

Techniques for Flexible Responsive Visualization Design

Jane Hoffswell
University of Washington
Seattle, Washington
jhoffs@cs.washington.edu

Wilmot Li
Adobe Research
Seattle, Washington
wilmotli@adobe.com

Zhicheng Liu
Adobe Research
Seattle, Washington
leoli@adobe.com

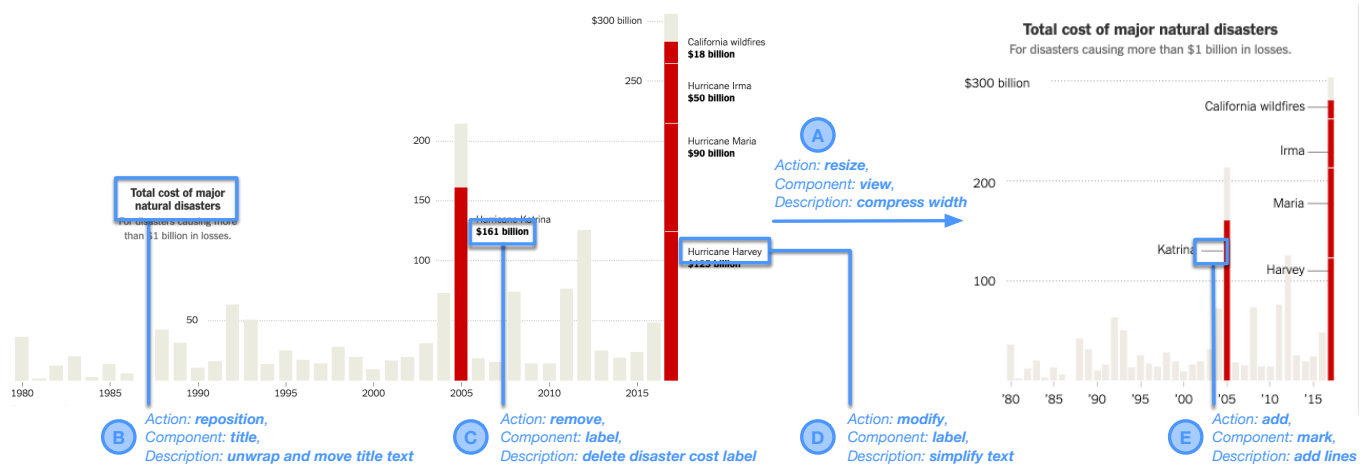


Figure 1. Desktop (left) and mobile (right) visualizations from the New York Times article “The Places in the U.S. Where Disaster Strikes Again and Again” [A13]. This example demonstrates responsive techniques that: (A) *resize* the view to compress the width; (B) *reposition* content (e.g., axes, labels, and title); (C) *remove* unnecessary labels; (D) *modify* the text and axis labels to reduce complexity; and (E) *add* new line marks to annotate the bars.

ABSTRACT

Responsive visualizations adapt to effectively present information based on the device context. Such adaptations are essential for news content that is increasingly consumed on mobile devices. However, existing tools provide little support for responsive visualization design. We analyze a corpus of 231 responsive news visualizations and discuss formative interviews with five journalists about responsive visualization design. These interviews motivate four central design guidelines: enable simultaneous cross-device edits, facilitate device-specific customization, show cross-device previews, and support propagation of edits. Based on these guidelines, we present a prototype system that allows users to preview and edit multiple visualization versions simultaneously. We demonstrate the utility of the system features by recreating four real-world responsive visualizations from our corpus.

CCS Concepts

•Human-centered computing → Human computer interaction (HCI); Visualization systems and tools;

Author Keywords

Visualization; Responsive Design; News; Mobile Devices.

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 '20, April 25–30, 2020, Honolulu, HI, USA.

© 2020 Association of Computing Machinery.

ACM ISBN 978-1-4503-6708-0/20/04 ...\$15.00.

<http://dx.doi.org/10.1145/3313831.3376777>

INTRODUCTION

Mobile devices are now a more important platform than computers for consuming news articles [8]. While the text content may easily adapt to the device size, it is non-trivial to create responsive visualizations. Responsive visualizations must adapt the design so that content remains informative and legible across different device contexts. For example, designers may choose to resize certain visualization marks or swap the axis encodings so that a chart fits better on a mobile screen.

To understand current practices for responsive visualization design, we examine 53 news articles gathered from 12 sources; in these articles we identify 231 visualizations, and label the visualization type and responsive techniques used by the article. We classify the techniques into six high-level actions: *no change*, *resize*, *reposition*, *add*, *modify*, and *remove*. The most common action in our corpus is to remove content from the mobile visualization; however, visualizations often exhibit multiple techniques, including more complex customizations such as completely redesigning the visualization encoding or adding clarifying marks. While a few designs take into account device-specific interaction capabilities, the vast majority of adaptations focus on creating legible charts at different sizes corresponding to different device categories (e.g., desktop, tablet, or portrait mobile). Thus, in this paper, we use the terms “device size/context,” and “chart size” interchangeably.

We conduct formative interviews with five authors of the coded visualizations to better understand their development processes and the responsive visualization techniques used in their work. We find that responsive designs regularly requires authors to

maintain different artboards for different device sizes. Resizing the design often requires device-specific customizations (e.g. to *reposition* the visual content [A13, A36], *add* clarifying information [A5], *modify* annotations to change or shorten the text [A10, A13], *remove* visualization details [A13, A50, A52], or *remove* interactivity altogether [A1, A14]). In rare cases, authors completely redesign the visualization and/or interaction for different device contexts [A19, A23, A36].

Despite the necessity of responsive visualizations, the process of developing and maintaining multiple designs requires extensive user time and effort. Responsive visualizations therefore become a burden on the development workflow. While responsive considerations may be discussed in the abstract throughout the visualization design process, implementation of the responsive design often occurs only in the final stages.

From the formative interviews, we identify four central design guidelines to inform the development of new systems for responsive visualization design: (1) **enable simultaneous cross-device edits** to facilitate design exploration for multiple target devices; (2) **facilitate device-specific customization** to address the need for adaptive designs; (3) **show cross-device previews** to provide an overview of customizations applied across devices; and (4) **support propagation of edits** to reduce user effort and accelerate design iterations.

Based on these guidelines, we contribute a set of core system features that allow designers to view, create, and modify multiple device-dependent visualizations. Our system displays separate views for each chart size and supports simultaneous editing of multiple views. The system enables generalized selections and view control to support robust customization of marks. The system also foregrounds all the variations between visualizations to help designers assess the full picture of the applied modifications and propagate changes across views.

To demonstrate the utility of our system, we recreate four real-world examples from our corpus [A13, A36, A50, A52]. These examples represent a range of visualization types (bar chart, dot plot, line chart, and symbol map) and demonstrate our core system features. For each example, we provide a step-by-step walkthrough of the development process for the visualization design. These walkthroughs demonstrate how a designer can construct, compare, customize, and iterate on different visualizations using a flexible development workflow.

RELATED WORK

This research is informed by related work on responsive visualization, visualization authoring, and responsive web design.

Responsive Web Design

While responsive visualization is still a nascent area, responsive web design has received more attention. Patterns and principles of responsive web design have been studied [15, 16]. HTML5 and CSS3 are popular standards to implement responsive designs [9]. Techniques for responsive web design, however, are not directly transferable to visualization: webpages primarily employ text wrapping, image resizing, and document reflow to achieve responsiveness; these approaches offer little insight on visualization challenges such as data encoding, scale adjustment, or annotation placement.

Responsive Visualization

Responsive visualization becomes particularly necessary for a journalism context in which readers often consume content on mobile devices. Conlen et al. [6] describe techniques to examine reading behaviors for interactive articles, with an implementation targeted at Idyll [5]; the articles analyzed by Conlen et al. [6] were primarily designed for desktop use, but 30%-50% of readers consumed and interacted with the content on a mobile device despite the limitations.

Despite the need for responsive articles, there is limited support for designing responsive visualizations. Journalists often combine a variety of approaches including data analysis in R and python, dynamic visualization development using D3.js [2], and customization in Adobe Illustrator. Static visualization approaches require designers to implement and maintain multiple artboards, which can be time consuming and labor intensive. The New York Times developed ai2html [4], which converts Adobe Illustrator documents into a web format by separating the text and graphic components; this approach ensures that the visualization text remains legible by supporting dynamic placement and scaling, but does not explicitly promote considerations for mobile visualization [19].

Visualization Authoring Approaches

There is a wealth of related work around visualization authoring approaches. Satyanarayan et al. [22] provide an overview of this space and reflect on the design of different visualization authoring systems. Satyanarayan et al. primarily compare three systems, which they classify as *visual builders*: Lyra [21], DataIllustrator [14], and Charticulator [18]. Such systems allow for fine-grained control of the visualization design, often via direct manipulation. Other visualization authoring systems utilize a *shelf construction* model; for example Tableau, formerly Polaris [24], and Voyager [25]. Our proposed system primarily employs the simpler shelf construction approach in which designers assign data fields to encoding channels via drag-and-drop, with some minor modifications possible via direct manipulation; this simplification allows us to focus on representing and communicating the responsive aspects of the visualization design as the primary contribution of this work.

Datawrapper [11] is a tool for journalists to design interactive and responsive visualizations based on a set of templates and device sizes; Datawrapper makes it easy to preview the design across devices, but limits the customization options available for the visualization designs and narrative content. Power BI has also introduced an automatic approach to responsive visualization design for mobile dashboards [10]. ViSizer [26] is a framework for applying local optimizations to more effectively resize a visualization. Vistribute [13] is a system for assigning interactive visualizations amongst multiple devices based on properties of the visualization and device. Recent work discusses the application of responsive web design techniques for responsive visualization [1] and strategies for designing visualizations for both desktop and mobile devices [3].

Charticulator [18] enables automatic chart layout using constraints and can constrain the layout to fit within a particular artboard size. Vega-Lite [23] is a lightweight language for specifying visualizations, which automatically sizes the can-

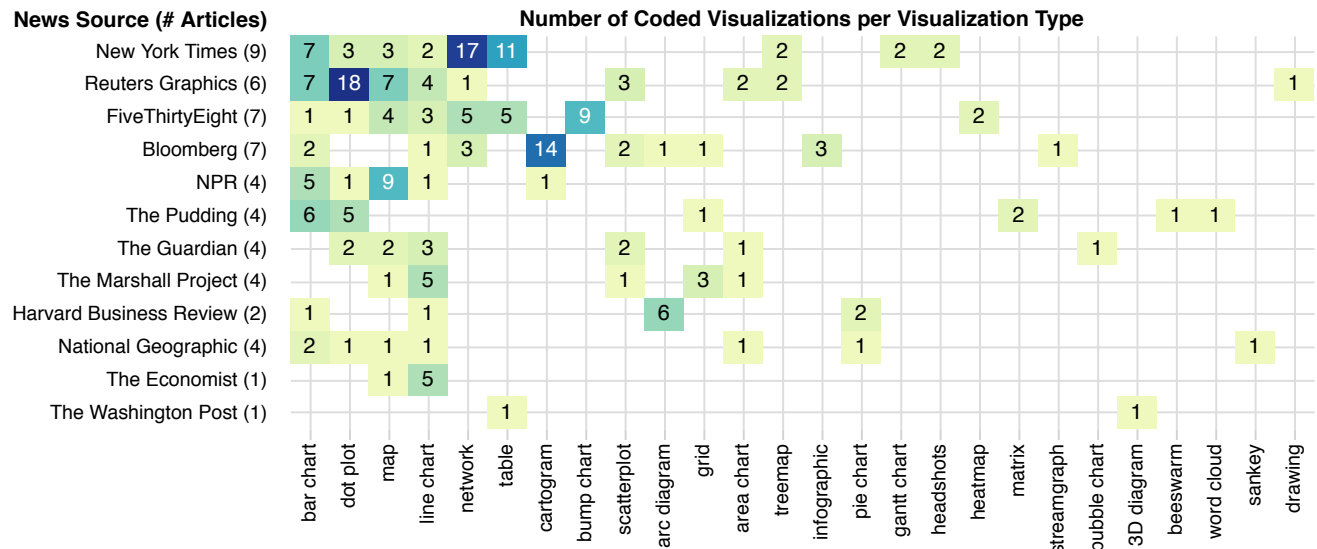


Figure 2. We examined 231 visualizations from twelve sources to inform our analysis of responsive visualization techniques. The number of articles per source is shown in parentheses. For the list of articles analyzed in this paper, see Appendix A. We labeled each visualization with the core visualization type. However, some visualizations were more complex (e.g., a normalized, stacked bar chart); 46 of the 231 visualizations were small multiples.

vas for the data, or can resize the visualization elements for a particular view size. D3.js [2] is often used for constructing dynamic visualizations that can be resized based on the available space. Ellipsis [20] is a tool for authoring narrative visualizations without programming by describing the narrative structure through distinct scenes. ChartAccent [17] enables free-form and data-driven annotation of visualizations based on a design space of chart annotations. Idyll [5] is a language for authoring interactive articles for the web, including the design and parameterization of visualizations.

RESPONSIVE VISUALIZATION CORPUS

To inform our exploration of responsive visualization design, we collected a corpus of 53 news articles gathered from twelve sources, to produce a corpus of 231 visualization examples. We then labeled each visualization instance with the visualization type and responsive techniques used between desktop and mobile versions of the visualization (Figure 2). To view the mobile version of the visualization, we used the Device Mode¹ provided by Chrome DevTools to simulated an iPhone X device. We then examined the responsive techniques used for both the portrait and landscape orientation of a phone.

We surveyed best-of lists and selected articles that included at least one visualization exhibiting responsive techniques, and ensured that the corpus covered a wide range of visualization types (Figure 2). We then performed an open-coding of the responsive techniques for the visualization design and interactive techniques used. Two of the authors coded and discussed a set of overlapping visualizations to ensure inter-coder agreement. When labeling the responsive techniques, we identified changes from the desktop to the mobile version of the visualization. Figure 1 shows several of the open-coding labels generated for the visualization (e.g., the description); we provide the full list of open-coding labels generated for each example in the supplemental material.

¹<https://developers.google.com/web/tools/chrome-devtools/device-mode/>

We then grouped the codes based on their behavioral similarity to determine the core *editing action*, and we associated the action with a particular *visualization component*. The responsive techniques generally fall along a spectrum of simple editing actions: *no changes*, *resize*, *reposition*, *add*, *modify*, and *remove*. These techniques may independently impact different visualization components (e.g., *axes*, *legends*, *marks*, *labels*, and *title*), allowing for complex and varied modifications based on the underlying device context. The modifications may also apply to either a single component, several components, or all components in the view. While most changes reflected small shifts in either layout or content, a subset of visualizations drastically changed the encoding representation (e.g., [A19, A23, A36]). The coded results are shown in Figure 3.

From our analysis, we found that a larger range of responsive techniques were used for the portrait orientation than the landscape orientation of a phone (Figure 3). For the landscape orientation, 69 of the visualizations exhibited *no changes* (29.9%) as opposed to only 6 in the portrait orientation. We also found that it was much more common to *remove* elements from the view (87 or 37.7%, portrait orientation) than to *add* new elements (26 or 11.3%, portrait orientation).

We also examined the end-user interactions included in the visualizations. Most visualizations were static or did not change the core interaction type, aside from using tap rather than click. Similar to the visual techniques, many visualizations removed the interactivity completely from the mobile version rather than redesigning the interactive capabilities (e.g., [A1, A14]). However, a small subset introduced or updated the interaction to improve the experience on mobile (e.g., [A2, A23]).

FORMATIVE INTERVIEWS WITH JOURNALISTS

We conducted semi-structured interviews with five journalists selected from the responsive visualization corpus about their responsive design practices. All participants had previously published at least one article that exhibited responsive visualization techniques and were personally responsible for the

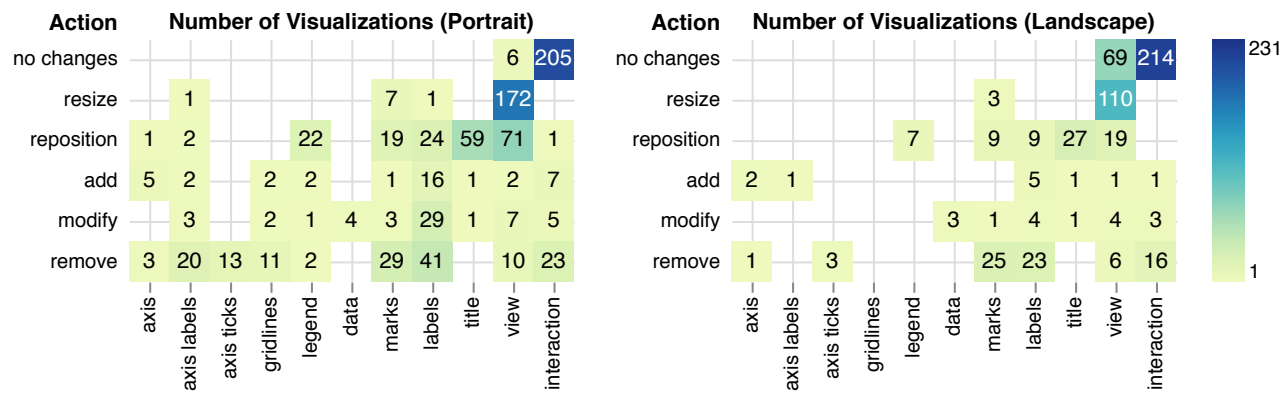


Figure 3. We performed an open coding of the responsive techniques used for both the portrait (left) and landscape (right) orientation of a phone. The labeled techniques reflect the changes made from a desktop visualization to the mobile visualization. We then clustered the techniques to indicate the type of action and the component to which it applies. Responsive techniques were used much more frequently to customize the portrait visualizations than the landscape visualizations. It was also more common to remove or reposition content, than to add new content for the mobile version.

visualization design. Participants were asked to describe their general process when developing a visualization for a news article and the responsive techniques used in one of their published articles; over the course of all interviews, we discussed ten different articles from five different news organizations. Finally, participants were asked to describe the major challenges they face when designing responsive visualizations. The interviews lasted about one hour. An anonymized version of the interview questions is included in the supplemental material.

Desktop-First or Mobile-First Development

Our participants noted that while desktop development was often their primary focus, they kept the mobile version of the visualization in mind throughout the development process. One participant explained that “when we’re sketching something and deciding whether something is gonna work, the question of... how is it gonna work on a phone comes up before we’ve gone too far” (P3). Another participant noted that “I guess it is always in the back of our minds, like ‘how will this work on mobile’ and often we will use that as a rationale to simplify ideas early on in the process because we know that they won’t really work on mobile” (P1). While designers may think about the mobile version, they are not necessarily exploring the mobile designs in a practical sense.

Part of the rationale for desktop-first development was that “by virtue of sort of sketching graphics on my laptop or on my desktop screen, often the first iteration of something works best at those screen widths” (P3). Another participant explained that “It’s easier to try things and to come up with an idea... on the desktop, cause that’s where we work” (P2). For the visualizations, one participant noted that “I think it’s easier to sort of be ambitious when you have a larger palette” (P1). Designers were generally motivated by the flexibility and ease provided by a desktop development environment, such that mobile designs were not at the forefront of their minds.

Some participants did explain that mobile-first development could be advantageous by encouraging more careful design and simplification of the content. In particular, mobile-first development can help designers “focus on what’s essential” (P2) and “it makes us more concise and it makes us get to the point quicker” (P4). When reflecting on the trade-offs of mobile-first or desktop-first development, one participant noted that

their focus was “Aspirationally, certainly mobile phones. I think in practice, that doesn’t really happen” (P3). Another participant observed that “much of the programs we use are geared towards desktop first or feel that way, anyway, so if all of them had a slight shift in default or in tone I feel like that would also help us to think that way” (P4).

Adapting Desktop Visualizations for Mobile

When producing a responsive visualization, participants often noted that they would first finalize the desktop design before creating the mobile visualization. One participant explained that “the mobile version comes after, when I’m happy with the desktop version, to avoid too many changes” (P2). To adapt the visualization to a mobile context, our participants often mentioned the need to prioritize information and remove unimportant content. For one example, the participant explained that “I do remember now removing all of the annotations from that map and I think that was because those annotations weren’t fundamental” (P1). Another participant explained that “There’s a hierarchy of information, right? So as you go down in the artboard size you make the decision about what information can be cut first” (P3). When reflecting on the adaptation process, another participant explained that “I think it’s easier to eliminate things when you have everything” (P2). For many of our participants, the most common workflow was to start with the full desktop visualization and to select what content could be removed when scaling visualizations down to mobile sizes; this trend also matches the overall preference for removing rather than adding content, as described in the section: “Responsive Visualization Corpus.”

Artboards, Dynamic Designs, and Automatic Techniques

To produce responsive visualizations, many of our participants chose to focus on a set of predefined artboard sizes. However, a major challenge with multiple artboards is maintaining and propagating changes to the design. One participant noted that “It’s annoying when you have to make changes to three or four or five different artboards and that usually introduces mistakes... so that’s one of the reasons why the design for mobile comes later” (P2). To produce multiple designs can be a time consuming and labor intensive process for the designers. One participant noted that it is “not the most intellectually stimulating exercise to redesign or make your graphic work...

but it is something that needs to be done for every single graphic” (P3). Another participant explained that “it feels like a chore... You want to be working on the story; you want to not be working on polishing things for small audiences” (P1). While there are clear benefits to having a responsive design, the process of producing these alternative designs can feel like a hindrance to the overall development process.

Several participants discussed the use of D3.js for easily producing responsive visualizations. One participant mentioned using D3 for a design and the need to dynamically resize the window to test the responsiveness: “We more just change the width of the screen pixel by pixel to make sure every pixel is properly looking okay” (P5). However, one participant expressed a hesitance towards dynamic artboard resizing because “dynamically positioning things like labels and annotations at every possible screen width is very easy for that to go wrong and having a fixed number of breakpoints tends to be a little bit less error-prone” (P3). While the ability to make designs dynamic could be helpful for producing visualizations that work for any screen size, testing all possibilities was a common source of difficulty and undesirable user effort.

While there are a variety of common approaches for responsive visualization design (e.g., removing content or simplifying labels), one participant explained that “I think that it’s usually a pretty iterative, ad hoc process. It takes a bit of thinking. It’s usually not the same solution for any two graphics” (P3). Participants often noted that responsive designs were an essential component to their work, but that the development process was currently underserved by existing tools.

Takeaways from the Formative Interviews

Participants in our formative interviews mentioned benefits of both a mobile-first and desktop-first design approach. Mobile-first development encourages designers to focus on only the most important aspects of the data whereas desktop-first development enables more complex, creative, or impressive designs. Since development often happens on a desktop computer, the designs tend to reflect this default. Participants felt that designing for multiple screen sizes (especially mobile) early in the process can lead to better and more consistent cross-device design decisions. More specifically, working through the challenges of visualizing data for various devices helps designers decide what information is most critical, how to effectively highlight key characteristics, and how to effectively encode or layout the data. Empirical evidence also suggests that working on multiple prototypes in parallel leads to better and more diverse designs, and increased self-efficacy [7].

However, cross-device design with existing tools is tedious and error-prone because each visualization is treated as a separate artifact, which requires every edit to be manually duplicated across designs. While having direct control is important for ensuring that designs meet publication standards, too much repetition in the process discourages iterative design modifications. As a result, most workflows start with a fully-executed desktop design that designers modify to better fit mobile screen sizes. While expedient, this approach limits the amount of cross-device design exploration and can sometimes lead to inconsistencies between the designs for various screen sizes.

RESPONSIVE VISUALIZATION SYSTEM

Based on our investigation of existing responsive visualizations and current development practices, we propose a new responsive visualization design system that facilitates flexible, cross-device design workflows. To realize this goal, our system adopts four key design guidelines.

- (1) **Enable simultaneous cross-device edits.** Simultaneous editing accelerates iteration by reducing the time it takes to experiment with different design ideas across multiple target sizes. This capability also reduces the chance of errors and inconsistencies from repeated manual application of edits.
- (2) **Facilitate device-specific customization.** Adaptation of the visualization content to particular device contexts is central to producing effective responsive designs. Our system therefore enables the application of device-specific customizations by focusing editing operations on a particular view or mark.
- (3) **Show cross-device previews.** Providing immediate, visual feedback across multiple designs allows designers to evaluate their choices in the context of all target chart sizes. Such previews help designers determine which choices should be consistent across devices and which should be customized for a particular view. Foregrounding design variation provides a complete picture of the customizations that have been applied.
- (4) **Support propagation of edits.** During the development process, designers may focus on refining the visualization design for one specific chart size. Tools that propagate edits to other chart sizes enable designers to quickly transfer ideas that work well for a particular size to other device contexts.

Responsive Visualization System Overview

To realize these goals, we implemented a responsive visualization design tool that maintains a synchronized representation of a design across multiple target screen sizes. Figure 4 shows an overview of our system. The *main panel* displays a different visualization for each specified chart size. The *toolbar* and other system panels display information about the data and visualization components introduced for each view. Our tool supports generalized selections of visualization components both within and across views to facilitate simultaneous editing operations and customization of specific designs. Motivated by prior work [12], these generalized selections allow designers to refine the selection based on properties of the underlying data or mark encoding values. From the system panels, designers can edit and propagate customizations to multiple visualizations at once, thus reducing the need for repeated work. The *attribute panel* and *layers panel* foreground design variations between views to provide an overview of the customizations that have been applied to different designs.

In contrast to the “desktop-first” strategy most designers currently adopt, our system enables more flexible and iterative workflows. For example, when first developing a visualization, designers can leverage **simultaneous editing** to quickly explore the impact of high-level design decisions (e.g., overall layout or what information should be displayed) across multiple target devices. Designers can immediately **preview the design for all device contexts** while making these global edits. To resolve layout concerns for particular views, designers

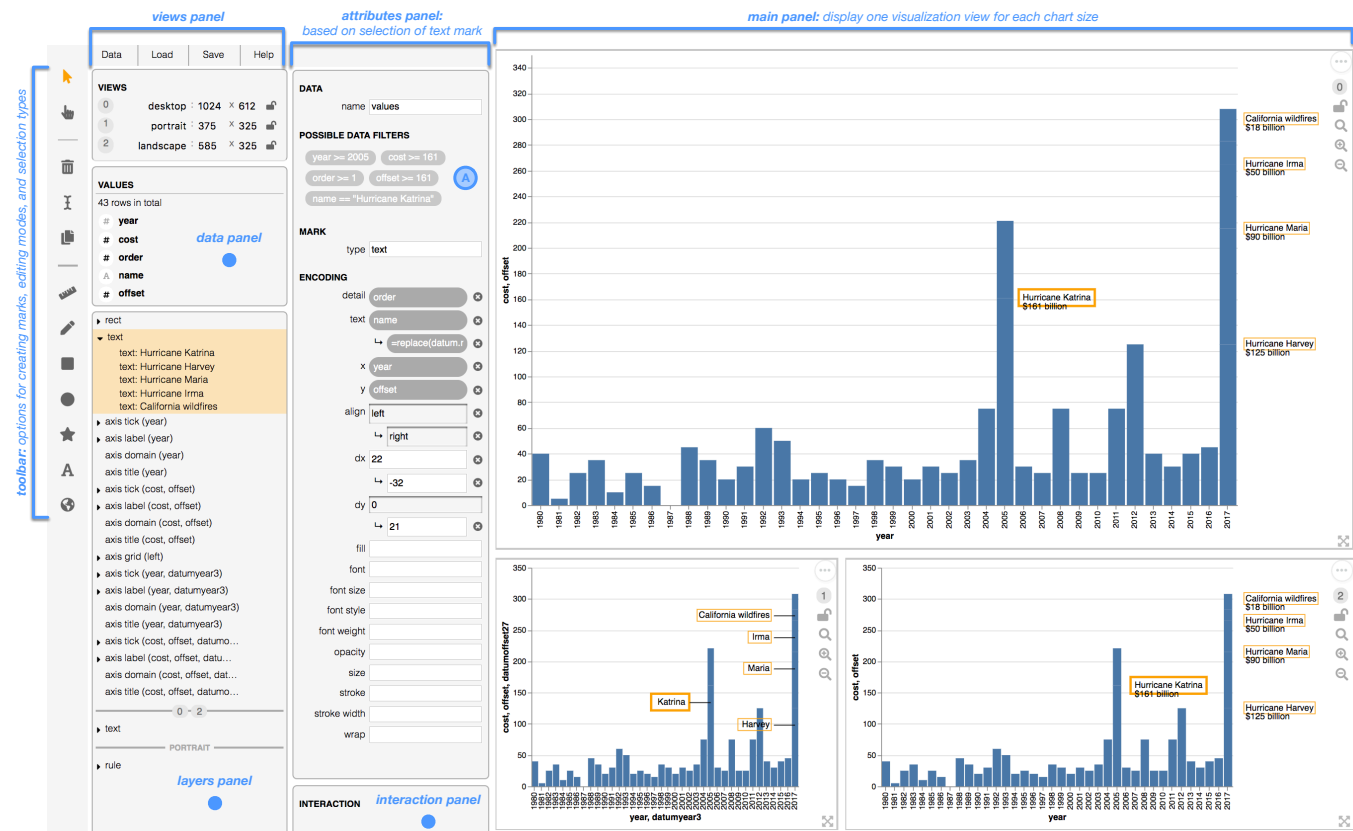


Figure 4. The designer creates a visualization mark by dragging a mark icon from the *toolbar* to a visualization canvas in the *main panel*. The *main panel* displays one visualization view for each device context specified by the designer. The size and name of each view is displayed in the *views panel*. The marks in the visualizations are shown in the *layers panel*. Designers can select a mark from the *layers panel* or directly on the visualization; the encodings for the mark are then displayed in the *attributes panel*. The backing data fields for the visualization are displayed in the *data panel*. To define new encodings, the designer can drag fields from the *data panel* to the *attributes panel*. This screenshot shows the intermediate state of the responsive visualization design process described in the section “An Iterative Workflow for Simplifying a Mobile Design” with the text marks selected.

can **apply device-specific customizations** by selecting visualization components in a subset of views and applying local edits. Designers can also **propagate edits** to different views if a device-specific refinement works well for other device contexts. A key benefit of our system is that it allows designers to iterate fluidly back-and-forth between these global and local editing modes. The result is a more flexible workflow that promotes design exploration, view-specific adaptations, and consistency across devices. The following sections describe our system features in the context of this basic workflow.








System Startup: Viewing Multiple Device Visualizations



When constructing a new visualization, the *main panel* displays a blank canvas for each default chart size. Designers may also import a previously constructed visualization which automatically resizes the design for these default sizes. Designers can customize the default chart sizes to match the standard artboard sizes used by their organization. Automatically displaying multiple, device-specific visualizations allows designers to **preview the design for all device contexts** and thus better incorporate considerations for the responsiveness of the design earlier in the development cycle.

The view names and sizes are displayed in the *views panel*. From this panel designers can rename, resize, select, and create new \oplus views at any point. The system can support an arbitrary

number of different views, up to the discretion of the designer. Designers may also select or resize views directly from the *main panel*. The *data panel* shows the datasets and data fields that have been loaded for the visualization. Each data field is labeled with the automatically detected data type; clicking the symbol for the data type allows designers to change the type.

Simultaneous Editing of the Basic Visualization Structure

To begin constructing a visualization, designers first create a new mark. The system uses Vega-Lite [23] as the underlying language for producing the visualizations and currently supports seven mark types: “rule” , “line” , “bar” , “circle” , “symbol” , “text” , and “geoshape” . Designers can create a mark by dragging the icon from the *toolbar* to one of the visualization canvases or by using keyboard shortcuts to select the mark type, followed by the view.

Our system **enables simultaneous editing** so that designers can construct multiple visualizations concurrently. When a mark is first created it is added to all active views. By default, all user-defined views are labeled as “active”. To apply device-specific customizations, designers can focus on only a single visualization by selecting the view number (1) on the canvas or from the *views panel*. Designers can select multiple views by locking  or unlocking  particular views. Locked or “inactive” visualizations are partially grayed out in the system.

Designers must first link the mark to one of the backing datasets in order to use data fields for the encodings. Similar to other *shelf construction* [22] visualization systems, designers can then drag data properties from the *data panel* to encoding shelves in the *attributes panel* to specify the visualization encodings. As the encodings are specified, the system automatically adds axes and legends as appropriate. Once again, the edits apply to all “active” views in the system.

When the mark has been bound to a dataset, all the individual elements (e.g. all the bars in a bar chart) are placed in a group, similar to the notion of a collection in Data Illustrator [14]. The newly created marks and their group are shown in the *layers panel*, which displays both the user-specified marks and marks for auto-generated axes and legends in any of the views.

Applying Customizations Using Flexible Selections


Designers may want to customize particular marks in a visualization design or customize the visualization design for particular chart sizes. The system provides several strategies for performing flexible selection of visualization elements.



Designers can select marks directly on the visualization or from the *layers panel*. When designers first click a mark, the system selects all marks in its group; designers can double click a mark to select only a particular item in the group. Designers can also toggle the selection mode from the *toolbar*. When editing visualizations, the customizations only apply to the selected marks (those highlighted with an orange stroke).

Designers can refine the selection using data filters on the mark. Data filters are displayed in the *attributes panel* (Figure 4a) based on the selected mark. Regardless of the selection type (group or item), the particular element that was selected acts as the anchor for the data filters; for each data field in the underlying dataset, a filter is suggested using the value of that field for the selection anchor. When selecting a data filter the system refines the selection to only the marks where the condition holds. Repeatedly selecting a filter toggles the comparison operator (e.g., \geq , \leq , $=$, and so on). For example, in Figure 4, the text “Hurricane Katrina” acts as the anchor for the labels; for each of the five data fields, a simple filter is created using the underlying values of this anchor point (e.g., `cost \geq 161 and year \geq 2005`). Selecting a different anchor changes the suggested filters (e.g., selecting “Hurricane Harvey” suggests `cost \geq 125 and year \geq 2017`).

Enabling Customizations and Displaying Design Variation

When designers apply customizations to particular marks or views, the system helps to **show cross-device previews** by foregrounding the variation in the *layers panel* and *attributes panel*. When a mark is added or removed from a particular view, the contents of the *layers panel* are reordered to sort the elements based on which views they apply to. For example, in Figure 6 a separator shows that each view (“portrait” and “landscape”) has a custom axis; the marks displayed above these separators apply to all views (e.g., the two text marks). Customizations are also displayed in the *attributes panel*; in Figure 6, the selected “bar” mark originally had the color encoding set to “#f0f0f0”. A modification to this encoding has subsequently been applied to update the color to “firebrick”.

The *attributes panel* allows designers to view design variation for the mark encodings of the currently active views. Designers can select an icon next to a customization to refine the selection to only include marks where the customization applies. For example, in Figure 6, clicking the  icon beside the “firebrick” customization refines the selection to only update the marks with this color when subsequent edits are applied. This functionality exhibits the system’s support for flexible workflows; designers can edit particular customizations even when multiple views with different encodings are active.

Designers may also **propagate edits across views** by identifying and deleting design variations from the system. To propagate edits, designers can delete encodings that should no longer apply to the active visualizations. When hovering over the “delete”  symbol, the system **shows cross-device previews** of the change. This functionality helps designers to quickly preview the result of different modifications across multiple visualization designs and update concurrent designs with different encoding decisions. For example, in Figure 4, imagine that the designer wants to propagate the label simplifications in the “portrait” visualization to the “landscape” visualization. To do this, the designer marks both views as “active” (e.g., by locking  the “desktop” visualization) then deletes the encoding elements that are no longer desirable (e.g., `align→left` and `text→name`), which changes the alignment to `right` and the text to `replace(datum.name(‘Hurricane’, ‘’))` for all of the selected elements across both the “active” views.

Enabling and Customizing End-User Interactions

When a mark is selected, the *interaction panel* allows designers to specify the functionality of end-user interactions with the visualizations. Designers can first define the interaction type and then use the interaction to customize the display and encodings of visualization marks. Similar to other encoding customizations, when a customization applies to an interaction it is still displayed in the *attributes panel* (Figure 7).

REPRODUCING RESPONSIVE NEWS VISUALIZATIONS

To demonstrate the utility of the proposed techniques for responsive visualization design, we reproduced four real-world examples using our system [A13, A36, A50, A52]. These examples illustrate four visualization types: bar chart [A13, A36], map [A52], dot plot [A36], and line chart [A50]. Each example also demonstrates a major component of our system: (1) visualization design and customization as an iterative workflow [A13]; (2) flexible data filtering for customizing annotations [A52]; (3) designing drastically different visualizations via linked editing [A36]; (4) customizing end-user interactions for different chart sizes [A50]. The following sections explore each example in terms of the design guidelines. Demo videos for each of the four walkthroughs are included in the supplemental material. We also introduce three additional examples in the supplemental material; for all seven examples, we include a webpage showing the different device-specific visualizations that were designed with our system.

An Iterative Workflow for Simplifying a Mobile Design

The New York Times article “The Places in the U.S. Where Disaster Strikes Again and Again” [A13] exhibits several re-

sponsive techniques to simplify the mobile version of the “Total Cost of Major Natural Disasters” visualization (Figure 1).

The designer starts by creating the basic visualization: a bar chart with the year on the x-axis and the disaster cost on the y-axis. Next, the designer decides to annotate the major natural disasters contained in the data. The designer duplicates the bar mark using the “copy” option from the *toolbar* to create a new layer with the same encodings, and then customizes the mark to display the disaster name, and position the marks appropriately. The designer then decides to duplicate this text mark and update the text to display the disaster cost. Using **simultaneous cross-device edits**, the visualization is constructed for each specified view size.

With **cross-device previews** of how this design looks for all device contexts, the designer can easily notice that the current layout is ineffective on the “portrait” orientation of the phone. For the phone, the large whitespace margin wastes too much space and the bars become too narrow. Before finalizing the design, the designer can apply **device-specific customizations** to the portrait visualization including modifying and repositioning the labels, and adding rule marks to more clearly label the bars (Figure 1c-e). The state of the system after customizations have been applied is shown in Figure 4.

After applying customizations, the designer can use **simultaneous cross-device edits** to finalize the axis styles, bar colors, and font styling. These updates apply to all views, even though customizations have been performed for the portrait visualization. Finally, the designer can add a title to the top of the view. For the desktop and landscape visualizations however, the designer might decide to reposition the title in the whitespace of the chart area instead. After making this change, the designer decides that the title was actually preferable above the chart area for the landscape visualization. To reapply this design, the designer activates both the “portrait” and “landscape” visualizations and **propagates edits** from one view to the other by deleting the customizations associated with the title.

Discussion. This example illustrates a flexible, iterative workflow in which the designer can switch between different device views to apply unique customizations. By still maintaining links between visualization marks, the system can also support global customizations even after modification have been applied. By displaying all device views and the specific customizations of these views, the designer can maintain a complete picture of the responsive visualization design.

Applying and Refining Data-Driven Customizations

The Reuters article “Oil Spilled at Sea” [A52] includes a map visualization that reduces the number of text annotations, rescales the text, and updates the size encoding and legend. In our formative interviews, several participants noted that maps present a particular challenge because the aspect ratio of the chart is predetermined by the map itself. One participant explained that the “US map is a nightmare for responsiveness... you can have a very beautiful, detailed US map on the desktop but when you shrink it down to the phone you can barely see like 5 cities named in it” (P2).



Figure 5. Recreating the map “Incidents at Sea” from Reuters [A52]. (a) The visualization is cluttered with all of the symbol and text marks; inset: encodings for the text mark in the *attributes panel*. (b) Removing the text marks where size < 252,000 emphasizes the major spills; inset: the possible and selected data filters in the *attributes panel*. (c) Reintroducing annotations for notable, historical spills provides a point of comparison; inset: the text marks as displayed in the *layers panel*.

For this visualization, the designer can start by producing the basic visualization design using **simultaneous cross-device edits**. However, this process produces a cluttered map exhibiting many overlapping labels (Figure 5a). To reduce the clutter in this map, the designer can select the mark elements and use a data filter to remove elements below a particular threshold (e.g., size < 252,000). This action removes most of the labels in the view except for the largest ones (Figure 5b). However, the designer wants to compare these spills to others of historical notability. The designer can use the “annotate” mode from the *toolbar* to reintroduce labels by clicking the points for the Sanchi and Exxon Valdez spills (Figure 5c).

Due to the shape of the map and density of the data, the designer might feel at this point that the mobile versions could be improved. The designer starts with the “landscape” phone orientation (the more natural fit for the map) and applies **device-specific customization** to rescale and reposition the map projection to only include one of the comparison points: the Sanchi disaster. The designer also wants to be able to include the Sanchi disaster in the “portrait” version, but due to the distance of this point from the core data, it is hard to make the visualization fit. The designer therefore decides to rotate the map for the “portrait” orientation; this change maintains

the text direction of the labels for proper reading on a phone. To view the map in the unrotated orientation, the reader could change to the landscape orientation, which would adapt the view to the other version of the customized visualization.

To better support this reading experience, the designer wants to ensure that the two mobile visualizations are similar in all design decisions besides the rotation. The designer thus reactivates the “landscape” visualization, and can immediately see the **cross-device previews** in the *attributes panel*, which shows that the `scale` and `translate` properties do not match. The designer can thus **propagate edits** from the landscape visualization by deleting the undesirable customizations that applied to the “portrait” visualization.

Discussion. Our system enables flexible selection behaviors to support customization of the visualizations. The data filter selection provides a lightweight mechanism for refining selections based on the underlying data. This functionality facilitates data-driven customizations. The “annotate” \mathbb{I} mode then provides a lightweight mechanism for reincorporating deleted annotations to the view. This example also highlights the challenges with responsive visualization design for maps, but shows how a designer can mitigate this difficulty by considering the final experience of the reader.

Producing and Reusing Radically Different Designs

The New York Times article “With Kennedy’s Retirement, the Supreme Court Loses Its Center” [A36] includes a radical responsive redesign for the “In close decisions, Kennedy voted in the majority 76 percent of the time” visualization. In this example, the desktop version uses a horizontal dot plot whereas the mobile version uses a vertical bar chart; each design makes use of the device orientation while displaying the same data.

To produce these visualizations, the designer can start by creating either the bar chart or dot plot visualization, including the marks and text labels. The designer can then focus on applying **device-specific customization** by setting the active view and changing the underlying encodings for the mark and text (e.g., to change the mark from “bar” to “circle” and the text from `percentage` to `justice`). The designer can switch amongst the visualization views to further customize the encoding.

For these visualizations, the designer wanted to emphasize Kennedy’s position in the Supreme Court with respect to decisions by the majority and therefore customized some of the individual marks accordingly. Down the line, the designer might decide to reuse this visualization to emphasize the position of a future justice. To do this, the designer could swap out the underlying dataset to include updated numbers; since the customizations are applied relative to the device context and data fields, the visualizations could therefore easily adapt to a new set of data. The designer might then decide to **propagate edits** representing the highlight to a new justice in the dataset.

Discussion. While the visualizations for this example are drastically different and therefore highly customized, our system can help the designer maintain a clear overall picture of the two versions: how they vary and how they are the same. Figure 6 shows the state of the system panels when one of the marks has been selected. From this view, the designer can see

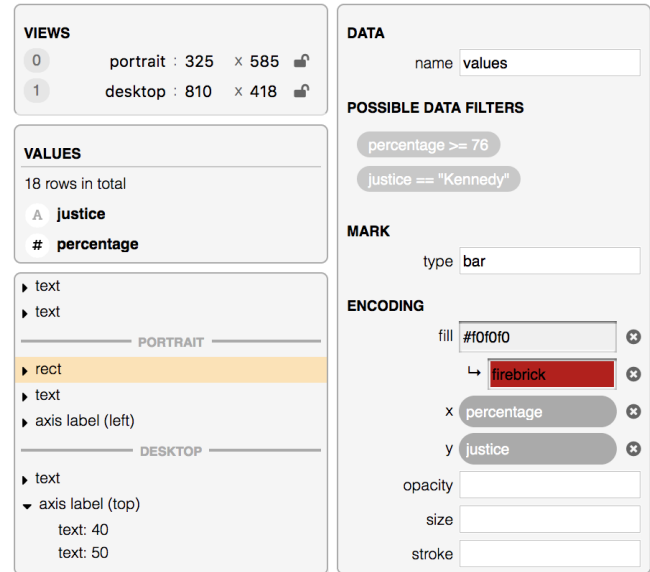


Figure 6. The systems panels for recreating the New York Times visualization: “In close decisions, Kennedy voted in the majority 76 percent of the time” [A36]. The *layers panel* shows that while the title and subtitle appear in both views, each device has a custom set of axes and marks. The *attributes panel* shows variation in the encoding properties such as the “color”, which has been used to highlight a particular bar.

that the mark “color” has changed from ‘#f0f0f0’ to ‘firebrick’ for some of the visualization marks. The layers panel also shows variation in which mark elements are visible on which views (e.g., to show that each device has a customized axis).

Custom End-User Interactions and Interactive Encodings

The National Geographic article “See Where Access to Clean Water Is Getting Better – and Worse” [A50] exhibits a variety of responsive techniques between the desktop and mobile version of the “Percentage of population without access to improved water” visualization. This example demonstrates the need for customized end-user interactions; the visualization uses a dropdown with fewer options on mobile and customized visual encodings for elements that have been interacted with.

For this example, the designer starts by customizing how the visualization should look when no interaction is applied. The designer creates the basic visualizations using **simultaneous cross-device edits** and applies **device-specific customizations** to the mobile design to remove excessive marks and update the encodings. To define the end-user interaction behavior, the designer can select the mark that should update when interacted with, and add a new interaction from the *interaction panel*. The designer can then specify how the end-user interaction should behave and update the mark encodings. Similar to the device-specific customizations, encodings or marks associated with an interaction are displayed in the hierarchical structure of the system. Figure 7 shows the state of the system panels after defining a dropdown interaction to select a line and display a corresponding text mark.

Discussion. Variation from end-user interaction is similarly displayed alongside other encoding modifications within the system to provide **cross-device previews** of the design vari-

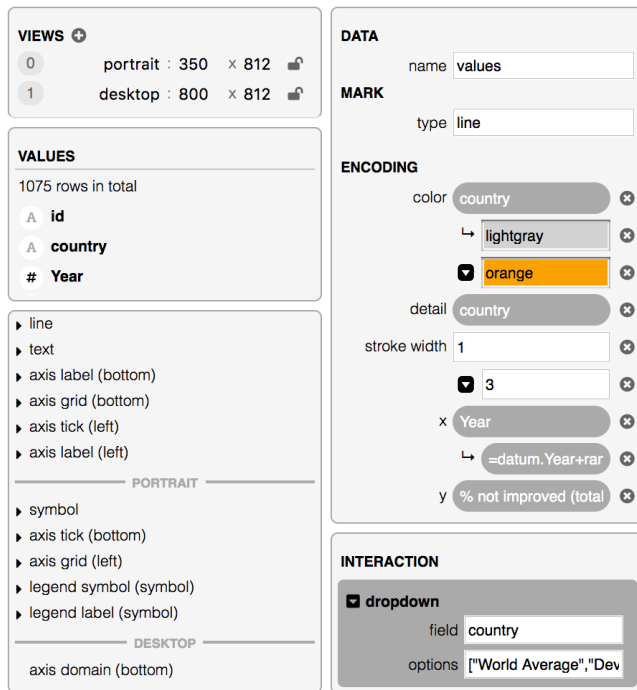


Figure 7. Designers may also customize the behavior of end-user interactions with the visualization. In this case, the designer specifies a dropdown interaction to update the line color and stroke width of the visualization. These customizations appear when the end-user interacts with a dropdown; however, the customizations are similarly displayed alongside the other encodings and customizations in the *attributes panel*.

ation. The *attributes panel* and *layers panel* provide a clear overview of all encoding decisions for a particular mark and thus allow the designer to easily update any encoding decisions for any use case throughout the system.

DISCUSSION, LIMITATIONS, AND FUTURE WORK

We built our system on top of Vega-Lite [23] to leverage its ability to express parameterized graphical elements (marks and axes) that exhibit reasonable default behavior when scaled to different devices. While this decision reduced the effort required to implement a functional visualization design tool, the capabilities of our current prototype are coupled with the underlying representation, sometimes resulting in awkward user experiences. For example, axis labels and text marks exhibit different sets of editable properties due to differences in the Vega-Lite specification. Furthermore, elements in one view will only match those in other views if they were specified simultaneously. Future work should update the system to decouple the front end from low-level details of the core visualization machinery. In this case, matches would be computed based on the visual similarity and underlying data of the elements, rather than the internal structure.

The current system allows designers to specify an arbitrary number of views based on their design needs. Similar to designers' current practice of focusing on a limited set of artboard sizes, our examples typically use three or four views targeting different device sizes. While it is possible to create more views, there are limitations on the amount of screen space available for development and the number of views that

a designer could feasibly view or comprehend at one time. Future work should more closely explore the development patterns of designers and how they would work with multiple views simultaneously. New techniques to cluster and summarize views could prove useful for alleviating challenges that arise when working with a larger space of designs.

Due to hesitance from our interview participants regarding automatic techniques, we chose to focus on an ad hoc, user-driven visualization design, which does produce a largely manual process. Exploring new techniques to increase the number of views or better summarize the space of designs could help designers transition to more automated or dynamic procedures. Future work should explore how best to ensure the transparency of automation within the development processes.

Our system includes a variety of interaction techniques such as keyboard shortcuts, toolbar menus and different types of mouse clicks for refining selections. Since generalized selections are a key part of our proposed workflow, it would be valuable to refine the usability and performance of such interactions. In particular, updating the data filter procedure to provide more intricate or generalizable suggestions could improve the utility of such a feature. Similarly, there are opportunities to explore new ways for designers to propagate customizations across views, perhaps through analysis of how the graphics themselves are arranged in each visualization.

In this work, we reproduce four real-world examples [A13, A36, A50, A52] to demonstrate the benefits of our system for responsive visualization design. To be clear, the procedures presented do not reflect the actual design process for the published visualizations; instead, they highlight a set of flexible, iterative workflows that contrast the more linear design process that designers typically adopt. Future work should explore the nuances of this system and the overall impact on the design of responsive visualizations in practice.

CONCLUSION

This paper explores the design of responsive visualizations that adapt the visualization design to different device sizes. We perform a survey of 231 responsive visualizations from twelve news organizations to examine existing responsive practices and further describe formative interviews with five authors about their design process and rationale. From the formative interviews, we identify four design guidelines and contribute a set of core system features to support the design of responsive visualizations. Our system displays multiple views for different device contexts and foregrounds design variation to provide a complete picture of the responsive techniques applied. Designers can construct visualizations using both simultaneous global edits or local customizations to the designs. Finally, we reproduce four real-world examples selected from our earlier survey. In contrast to the existing linear workflows described in the formative interviews, these examples demonstrate the expressiveness and flexibility of our system for supporting the iterative design of responsive visualizations.

ACKNOWLEDGMENTS

We would like to thank our reviewers, members of Adobe Research, the UW Interactive Data Lab, and many others for their for their interest and feedback on this work.

REFERENCES

- [1] Keith Andrews. 2018. Responsive Visualisation. *Data Visualization on Mobile Devices Workshop at CHI 2018 (MobileVis)* (2018). https://mobilevis.github.io/assets/mobilevis2018_paper_4.pdf
- [2] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics* 17, 12 (2011), 2301–2309.
- [3] Nadieh Bremer. 2019. Techniques for Data Visualization on both Mobile & Desktop. (2019). <https://www.visualcinnamon.com/2019/04/mobile-vs-desktop-dataviz>
- [4] The New York Times Company. 2017. ai2html. <http://ai2html.org/>. (2017).
- [5] Matt Conlen and Jeffrey Heer. 2018. Idyll: A Markup Language for Authoring and Publishing Interactive Articles on the Web. In *ACM User Interface Software & Technology (UIST)*. <http://idl.cs.washington.edu/papers/idyll>
- [6] Matt Conlen, Alex Kale, and Jeffrey Heer. 2019. Capture & Analysis of Active Reading Behaviors for Interactive Articles on the Web. *Computer Graphics Forum (Proc. EuroVis)* (2019). <http://idl.cs.washington.edu/papers/idyll-analytics>
- [7] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. 2010. Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-efficacy. *ACM Trans. Comput.-Hum. Interact.* 17, 4, Article 18 (Dec. 2010), 24 pages. DOI: <http://dx.doi.org/10.1145/1879831.1879836>
- [8] Sophia Fedeli and Katerina Eva Matsa. 2018. Use of mobile devices for news continues to grow, outpacing desktops and laptops. *Pew Research Center* (2018). <https://pewrsr.ch/2uvqS04>
- [9] Ben Frain. 2015. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing.
- [10] Roy Gal. 2017. Responsive visualizations coming to Power BI. *Microsoft Power BI Blog* (2017). <https://powerbi.microsoft.com/en-us/blog/responsive-visualizations-coming-to-power-bi/>
- [11] Datawrapper GmbH. 2019. Datawrapper. <https://www.datawrapper.de/>. (2019).
- [12] Jeffrey Heer, Maneesh Agrawala, and Wesley Willett. 2008. Generalized Selection via Interactive Query Relaxation. In *ACM Human Factors in Computing Systems (CHI)*. 959–968. <http://idl.cs.washington.edu/papers/generalized-selection>
- [13] Tom Horak, Andreas Mathisen, Clemens N. Klokmoose, Raimund Dachselet, and Niklas Elmqvist. 2019. Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA. DOI: <http://dx.doi.org/10.1145/3290605.3300846>
- [14] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. Data Illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 123.
- [15] Ethan Marcotte. 2014. *Responsive Web Design*. A Book Apart Publ.
- [16] Ethan Marcotte. 2015. *Responsive Design: Patterns and Principles*. A Book Apart Publ.
- [17] Donghao Ren, Matthew Brehmer, Bongshin Lee, Tobias Höllerer, and Eun Kyoung Choe. 2017. ChartAccent: Annotation for Data-Driven Storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 230–239.
- [18] Donghao Ren, Bongshin Lee, and Matthew Brehmer. 2018. Charticulator: Interactive construction of bespoke chart layouts. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 789–799.
- [19] Cedric Sam. 2018. Ai2html and Its Impact on the News Graphics Industry. *Data Visualization on Mobile Devices Workshop at CHI 2018 (MobileVis)* (2018). https://mobilevis.github.io/assets/mobilevis2018_paper_20.pdf
- [20] Arvind Satyanarayan and Jeffrey Heer. 2014a. Authoring Narrative Visualizations with Ellipsis. *Computer Graphics Forum (Proc. EuroVis)* (2014). <http://idl.cs.washington.edu/papers/ellipsis>
- [21] Arvind Satyanarayan and Jeffrey Heer. 2014b. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum (Proc. EuroVis)* (2014). <http://idl.cs.washington.edu/papers/lyra>
- [22] Arvind Satyanarayan, Bongshin Lee, Donghao Ren, Jeffrey Heer, John Stasko, John R. Thompson, Matthew Brehmer, and Zhicheng Liu. 2020. Critical Reflections on Visualization Authoring Systems. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2020). <http://idl.cs.washington.edu/papers/reflections-vis-authoring>
- [23] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2017). <http://idl.cs.washington.edu/papers/vega-lite>
- [24] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (2002), 52–65.

- [25] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 649–658.
- [26] Yingcai Wu, Xiaotong Liu, Shixia Liu, and Kwan-Liu Ma. 2012. ViSizer: a visualization resizing framework. *IEEE Transactions on Visualization and Computer Graphics* 19, 2 (2012), 278–290.
- ## APPENDIX A
- [1] Gregor Aisch. 2014. The Clubs That Connect the World Cup. *The New York Times* (2014). <https://nyti.ms/2nCTGXG>
- [2] Gregor Aisch and Karen Yourish. 2015. Connecting the Dots Behind the 2016 Presidential Candidates. *The New York Times* (2015). <https://nyti.ms/2ykNIYP>
- [3] Susanne Barton and Hannah Recht. 2018. The Massive Prize Luring Miners to the Stars. *Bloomberg* (2018). <https://www.bloomberg.com/graphics/2018-asteroid-mining/>
- [4] David Batty, Caelainn Barr, and Pamela Duncan. 2018. What Lies Beneath: The Subterranean Secrets of London’s Super-Rich. *The Guardian* (2018). <https://www.theguardian.com/money/2018/may/07/going-underground-the-subterranean-secrets-of-londons-super-rich>
- [5] Gurman Bhatia. 2018. India’s Premium Price of Petrol. *Reuters Graphics* (2018). <https://fingfx.thomsonreuters.com/gfx/rngs/INDIA-ELECTION-FUEL/010080DM0SB/index.html>
- [6] Gurman Bhatia and Manas Sharma. 2019. The Figures Behind the Faces. *Reuters Graphics* (2019). <https://graphics.reuters.com/INDIA-ELECTION-CRIMINAL-CANDIDATES/0100925031T/index.html>
- [7] Seth Blanchard and Reuben Fischer-Baum. 2018. One of the World Cup’s Best Goals was Even Crazier Than You Thought. *The Washington Post* (2018). https://wapo.st/2Nu5yk9?tid=ss_tw
- [8] Jay Boice and Rachael Dottle. 2018. 2018 March Madness Predictions. *FiveThirtyEight* (2018). <https://projects.fivethirtyeight.com/2018-march-madness-predictions/>
- [9] Larry Buchanan and Karen Yourish. 2018. From Criminal Convictions to Ethical Lapses: The Range of Misconduct in Trump’s Orbit. *The New York Times* (2018). <https://nyti.ms/2N3WGVj>
- [10] Weiyi Cai. 2018. War of Words. *Reuters Graphics* (2018). <http://fingfx.thomsonreuters.com/gfx/rngs/NORTHKOREA-USA-KIM-TRUMP/010070JM16P/index.html>
- [11] Manuel Canales and Sean McNaughton. 2019. See Which Countries Fund the Most Scientific Research. *National Geographic* (2019). <https://www.nationalgeographic.com/magazine/2019/05/data-show-why-china-science-research-is-booming/>
- [12] Pia Catton and Gus Wezerek. 2018. Nearly Half The Kentucky Derby Field Is Racing Against A Half-Brother. *FiveThirtyEight* (2018). <https://53eig.ht/2H0JSQH>
- [13] Sahil Chinoy. 2018. The Places in the U.S. Where Disaster Strikes Again and Again. *The New York Times* (2018). <https://nyti.ms/2GJjoe4>
- [14] Sahil Chinoy and Jessica Ma. 2019. How Every Member Got to Congress. *The New York Times* (2019). <https://www.nytimes.com/interactive/2019/01/26/opinion/sunday/paths-to-congress.html>
- [15] Matt Daniels. 2019. The Largest Vocabulary In Hip Hop. *The Pudding* (2019). <https://pudding.cool/projects/vocabulary/>
- [16] Fathom Information Design. 2014. What the World Eats. *National Geographic* (2014). <https://www.nationalgeographic.com/what-the-world-eats/>
- [17] Anna Flagg. 2017. The Opposite of Sanctuary. *The Marshall Project* (2017). <https://www.themarshallproject.org/2017/02/20/the-opposite-of-sanctuary>
- [18] Anna Flagg. 2018. The Myth of the Criminal Immigrant. *The Marshall Project* (2018). <https://www.themarshallproject.org/2018/03/30/the-myth-of-the-criminal-immigrant?ref=hp-1-112>
- [19] Walter Frick. 2016. The Decline of Yahoo in Its Own Words. *Harvard Business Review* (2016). <https://hbr.org/2016/06/the-decline-of-yahoo-in-its-own-words>
- [20] Russell Goldenberg. 2018. The World Through the Eyes of the US. *The Pudding* (2018). <https://pudding.cool/2018/12/countries/>
- [21] Russell Goldenberg and Amber Thomas. 2019. How Many High School Stars Make It in the NBA? *The Pudding* (2019). <https://pudding.cool/2019/03/hype/>
- [22] Kirk Goldsberry. 2019. How Mapping Shots In The NBA Changed It Forever. *FiveThirtyEight* (2019). <https://53eig.ht/2PF20gE>
- [23] Jackie Gu. 2018. Women Lose Out to Men Even Before They Graduate From College. *Bloomberg* (2018). <https://www.bloomberg.com/graphics/2018-women-professional-inequality-college/>
- [24] Eelke Heemskerk. APRIL 21, 2016. How Corporate Boards Connect, in Charts. *Harvard Business Review* (APRIL 21, 2016). <https://hbr.org/2016/04/how-corporate-boards-connect-in-charts>
- [25] Walt Hickey and Gus Wezerek. 2016. The Definitive Analysis Of ‘Love Actually,’ The Greatest Christmas Movie Of Our Time. *FiveThirtyEight* (2016). <http://53eig.ht/2he9BVh>
- [26] Josh Holder. 2018. How the NHS Winter Beds Crisis is Hitting Patient Care. *The Guardian* (2018). <https://www.theguardian.com/society/ng-interactive/2018/jan/11/how-the-nhs-winter-beds-crisis-is-hitting-patient-care>

- [27] Josh Holder and Alex Hern. 2018. Bezos's Empire: How Amazon Became the World's Most Valuable Retailer. *The Guardian* (2018). <https://www.theguardian.com/technology/ng-interactive/2018/apr/24/bezoss-empire-how-amazon-became-the-worlds-biggest-retailer>
- [28] Josh Katz, Kevin Quealy, and Margot Sanger-Katz. 2019. Would 'Medicare for All' Save Billions or Cost Billions? *The New York Times* (2019). <https://nyti.ms/2UFGVaV>
- [29] Ella Koeze and Neil Paine. 2019. The Story of the NBA Regular Season in 9 Charts. *FiveThirtyEight* (2019). <https://53eig.ht/2KHkqL>
- [30] Niko Kommenda, Caelainn Barr, and Josh Holder. 2018. Gender Pay Gap: What We Learned and How To Fix It. *The Guardian* (2018). <https://www.theguardian.com/news/ng-interactive/2018/apr/05/women-are-paid-less-than-men-heres-how-to-fix-it>
- [31] Daniel Lathrop and Anna Flagg. 2017. Killings of Black Men by Whites are Far More Likely to be Ruled "Justifiable". *The Marshall Project* (2017). <https://www.themarshallproject.org/2017/08/14/killings-of-black-men-by-whites-are-far-more-likely-to-be-ruled-justifiable>
- [32] Lauren Leatherby and Paul Murray. 2019. A Staggering Number of Candidates Are Running for U.S. President. *Bloomberg* (2019). <https://www.bloomberg.com/graphics/democratic-party-candidates-running-2020-election/>
- [33] Denise Lu and Karen Yourish. 2019. The Turnover at the Top of the Trump Administration. *The New York Times* (2019). <https://nyti.ms/2FQ0KBq>
- [34] Yolanda Martinez. 2018. Sending Even More Immigrants to Prison. *The Marshall Project* (2018). <https://www.themarshallproject.org/2018/05/20/sending-even-more-immigrants-to-prison>
- [35] Dave Merrill and Lauren Leatherby. 2018. Here's How America Uses Its Land. *Bloomberg* (2018). <https://www.bloomberg.com/graphics/2018-us-land-use/>
- [36] Alicia Parlapiano and Jugal K. Patel. 2018. With Kennedy's Retirement, the Supreme Court Loses Its Center. *The New York Times* (2018). <https://nyti.ms/2IyGAWh>
- [37] Adam Pearce and Dorothy Gambrell. 2016. This Chart Shows Who Marries CEOs, Doctors, Chefs and Janitors. *Bloomberg* (2016). <https://www.bloomberg.com/graphics/2016-who-marries-whom/>
- [38] Adam Pearce and Joe Ward. 2018. LeBron James Is Carrying the Cavaliers in a Historic Way. *The New York Times* (2018). <https://nyti.ms/2JojZ70>
- [39] Oliver Roeder. 2019. The Man Who Solved 'Jeopardy!'. *FiveThirtyEight* (2019). <https://53eig.ht/2UzjXsS>
- [40] Simon Scarr and Marco Hernandez. 2019. A Network of Extremism Expands. *Reuters Graphics* (2019). <https://graphics.reuters.com/SRI%20LANKA-BLASTS-PLOTTER/010091W52YP/index.html>
- [41] NPR Staff. 2015. The Urban Neighborhood Wal-Mart: A Blessing Or A Curse? *NPR* (2015). <https://n.pr/1IP5XF2>
- [42] Ashlyn Still and Howard Schneider. 2018. Looking for Workers. *Reuters Graphics* (2018). <http://fingfx.thomsonreuters.com/gfx/rngs/USA-ECONOMY-JOBS/010062VB4V2/index.html>
- [43] Jessica Taylor, Katie Park, Tyler Fisher, and Alyson Hurt. 2017. Health Care Plan Championed By Trump Hurts Counties That Voted For Him. *NPR* (2017). <https://n.pr/2n1am50>
- [44] The Data Team. 2018. The Global Slump in Press Freedom. *The Economist* (2018). <https://www.economist.com/graphic-detail/2018/07/23/the-global-slump-in-press-freedom>
- [45] Amber Thomas. 2019. Sing My Name. *The Pudding* (2019). <https://pudding.cool/2019/05/names-in-songs/>
- [46] Amelia Thomson-DeVeaux and Gus Wezerek. 2019. Here's Why The Anti-Abortion Movement Is Escalating. *FiveThirtyEight* (2019). <https://53eig.ht/2WVfYh0>
- [47] Jason Treat and Anna Scalamogna. 2014. We'll Have What They're Having. *National Geographic* (2014). <https://www.nationalgeographic.com/foodfeatures/diet-similarity/>
- [48] Cory Turner, Jennifer Guerra, Sam Zeff, Kate McGee, Aaron Schrank, Jenny Brundin, Rob Manning, Ana Tintocalis, and Paul Boger. 2016a. Is There A Better Way To Pay For America's Schools? *NPR* (2016). <https://n.pr/1SPMXfa>
- [49] Cory Turner, Reema Khrais, Tim Lloyd, Alexandra Olgin, Laura Isensee, Becky Vevea, and Dan Carsen. 2016b. Why America's Schools Have A Money Problem. *NPR* (2016). <https://n.pr/1p1NMag>
- [50] Nicole Washington, Jason Treat, John Kondis, and NG Staff. 2016. See Where Access to Clean Water Is Getting Better—and Worse. *National Geographic* (2016). <https://www.nationalgeographic.com/clean-water-access-around-the-world/>
- [51] Alex Webb and Chloe Whiteaker. 2016. Technology and Car Companies Are More Intertwined Than Ever. *Bloomberg* (2016). <https://www.bloomberg.com/graphics/2016-merging-tech-and-cars/>
- [52] Jin Wu, Weiyi Cai, and Simon Scarr. 2018. Oil Spilled at Sea: Putting the Sanchi Disaster Into Perspective. *Reuters Graphics* (2018). <https://graphics.reuters.com/OIL-SPILLS/010060SL1GQ/index.html>
- [53] Steven Yaccino, Jeremy Scott Diamond, and Mira Rojasasakul. 2015. This is Who Republican Presidential Contenders Follow on Twitter. *Bloomberg* (2015). <https://www.bloomberg.com/politics/graphics/2015-who-republican-candidates-follow/>