Beagle: Automated Extraction and Interpretation of Visualizations from the Web

Leilani Battle
University of Washington
Seattle, USA
leibatt@cs.washington.edu

Peitong Duan
MIT
Cambridge, USA
peitong.duan@gmail.com

Zachery Miranda
MIT
Cambridge, USA
zmiranda@mit.edu

Dana Mukusheva
MIT
Cambridge, USA
d.mukusheva@gmail.com

Remco Chang
Tufts University
Medford, USA
remco@cs.tufts.edu

Michael Stonebraker
MIT
Cambridge, USA
stonebraker@csail.mit.edu

ABSTRACT

"How common is interactive visualization on the web?" "What is the most popular visualization design?" "How prevalent are pie charts *really*?" These questions intimate the role of interactive visualization in the real (online) world. In this paper, we present our approach (and findings) to answering these questions. First, we introduce Beagle, which mines the web for SVG-based visualizations and automatically classifies them by type (i.e., bar, pie, etc.). With Beagle, we extract over 41,000 visualizations across five different tools and repositories, and classify them with 85% accuracy, across 24 visualization types. Given this visualization collection, we study usage across tools. We find that most visualizations fall under four types: bar charts, line charts, scatter charts, and geographic maps. Though controversial, pie charts are relatively rare for the visualization tools that were studied. Our findings also suggest that the total visualization types supported by a given tool could factor into its ease of use. However this effect appears to be mitigated by providing a variety of diverse expert visualization examples to users.

CCS Concepts

•Information systems → Web mining; •Human-centered computing → Information visualization;

Author Keywords

web mining; design mining; visualization classification

INTRODUCTION

The Visualization community has made tremendous strides in the past decade or so in bringing data visualization to the masses, and recently in creating and sharing visualizations online. Tools such as Tableau, D3, Plotly, Exhibit and Fusion Charts have made the design and publication of interactive visualizations on the web easier than ever. Popular venues such as the New York Times, the Guardian, and Scientific American then utilize these tools as an effort to democratize data, resulting in unprecedented advances in data journalism, visual storytelling, and browser-based rendering techniques.

However, while it is generally believed that visualization has reached the public, it remains unclear just how wide the reach is. Although the Visualization research community continues to design and develop new interactive visualization techniques (particularly for the web), we still have little idea as to how frequently these techniques are used. For example, many have advocated for the "death" of pie charts [14, 5, 25], but just how commonly used are pie charts compared to the recommended alternative, the bar chart? We aim to investigate these important questions in the context of rendering and publishing visualizations on the web.

In this paper, we present our approach and findings to these questions for web-based visualizations. In order to mine and classify visualizations on the web, we developed Beagle. Beagle is an automated system to extract SVG-based visualizations rendered in the browser, label them, and make them available as a query-able data store. Beagle consists of two major components: a Web Crawler for identifying and extracting SVG-based visualizations from web pages, and an Annotator for automatically classifying extracted visualizations with their corresponding visualization type.

To date, we have used Beagle to extract visualizations from five different web-based visualization tools and repositories, totaling over 41,000 visualizations. We evaluate Beagle's classification accuracy using our extracted visualization collections. We find that Beagle can correctly classify SVG-based visualizations with 85% accuracy, in a multi-class classification test across 24 visualization types observed on the web.

In an analysis of our visualization collections, we find that the vast majority of visualizations fall under *only four types*: bar charts, line charts, scatter charts, and geographic maps. We

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.

CHI 2018, April 21–26, 2018, Montréal, QC, Canada.
Copyright © 2018 ACM ISBN 978-1-4503-5620-6/18/04 ...$15.00.
http://dx.doi.org/10.1145/3173574.3174168

also find that the most popular visualizations varied across visualization tools, indicating that some tools may be more accessible or appealing for certain visualization types. Despite this variation, bar and line charts clearly dominate usage of the visualization tools and repositories that we studied.

Except for D3, we only observed 14 or fewer visualization types for each tool. Further analysis of usage statistics suggests that the total visualization types supported by a given visualization tool could be linked to a design tradeoff between flexibility (more types supported) and ease of use (fewer types supported). However, when extracting the D3 collection, we found that users often copied established D3 examples as a starting point for new visualizations, showing promise as a technique to encourage users to try new visualization types.

Unlike other mediums, such as presentations and reports, we find that pie charts are relatively rare in the extracted collections. As such, the controversy surrounding pie charts appears to be a moot point for the SVG-based, web-driven visualizations that we studied.

To summarize, we make the following contributions:

- a new technique for classifying SVG objects. When performing five-fold cross validation, our classification technique provides: 1) 82%-99% classification accuracy *within* collections, and 2) 85% accuracy *between* collections.
- we analyzed the result of mining over 41,000 pages that contain SVG visualizations, out of roughly 20 million pages that were visited. We found that SVG-based interactive visualizations still represent a small number of web pages on the internet. We present our analysis results in the Section titled “Discussion: Visualization Usage on the Web”.

FINDING AND EXTRACTING VISUALIZATIONS

In this section, we describe how we used our Web Crawler to collect thousands of visualizations from the web.

We initially performed a general, unguided crawl from the web to discover visualizations. However, after crawling 20 million webpages, we only found roughly 10,000 pages with visualizations, or 0.05%. The majority of these webpages were user profile pages from stackoverflow.com (and stackexchange websites), each with a single line chart showing user activity over time, resulting in thousands of redundant visualizations.

As a result, we sought out specific “islands” of usage on the web, where users frequently deposit their visualizations at a single, centralized source website. We performed a broad search for islands that consistently contained SVG-based visualizations, investigating sites such as Tableau Public [24], Many Eyes [26], and bl.ocks.org [2] for D3 examples [3]. After filtering out the islands that use raster formats instead of SVG, five islands were successfully mined using our Web Crawler: bl.ocks.org, Plotly [16], Chartblocks [4], Fusion Charts [13], and the Graphiq knowledge base [9].

For each URL from the visualization islands, the Web Crawler identifies any SVG objects on the corresponding webpage, and extracts the raw SVG specification for each object, as well as a snapshot of the object.

Web Crawler Results

Using the urls collected from our targeted web search, we ran the Web Crawler to visit the corresponding webpages to extract SVG objects. We label the resulting collections by their corresponding visualization tool or repository name (D3, Plotly, Chartblocks, Fusion Charts, and Graphiq). Given that our islands are websites dedicated to a specific tool or repository, each url collected from our islands was likely to produce SVG-based visualizations, allowing us to quickly collect thousands of visualizations per run. The crawls resulted in over 42,000 total SVG-based visualizations. Per island, we found: over 2000 visualizations for D3, 15000 visualizations for Plotly, 22000 visualizations for Chartblocks, 500 visualizations for Fusion Charts, and over 2500 visualizations for Graphiq.

Challenges

Here, we discuss specific challenges in extracting SVG visualizations that influence the data collected by the Web Crawler.

Website Differences: We found significant differences in the interfaces provided by each website to explore visualizations, which made these sites challenging to crawl. For example, the bl.ocks.org website had a keyword search interface, where relevant visualizations were returned based on search results. The Plotly website had a single exploration page that revealed more visualizations as one scrolled down the page. In contrast, the Chartblocks website was more straightforward in structure, providing a complete list of the available visualizations.

Time Sensitive Crawls: As found with other web crawling projects, such as the Common Crawl [1], our web crawls represent a specific point in time for the websites that were visited. For a more comprehensive view of each website, including the evolution of visualization tools over time (e.g., capturing visualizations made with the most recent version of D3), multiple crawls would need to be executed.

Focus on SVG: Visualizations on the web can appear in many different file formats: PNG, JPEG, GIF, SVG, HTML canvas, etc. Currently, our results will only provide insight into the use of SVG-based visualizations on the web. As opposed to static visualizations in JPG, GIF, etc., SVG-based visualizations are often interactive, which more accurately reflect the trajectory of InfoVis designs. However, we recognize that a comprehensive analysis of all visualizations on the web can also be useful. We plan to extend our analysis to bitmap images and other file formats in the future, to broaden this analysis.

AUTOMATICALLY LABELING VISUALIZATIONS

Labeling visualizations by type helps us gain a sense for how different visualization types are designed and shared on the web. Given that the Web Crawler can extract thousands of visualizations, an automated process is needed to efficiently label visualizations. In this section, we explain how the Beagle Annotator uses a new SVG-based classifier to automatically classify extracted visualizations by type.

The Annotator calculates basic statistics over SVG elements to discern visualization types. Examples of some statistics are: the average x position of SVG elements (*e.g.,* `rect`’s,

[1] http://commoncrawl.org/

circles, etc.), average width of SVG elements, and count of unique colors observed. Each basic statistic represents a single classification feature.

The features are then passed to an *off-the-shelf classifier* (Python's Scikit-Learn [15]) to determine visualization types.

We initially tested four classifier types: Multinomial Naive Bayes, Gaussian Naive Bayes, decision tree, and support vector machine (SVM). Default Scikit-Learn parameters were used for all classifiers. We did not observe a significant difference in performance amongst the highest performing classifiers (decision tree, SVM, and Multinomial Naive Bayes). We used decision trees as a starting point, for ease of interpretation of the resulting models. To take overfitting into account, we use a Scikit-Learn random forest classifier, an ensemble learning technique that trains multiple decision trees (see Section "Accuracy Results for Labeling Visualizations" for details).

Our feature extraction code computes statistics over the positions, sizes, and basic styles for SVG elements (114 features total). These features cannot be individually covered here due to space constraints. Instead, we summarize the groups of features we extract, and provide intuition for why they were selected. Features belong to one of three groups: general (6 features), stye (19 features), and per-element (89 features). General features measure the occurrence of element *types* in a visualization. Style features track how fill, border, and font styles are applied for all elements in the visualization, regardless of element type. Per-element features track the positions, sizes, and styles of five element types: `circle` (16), `rect` (20), `line` (15), `path` (35), and `text` (3). For the rest of this section, we explain how we calculate each feature group.

General features

The intuition behind the general features is to summarize the prevalence of certain elements within each visualization. This may be an early indicator for certain visualization types. For example, heavy use of circles may indicate a scatter or bubble chart. These features consist mainly of counts for the five SVG element types. For each SVG object (*i.e.,* each extracted visualization), we count the instances of each element type.

We also count the number of horizontal and vertical axis lines, to differentiate visualizations that often contain axes (*e.g.,* bar charts), from ones that do not (*e.g.,* geographic maps).

Style features

Differences in visualization types can be found in the way that SVG elements are styled. For example, how they are colored, and given border and line thicknesses. Styling is often treated as orthogonal to the layout of elements in the DOM tree, so we analyze styling using separate features.

Counting the unique colors observed can be useful, because many visualizations have color applied in a systematic way (*e.g.,* all bars in a bar chart are one color). We count the number of unique fill and border colors as two separate features. We also consider the maximum and minimum stroke widths, as well as maximum and minimum font size. Given that text can vary widely in font size, such as for word clouds, we calculate the number of unique font sizes, and font size variance. Note that we also take into account the inheritance of styles through parent objects and embedded CSS style properties.

Per-Element features

We analyze each SVG element type to further differentiate visualization designs. For example, rect elements with the same y position might indicate a bar chart, whereas circles with identical radii might indicate a scatterplot. Note that all features are normalized: x positions are divided by visualization width, y positions by visualization height, all line and path lengths by visualization diagonal, and all shape widths by either width or height (whichever is larger).

Features extracted for all elements: For all element types, we calculate a standard set of statistics. First, we calculate the maximum, minimum, variance, and total unique x positions, and then repeat for y position. We then calculate the average number of elements that share positions, and the total unique CSS class names. Tracking x and y positions allows us to identify layout patterns. For example, vertical bar charts have rectangles with identical y position, and periodic x positions. In contrast, scatter charts have circles with many unique x and y positions, helping to discern bar charts from scatter charts.

Circle Features: We calculate the maximum, minimum, and variance in the radii. We also consider the maximum number of circles with identical radii. Radii variance helps to discern visualizations with equal-size circles (*e.g.,* scatter charts), from those with varying circle sizes (*e.g.,* bubble and radial charts).

Rect Features: We calculate: the maximum, minimum, and variance in rect widths; the maximum number of rects with identical width; and the number of unique widths observed. We repeat these calculations for rect heights. Similar to positioning, tracking widths and heights helps to identify size patterns, such as equal-width or equal-height bars in bar charts.

Path Features: We mainly consider the total characters used to specify a path. To do this, we analyze the "d" attribute, which contains commands for drawing the path (*e.g.,* move to point A, draw a line to point B, etc.). We calculate the maximum, minimum, mean and variance in the length of the "d" attribute. This feature is useful for paths, because the longer a "d" attribute is, the more detailed it is. Complex shapes require detailed paths, such as countries in geographic maps. We also calculate the Euclidean distance between a path's start and end points, to help discern short paths (*e.g.,* parallel coordinates) from long paths (*e.g.,* line charts).

However, path elements are complex, and can be used in place of other SVG elements, requiring some additional statistics. To find polygon-heavy visualizations (*e.g.,* voronoi visualizations), we compute the above statistics specifically for paths that contain polygons (*i.e.,* paths that start and end in the same place). We also compute the number of arc calls within a path element, which can help to identify circles in path elements.

Line Features: For lines, we calculate the maximum, minimum, and variance in line length.

ACCURACY RESULTS FOR LABELING VISUALIZATIONS

One of Beagle's important features is its ability to automatically label visualizations. As such, it is necessary to measure

Collection	Size	Total Types	Visualization Type Labels
D3	1247	22	area (32), bar (154), box (11), bubble (70), chord (34), donut (31), heatmap (32), **geographic map (379)**, graph (60), hexabin (21), line (157), radial (13), pie (7), sankey (11), scatter (118), treemap (10),voronoi (25),waffle (12),word cloud (6), sunburst (28), stream graph (13), parallel coordinates (23)
Chartblocks	22730	4	pie (5514), **line (8065)**, bar (7402), scatter (1749)
Fusion Charts	530	10	area (14), **bar (224)**, box (22), donut (54), geographic map (48), heatmap (12), line (84), pie (26), scatter (29), sunburst (6)
Graphiq	2727	11	bubble (9), donut (18), area (210), graph (5), geographic map (244), line (655), waffle (4), box (6), **bar (1542)**, treemap (6), scatter (28)
Plotly	6544	11	area (10), bar (1364), box (259), contour (118), donut (193), filled-line (126), geographic map (184), line (1198), pie (26), radial (17), **scatter (3049)**

Table 1. General information about each visualization collection in our evaluation.

Beagle's accuracy, to ensure that people can rely on the classification labels. In this section, we evaluate Beagle's SVG-based classifier in two ways. First we perform a "within-group" evaluation where we trained and evaluated the classifier using the five visualization collections extracted by the Beagle Web Crawler. Second, we conduct a "between-group" evaluation where we randomly mixed visualizations from the different collections and evaluated the accuracy of the classifier.

Within-Group Evaluation

We use all five of our visualization collections for our validation experiments. Table 1 provides a summary of these details for each visualization collection. Each collection was created by mining the corresponding website using the Beagle Web Crawler (see Section 2 for more details). The collections range in size from 530 visualizations (Fusion Charts), up to 23,270 visualizations (Chartblocks), and from 4 visualization types in one collection (Chartblocks) to 22 types (D3).

Note that for our analyses, we omit roughly one quarter of the visualizations extracted by the Web Crawler. This is because the corresponding webpages contained complex features that interfered with the extraction process, such as animations. The total visualizations used in each experiment is recorded in column 2 of Table 1.

Dataset Labels

We apply classification labels to every visualization used in our analyses (*i.e.,* the visualization type). We considered the visualization types that appeared in all of our data collections, and created a superset of labels to cover them. Only visualization types with too few samples (*e.g.,* only one example) were omitted from the list. The final set of labels is provided in Table 1, along with the number of samples observed for each visualization type and each data collection.

The superset of labels was created in three steps: 1) the usage documentation (and visualization gallery, if available) was reviewed for each website to form an initial set of labels; then 2) each set of labels was compared against the visualizations in the corresponding collection to identify missing labels from the initial set; and 3) the resulting label sets were consolidated into the final superset (e.g., bar chart labels from the collections are consolidated into a single "bar" label in the superset, line charts into a single "line" label, etc.).

Visualizations were then labeled by the authors either by hand (for bl.ocks.org/D3, Fusion Charts, and Graphiq) or through code (for Plotly and Chartblocks) using the final superset. To extract visualization labels through code, the structure of each website was inspected for indicators of visualization type. The Plotly website has a separate webpage for each visualization, with visualization types contained in the title of each page. Similarly, the main page of the Chartblocks website contained links to all of its visualizations, organized by visualization type. However, the other three websites lacked useful metadata for automatic labeling, and had to be labeled by hand. In general, we observed a lack of consistent metadata across websites, making this form of automatic labeling of limited use.

We found strong similarities between certain visualization types within the following two groups, and consolidated each group: geographic maps (e.g., choropleth and map projections), and graphs and trees (e.g., dendrograms and trees). We also grouped visualization types with style variations (e.g., stacked and grouped bar charts were consolidated).

A small fraction of visualizations in the D3, Fusion Charts, and Graphiq collections had a mixed design, where more than one visualization type could apply. In these cases, we assigned each visualization a primary visualization type, and added secondary labels as necessary. Note that all classifiers are trained and tested on the *primary labels only*.

Experimental Setup and Results

We used the Scikit-Learn `RandomForestClassifier` for our experiments. We set the number of decision trees to 14 (`n_estimators=14`). The default values were used for all other input parameters.

We calculate the overall accuracy as the fraction of correct answers across ten runs of stratified, five-fold cross validation (i.e., weighted accuracy column in Table 2). We also calculate the average accuracy across all classes, giving each class equal weight (i.e., non-weighted accuracy column in Table 2). With five-fold cross-validation, each dataset is shuffled and partitioned into five groups (or folds). For each fold, the classifier is trained on the other four folds (or 80% of the data), and tested on the fold that is left out (20% of the data). With some classes being quite small, we chose fewer folds in our cross validation in an effort to better balance the ratio of testing and training samples. Each 5-fold evaluation was run 10 times, and the results were averaged across all 10 runs.

	Accuracy		F1 Score	
	Non-Weighted	Weighted	Non-Weighted	Weighted
D3	0.7155	0.8193	0.7473	0.8149
Plotly	0.9159	0.9721	0.9290	0.9717
Chartblocks	0.9955	0.9955	0.9955	0.9955
Fusion Charts	0.8753	0.9258	0.9020	0.9253
Graphiq	0.9726	0.9873	0.9799	0.9873
Mixture	0.7817	0.8527	0.7899	0.8503
Mixture (Revision)	0.6025	0.7952	0.6261	0.7892
Mixture + Non-Vis (binary)	0.9411	0.9270	0.9467	

Table 2. Weighted and non-weighted (i.e., equal class weight) multi-class classification accuracy (between 0 and 1) and F1 score (between 0 and 1) for individual collections, and for combining the collections (Mixture, Mixture + Non-Vis). Each set was evaluated using 5-fold cross-validation (80% training set size, 20% test set size per fold). Note that the "Mixture + Non-Vis" experiment shows binary classification results.

The goal of these experiments is to test Beagle's ability to accurately label visualizations across a variety of rendering environments. Here, we have five separate visualization collections, represented by the visualization tools used to create the visualizations (D3, Plotly, Chartblocks, Fusion Charts, and Graphiq). The results of our experiments are provided in Table 2. We see that in all five cases, Beagle provides 81.9% classification accuracy or higher. In four of five cases, Beagle provides 92%-99% accuracy. Through these results, we confirm that Beagle is able to capture the defining characteristics of different visualization types across collections, simply by calculating basic statistics over the SVG objects.

Between-Group Evaluation

In the previous section, we found that the Beagle Annotator achieves high classification accuracy for visualizations rendered using a particular tool (i.e., classification *within groups*). However, on the web, Beagle has to contend with a mixture of SVG objects, where some may have been created by different tools, and others may not even be visualizations (e.g., logos). Thus, it is important to also test Beagle's performance when faced with a mix of different SVG outputs. To test this, we formed a new collection by randomly selecting 500 visualizations from each visualization collection. All visualization types for each collection are represented in the samples. We performed 10 runs of five-fold cross validation on the mixture of 2500 visualizations, and found that Beagle provides 85.2% classification accuracy (labeled as "Mixture" in Table 2). Non-weighted (0.7899) and weighted (0.8503) F1 scores are also high, reinforcing Beagle's high performance.

To test how Beagle discerns between visualizations and other non-visualization SVG objects, we added 1,000 non-visualizations to the mixture (3,500 total SVG objects). We trained a binary classifier using Beagle's features, and found that Beagle provides 92.7% classification accuracy (F1 score 0.9467). (labeled as "Mixture + Non-Vis" in Table 2).

We found that Beagle is comparable to the reported performance of related classification techniques, such as those used in Revision [21] (80-90% accuracy), FigureSeer [23] (86% accuracy), and ChartSense [10] (76.7-91.3% accuracy). However, an advantage of Beagle is its ability to classify different visualization types with a limited number of training examples. Computer vision and deep learning techniques generally require large training sets to develop their classification models, which may not be possible when building a new visualization collection from scratch.

Nevertheless, to provide a direct comparison, we also ran the Revision[2] classifier with our mixed visualization collection (79.5% accuracy), and found that Beagle provides a 5.7% improvement in classification accuracy, with similar improvements to weighted F1 score. We also found that Beagle provides significantly better results across all classes, with an 18% improvement to non-weighted classification accuracy, and similar improvements to non-weighted F1 score. Note that for each visualization extracted by the Beagle Web Crawler, we collected both the SVG output and a snapshot of the visualization. Thus we had an accurate image representation of the visualization from its original environment, to ensure that Revision had a competitive point of comparison with Beagle.

DISCUSSION: VISUALIZATION USAGE ON THE WEB

In this section, we highlight interesting insights from our extracted collections. Note that these results are specific to the SVG visualizations analyzed, and may not translate across other output formats, such as Excel files or raster image files.

Supporting Fewer Visualization Types is Common

Here, we discuss our observations in the diversity versus usage of visualization types. For some collections, we observed more visualization types than are represented in our evaluation (i.e., Table 1). For example, we observed exactly one waffle chart from our crawl of the Fusion Charts website. The total observed visualization types are provided in Table 1. Except for D3, the other visualization collections have limited variety in visualization types. Furthermore, we find that aside from a small set of visualization types, most have few examples from users. When analyzing these more obscure visualization types across the collections, we find that they generally: 1) illustrate complex relationships between data groups (*e.g.,* graphs, sunbursts, treemaps, and arc diagrams); or 2) have more popular (and often simpler) equivalents, such as waffle charts and word clouds, which could be replaced by bar charts. These findings suggest that when more complex visualization types are supported, there may still be challenges to overcome in getting general lay-users to try them out.

[2] Note that Revision requires bitmap images as input.

Collection	All Types Observed	Most Popular	2nd Most Popular	% Bar	% Line	% Pie
Chartblocks	4	line, 34.7%	Bar, 31.8%	31.8%	34.7%	23.2%
D3	25	Map, 30.4%	Line, 12.6%	12.3%	12.6%	0.6%
Fusion Charts	14	Bar, 42.3%	Line, 15.8%	42.3%	15.8%	4.9%
Graphiq	12	Bar, 56.5%	Line, 24.0%	56.5%	24.0%	0%
Plotly	11	Scatter, 46.6%	Bar, 20.8%	20.8%	18.3%	0.4%

Table 3. Relevant totals for visualization types across the visualization collections (raw data for columns 3-7 available in Table 1).

Line and Bar Charts Dominate the Collections

We see in Table 3 that the most popular visualization type varies across visualization collections, with a steep drop in usage between the most popular and second most popular visualization types. We found geographic maps, line charts, bar charts, and scatter charts to be the most popular types across the visualization collections. Together, these four visualization types represent a large fraction of the visualization collections: 64.8% for D3, 72.6% for Fusion Charts, 88.6% for Plotly, and 90.5% for Graphiq. Furthermore, line and bar charts are always within the top 3 most popular visualization types across all collections (usage reported in columns 5 and 6 of Table 3).

As such, it seems that the vast majority of the time, simple visualization types are a suitable solution for helping online users to make sense of their data. Our findings thus far may suggest a design tradeoff between flexibility (*e.g.,* D3) and ease of use (*e.g.,* Chartblocks), and thus a need for a spectrum of tools targeting different visualization needs and expertise.

However, having easy-access examples that are well documented, such as the D3.js visualization gallery and tutorials pages, could encourage users to experiment with new visualization types. Throughout the course of our crawl of the bl.ocks.org website (and subsequent analysis), we did find that many visualizations created by users were very similar to the available D3 examples.

Pie Charts Have Limited Usage in the Collections

Interestingly, though pie charts are generally considered to be a well-known and relatively simple visualization type, they represent only a small fraction of the visualizations that we observed in the collections. For four of the five visualization tools and repositories that we studied, pie charts represent less than 5% of the visualizations we observed. The only exception was for Chartblocks, where only four visualization types are observed (line, bar, scatter, and pie). In this case, pie charts are 23.2% of the collection. However, even in the case of Chartblocks, we find that line charts are 34.7% of the collection (12% higher than pie charts), and bar charts are 31.8% of the collection (8.6% higher than pie charts). It is unclear whether this stems from existing visualization conventions (e.g., [14, 5, 25]), or other factors, such as the design or presentation of the visualization tools themselves.

Dataset Limitations and Future Work

We see Beagle as a useful starting point for exploring visualization usage on the web. However, there are limitations to what analyses are possible with our dataset. Here, we outline four aspects of the dataset we aim to extend in the future.

More websites: Five different websites have been crawled using Beagle, but there are certainly more than five websites dedicated to creating and sharing visualizations online. We plan to continue our data collection process, to provide a broader context for analyzing online usage of visualization tools. We also encourage others to extend the dataset with more examples as a community effort in the future.

More metadata: As shown in our analysis, it can be helpful to know more information beyond only the visualization type, such as the specific tool that was used to create the visualization (e.g., D3 versus Chartblocks), and any available examples of tool usage (e.g., the D3 image gallery). We plan to extend Beagle to capture more supplemental data, such as documentation and other information about the visualization tools themselves (e.g., the code repository for the tool), and when possible, the raw data used as input for the visualizations.

More complex design analysis: currently, our analysis emphasizes only the types of visualizations created on the web. In the future, we plan to extend the project to analyze more complex interaction designs, such as cross filtering across coordinated views.

RELATED WORK

Our goal is to better understand how different tools are utilized to create and share visualizations on the web. In pursuit of this goal, we developed visualization extraction and classification techniques to harvest a collection of visualizations for analysis. In this section, we highlight related projects in these areas.

Beagle was inspired by existing projects to mine and analyze website designs, such as Webzeitgeist [11] and D.Tour [19]. Some projects within the visualization community also incorporate web mining and visualization extraction, but for different goals. For example, Harper and Agrawala extract data from D3 visualizations to re-style visualizations [6], as well as to generate visualization templates [7]. Saleh *et al.* extract infographics from Flickr to develop and evaluate techniques for similarity-based search of infographics [20].

Beagle also extends existing work in visualization classification [22, 21, 18, 8, 23, 12, 10]. Most projects focus on classifying raster images of charts [21, 18, 8, 23, 12, 10]. Prasad *et al.* [18] and Savva *et al.* [21] apply computer vision techniques to extract features from raster images for classification. Huang and Tan [8] extract graphical marks from raster images, and use them to classify visualizations. Jung *et al.* [10] and Siegel *et al.* [23] apply deep learning techniques to classify raster visualizations. Poco and Heer revisit the approach of raster image classification from a new perspective by extracting visualization *specifications* from images [17], instead of visualization types. Shao and Futrelle [22] analyze vector images from PDF's to classify images across five visualization types. Beagle calculates basic statistics over SVG as classification features. Beagle's classification features apply to any SVG image, and can accurately classify thousands of visualizations, and as many as 24 different visualization types.

Surprisingly, though many techniques have been proposed to classify visualization images, few projects move beyond the extraction and classification phase to perform broader studies of visualization usage, particularly on the web. Here, we highlight two relevant projects. Benson and Karger study usage of the Exhibit tool for publishing data online [1], and also find that authors' design decisions are influenced by available design examples. In a related area, Lee *et al.* developed a platform for extracting and analyzing visualizations from scientific papers [12], and find that papers with higher densities of images tend to have higher research impact (visualizations comprise 35% of these images). In contrast, we find that only a small fraction of webpages contain SVG-based visualizations.

CONCLUSION

In this paper, we presented Beagle, an automated system for collecting, labeling and analyzing visualizations created on the web. Beagle supports a flexible design with two standalone components. The Web Crawler extracts SVG-based visualizations from webpages, and was able to extract over 41,000 visualizations from the web. The Annotator uses SVG-focused classification techniques to label visualizations, and achieves 86% classification accuracy, in a multi-class classification test with 24 different visualization types. We then use the resulting visualization collection to study usage of the different visualization types across tools. We found that only a small fraction of webpages (0.05%) contain SVG visualizations created using browser-based tools. Furthermore, the vast majority of visualizations in the collection are covered by just four visualization types: bar charts, line charts, scatter charts, and geographic maps. And though they are hotly debated in the visualization community, pie charts are somewhat rare for the particular visualization tools that we studied. Our findings indicate that in addition to tools that provide flexibility (*e.g.,* D3), users may also benefit from visualization tools that facilitate fast ease of use by focusing on supporting a small set of visualization types.

ACKNOWLEDGMENTS

This work was supported by the Intel Science and Technology Center for Big Data, the National Science Foundation under Grant No. IIS-1452977, and DARPA under Grant No. FA8750-17-2-0107.

REFERENCES

1. Edward Benson and David R. Karger. 2014. End-users Publishing Structured Information on the Web: An Observational Study of What, Why, and How. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1265–1274. DOI: http://dx.doi.org/10.1145/2556288.2557036

2. Michael Bostock. 2016. Popular Blocks - bl.ocks.org. (Sept. 2016). http://bl.ocks.org/

3. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309. DOI: http://dx.doi.org/10.1109/TVCG.2011.185

4. Chartblocks. 2017. Online Chart Builder - ChartBlocks. (2017). https://www.chartblocks.com/en/

5. Stephen Few and Perceptual Edge. 2007. Save the pies for dessert. *Visual Business Intelligence Newsletter* (2007), 1–14.

6. Jonathan Harper and Maneesh Agrawala. 2014. Deconstructing and Restyling D3 Visualizations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 253–262. DOI: http://dx.doi.org/10.1145/2642918.2647411

7. J. Harper and M. Agrawala. 2017. Converting Basic D3 Charts into Reusable Style Templates. *IEEE Transactions on Visualization and Computer Graphics* PP, 99 (2017), 1–1. DOI:http://dx.doi.org/10.1109/TVCG.2017.2659744

8. Weihua Huang and Chew Lim Tan. 2007. A System for Understanding Imaged Infographics and Its Applications. In *Proceedings of the 2007 ACM Symposium on Document Engineering (DocEng '07)*. ACM, New York, NY, USA, 9–18. DOI: http://dx.doi.org/10.1145/1284420.1284427

9. Graphiq Inc. 2017. Graphiq | Knowledge Delivered. (2017). https://www.graphiq.com/

10. Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive Data Extraction from Chart Images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6706–6717. DOI: http://dx.doi.org/10.1145/3025453.3025957

11. Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3083–3092. DOI: http://dx.doi.org/10.1145/2470654.2466420

12. Poshen Lee, Jevin West, and Bill Howe. 2016. Viziometrics: Analyzing Visual Patterns in the Scientific Literature. *Journal of the Association for Information Science and Technology (JASIST) (in prep)* (2016).

13. InfoSoft Global Pvt. Ltd. 2017. JavaScript Charts for Web, Mobile & Apps. (2017). http://www.fusioncharts.com/

14. Cole Nussbaumer. 2011. death to pie charts. (July 2011). http://www.storytellingwithdata.com/blog/2011/07/death-to-pie-charts

15. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

16. Plotly. 2016. Plotly | Make charts and dashboards online. (2016). https://plot.ly/

17. Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum (Proc. EuroVis)* (2017). http://idl.cs.washington.edu/papers/reverse-engineering-vis

18. V. S. N. Prasad, B. Siddiquie, J. Golbeck, and L. S. Davis. 2007. Classifying Computer Generated Charts. In *2007 International Workshop on Content-Based Multimedia Indexing*. 85–92. DOI: http://dx.doi.org/10.1109/CBMI.2007.385396

19. Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 165–174. DOI: http://dx.doi.org/10.1145/2047196.2047216

20. Babak Saleh, Mira Dontcheva, Aaron Hertzmann, and Zhicheng Liu. 2015. Learning Style Similarity for Searching Infographics. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 59–64. http://dl.acm.org/citation.cfm?id=2788890.2788902

21. Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. ReVision: Automated Classification, Analysis and Redesign of Chart Images. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 393–402. DOI: http://dx.doi.org/10.1145/2047196.2047247

22. Mingyan Shao and Robert P. Futrelle. 2006. Recognition and Classification of Figures in PDF Documents. In *Proceedings of the 6th International Conference on Graphics Recognition: Ten Years Review and Future Perspectives (GREC'05)*. Springer-Verlag, Berlin, Heidelberg, 231–242. DOI: http://dx.doi.org/10.1007/11767978_21

23. Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. 2016. FigureSeer: Parsing Result-Figures in Research Papers. In *Computer Vision – ECCV 2016 (Lecture Notes in Computer Science)*. Springer, Cham, 664–680. DOI: http://dx.doi.org/10.1007/978-3-319-46478-7_41

24. Tableau Software. 2016. Tableau Public. (2016). https://public.tableau.com/

25. Edward R. Tufte. 1986. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA.

26. Fernanda B. Viegas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. 2007. ManyEyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov. 2007), 1121–1128. DOI: http://dx.doi.org/10.1109/TVCG.2007.70577