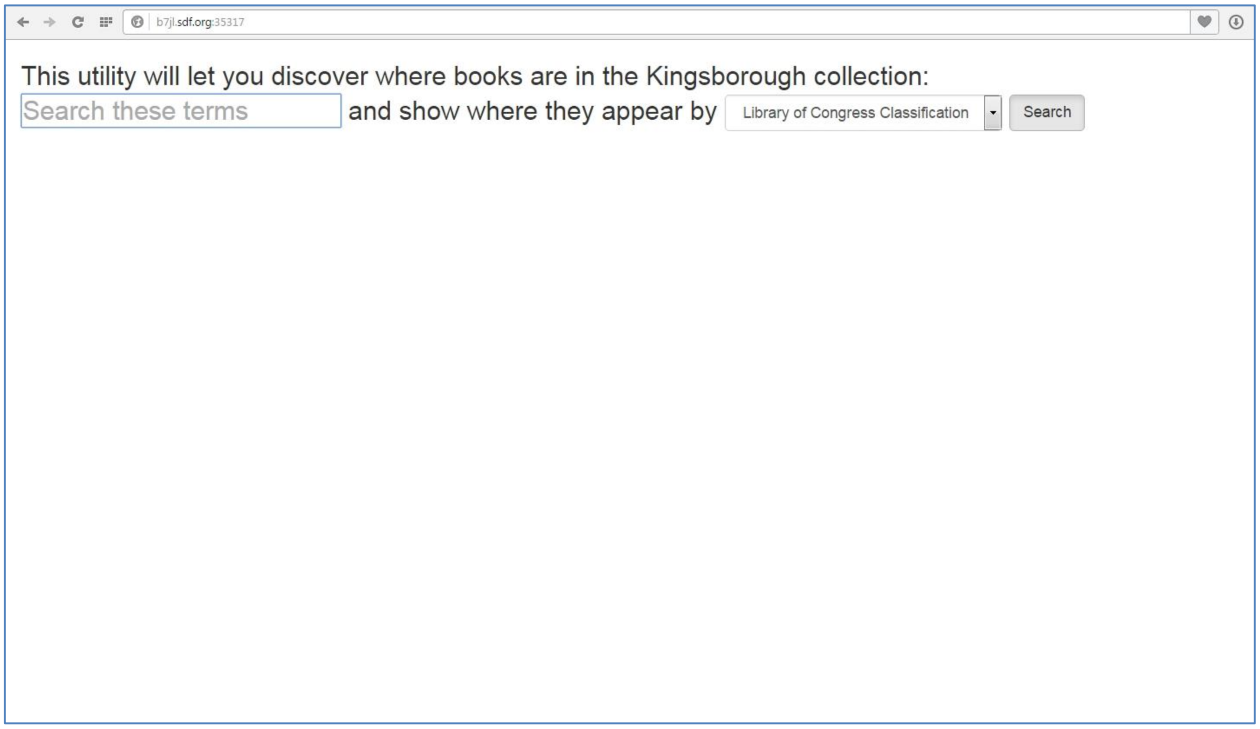
SeeCollections: A Visualization Project

SeeCollections visually represents the book collections of City University of New York (CUNY) libraries. Our college, Kingsborough Community College, is one of 24 CUNY schools. The Kingsborough library has a collection of approximately 161,000 books and 459,000 e-books. This project was developed to give librarians at CUNY colleges a deeper understanding of what is in their collections through visualization. SeeCollections seeks to make intuited knowledge about the collections more tangible by representing these collections visually. Rather than restrict those visual insights to a local network or intranet, SeeCollections is accessible on the public web. While primarily intended for CUNY librarians, the general public and researchers are welcome to use this tool to visualize our library's collections.

SeeCollections is also a prototype, proof-of-concept project, meant to demonstrate the value of data visualization to my colleagues at Kingsborough Community College. Part of the rhetorical agenda of SeeCollections is to show that there is a great deal of useful, interesting, and actionable data available to our library. The project is therefore meant to convey the potential utility of library data sources to CUNY co-workers and peers.

What does SeeCollections do? It is a web application that visualizes the library's holdings of books and e-books according to certain facets and keywords. Users can visualize whatever topics they want, by selecting keywords and facets that interest them. Before discussing the technical processes that make this possible, I will walk through how the application is approached from a user's perspective.

The user interface for SeeCollections is minimal, with a focus on clear presentation. When a user loads the application page, they are prompted to select their college from a dropdown list of CUNY colleges. Once a college is selected, the user is presented with a search screen where they can search the collection by keyword and facet (see figure 1). Keywords can be any combination of words, while a dropdown menu lets the user choose a facet to visualize.


b7jl.sdf.org:35317

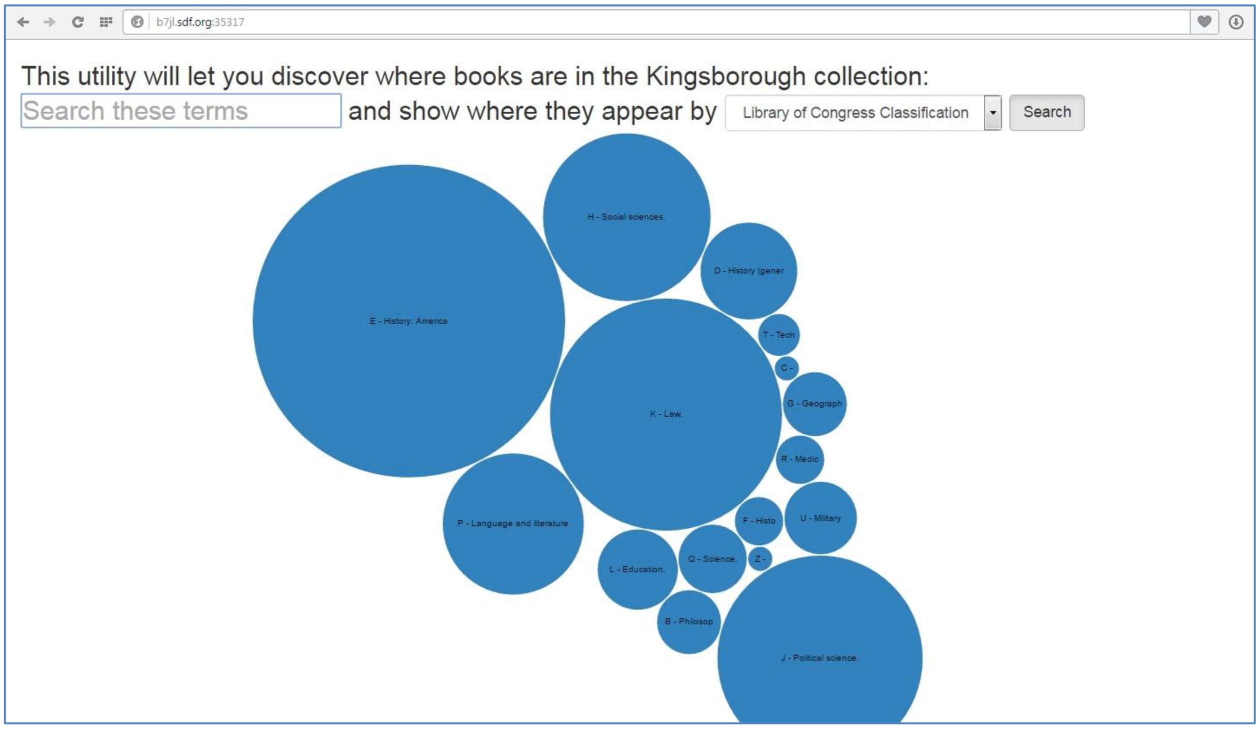
This utility will let you discover where books are in the Kingsborough collection:

Search these terms and show where they appear by Library of Congress Classification Search

FIGURE 1

The facet choices are: (1) Library of Congress (LC) Classification, (2) Creation Date, and (3) Topic. These particular facets were selected because they would be relevant to librarians, and because they were easy to extract from the data. Other readily available facets were not selected, either because they would likely not be interesting to the user, or because they produced visualizations that were not intelligible using the visualization script adapted by this project. For example, visualizations by Language or Resource Type did not produce useful visualizations, because one category within each of those facets overwhelmingly dominated all others.

After the user picks from the facet options, running the search produces a visualization of the number of print and electronic books at the selected college by the selected keywords and facet. A blue bubble represents each category within the facet. A search by the Library of Congress Classification facet, for example, would return bubbles for each relevant top-level LC class (see figure 2). The sizes of the bubbles represent the number of results for each category within that facet. The bubbles are labeled, and hovering over the label displays a tooltip with more details. Only categories with results are visualized. For example, a search by LC class that returns results in both Music and Literature would have correspondingly sized bubbles representing the M and P classes. Thus, SeeCollections works programmatically with data to produce visualizations of the library's book and e-book holdings.


This utility will let you discover where books are in the Kingsborough collection:

Search these terms and show where they appear by Library of Congress Classification

FIGURE 2

Interactivity is built into the application. A librarian can run multiple searches on SeeCollections, with differing keywords or facets, and corresponding visualizations will be generated dynamically. This has important implications: "Suddenly, the creator of a visualization does not have a monopoly over its meaning because the viewer can unveil patterns on their own"

(Phetteplace 2012, 95). The viewer can participate in the production of meaning because the tool can access data on demand, and process it instantly.

For our librarians, interactive visualization has important benefits. It has contributed to problem solving in our library in interesting ways. For example, early on in the development of SeeCollections, a Kingsborough librarian discovered that running a search for “computer science” according to “creation date” at our college produced a visualization that was extremely skewed toward books from 2009. We did some further investigation, and it turned out that the reason for this anomaly was due to some rather uncontroversial cataloging considerations. Despite our original hypothesis, it was not due to a disproportionate number of computer science texts being purchased that year. Although the issue was resolved rather easily and satisfactorily, it is clear that the librarians would have never identified this problem, nor reached these accurate conclusions, without a visual tool to show us these previously unseen patterns in the data.

How Does SeeCollections Work?

SeeCollections uses data from the Primo X-Services API (X-Services 2016). Primo is the discovery tool that is used across CUNY libraries. An API is an “Application Programming Interface.” The API allows librarians to write scripts that can interact programmatically with Primo, helpfully letting them build programs that expand upon the existing functionality of Primo. In particular, this Primo API provides data about our libraries’ holdings in a machine-readable format. This is a particularly useful starting point for library visualizations, as it allows programs to be written that can access a huge amount of holdings data.

However, working with the X-Services API can also be somewhat frustrating, since the documentation (at the time of writing) does not cover all of the API's functionality. Most notably, some parameters that can be passed to the API are not described in the X-Services API documentation (X-Services 2016). As a result, the API documentation was at times an obstacle to the development of SeeCollections. Thankfully the Primo community is very supportive, and CUNY is fortunate to employ librarians with deep knowledge of Primo. The community of users, at CUNY and beyond, was essential to this project's success.

SeeCollections' use of the Primo X-Services API is rather straightforward. The program queries the API on the fly for the specific records that match the user's request. However, it does not tap into the full range of possible query parameters, nor does it make full use of the rather exhaustive data returned by the API. Indeed, some portions of the data returned by the API are simply discarded by the application. In particular, SeeCollections' visualizations draw specifically from the header data returned by the API, rather than from the search results themselves. In this way, SeeCollections takes a fairly minimal approach to API data. Rather than aiming for comprehensive functionality, SeeCollections' use of the API is kept as simple as possible.

This is by design, for a couple reasons: First, this tool is a prototype. It is not intended to exhaust the possible uses of the X-Services API. Second, there is merit to having a tool that is easily explained to librarian colleagues. Using a straightforward API call makes this easier. Librarians – our primary audience – often ask probing questions about where specific data come from. They want clarity about which underlying collections are reflected by the visualization. However, clearly representing diverse Primo data can be a challenge. Keeping the API call simple makes it easier to explain the data underlying the visualization. The intention is to have the data visualization's workings be as transparent as possible.

Structure of the Application

To access the Primo API data, the application takes user input in order to dynamically generate an API call. Primo provides the choice of whether the data should be returned as JSON or XML. For this application, I chose to use JSON, because this is the format that is ultimately required for the front-end visualization script. JSON, or JavaScript Object Notation, is a "lightweight data-interchange format" (JSON 2016) that is easy to manipulate in a number of programming languages. This inter-operability makes it advantageous to retrieve and keep the data in JSON format.

Much of the application is devoted to handling and manipulating JSON, and ultimately displaying that data on the web. This is accomplished using Flask, a very flexible Python web micro-framework (Flask 2016). There are other, more elaborate web frameworks for Python, but Flask is more than sufficient for this project. The minimal overhead of a Flask application was appealing for this project, especially because the project is intended as a lightweight prototype. Flask was also chosen because it allows most of the data manipulation to happen in Python, a language that I am familiar with. Flask is particularly well suited to this project because it makes it easy to integrate Python logic with the web. Flask coordinates all of the various aspects of our SeeCollections application; it allowed me to work on the data on the back end with Python, transform it, and pass it programmatically to the front-end code that delivers the visualization.

Flask uses “routes,” which are functions, to handle HTTP requests. Much of the work in the SeeCollections application is done by a route that handles the data submitted by the user. First, this route parses and validates the user’s input. For example, keyword input is controlled for length, and limited to certain characters with a regular expression. The route then dynamically creates an API call and sends it. It then passes the resulting JSON data to a Python module (which I also wrote) for processing. This module transforms the raw API JSON data into more helpful, better-structured data by checking it and putting it through a series of manipulations. These manipulations are fairly straightforward. For example, they ensure that all of the Python dictionaries present have appropriately named keys; they wrap all of the data in a Python list; they catch and handle exceptions correctly; and so on.

In addition to creating the visualization, SeeCollections also makes this data available on the web. JavaScript is the front-end technology that ultimately presents data to the SeeCollections user. JavaScript is a cornerstone of contemporary web development; a great deal of today’s interactive web content relies upon it. Many popular code libraries have been written for JavaScript. This project draws upon jQuery, Bootstrap and d3.js. These libraries are helpful additions to plain JavaScript, allowing for the execution of complicated operations with relatively simple bits of code.

For example, in SeeCollections, jQuery is primarily used as a way to display messages. These include helpful updates for the user like “Loading your data…”, as well as error messages, when necessary. jQuery is especially useful because it provides straightforward event handling: it can capture users’ interactions with the page and it provides helpful tools for responding to those events (jQuery 2016). This allows for the display of appropriate messages in response to users’ interactions with SeeCollections. For this project, the main benefit of jQuery is that it greatly simplifies the process of writing JavaScript for the web, which is no doubt in part why jQuery is such a popular JavaScript library.

To give SeeCollections a unified visual theme, I have used Bootstrap. Bootstrap is most commonly used to make webpages responsive to different devices (Bootstrap 2016). However, this project did not use much of this responsiveness functionality; instead the Bootstrap library was primarily used to style the elements of the page in a consistent and appealing way. For example, the expandable help menu in the bottom corner of the results screen was entirely styled with Bootstrap. Bootstrap was used because it provides a minimal, easy-to-implement style for our web content.

Most critically, SeeCollections draws upon the d3.js library to display the data. D3.js is an immensely popular library that provides useful building blocks for making data visualizations (Bostock 2016b). D3.js facilitates the binding of data to the content of a web page, which allows manipulation of the web content based on the underlying data. I adapted a d3.js script called “Bubble Chart,” which was originally published by Mike Bostock under the GNU General Public License (2016a). This fairly accommodating license permits the use of his code in this project. Bostock is a prolific creator of d3.js scripts, and a number of his other d3.js visualizations could also have plausibly been used for this project; “Bubble Chart” was chosen based on the simplicity of its JSON data structure and the clarity of the resulting visualization.

The “Bubble Chart” script, as originally written, draws its data from a static JSON file (Bostock 2016a). While this may have been useful for Bostock, using static JSON does not meet the needs of this application. SeeCollections generates visualizations dynamically, and a static JSON file is not ideally suited to this. As a result, I altered the d3.js script so that JSON could be directly passed to the script as a variable. This is possible thanks to Flask's templating feature, which allows the application to drop bits of Python, such as variables and logic, into the front-end code. For this purpose, Flask relies on Jinja2, "a modern and designer-friendly templating language for Python" (Jinja2 2016). Jinja2 allows for the creation of HTML templates that integrate the results of our Python logic. In this way, the restructured JSON data can be directly included in the front-end code that powers the visualization. Bostock's d3.js script is then able to transform this JSON into appealing visualizations for the web.

The crux of the programming problem for this project was getting the JSON from the API to work with the d3.js script. As I figured out this programming challenge, there were a number of corner cases along the way that needed to be addressed (what happens if there is no data, or if only one item from the collection is returned?). Once these were resolved, the Flask application was successfully passing data to the d3.js script.

Conclusion

SeeCollections is an example of a small application for producing data visualizations from a vendor API. It dynamically draws on API data to responsively produce custom visualizations. The basic principles outlined here – structuring an application with Flask, visualizing with d3.js, and drawing data from an API – could be applied to many other library data projects. This is important because librarians usually have access to data through numerous APIs, either freely available or through contracts with vendors. There are substantial opportunities to better utilize these data sources for the benefit of the library community.

Of course, these initiatives require librarian labor. However, Chapman and Lown have convincingly made the point that “although there is undeniably a time-commitment involved in initiating this kind of effort, especially in cases where data … is not a part of one’s usual responsibilities, we think the short term and long term benefits are significant” (2010, ¶25). These benefits include improved library decision-making and expanded librarian skill sets. It is in librarians’ interests to programmatically dig into library data sources, as there are clear benefits in making data more visible.

One function of SeeCollections is as a proof-of-concept project for the Kingsborough library. It is intended to demonstrate the benefits of library data work to colleagues at Kingsborough Community College. This project has shown that useful data projects are within reach for the library, using sources the library already has access to. SeeCollections only scratches at the surface of the data available to the library; more can be done to take advantage of the numerous existing data sources.

SeeCollections may also be helpful and interesting to colleagues who may have no interest in developing digital tools. SeeCollections has proved useful for identifying patterns in holdings data; for example, in analyzing the computer science books in the library’s collection. These data-driven insights have clear value, and SeeCollections makes them accessible to librarians across CUNY and beyond. In this way, the tool may prove useful to CUNY librarians who may be approaching the data from a number of non-technical perspectives.

Lastly, SeeCollections is also meant to open a conversation with librarians at other institutions. Hopefully this project can serve as a point of reference for librarians considering visualization projects for their own collections. To encourage adoption of these ideas and the code, the project is open source and available on GitHub (Eaton 2016). The code is permissively licensed, and can be freely adapted and modified. In the hands of other librarians, ideally this work may lead to further, more well-elaborated applications. SeeCollections is meant to serve as a useful example of a d3.js visualization using a Flask framework, within a library context. It shows that useful data visualization applications are possible, even with fairly simple, librarian-built web applications. Hopefully this project suggests viable paths towards future library data projects.

References

"Bootstrap: The World's Most Popular Mobile-First and Responsive Front-End Framework." 2016. Accessed July 19. https://getbootstrap.com/.

Bostock, Mike. 2016a. "Bubble Chart." Accessed February 23. http://bl.ocks.org/mbostock/4063269.

———. 2016b. "D3.js - Data-Driven Documents." Accessed May 29. https://d3js.org/.

Braun, Steven. 2015. "Manifold: A Custom Analytics Platform to Visualize Research Impact." *Code4Lib Journal*, 30.

Cairo, Alberto. 2013. *The Functional Art: An Introduction to Information Graphics and Visualization*. Berkeley, CA: New Riders.

Chapman, Joyce, and Cory Lown. 2010. "Practical Ways to Promote and Support Collaborative Data Analysis Projects." *Code4Lib Journal*, 12.

Chapman, Joyce, and David Woodbury. 2012. "Leveraging Quantitative Data to Improve a Device-Lending Program." *Library Hi Tech* 30 (2): 210–34.

Eaton, Mark. 2016. "Primo-Visualization." *GitHub*. Accessed May 19. https://github.com/MarkEEaton/Primo-Visualization.

"Flask." 2016. Accessed February 23. http://flask.pocoo.org/.

"Jinja2." 2016. Accessed July 18. http://jinja.pocoo.org/docs/dev/.

"jQuery." 2016. Accessed July 19. https://jquery.com/.

"JSON." 2016. Accessed February 23. http://www.json.org/.

Knaflic, Cole Nussbaumer. 2015a. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Hoboken, NJ: John Wiley and Sons, Inc.

———. 2015b. “The Great Pie Debate.” *Storytelling with Data*. March 31. http://www.storytellingwithdata.com/blog/2015/03/the-great-pie-debate.

Morton-Owens, Emily, and Karen L. Hanson. 2012. “Trends at a Glance: A Management Dashboard of Library Statistics.” *Information Technology and Libraries*, September.

Murphy, Sarah Anne. 2013. “Data Visualization and Rapid Analytics: Applying Tableau Desktop to Support Library Decision-Making.” *Journal of Web Librarianship* 7 (4): 465–76.

———. 2015. “How Data Visualization Supports Academic Library Assessment: Three Examples from the Ohio State University Libraries Using Tableau.” *College & Research Libraries News* 76 (9): 482–86.

Phetteplace, Eric. 2012. “Effectively Visualizing Library Data.” *Reference & User Services Quarterly* 52 (2): 93–97.

Tufte, Edward R. 1983. *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

———. 1990. *Envisioning Information*. Cheshire, CT: Graphics Press.

Wong, Dona M. 2010. *The Wall Street Journal Guide to Information Graphics: The Do’s and Don’ts of Presenting Data, Facts, and Figures*. New York: W.W. Norton & Co.

“X-Services.” 2016. Accessed February 23. https://developers.exlibrisgroup.com/primo/apis/webservices/xservices.

Yau, Nathan. 2013. *Data Points: Visualization That Means Something*. Somerset, NJ: John Wiley & Sons.