



NORTHWESTERN UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2024-06

June, 2024

Interactive Systems for Data Visualization for Multiple User Contexts

Hyeok Kim

Abstract

Data visualization helps people to understand and communicate about and around the world. How, when, and where people access and use visualization is becoming more and more diverse with respect to device types, perceptual abilities, levels of data/visualization literacy, etc. While creating visualizations for different viewing conditions, or contexts, is necessary to accommodate them, it is challenging to do so because of tediousness in design exploration for multiple versions and management of consistency across those versions. To find optimal designs, authors might have to apply different strategies per context-specific version. For example, an author might aggregate data points to prevent overplotting on smartphone screens, while keeping them disaggregated on larger screens. At the same time, the author needs to ensure each version is delivering consistent takeaways to achieve their communicative goals. Current visualization systems provide limited support for those challenges in multiple visualization user contexts, leading to inappropriate designs for users in non-conventional contexts, such as screen overflows on mobile screens or inaccessible charts for people with low vision.

Therefore, my dissertation aims to propose and demonstrate approaches to multi-context data visualization. I define a visualization user context as the viewing conditions of an audience that are relevant to their use of a visualization. I focus on two application areas of multi-context data visualization design: (1) *responsive visualization* for different device types and (2) *data sonification* that encodes data using auditory channels for immersive storytelling and accessibility.

Responsive visualization refers to adapting a visualization design for different device and screen types. Responsive visualization is essential to serve readers from diverse device types like desktops, smartphones, and tablets. To inform visualization tools, I first identify key design tradeoffs between maintaining implied takeaways and adjusting graphical density. I formalize loss measures that approximate changes to task-oriented insights based on the design tradeoff characterization to allow automated design tools to reason about consistency in takeaways across responsive versions while enumerating possible designs. To enable consistently expressing diverse responsive design strategies in authoring pipelines like coding, GUI, and automated recommenders, I contribute *Cicero*, a declarative grammar that expresses responsive design transformations. Lastly, to support automated design exploration within an authoring pipeline, I contribute *Dupo*, a mixed-initiative authoring tool for responsive visualization by consolidating these abstractions. I evaluate *Dupo* with six experienced graphics reporters who brought their prior responsive visualization cases. While using *Dupo*, participants could explore different design ideas without manual prototyping, thereby avoiding premature fixation on a particular design. They created designs that they were satisfied with but which they had previously overlooked.

Data sonification refers to mapping data variables to auditory variables, such as pitch, volume, and duration. Although data sonification is a widely applicable method for accessible visualization as well as immersive data experiences, developing sonifications is difficult due to limited software support. To enable easier creation of data sonifications, I present *Erie*, a declarative grammar for data sonification that expresses a sonification design in terms of data, tone, and encoding. Using *Erie*, I present novel sonification designs for data analysis like residual plots, Q-Q plots, and kernel density estimates. I also replicate prior sonification-based accessibility applications and extend them by allowing users to customize auditory encodings according to their preferences.

Ideally, different user context types are intertwined, such as accessible responsive visualization or audience retargeting on multiple device types. To effectively combine findings and advancements in specific user context types into useful tools, a consistent method for abstractly expressing relevant design tradeoffs and constraints is necessary. Based on my contributions and prior approaches, I propose a blueprint for a generalized approach to abstracting visualizations on multiple user contexts. This proposal characterizes a multi-context data visualization problem in terms of accommodation constraints for context-specific adjustments and preservation constraints for consistency across contexts. To show how this proposal will help future work to capture different combinations of user contexts, I demonstrate a prototype recommender for accessible responsive visualization and describe further application areas like accessible statistics and streaming data observations. I conclude this dissertation by outlining a future research agenda to generalize and extend my contributions to other user context types.

Keywords

Data visualization, data sonification, responsive visualization, and declarative grammar.

NORTHWESTERN UNIVERSITY

Interactive Systems for Data Visualization for Multiple User Contexts

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Hyeok Kim

EVANSTON, ILLINOIS

June 2024

© Copyright by Hyeok Kim, 2024
All Rights Reserved

Abstract

Data visualization helps people to understand and communicate about and around the world. How, when, and where people access and use visualization is becoming more and more diverse with respect to device types, perceptual abilities, levels of data/visualization literacy, etc. While creating visualizations for different viewing conditions, or *contexts*, is necessary to accommodate them, it is challenging to do so because of tediousness in design exploration for multiple versions and management of consistency across those versions. To find optimal designs, authors might have to apply different strategies per context-specific version. For example, an author might aggregate data points to prevent overplotting on smartphone screens, while keeping them disaggregated on larger screens. At the same time, the author needs to ensure each version is delivering consistent takeaways to achieve their communicative goals. Current visualization systems provide limited support for those challenges in multiple visualization user contexts, leading to inappropriate designs for users in non-conventional contexts, such as screen overflows on mobile screens or inaccessible charts for people with low vision.

Therefore, my dissertation aims to propose and demonstrate approaches to multi-context data visualization. I define a *visualization user context* as the viewing conditions of an audience that are relevant to their use of a visualization. I focus on two application areas of multi-context data visualization design: (1) responsive visualization for different device types and (2) data sonification that encodes data using auditory channels for immersive storytelling and accessibility.

Responsive visualization refers to adapting a visualization design for different device and screen types. Responsive visualization is essential to serve readers

from diverse device types like desktops, smartphones, and tablets. To inform visualization tools, I first identify key design tradeoffs between maintaining implied takeaways and adjusting graphical density. I formalize loss measures that approximate changes to task-oriented insights based on the design tradeoff characterization to allow automated design tools to reason about consistency in takeaways across responsive versions while enumerating possible designs. To enable consistently expressing diverse responsive design strategies in authoring pipelines like coding, GUI, and automated recommenders, I contribute CICERO, a declarative grammar that expresses responsive design transformations. Lastly, to support automated design exploration within an authoring pipeline, I contribute DUPO, a mixed-initiative authoring tool for responsive visualization by consolidating these abstractions. I evaluate DUPO with six experienced graphics reporters who brought their prior responsive visualization cases. While using DUPO, participants could explore different design ideas without manual prototyping, thereby avoiding premature fixation on a particular design. They created designs that they were satisfied with but which they had previously overlooked.

Data sonification refers to mapping data variables to auditory variables, such as pitch, volume, and duration. Although data sonification is a widely applicable method for accessible visualization as well as immersive data experiences, developing sonifications is difficult due to limited software support. To enable easier creation of data sonifications, I present ERIE, a declarative grammar for data sonification that expresses a sonification design in terms of data, tone, and encoding. Using ERIE, I present novel sonification designs for data analysis like residual plots, Q-Q plots, and kernel density estimates. I also replicate prior sonification-based accessibility applications and extend them by allowing users to customize auditory

encodings according to their preferences.

Ideally, different user context types are intertwined, such as accessible responsive visualization or audience retargeting on multiple device types. To effectively combine findings and advancements in specific user context types into useful tools, a consistent method for abstractly expressing relevant design tradeoffs and constraints is necessary. Based on my contributions and prior approaches, I propose a blueprint for a generalized approach to abstracting visualizations on multiple user contexts. This proposal characterizes a multi-context data visualization problem in terms of accommodation constraints for context-specific adjustments and preservation constraints for consistency across contexts. To show how this proposal will help future work to capture different combinations of user contexts, I demonstrate a prototype recommender for accessible responsive visualization and describe further application areas like accessible statistics and streaming data observations. I conclude this dissertation by outlining a future research agenda to generalize and extend my contributions to other user context types.

Acknowledgment

Becoming a PhD has been an humbling experience for me to realize the support from my mentors, my family, and my friends.

I thank my mentors for recognizing my possibilities. First, my advisor, Jessica Hullman, has offered me tremendous support since I started my PhD program. If I were hesitant to graduate, the only reason would be that I will no longer be able to get as much of her feedback as I have gotten. As a researcher, she demonstrated how to set higher standards in doing research along with trust and careful feedback. *Jessica's feedback is touching.* As a person, she has cared my well-being so that I was always and confidently able to discuss with her when I struggled. Her caring has kept me from exhausting myself. As an advisor, she let me prioritize research as if there were nothing a PhD student should be worried about other than research.

Next, I was very fortunate to work with Jane Hoffswell through my internship at Adobe Research and subsequent collaborations. Whenever I communicate with her, I realize how brilliant and thoughtful she is. Although it was mostly over Zoom and Overleaf due to the pandemic, her thoughtfulness has been always communicated through the screen. I hope to continue collaborating with her in the future.

Matthew Brehmer showed me how to do research with people. If I could legally steal anything from him, I would steal his human skills along with his research ambitions. My internship with him at Tableau Research helped me explore spaces where researchers, practitioners, and users cowork to enable useful technologies.

I thank Matthew Kay and Michael Horn for being on my committee with thoughtful, critical, but kind feedback.

My dissertation and PhD was possible with my collaborators. First, Dominik Moritz introduced me to the pure joy of logic in programming, influencing how I shaped my research. Next, Ryan Rossi intellectually challenged me, which led to high-quality research outcomes. Yea-Seul provided insights that helped me to make a smooth transition to sonification research. Lastly, Arjun Srinivasan has been a supportive and *fun* collaborator.

My family has backed my PhD. 가족들은 항상 나의 진로를 지지해주었습니다. 어께 너머로 엄마에게 배운 것들을 통해서 스스로를 정돈할 수 있었습니다. 엄마는 본인 오랜 시간 부모님과 떨어져 지냈지만, 자식인 내가 타국에서 지내기로 한 결정을 지지해주었습니다. 쉽지 않은 마음입니다. 아빠 건강만 해. 아마도 누나가 묵묵히 두 남매의 몫을 해주고 있어서 내가 유학 생활을 편하게 할 수 있었습니다. 모두 너무 고맙습니다. 그리고 미국에 있는 이모들과 이모부, 형들과 누나, 멀리 나와 있음에도 집이 되어 주어서 감사합니다.

My friends from MU Collective, Northwestern CS, and HCI/VIS community made me feel belonging to communities. I tremendously thank Paula Kayongo, Stephanie Jones, and Lukas Lazarek for being my best friends, inspiration, and a safe place. My lab mates, Priyanka Nanayakkara, Abhraneel Sarma, Fumeng Yang, Maryam Hedayati, Dongping Zhang, Lily Ge, Taewook Kim, Mandi Cai, Sheng Long, Ziyang Guo, Charles Cui, and Negar Kamali, together contributed to a supportive research environment. My graduate life was much more fun with Tommy, Nick, Stefany, Vaidehi, Madhav, Nathan, Nathaniel, Melissa, Atmn, and Sanchit. I thank Alex Kale, Cindy Xiong Bearfield, Alireza Karduni, and Inha Cha for friendship, advice, and support.

I also thank Nicholas Gobble for being my best friend in the town and supporting me when it was hardest.

Dedication

To Kims

Table of Contents

Abstract	3
Acknowledgment	6
Table of Contents	9
List of Figures	14
List of Tables	22
1 Introduction	25
1.1 Challenges in Authoring Data Visualization for Multiple Contexts . . .	27
1.2 Thesis Contributions	33
1.3 Prior Publications and Authorship	38
2 Related Work and Background	40
2.1 Visualization User Contexts	40
2.2 Responsive Visualization	46
2.3 Accessible Visualization and Data Sonification	50
2.4 Declarative Grammars for Data Visualization	57
2.5 Visualization Design Recommendation	59

	10
3 Characterizing Tradeoffs in Responsive Visualization	62
3.1 Responsive Visualization Design Patterns	64
3.2 Trade-offs in Responsive Visualization	81
3.3 Discussion	88
3.4 Conclusion	91
4 Formalizing Changes to Task-oriented Insights under Responsive Transformations	93
4.1 Related Work	95
4.2 Problem Formulation	96
4.3 Task-oriented Insight Preservation Measures	98
4.4 Prototype Responsive Visualization Recommender	108
4.5 Model Training and Evaluation	114
4.6 Discussion and Future Work	124
4.7 Conclusion	128
4.8 Acknowledgement	128
5 CICERO: a Declarative Grammar for Responsive Visualization	129
5.1 Background: Responsive Web and Visualization	132
5.2 Design Guidelines for a Responsive Visualization Grammar	134
5.3 Responsive Visualization Grammar	137
5.4 Principles for CICERO Compiler	149
5.5 Reproducing Real-World Examples	157
5.6 Potential Applications for CICERO	160
5.7 Limitations	167
5.8 Conclusion	168

	11
6 DUPO: a Mixed-initiative Authoring Tool for Responsive Visualization	169
6.1 Usage Scenarios	171
6.2 Design Considerations	173
6.3 Recommendation Pipeline	175
6.4 User Interface	181
6.5 User Study	192
6.6 Discussion	200
6.7 Conclusion	202
7 ERIE: a Declarative Grammar for Data Sonification	203
7.1 Background: an Overview of Auditory Channels	205
7.2 Formative Study: Gaps in Sonification Development Practices	206
7.3 Design Considerations	209
7.4 ERIE Grammar	211
7.5 ERIE Compiler and Player for Web	225
7.6 Demonstration	230
7.7 Discussion	248
7.8 Conclusion	252
8 Discussion: Toward a Framework for Multi-context Data Visualization	254
8.1 Problem Formulation	257
8.2 Example Cases	263
8.3 Potential Applications	269
9 Conclusion	277
9.1 Summary of Contributions	277
9.2 Reflections	280

	12
9.3 Future Research Directions	282
9.4 Concluding Remarks	288
References	289
A Responsive Visualization Cases for the Design Space and Tradeoff Analyses	317
B Survey of Responsive Visualization Practitioners	324
B.1 Methods	324
B.2 Results	325
B.3 Actual Questionnaire	330
C Detailed Results for Task-oriented Insight Loss Measure Evaluation	334
C.1 Rank Correlation Results	334
C.2 Training Results	336
D Formal Definitions of Cicero and Extended Vega-Lite	340
D.1 Cicero Formal Definition	340
D.2 An Extended Version of Vega-Lite	343
E Technical Details for Cicero	347
E.1 Cicero Compiler API for Extended Vega-Lite	347
E.2 Recommender Prototype for Responsive Visualization	349
F Additional Walkthrough Cases for Cicero	355
F.1 Aid Budget	355
F.2 Disaster Cost	358
G Applying Cicero to MobileVisFixer	362

H	Technical Details for Dupo	364
H.1	Rationales for Exploration Suggestions	364
H.2	Rationales for Quick Edits	370
H.3	Loss Measures for the Recommender	371
I	Dupo Pilot Study	376
I.1	Method	376
I.2	Changes to the System Design	377
I.3	Changes Made to the Study Design	381
J	Dupo User Study Details	382
J.1	Study Protocols	382
J.2	Analysis Details	392
K	Sonification Tutorials and Design Cases for the Formative Study of ERIE	398
K.1	Collected Tutorials	398
K.2	Collected Sonification Designs	401
L	Technical Details of Erie	409
L.1	Customizing a Tone	409
L.2	Encoding	410

List of Figures

- 1.1 Example visualization designs that are not suitable for multiple user contexts. (Left) A visualization dashboard (Credit: Iaroslava Mizai) where the contents overflow on smartphone screens. The screenshot was taken on a smartphone. (Right) A communicative visualization that is not fully accessible (Credit: Fivethirtyeight). The only screen-readable part was the title of the table. 26
- 1.2 An example responsive design where the desktop (left) and smartphone (right) version use different encodings for the same information to adjust visual density (Credit: the New York Times). The desktop version encodes the ‘increase in homicide rate in 2015, compared with the average rate’ using the heights of arrows positioned on the corresponding states (i.e., an interval type channel). The smartphone version encodes the same variable by listing the states in a descending order (i.e., an ordinal type channel). 28
- 1.3 Data updates can affect visualization designs by adding more visual marks. While the effect is relatively marginal on desktop devices (a1 → a2), it can be bigger on smartphone screens (b1 → b2), causing screen overflows (b3). If the author wants to produce a more compact view on smartphone screens, then they could choose a different set of encodings (b4). 30
- 1.4 An example responsive visualization where the desktop (left) and smartphone (right) versions may imply different relationships between variables (Credit: the New York Times). The desktop version may imply a weaker effect of the independent variable (parent’s socioeconomic status) on the dependent variable (children’s academic progress), while the mobile version may imply a stronger effect. 31
- 3.1 Properties of our visualization sample. We reconstructed cell count by multiplying the number of tuples (records) and the number of fields (columns) and grouped cell sizes by k-means clustering (k = 5). 65

3.2	Screenshots of <i>Bond Yield</i> 's desktop and smartphone view pair that illustrates <i>remove records</i> , <i>remove annotations</i> , <i>remove emphases</i> , and <i>reduce width</i> . Blue highlights indicate parts of the desktop view that are removed in smartphone.	68
3.3	Screenshots of <i>U.S. Cabinet</i> 's desktop and smartphone view pair that demonstrates <i>disable hover interaction</i> , <i>remove encoding</i> , <i>serialize label-marks</i> , <i>transpose axes</i> , and <i>fix tooltip position</i>	70
3.4	Screenshots of <i>French Election</i> 's desktop and smartphone view pair that demonstrates <i>add encoding</i> , <i>externalize annotations</i> , and <i>number annotations</i> . Yellow highlights indicate parts that are added or repositioned in smartphone.	72
3.5	The dimensions of design patterns for responsive visualization.	74
3.6	Examples of design patterns for responsive visualization, grouped by the target of the change (columns). The left and right side for each pattern denote desktop and smartphone views, respectively. Orange is used to highlight those elements that change from desktop to smartphone.	75
3.7	The distribution of responsive visualization strategies observed in our sample. Each strategy includes a Target (rows) and Action (columns), each of which we further categorized (bold labels). The gray-shaded cells denote combinations of target and action that we have not observed, and the dark gray cells indicate impossible combinations by definition. 'Ref.' stands for 'references.'	80
3.8	Motivating cases for trade-off analysis. From desktop line or area chars with a wide landscape aspect ratio, (A) <i>Bond Yield</i> removes records, (B) <i>Cold Math</i> reduces width, and (C) <i>Megabank</i> split states into panels for smartphone views.	82
3.9	Examples of how transforming visualizations to fit narrower screen dimensions can change graphical perception.	86
4.1	Example responsive transformations for small screen generated from a large screen design: (a) proportionate rescaling, (b) disproportionate rescaling, (c) transposing axes, and (d) increasing bin size.	94
4.2	A pipeline for a responsive visualization recommender	97
4.3	Our notation for a visualization. Rendered values are defined in the space implied by the visual variable (e.g., pixel space for position or size, color space for color).	98
4.4	Responsive transformations that may cause identification loss.	100
4.5	Responsive transformations that may cause comparison loss.	101
4.6	Responsive transformations motivating trend loss.	104

4.7	Components of computing trend loss. (a) Calculating area between curves by standardizing chart size and interpolating break points. (b) Dividing and matching subgroups. (c) Linearizing color scale. D indicates desktop view, and S denotes smartphone view.	106
4.8	Prototype pipeline. (1) The full specification of an input source view in ASP. (2) Enumerating targets by extracting a partial specification of the source view and generating a search space using an ASP solver. (3) Evaluating targets by computing our loss measures and ranking them using a model trained on human-produced rankings. (4) Ranked targets.	109
4.9	(a, b) Example target transformations enumerated by our prototype responsive visualization recommender (total size of search space per source given as #Targets). ♣ indicates rankings of each five targets per source view predicted by our best model (see Section 4.5.3). (c) Source visualizations for our user study (also includes a and b). Sources views have width of 600px and height of 300px. The width of targets is fixed as 300px. Data sets are from <i>Our World in Data</i> [210–212]. Continuous, Nominal, Temporal, Identification loss, Comparison loss, and Trend loss.	112
4.10	(a) Study design. (b) Task interface. (c) Quintile sampling of targets for task materials.	117
4.11	(a) Joint distributions of rankings of target views in each pair of aggregated features (the source visualization is shown in Figure 4.9a). (b) Kendall rank correlation coefficients for targets of our source views in Figure 4.9. Cmp (comparison).	120
5.1	Design specifications for label-mark serialization using (a1) Vega-Lite and (a2) CICERO and parallelization using (b1) Vega-Lite and (b2) CICERO.	131
5.2	Responsive transformation from axis labels to a legend accompanied by a layout change for smaller display.	135
5.3	Two possible approaches to specifying responsive transformations. (a) Decorating a specification with conditional statements. (b) Separately defining responsive transformations.	136
5.4	Examples and roles in the CICERO grammar. (a) An overview of a CICERO specification with a rule describing “modify the color of the marks to red”. (b) role expressions used in CICERO. (c) Example transformations referred to in Section 5.3.	139
5.5	The formal specification of CICERO. Section D.1 provides more detailed description.	140

5.6	An example CICERO rule describing partial transpose. The bars are grouped by columns in the left view (before) and by rows in the right view (after). The entire set of transformations for this case (Aid Budget) can be found in the Section F.1.	148
5.7	The pipeline for our CICERO compiler, developed for our extended version of Vega-Lite.	149
5.8	An example case (Aid Budget) for a downstream effect to the layout of elements (moving a column axis to a row axis; line 1–8) and applying a rule (reordering a nominal y axis) to the previously transformed view (line 9–15).	151
5.9	An example use case (Disaster Cost) for our CICERO compiler’s default behavior for replacing the title as an internal annotation (line 1–6) and for introducing newly added elements (axis labels and grid lines; line 7–9).	153
5.10	An example use case (Covid Spending 1) for our CICERO compiler’s treatment of multiple series of existing elements. In this case, our CICERO compiler adds new axis labels by mimicking the most similar type of the existing axis labels according to the number of subelements (text lines).	154
5.11	Rules to change the color of marks (a) by specifying the mark for the “Apparel” category and (b) by generally changing the color of all marks (independent of the data).	156
5.12	Thirteen responsive visualization use cases reproduced using CICERO. The blue- and gray-bordered views are the desktop and mobile versions, respectively. The mobile versions of the Oil Spills case are from (1) the original article and (2) the version suggested by Hoffswell et al. [99]. Full size images are included in the online gallery (https://see-mike-out.github.io/cicero-supplemental/).	157
5.13	A walk-through example case of Bond Yields from a desktop version (top left) to a mobile version (top right). Starting with the desktop version, we first resize the chart to fit to a mobile screen (line 2–4), remove a subset of data for earlier years (line 5–10), remove the area mark (line 11–14), update grid lines by rescaling the domain of the y position channel (line 15–21), and reposition the annotation (22–30).	159
5.14	Selected examples among top seven recommendations for Bond Yields case from desktop to mobile. The original design is shown in Figure 5.13.163	
5.15	Expressing transformation strategies of MobileVisFixer [93] in CICERO. Line 2–4: decreasing the range of the x axis by reducing the width of the chart. Line 7–9: decreasing the font size using <code>prod</code> keyword.	165

- 6.1 Two proposed usage scenarios for a mixed-initiative responsive visualization design tool. **A** A novice author creating responsive designs in a desktop-first manner. **B** An experienced author editing responsive artboards simultaneously. 172
- 6.2 DUPO's three recommendation pipelines: *Exploration*, *Alteration*, and *Augmentation*. Initiated by a user action, each pipeline gathers the input data, runs a search, and returns the output suggestions. **C** The user can run the *Exploration* pipeline to see high-level transformations (e.g., to data, marks, encodings, and layout) by pressing the *recommender button* from the *toolbar* or creating a new artboard **C1**. Then, DUPO takes the source design and user constraints as input **C2**. DUPO generates alternatives with high-level changes and ranks them **C3**. These design alternatives are presented in the *recommender window* **C4**. If the user clicks the *hide this* button of an alternative, DUPO stores the alternative design in the *hide this history* as a user constraint for next time the user runs the *Exploration* pipeline. **D** The user can run the *Alteration* pipeline to see low-level transformations (e.g., to references, text, interactions) by selecting *see similar* for an alternative design **D1**. DUPO takes the alternative of interest, the user constraints, and the source design as input **D2**. DUPO then populates alternatives for low-level changes and evaluates them **D3**. The *Alteration* suggestions then appear next to the initial *Exploration* design **D4**. **E** The *Augmentation* pipeline is initiated after a user makes a manual edit **E1**, at which point DUPO takes its CICERO rule as input **E2** and searches a pre-defined search space **E3** to suggest *quick edits* that are commonly applied with the manual edit **E4**. 176
- 6.3 A walkthrough example for creating a responsive visualization using DUPO in a mobile-first manner. Kris starts by drafting an initial bar chart for a mobile phone, and then uses DUPO's three recommendation pipelines to refine the design. 183

- 6.4 An overview of DUPO’s interface. **G** By selecting the *mark* menu (🔍) from the *toolbar* (G1), the user can edit marks (e.g., mark type and encodings) in the *editing panel* (G2). To set an encoding channel, the user drags a data field from the *data widget* (G3) and drops it on the desired field. In the *navigation bar* (G4), the user can undo or redo by clicking the arrow icons and search the system menus. The *artboard area* (G5) displays responsive *artboards* (G6). The user can load suggestions using the *recommender button* (G8), and set *responsive locks* on elements that the recommender must keep (G9). At the bottom right corner, the user can *quickly access particular artboards* (G10). **H** Each artboard consists of *header* (H1), *view area* (H2), and *status bar* (H4). The *header* provides options to manage its *status* and *settings* (H7) and indicates the *activeness* and *lock status* (H8). In the *view area*, the *device size indicator* (H3) helps the user to check if the content overflows the intended space. From the *status bar*, the user can access the *edit history* (H5) and *versions* (H6). **J** Some manual edits by the user prompt DUPO to recommend associated *quick edits*, appearing on the top right corner of the *editing interface* (G7). **K** The user has requested *design suggestions* (K1) generated for a smartphone version. From the *action widget* (K2), the user can *branch* each suggestion as a new artboard, *apply* it to the current artboard, request further *Alteration* suggestions, *hide* a suggestion (K4), and read the rationales for each suggestion. The *control panel* (K3) allows users to *apply their custom edits* to the suggestions and toggle the depth of descriptions in the *action widget*. **L** The user can review the *history* (L1) of their use of the recommender (L) and customize the recommender parameters (L2) (🔧) from the *editing panel* (G2). 184
- 6.5 Selective design outcomes from the user study with six expert responsive visualization authors. Participants provided their own visualizations for use in the study. For anonymization purposes, we replaced participants’ data sets while maintaining the cardinality and data type, edited the text content but preserved annotation length, and altered the color schemes. A star icon (★) indicates an edit from the recommendations, a pencil icon (✎) denotes a manual edit made by a participant, and a cursor icon (👉) represents a manual edit using direct manipulation. 193
- 7.1 The formal definition of Erie. For applicable elements, roughly analogous visualization elements are denoted by \approx signs. 216
- 7.2 Our replication and extension of Audio Narrative [64] using ERIE. In addition to the originally offered sequencing and speech description, we included options for using different encoding channels (A) and playing the scale description (B). 244

7.3	Our replication and extension of Chart Reader [72] using ERIE. We further included user options for toggling the hover/selection interaction (A) and aggregation (B).	246
7.4	Our replication of the <i>Nine Rounds a Second</i> article [245] using ERIE.	248
8.1	The pipeline of the prototype recommender builder for accessible responsive visualization. (a) A developer defines contexts and constraints that are translated to Answer Set Programming expressions. (b) The developer provides templates for visualization and sonification. (c) An author configures a dataset and columns to encode, which are used to populate candidate designs. At this stage, the builder computes the loss values between each pair of populated designs. (d) The author obtains design suggestions ranked by the cost caused by violating soft constraints. I assume that the cost model is provided by the developer while providing a room for designers to make minor adjustments.	270
8.2	A prototype design recommender for accessible responsive visualization. (a) A developer defines the set of contexts and relevant constraints. (b) A developer provides a set of design templates. (c) An author specifies the dataset and data fields to encode. (d) An author receives design suggestions.	271
C.1	Rank correlations between disaggregated loss measures.	335
D.1	The detailed formal specification of Cicero (part 1)	341
D.2	The detailed formal specification of Cicero (part 2)	342
D.3	The formal specification of Extended Vega-Lite (part 1)	345
D.4	The formal specification of Extended Vega-Lite (part 2)	346
E.1	A Cicero Compiler API use cases. Line 1–14: instantancing a <code>Cicero</code> class object. Line 16–22: pipelining the job using the <code>loadCicero</code> function.	349
E.2	The pipeline of our prototype recommender.	350
F.1	A walk-through example case of Aid Budget (Section F.1).	356
F.2	A walk-through example case of Disaster Cost (Section F.2).	361
G.1	Re-expressing actions in MobileVisFixer [93] using Cicero	363
H.1	An example density grid (B) approximated for a map with circle marks (A). The color in the density grid indicates the number of occupying visualization elements in each grid cell. Geographic shapes are excluded when it has no encoding.	373
I.1	Updates to duplicate and artboard creation	378
I.2	Updates to the recommender buttons and artboard area	379

I.3	Updates to the Control Panel	380
J.1	Training visualizations for Dupo's user study.	388
J.2	Dupo user study log data for overall usages.	395
J.3	Dupo user study log data for recommender usages (part 1).	396
J.4	Dupo user study log data for recommender usages (part 2).	397

List of Tables

2.1	Comparison of ERIE to prior sonification toolkits. Abbreviations: VL (VoxLens and Sonifier.JS) [127, 128], HC (Highcharts Sonification) [35], WSS (Web Sonification Sandbox) [129], SC (Sonification Cell) [130], AAG (Apple Audio Graph) [131], DGB (DataGoBoop) [132], PR (PlayIt-ByR) [133], XS (xSonify) [134], SS (Sonification Sandbox) [66], SY (Sonifyer) [36], IST (Interactive Sonification Toolkit) [135], SW (Sonification Workstation) [136], SA (SonArt) [137], L (Listen) [138], M (MUSE) [139], PS (Personify) [140], Str (Strauss) [141], Sta (StarSound) [142], SD (SODA) [143], Eq (Equalizer), Pow (Power function), Sqrt (Square-root function), Sym-Log (Symmetric log function).	54
4.1	The set of features for our ML models by each chart type. These features are either concatenated or differentiated for each pair of targets. Aggregated features are the sum of the corresponding Disaggregated features. Pink, bold-bordered circles represent required features, and yellow, light-bordered circles optional encoding-specific features. . .	115
4.2	(a) Prediction accuracy of our models, averaged over LOO cross validation. Other performance measures (AUC score and F1-score) appeared similarly to accuracy. e stands for the number of estimators of each random forest model. (b) Average importance of Disaggregated features (g = difference) measured by impurity-based importance from training a random forest model ($e = 50$) 10 times.	122
7.1	The results of data transforms in Section 7.4.1.	213
7.2	The sonification output for an auditory histogram in Section 7.4.1. “#” indicates the playing order of each part. Units: seconds (start, end, duration) and Hz (pitch). “Sine” means a sinusoidal oscillator.	215
7.3	The sonification stream order for the auditory histograms repeated by the <i>origin</i> and <i>cylinders</i> variables.	222
7.4	The sonification stream order for the auditory histograms sequenced by the <i>origin</i> field and overlaid by the <i>cylinders</i> field.	223

7.5	The default scale description provided by <i>Erie</i> for the walkthrough case. These items are played before the sonification in Table 7.2 by default.	227
7.6	The audio queue resulting from a sparsity sonification spec in Section 7.6.1.1. “Q” indicates the index of each sub-queue. “After prev.” means “play after the previous sound” within the same sub-queue. A tapping pattern, $[a, b] \times c$, means a tap sound for a seconds and a pause for b seconds are repeated c times (the last pause is omitted). A tapping pattern, $[a, b, c]$, means a pause for a seconds, a tap sound for b seconds, and a pause for c seconds.	232
7.7	A preview of the <code>penguins.json</code> dataset.	234
7.8	The audio queue resulting from a kernel density estimate sonification spec in Section 7.6.1.2. “Q” indicates the index of each sub-queue. The pitch values (range from 0 to 700) are low because they are representing the both-side tails of each estimated density distribution.	237
7.9	The audio queue resulting from a model fit sonification spec in Section 7.6.1.3. “Q” indicates the index of each sub-queue.	242
8.1	The accommodation constraints for the accessible responsive visualization problem described in Section 8.2.1. \$Difference means the gap between the constraint’s value and a candidate’s value	266
8.2	The preservation constraints for the accessible responsive visualization problem described in Section 8.2.1. *Loss should range from 0 to 1.	267
8.3	The accommodation constraints for the audience retargeting problem described in Section 8.2.2.	268
8.4	The preservation constraints for the audience retargeting problem described in Section 8.2.2. *Loss should range from 0 to 1.	268
A.1	List of responsive visualization cases used in Chapter 3 (part 1)	317
B.1	Directions for responsive visualization design process. Responses in “ are suggested by participants.	325
B.2	Frequency of considering mobile views.	325
B.3	Average rank indicating perceived importance (out of 7) for seven possible guidelines.	326
C.1	Training results for all model configurations (part 1)	336
H.1	Initial weight terms	375
I.1	Dupo pilot study participants	376

- K.1 The distribution of sonification use cases. *Audio tools refer to professional audio editing tools (e.g., Ableton Live) and electronic music equipment (e.g., sequencers). **Circuit-based sonifications mean devices that generate sonifications by physical inputs through some kind of electronic circuits. ***While I could sense that the sounds are electronically generated (e.g., sine-wave oscillator or synthesizers) but they did not provide clear creation methods. 401

Chapter 1

Introduction

Data visualization helps us to understand and communicate about the world. Visualizations offload cognition to perception [1–3], allowing people to find patterns and insights from data by effectively visualizing them. Such strengths make visualization essential across various domains and tasks, such as exploratory data analysis [4, 5], journalism [6], and health analytics [7], to name a few.

This ubiquitous use of data visualization means that its users are coming from different contexts of use. Readers may use different devices or displays like desktops, tablets, smartphones, and paper. They may have different visual perception abilities (e.g., Blindness and color-blindness). They may exhibit different levels of expertise in data analysis and domain knowledge. I use a (*visualization user*) *context* to refer to the viewing conditions of an audience group that are relevant to their use of visualization.

Providing visualizations for only a single context is unlikely to be successful for communicating data insights to a broader audience than a single team because doing so prevents readers in different contexts from having basic access to the contents. For example, analytic visualizations created only for desktops like the one in



Figure 1.1: Example visualization designs that are not suitable for multiple user contexts. (Left) A visualization dashboard (Credit: Iaroslava Mizai) where the contents overflow on smartphone screens. The screenshot was taken on a smartphone. (Right) A communicative visualization that is not fully accessible (Credit: Fivethirtyeight). The only screen-readable part was the title of the table.

Figure 1.1 (Left¹) fail to show and communicate all their information to mobile readers due to the screen overflow problem. As the number of readers on mobile devices increases [8], the lack of mobile support cause content providers to lose a significant portion of media coverage. In addition, visualizations in online news articles that have no alternative text or audio graph (e.g., Figure 1.1-Right²) hinder Blind readers from accessing the information, which threatens data equity [9]. Thus, providing visualizations on multiple user contexts is no longer optional, but mandated for authors to communicate data with a broad audience.

¹https://public.tableau.com/app/profile/iaroslava/viz/Dashboard_17066338442540/OVERVIEW

²<https://projects.fivethirtyeight.com/polls/governor/2022/illinois/>

1.1 Challenges in Authoring Data Visualization for Multiple Contexts

To serve readers in different contexts, authors³ need to create multiple versions of a visualization that are tailored for corresponding contexts. Designing context-specific visualization versions can be challenging because authors have to put more time and effort to explore designs that are suitable for each context and manage consistency across contexts at the same time. In some cases, the set of user contexts to support might not be straightforward, and new contexts can be added as the author identifies novel audiences. Yet, tool support is currently limited.

Some user contexts pose constraints that require authors to explore different, complex design strategies through iterations, compared to some “fixable,” straightforward constraints like applying color-blind friendly color scales, removing interactions for printing, and resizing charts to fit screen spaces. To deal with the amount of visible information at a time on different screen sizes, for example, authors may need to explore design strategies beyond resizing, such as using different encoding channels or visual marks. As shown in Figure 1.2 (Left)⁴, a map visualization for desktops is paired with a smartphone version of a ranked list (Right) because the information-rich map can result in an overly dense visualization on a smartphone. Furthermore, authors must reason about entirely different perceptual channels when creating a visualization with a sonification which describes data using auditory features like time, pitch, and volume.

³I use ‘user’ or ‘reader’ to refer to those who read and use visualizations and ‘author’ to indicate those who create them.

⁴<https://www.nytimes.com/interactive/2016/09/08/us/us-murder-rates.html>

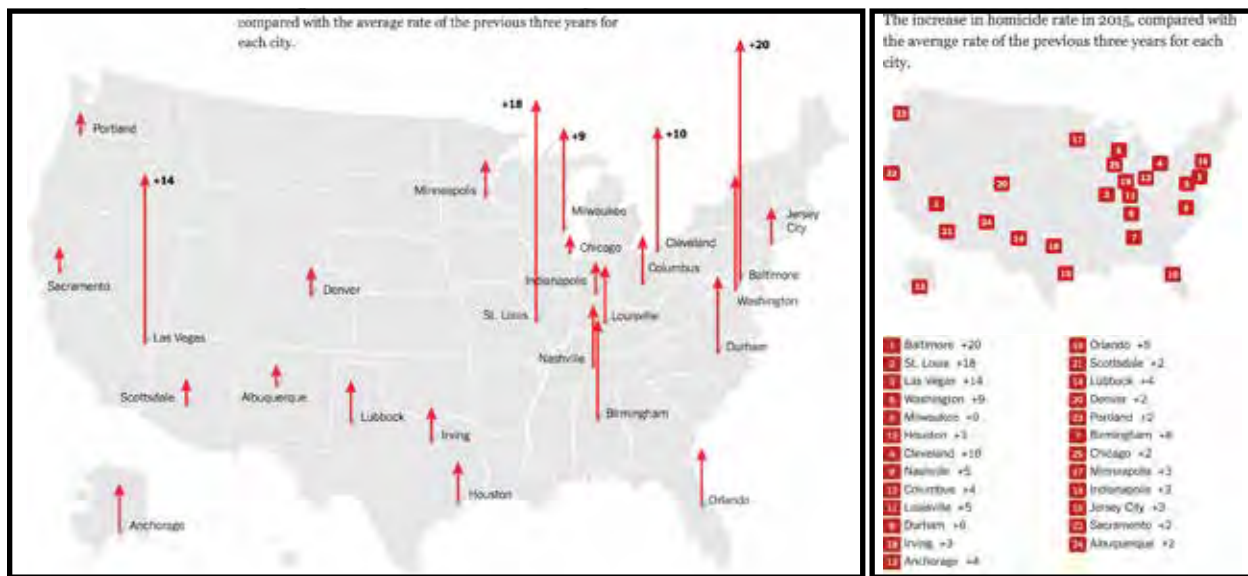


Figure 1.2: An example responsive design where the desktop (left) and smartphone (right) version use different encodings for the same information to adjust visual density (Credit: the New York Times). The desktop version encodes the ‘increase in homicide rate in 2015, compared with the average rate of the previous three years for each city.’ using the heights of arrows positioned on the corresponding states (i.e., an interval type channel). The smartphone version encodes the same variable by listing the states in a descending order (i.e., an ordinal type channel).

Exploring and testing various design ideas is often necessary to arrive at a successful final design. However, design exploration is often difficult in practice due to design fixation and limited design knowledge. On the one hand, design fixation—designers’ unquestioning adherence to prior design strategies—is a widely known phenomenon across various domains [10–13] as well as in visualization authoring [14]. When creating visualizations for multiple contexts, authors might adhere to one of the versions that they initially designed. Yet, a simple strategy like reusing the same format by resizing requires additional work, such as resizing texts, adjusting mark sizes, and rearranging elements, that is more complicated than just copying and pasting a design because not every element in a visualization scales homogeneously. For example, texts and visual marks need to be bigger than certain size limits to be perceived, and they may also need to be repositioned to avoid excessive

overlaps. In reality, authors often need to consider different design ideas for each context as such simple reformatting strategies might not be optimal.

On the other hand, authors may not be able to explore designs due to the lack of relevant knowledge. Bigelow et al. [14] find that the range of design exploration highly depends on individual authors expertise in visualization creation. Furthermore, it is hard to expect a single visualization author to have all the relevant design knowledge given that they are not necessarily visualization design experts.

This difficulty in design exploration for multi-context visualization becomes more problematic when authors' assumptions and knowledge about the data change. If any change is made to a version, authors have to check and adjust all the other versions. As shown in Figure 1.3, for example, updates to data may have effects on the layout of a visualization (e.g., increased marks to show). While the same amount of new data points has a marginal effects on a desktop design ($a1 \rightarrow a2$), it could make the smartphone counterpart more dense or hard to browse due to contents overflowing the screen size ($b1 \rightarrow b2 + b3$), where an author may need to choose a different set of encodings to provide a compact view ($b4$).

In managing consistency across multiple versions, authors must reason about is how to maintain consistent takeaways or “messages” across them. For example, resizing charts for different screen sizes by just fitting screen widths may cause different impressions about a trend for the same data (c.f. [15–17]). As shown in Figure 1.4⁵, a landscape aspect ratio for wider screens (Left) implies a weaker impact of the independent variable (parent's socioeconomic status) on the dependent

⁵<https://nyti.ms/3QHivk6> by Motoko Rich, Amanda Cox, and Matthew Bloch from the New York Times

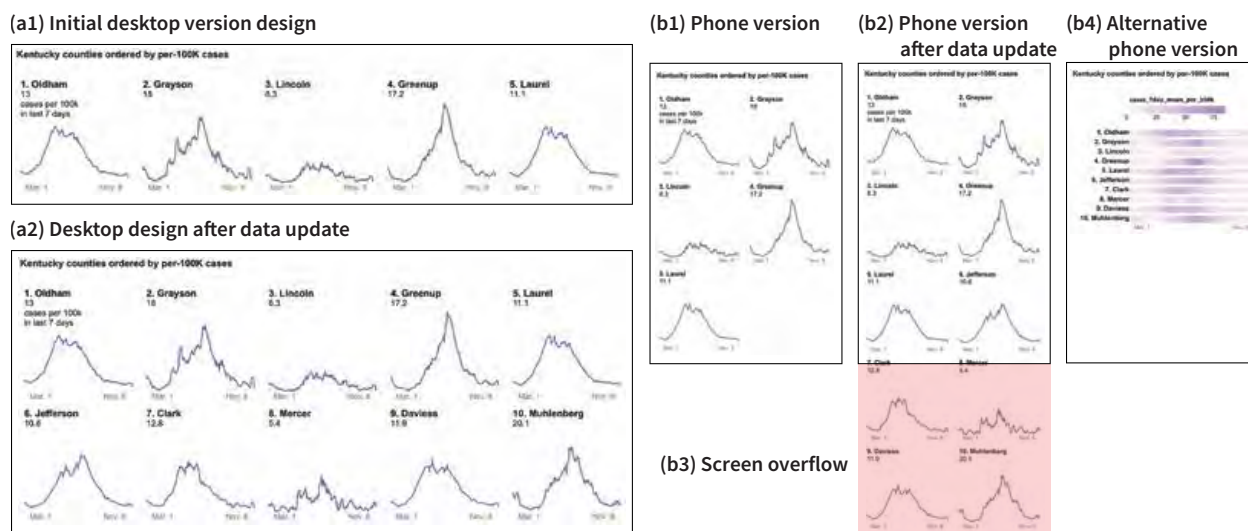


Figure 1.3: Data updates can affect visualization designs by adding by more visual marks. While the effect is relatively marginal on desktop devices (a1 \rightarrow a2), it can be bigger on smartphone screens (b1 \rightarrow b2), causing screen overflows (b3). If the author wants to produce a more compact view on smartphone screens, then they could choose a different set of encodings (b4).

variable (children’s academic progress), while a portrait ratio for narrower screens (Right) make the impact look stronger. The same strategy would not be an issue if the overall relationships appear smooth in both views. The preservation of similar takeaways across versions becomes more complicated under encoding changes like shown in Figure 1.2 because the differences in what readers would interpret are less straightforward. While readers will have slightly different conclusions from a visualization, arriving at completely different impressions may threaten its communication goal, particularly in collaborative settings.

Reasoning about consistency in a visualization’s “message” is non-trivial. Simply preserving every design choice across versions for different devices may result in overly dense visualizations that prevent readers from deciphering meaningful information. When perceptual modalities vary (e.g., visual vs. sound. vs. tactile), such design preservation is not even possible. Although necessary for accessibility

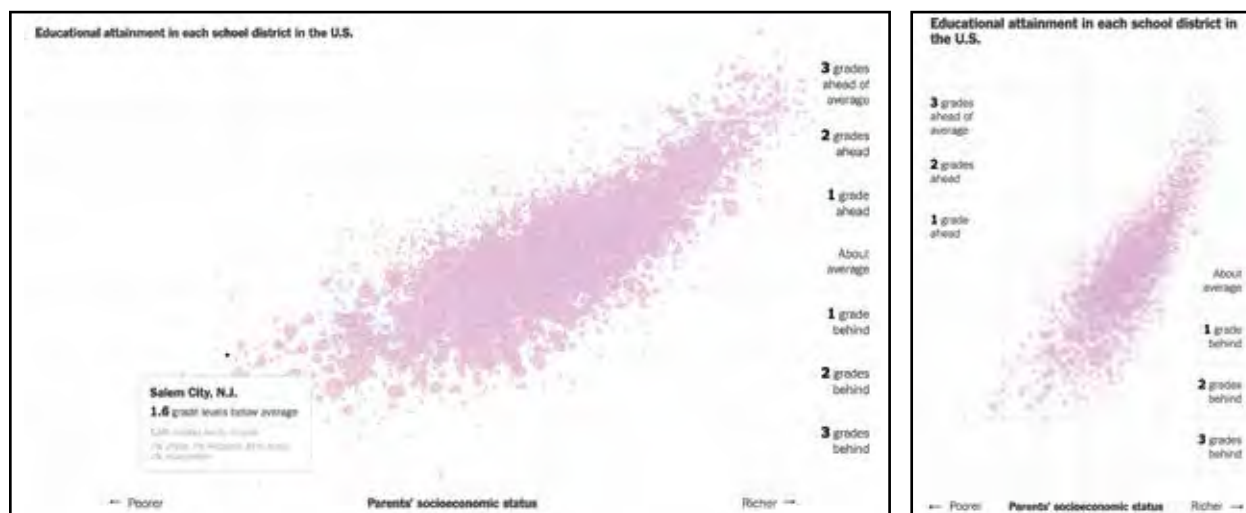


Figure 1.4: An example responsive visualization where the desktop (left) and smartphone (right) versions may imply different relationships between variables (Credit: the New York Times). The desktop version may imply a weaker effect of the independent variable (parent's socioeconomic status) on the dependent variable (children's academic progress), while the mobile version may imply a stronger effect.

support, non-visual channels like audio graphs and tactilization cannot inherently encode as many variables at the same time as visualizations because they use less accurate sensory channels than visualization [18, 19].

Prior work [20–22] applies statistical measures like correlation coefficients or p-values to approximate visualization messages for data analytics primarily in order to recommend visualization designs that are better associated with insights embedded in data. For example, Foresight [20] shows analytic visualizations ranked by the statistics of an underlying variable (e.g., for dispersion, the histogram for a variable with a larger variance is ranked higher). However, those measures are defined on a data space and one-dimensional, so they are invariant when applied to the context-specific versions of the same visualization. They are also high-level and pre-defined, limiting their applicability for cases where a visualization embeds patterns beyond their scope (e.g., linear regression cannot capture a curved trend). Thus, we need to

define a principled but flexible way to define and capture visualization “messages” across context-specific versions.

The aforementioned challenges—design exploration for multiple versions and management of consistency across them—may overwhelm authors. While commercial tools like Flourish [23] and Datawrapper [24] support multiple device types using templates, their template bundles are often too strict to allow authors to explore various possibilities for each device type. For higher flexibility in their design, they have to switch between different tools that are not originally intended for multiple contexts. For example, an author may first prototype visualizations using D3.js [25] and ggplot2 [26], import the graphics to Adobe Illustrator, and export it to web-friendly format using ai2html [27]. In doing so, authors need to come up with how to define, manipulate, and manage user contexts, as these tools do not provide explicit methods for dealing with contexts. In addition, tools for non-visual channels are far more limited currently, so authors have to manually convert data variables to auditory values (e.g., pitch in Hz or notes, height in a 3D space), assign them to computational models (e.g., digital instruments, 3D modeling) via creation toolkits, and post-process output artifacts. As data sonification is an emerging medium in public science communication (e.g., NASA⁶) and data accessibility [28, 29], relevant software support is increasingly necessary.

Facilitating data visualizations for multiple user contexts, therefore, warrants useful and effective tools that help authors easily explore design ideas and ensure consistent messages across contexts. An important first step is to characterize and abstract design spaces and design tradeoffs between information density and con-

⁶<https://science.nasa.gov/mission/hubble/multimedia/sonifications/>, particle physics⁷

sistent messages so that resulting tools can robustly support authors. Without concrete abstraction, resulting tools would be patchworks of ad-hoc solutions that are unlikely to scale well with novel techniques. In my dissertation, I demonstrate the following hypotheses:

THESIS STATEMENT

Characterizing and abstracting visualization user contexts and relevant trade-offs leads to useful interactive systems that help authors to explore diverse design ideas per context and reason about consistency in messages across context.

1.2 Thesis Contributions

To support the hypothesis and address the above challenges, my dissertation contributes abstractions and interactive systems for *responsive visualization* and *data sonification* as application areas. I focus on these context types because their design spaces are non-trivial, and they currently lack foundational technological support.

Referring to adapting a visualization design for different devices and screen types like desktops and smartphones, responsive visualization is essential as people’s device usage has become diversified. As shown in previous examples, responsive visualization authors may need to explore a larger set of designs than they initially have thought of. At the same time, responsive visualization authors need to manage consistent “messages” (e.g., reader’s ability to identify relationships or make comparisons) across different versions in order to ensure the readers obtain congruent takeaways at large. Thus, a systematic taxonomy for possible design strategies and methods for reasoning about message consistency are necessary to

help designers to explore various ideas. They can also be useful for intelligent authoring tools to comprehensively search across possible designs with principles.

Next, data sonification (or audio graphs) is one of the primary methods to enable immersive data storytelling, data-driven art, scientific observation, and data accessibility [30]. Data sonification maps data to auditory features [31–33] like time, pitch, and loudness, for instance. I focus on data sonification because of its wide applicability in the real world. Although auditory channels are harder to perceive than natural language speech or tactile representations [28], it is more flexible and affordable. Natural language descriptions are often limited to capture various distributional information in detail as they tend to rely on predefined formats [34]. Tactilization requires a tactile printer or 3D printer which is expensive for the general public to use every day.

A major challenge in practicing and researching data sonification is that we are not fully aware of how to design effective audio graphs. With compartmentalized support offered by existing tools, it is currently limited or extremely difficult to explore and test all the possible sonification designs. For example, Highchart Sonification [35] provides several encoding channels without screen-readable scale descriptions. Sonifyer.js [36] offers an application programming interface (API), but it only supports time and pitch channels. If authors want to explore designs that are not supported by those tools, they must switch between different tools from low-level audio APIs (e.g., Web Audio, PyMusic) to digital music software (e.g., Garage Band, Adobe Audition, Ableton Live) just to create one five-second sonification.

My dissertation offers the following contributions specifically:

Design tradeoffs in responsive visualization

To build an effective authoring tool, we need to characterize key design tradeoffs that authors have to navigate. A first step to deriving design tradeoffs is to identify a broad set of design strategies used by experts and understand motivations in using those strategies. In Chapter 3, I present an analysis of 378 expert-created responsive visualization cases. This analysis identifies 76 design strategies in terms of *target* (what is changed) and *action* (how to change). A key observation from this analysis was that visualizations sharing the same high-level desktop designs exhibit varying mobile views. This implies that a single “best” solution might not exist, but it depended on problems that the authors prioritized. For example, the same line chart can be resized to include all the information or cropped to prioritize a certain subset of the information while maintaining the graphical density. Furthermore, authors often use additional design steps to compensate the loss of information caused by a certain design technique. For instance, when cropping a chart, they add an interactive tab or link to a view that shows the entire information. Inspired by these observations, I reanalyzed the data to reconstruct motivations behind cases where desktop designs sharing similar representations result in different phone version designs. Through this analysis, I characterize a key design tradeoff in responsive visualization between preserving the same “message” and maintaining the appropriate graphical density.

Task-oriented insight loss measures for responsive visualization

Design tradeoffs like those above make it harder to explore designs as authors need to compare “messages” across visualization versions manually, relying a few rules of thumbs. In responsive visualization, authors often need to iteratively try out various visual encodings and mark types, which makes it much harder to reason

about maintaining the same “messages” or takeaways both cognitively and physically. Thus, intelligent tools like automated authoring systems need to consider how to algorithmically assess changes to visualization insights across different responsive versions. In Chapter 4, I describe a set of *task-oriented insight loss* measures that approximate visualization insights as what readers obtain by performing low-level tasks and estimate changes to support for task-oriented insights between two responsive versions. Then, I demonstrate their usefulness and feasibility an automated responsive design recommender pipeline by applying these measures into a prototype recommender. The recommender with these measures achieved more than 80% accuracy in reproducing rankings labeled by experts.

CICERO: a declarative grammar for responsive visualization

Interactive authoring systems and automated design recommenders must have systematic expressions for their targeted design spaces. Such expressions should be flexible and reusable so that the same high-level technique can generalize to various design cases rather than requiring ad-hoc adjustments. To further motivate, suppose that we want to develop a intelligent visualization authoring tool that allows both automated and manual edits to expedite design exploration. Then, the tool’s design expression method must be consistent for both input sources. In Chapter 5, I provide CICERO, a declarative grammar that expresses responsive visualization techniques in an expressive, flexible, and reusable way. By replicating real-world use cases and applying CICERO to a prototype recommender, I demonstrate its applicability in responsive visualization authoring practices.

DUPO: a mixed-initiative authoring tool for responsive visualization

In general, it is often difficult and tedious to explore a large design space when designing an artifact because human authors have limited time. For responsive visualization, even when applying simplest design strategies like resizing, authors have to adjust the design (e.g., text annotations, visual emphasis, overplotting, etc.) as visualization elements do not scale uniformly. While such post-hoc adjustments have multiple alternatives and hence can take more than a few minutes, they also need to assess whether such designs are okay to publish and consistent with other responsive versions. Such costs of prototyping and evaluation can overwhelm authors and lead them to giving up design exploration [14]. An automated design recommender can meaningfully reduce those costs only if it also allows authors to express their design intentions. In Chapter 6, I contribute DUPO, a mixed-initiative authoring tool for responsive visualization where authors can apply automated designs and make custom edits iteratively. A user study with expert graphics reporters demonstrates that DUPO can help them focus on design exploration per se with less burden of tedious manual input.

ERIE: a declarative grammar for data sonification

Data sonification is an affordable and widely applicable solution for making visualization accessible to Blind people. Sonifications are also useful producing immersive data experiences like in museums or online news articles by adding perceptual cues coordinated with visual materials. Research- and system-wise support for data sonification is, however, highly limited compared to data visualization. A main reason for this limitation is the lack of expressive and programmatic toolkits for authoring data sonification. In Chapter 7, I present ERIE, a declarative grammar for data sonification. Using ERIE, I replicate and extend prior sonification cases like accessi-

bility applications and online news articles, showing its feasibility with sonification practices. To further demonstrate its expressiveness and potential for accessible data analysis, I further showcase novel sonification designs for residual plots, Q-Q plots, and density plots.

Toward a framework for multi-context data visualization

As a concluding discussion, I offer a blueprint for a generalized framework for data visualization for multiple use contexts in Chapter 8. This proposal will help visualization authors and system developers to more explicitly reason about user contexts they want to support and design tradeoffs they need to navigate. Furthermore, this proposal will help ground future research with an abstraction for novel visualization user contexts and relevant sophisticated systems.

1.3 Prior Publications and Authorship

While I am the primary author of the work included in this dissertation, I acknowledge that each project was done through the collaboration with my advisor, Jessica Hullman, mentors and colleagues. For the design strategies and tradeoffs project (published at EuroVis 2021 [37]), the initial data analysis and strategy classification (Section 3.1) started as my master’s thesis project with the supervision of Joonhwan Lee from Seoul National University. Jessica Hullman supervised and helped me to reanalyze the data and characterize design tradeoffs. While I do not claim the design patterns as a contribution of this dissertation, I included them for readability. Next, I implemented the task-oriented message loss measures (published at IEEE VIS 2021 [38]) with Jessica Hullman, Dominik Moritz from Carnegie Mellon University, and Ryan Rossi from Adobe Research. Abhraneel Sarma from MU Collective

at Northwestern University assisted in data analysis. The CICERO grammar (published at ACM CHI 2022 [39]) was my internship project at Adobe Research with Jane Hoffswell (mentor). Ryan Rossi, Eunyee Koh, Fan Du, and Shunan Guo from Adobe Research contributed during the initial stage of this work. Jessica Hullman provided feedback during the later stage. I designed and developed the DUPO system (published at IEEE VIS 2023 [40]) with the mentorship of Jane Hoffswell, Jessica Hullman, and Ryan Rossi. Lastly, I developed the ERIE grammar and tools (published at ACM CHI 2024 [41]) through discussion with Jessica Hullman and Yea-Seul Kim from the University of Wisconsin—Madison. In this thesis, I use ‘we’ to reflect the help and support of my collaborators.

Chapter 2

Related Work and Background

This dissertation is based on prior work on responsive and mobile visualization, data sonification and accessible visualization, declarative programming paradigm, and visualization design recommendations. Prior work related to a specific project is discussed in the corresponding chapter.

2.1 Visualization User Contexts

Earlier, I defined a *visualization user context* (or context) as the viewing conditions of an audience group that are relevant to their use of visualization. Prior work has investigated how different user context factors affect visualization design, including device types, levels of expertise in domain and data analysis, physical abilities, and aesthetic styles. First, as the number of mobile readers increases [8] and extremely large and small screens (e.g., wall-sized screens and smart watches) became widely available, studies have looked at strategies for mobile visualization [42–44] and scalable visualization [45–53]. Next, data analysts often use complex visualizations that maximize the amount of contained information. When they are to deliver the results of their analysis to colleagues or the public who have limited domain knowl-

edge or data literacy, the analysts need to come up with easier representations, which is a common bottleneck in collaborative data work [54–58]. In addition, as visualization exploits human visual perception, it is inherently an inaccessible communication method for Blind people. Research has looked at alternative texts [59–63], data sonification [29, 64–72], braille [73], and tactilization [74] to enhance the accessibility of visualization. Furthermore, as a public communication channel, visualizations often need to be restyled to enhance contrasts for color-blind people or achieve certain stylistic goals (e.g., comic-styled, report-styled), which is called style transfer [75, 76]. There are other factors of users that affect their use of visualization, such as cultural backgrounds [77, 78], individual preferences [79, 80], and cognitive ability [81]. To provide overall background in designing visualizations for multiple user contexts, I discuss related considerations and prior approaches.

2.1.1 Characteristics of Designing Visualizations for Multiple User Contexts

Prior visualization research aims at finding a universal set of characteristics for “good” visualization to inform design practices. For example, empirical studies (e.g., Heer and Agrawala [17], Kim et al. [82]) find effects of visualization design on perception of visual information. Prior approaches (e.g., Voder [83], GraphScape [84]) offers design evaluation methods for ranking visualization designs for automated recommenders.

Nevertheless, prior work only looked at one context type at a time or did not explicitly define user contexts for their studies. By having visualization user contexts upfront, I shed light on their coexistence, fluidity, and specificity. First, visu-

alization user contexts coexist, so an author must prepare context-specific designs at the same time if they want to communicate with broad audiences. For example, when creating a visualization for journalism, an author often needs to expect that Blind and sighted people access it on desktops and smartphone devices. In addition, authors may need to identify a specific set of visualization user contexts to support given communication goals. For instance, an author may work on desktop and smartphone designs for internal reports from an analytic visualization. Later, they might need to create presentations and printed reports for a different dataset, considering style guidelines and the expertise levels of those audiences. Furthermore, each context requires different effectiveness constraints. For example, a mobile visualization is less usable when too many interactive features are densely arranged; or a sonification is considered too long if it is longer than a few seconds [29]. Therefore, a visualization author must support different user contexts by creating multiple versions together rather than creating them by assuming a single context. The next sections (Section 2.2 and Section 2.3) will detail specific design considerations regarding devices and accessibility.

As variants of the same visualization, multiple versions specific to different user contexts should share the same primary “messages,” takeaways, or insights. For example, if a visualization offers fundamentally different impressions about a trend on desktops and smartphones as shown in Figure 1.4, it can cause a significant communication problem among stakeholders. The same user might find the data has changed abruptly between moments only because they used different devices (e.g., a desktop at work and a smartphone while commuting). Moreover, if an accessible version of a visualization has to contain less information, then it provokes equity concerns. In response, prior work has attempted to formulate visualization

insights, characterize salient visual information, and compare perceptual structure of a visualization to enable scalable visualization, for example. Nevertheless, it is non-trivial to measure visualization insights in a principled and explainable way for the following reasons.

First, prior work on visualization recommendation employs statistical calculations to characterize properties of a visualization thought to relate to the insights a user can draw from it. Often these ‘insights’ are intended to capture how well a user can perform analytic tasks, such as recognizing trends or identifying and comparing data points. Tang et al. [22] suggest detecting ‘top-k insights’ from data using statistical significance testing (e.g., a low p -value of a linear regression coefficient for slope insight). Similarly, Foresight [20], DataSite [21], and Voder [83] use statistics calculated on the data, such as correlation coefficient and interquartile range, as proxies to corresponding analytic tasks. Then, they recommend visualization types predicted to better support extracting such information. However, statistics on data are invariant for views sharing the same dataset and hence of limited use for comparing different ways of visualizing the same underlying data.

Next, earlier work on visualization resizing introduces algorithms for maintaining key information at different size scales. For example, Di Giacomo et al. [50] propose a greedy algorithm that repeatedly removes the pixels determined to be least important to rescale a graph visualization. Visizer by Wu et al. [85] takes a similar approach that iteratively minimizes scaling in more salient regions for visualizations with force-based layout (e.g., network graph and word cloud). Yet, the scope of these approaches are often too limited to certain mark types and encoding channels. Thus, a flexible approach to ensuring message consistency across con-

texts is necessary to consider diverse visualization designs.

Lastly, signal processing-based approaches analyze the underlying visual or perceptual structure of a visualization to enable multi-scale visualizations (i.e., providing different insights at different scales) and to enhance visualization effectiveness. Prior work has attempted to enable multi-scale views through perceptual organization analysis of an information graphic at each scale [86, 87] and hybrid-image visualization that displays views with different aggregation levels at different viewing distances [45], for example. With these approaches, zoomed-out views prioritize overall structure of elements while zoomed-in views keep details of the elements identifiable. Signal processing approaches have also been applied to improve the effectiveness of a visualization, for instance, by measuring the difference between the visual salience of a representation and salience of signals in data [88, 89], comparing kernel density estimations between a LOESS curve and different representations [90], and extending a structural similarity index for image compression to data visualization [91]. Signal processing-based approaches have typically been applied to single views, and are generally confined to a predefined set of marks and visual variables (e.g., a line chart, a scatterplot), restricting their applicability for settings with multiple user contexts.

2.1.2 Design Exploration for Multiple User Contexts

To supporting visualizations in multiple user contexts, authors need to have foundational understandings of effective design strategies across contexts and be skilled with tools that enable reifying them. From the perspective of producing them, visualization authors are often encouraged to explore as many strategies as possible. Even if it is possible, however, exploring all the possible design choices is likely

to be overwhelming in addition to learning relevant skills and techniques. When a design space is large and effortful to navigate, designers typically exhibit design fixation across different domains [10, 12] including visualization authoring [14].

Prior research has framed support for multiple visualization user contexts as a *retargeting* problem. Visualization retargeting refers to reproducing or repurposing an existing visualization for a different context. For example, scalable visualization work has provided algorithms to adjust levels of details [48, 49, 92] or resize the chart with different aspect ratios [50, 85, 93]. As noted earlier, a universal set of visualization contexts may not exist, but it can change for different communication purposes. Authors may find contexts that were unknown at first, or future technology could introduce nascent contexts. My approach to authoring visualizations for multiple user contexts, or *multi-context visualizations*, motivates considering user contexts flexibly throughout the process of creating a visualization.

Prior empirical work and systems research on visualization have overlooked the multiplicity of user contexts. Many empirical studies typically assume desktop settings where people use keyboard and mouse for interaction. Furthermore, many visualization authoring tools like Data Illustrator [94, 95] and programming toolkits like Vega-Lite [96], D3 [25], and ggplot2 [26] presume single-context charts. Commercial authoring tools like Datawrapper [24] and Zing Chart [97] offer some authoring modes for different device types. However, these tools have limited flexibility in terms of choosing different encodings per version (e.g., having to change data structure for different mark types) or do not provide robust low-level expressions, making it hard to build sophisticated or intelligent systems, such as recommenders, out of them.

2.2 Responsive Visualization

Responsive visualization means adapting a visualization's design to other device types [98, 99] (e.g., a laptop with a keyboard and track pad; a smartphone with a touch interface), drawing an analogy to responsive web design [100]. Responsive visualization has become an important visualization problem since the number of mobile users of visualization has increased over time [8]. Below, I focus on desktop and smartphone devices as they are primary device types for typical responsive visualization design.

2.2.1 Considerations for Responsive Visualization

The need for responsive visualization stems from the physical and environmental differences between various device types [101]. On the one hand, desktop and smartphone devices are physically different in terms of screen size, aspect ratio, input modality, and computational power. The smaller screen size and portrait aspect ratio of smartphone devices require different visualization specifications, primarily because visual marks and letters need a certain minimum pixel-space difference (e.g., size, position, hue, etc.) to be recognized. While desktop devices receive inputs through keyboard and pointing devices, smartphone devices usually use touch interfaces. Because touch interactions are less accurate on mobile devices due to the fat-finger problem [102, 103], interactions often must be altered. The reduced computational power of smartphone devices creates problems rendering dynamic and complex visual representations and interactions.

On the other hand, designers should take environmental characteristics, such

as the conditions, purpose, and length of use into account in responsively transforming a visualization. People often use smartphone devices under conditions that make it hard to focus (e.g., walking [104] or using them with other devices [105]). People are likely to use smartphone devices for simpler purposes (e.g., instant messaging or pickups) for shorter amounts of time [106]. These environmental differences between desktop and smartphone devices imply that authors often need to tune their smartphone visualization according to a more focused subset of their intentions (e.g., simplifying or emphasizing elements in terms of importance) to prevent them from overwhelming users.

2.2.2 Scalable and Mobile Visualization

Responsive visualization involves several visualization scalability issues including display scalability and level of detail. Display scalability is known as a key challenge in designing visualization, referring to how well a visualization design scales for multiple device types with varying screen sizes and interaction methods [107]. Dealing with display scalability often requires more than simply rescaling to different screen sizes (e.g., everyday devices like desktops and smartphones for responsive visualization). For example, scalability challenges arise from how well different viewing factors (e.g., viewing distance [45], chart size [46]) support presenting different levels of details. To enhance scalability, prior work contributes algorithms for managing levels of detail through progressive refinement [48, 49] or by limiting “the number of visual entities” [92]). Visualization retargeting studies (e.g., Wu et al. [85], Di Giacomo et al. [50]) provide algorithms for resizing charts while keeping visually salient information. Scalability concerns arise across various device types, such as scaling up desktop visualizations to wall-sized displays [47, 51] and

non-rectangular devices (e.g., circular tabletops [52], smart watches [53]).

2.2.3 Techniques for Responsive Visualization

Prior work identifies *programmatic difficulties*, *artboard management*, *design exploration*, and *changes to takeaways* as some of the major challenges in adapting visualizations for different screens. While creating responsive visualizations demands both cross-device design and development expertise, earlier work mainly focused on programmatic techniques using general purpose tools like D3.js [25]. This programmatic support was important particularly when there was limited tooling for responsive visualization. For example, Hinderman [100], Körner [108], and Andrews [98] demonstrate the technical feasibility of creating responsive visualizations using D3.js. The JavaScript library R3S.js [109] provides direct programming interfaces for responsive visualization using D3.js.

When using GUI-based authoring tools (e.g., Adobe Illustrator with ai2html [27], Datawrapper [24], or Flourish [23]), authors need to manage multiple *artboards* (drawing or design areas) for different device types. Authors often have a set of predefined screen breakpoints for responsive versions (commonly including desktops, tablets, and smartphones for the Web environment) to avoid unpredictable errors arising due to dynamic resizing on different devices [98, 99]. An easy approach to handling these multiple artboards is to simply finalize an initial design for a certain screen type, and only then create other responsive versions that exhibit a small set of transformations, such as resizing [99]. However, this sequential approach makes it burdensome to run design iterations on non-initial versions of the visualization, even though they are not necessarily less important to the overall success of the design. Instead, Hoffswell et al. [99] suggest transferring edits from one art-

board to the other artboards and flexibly toggling such edit transfers in order to support simultaneous design iterations on multiple artboards. Commercial tools like ZingCharts [97] and Datawrapper [24] allow authors to specify conditional settings (e.g., altering axis positions for smartphone screens) for a bundle of templates for different screen types.

Ideally, design exploration can help an author to identify better ideas [12, 110, 111], yet manually drafting and managing more than a few design alternatives is often time-consuming. Thus, computational (semi-) automation is often recommended [110], and prior work has proposed automation approaches to responsive visualization [93, 112]. Business intelligence tools like Power BI [113] and Tableau [114] offer presets to make it easier to create day-to-day responsive visualizations for data analytics. MobileVisFixer [93] uses machine learning (ML) techniques to retarget non-responsively created desktop visualizations for mobile screens. LQ^2 [112] introduces an ML-based (pairwise) ranking model for visualization layout given a chart size (e.g., a larger number of bars or thicker bars for a wider chart size).

2.2.4 Gaps in Tooling for Responsive Visualization

While prior work individually addresses various challenges, such as technical feasibility, retargeting, and simultaneous authoring, in authoring responsive visualizations, there are still significant gaps in relevant systems research. First, we lack a systematic understanding and expression that can cover a variety of responsive visualization strategies rather than applying ad-hoc solutions for particular encoding channels and mark types. Next, prior approaches are limited in characterizing and hence considering key design tradeoffs between consistent “messages” and device-specific characteristics. To build intelligent tools like design recommenders, it is

important to operationalize such tradeoffs. In addition, systematically translating various responsive design techniques into an authoring system is necessary to actually support visualization authors. Otherwise, such tools would need to rely on a collection of ad-hoc solutions, making it difficult to extend them with novel techniques. Lastly, how automated design recommendations are communicated to authors matters with respect to the usefulness and feasibility of intelligent authoring systems. For example, if an automated recommender produces only one final outcome or too many outcomes with redundancy, it will limit authors' ability to explore designs. My work contributes a characterization of design tradeoffs, a set of loss measures for changes to insights, a declarative grammar, and a mixed-initiative authoring system for responsive visualization.

2.3 Accessible Visualization and Data Sonification

Accessibility support for data visualization mainly targets Blind people through auditory and tactile channels [62]. Auditory methods include *alternative text* that describes a visualization with natural language speech (spoken by a screen reader) and *data sonification* that uses non-speech sound like a sinusoidal oscillator or piano. For tactile perception, tactile and braille printers are commonly used, supported by packages like TactileR [115] and BrailleR [73]. Chundury et al. [74] mix sonification and tactilization by playing corresponding sound and screen vibration when a user touches a visualization's mark on a touch screen device like iPad.

Extensively reviewing literature in accessible visualization, Kim et al. [62] propose a set of principles for creating accessible visualization. Those principles include: notifying whether a chart exists, providing its overview, allowing for on-

demand detail, offering necessary or useful relevant information. In practice, data-heavy organizations like European Union Data [116] and WHO Data Design Language [117] provide guidelines like using semantic HTML tags by following Web Content Accessibility Guidelines [118] and using multi-sensory representations like those described above.

However, practicing accessible visualization is often undermined or challenging. Via an online survey with 144 visualization practitioners, Joyner et al. [28] find that only a few of them (24%) were required and guided to make accessible visualizations. Furthermore, 42% of the respondents had to work on accessibility by themselves while 8% of them had professional personnel for accessibility techniques. Joyner et al. [28] remarked that expertise in visualization authoring played an important role in acknowledging relevant practices and actually applying those skills in their creation; yet, they tended to rely more on manual efforts in creation except the use of accessibility check software. By analyzing 76 Covid-19 visualizations that appeared on Google search's top results, Fan et al. [119] show that 70 to 90% of them failed to pass different accessibility criteria proposed by Elavsky et al. [120]. Therefore, tooling for accessible visualization is important to make relevant practices easier and more doable.

Furthermore, through online survey and contextual inquiry, Fan et al. [119] identify the gap between the usage and familiarity of a format as important challenges in creating accessible visualization. For instance, while 50 out of 171 Blind respondents said they would be familiar with tactile graphs but five of them used them primarily. The ratio was 7 (familiar) to 20 (primary) for screen-reader, and 15 to 17 for sonification. A rationale for this gap could be expensive prices of tactile

devices while auditory channels typically do not need extra devices but earphones. My thesis focuses on tooling for data sonification given the wide adoptability of auditory channels.

2.3.1 Data Sonification

Data sonifications or audio graphs encode data values as auditory values [31–33]. For example, a Geiger counter maps ionizing radiation to the loudness of a sound. Sonification is considered as one of the primary methods for data accessibility or accessible data visualization for people with Blindness [28]. For instance, web-based data visualization can be coupled with sonification along with alternative text descriptions. Yet, accessibility is not the only venue for sonification. Various other fields, such as scientific data representation [121–123], data-driven art [124], and public engagement with science (e.g., learning [125], museums [67, 126]), use data sonification.

2.3.1.1 Empirical studies in data sonification for accessibility

Prior work on sonification for accessibility has focused on how Blind people interpret different auditory mappings. Walker et al. [65, 68, 69] extensively compared how sighted and Blind people perceive various auditory channels and the polarity of mappings for different quantitative data variables (e.g., dollars, temperature). Recent work extends focus to other qualities of auditory mappings. For instance, Hoque et al. [70] used natural sound (e.g., bird sound) to support enhanced distinction between categorical values. Wang et al. [29] show that Blind readers find certain audio channels to be more intuitive given visual encodings (e.g., pitch for bar heights) and given data type (e.g., quantitative, ordinal). In their experiment, partic-

ipants indicated a need for an overview of auditory scales [29]. Furthermore, Blind users may have different preferences for auditory encodings, so an ability to personalize auditory according to their preferences is a key aspect in designing sonifications [127]. These empirical studies indicate the needs for supporting diverse sonification designs.

2.3.2 Sonification Tools and Toolkits

Prior work has proposed sonification tools for accessibility support for data visualizations. For example, iSonic [144], a geospatial data analytic tool, offers various audio feedback for browsing maps, such as using stereo panning to provide a spatial sense of the geospatial data point that a user is browsing. iGraph-Lite [145] provides keyboard interaction for reading line charts, and Chart Reader [72] extends this approach to other position-based charts and supports author-specified “data insights” that highlight certain parts of a given visualization and read out text-based insight descriptions. Siu et al. [64] propose an automated method for splitting a line chart into several sequences and adding a template-based alternative text to each sequence. Agarwal et al. [71] provide a touch-based interaction method for browsing data sonifications on mobile phones. While prior sonification research has focused on use of non-speech sound, accessibility studies underscore combining speech and non-speech sound to design audio charts.

Beyond supporting accessibility, others proposed sonification toolkits created for developers or creators to directly make data sonifications. This prior tooling motivates a design space for sonification toolkits, such as the distinction between instrument and audio channels, needs for programming interfaces, and the utility of audio filters. However, existing tools often provide compartmentalized support for

Table 2.1: Comparison of ERIE to prior sonification toolkits. Abbreviations: VL (VoxLens and Sonifier.JS) [127, 128], HC (Highcharts Sonification) [35], WSS (Web Sonification Sandbox) [129], SC (Sonification Cell) [130], AAG (Apple Audio Graph) [131], DGB (DataGoB-ooop) [132], PR (PlayItByR) [133], XS (xSonify) [134], SS (Sonification Sandbox) [66], SY (Sonifyer) [36], IST (Interactive Sonification Toolkit) [135], SW (Sonification Workstation) [136], SA (SonArt) [137], L (Listen) [138], M (MUSE) [139], PS (Personify) [140], Str (Strauss) [141], Sta (StarSound) [142], SD (SODA) [143], Eq (Equalizer), Pow (Power function), Sqrt (Square-root function), SymLog (Symmetric log function).

Category		Property	Erie	VL	HC	WSS	AAG	DGB	PR	XS	SS	SY	IST	SW	SA	L	M	PS	Str	Sta	SD	
Environment			Web	Web	Web	Web	Swift	R	R	Java	Java	Pure Data	C++	C++	C++	C		Python				
Year			2023	2022	2021	2017	2021	2020	2011	2006	2004	2008	2004	2019	2002	1996	1997	1995	2021	2020	2014	
Data	Transform	Aggregate	○							○												
		Bin	○																			
		Filter	○																			
		Calculate	○																			
		Density	○																			
		Fold	○																			
Sound tone	Speech	○	○	○		○																
	Instrument type	Oscillator	○		○						○	○	○	○		○		○	○			
		FM Synth	○				○					○	○	○								○
		AM Synth	○				○							○						○		○
		Musical	○		○	○					○						○	○			○	○
		Wave	○										○	○								
		Noise	○		○	○					○		○	○						○		
		Natural	Via sampling																			
		Human vowel	Via sampling																○			
	Instrument sampling	○																		○		
	Continuous/discrete sounds	○												○								

creating expressive and data-driven sonifications as summarized in Table 2.1. For example, sonification designs supported by DataGoBoop [132] and PlayItByR [133] are strongly tied to underlying chart type (e.g., histogram, box plot), limiting the author’s freedom in choosing auditory encoding channels. Sonifier.js [127] offers limited audio channels, time and pitch. Sonification Sandbox [66] and its successors [35, 129] support more encoding channels, but developers need to use external sound editors to sequence or overlay multiple sonifications that they created using the interface, requiring a different stack of skills. Furthermore, many existing tools lack application programming interface (API), making it difficult for users to personalize or customize sonification designs with their preferred encoding channels or instruments. To achieve greater expressiveness with APIs, developers could use audio programming libraries, such as Tone.js [146], but they have to manually scale data values to auditory values, which can be a substantial hurdle for those with limited audio skills. These tools also lack support for scale references (e.g., tick, scale description), making it harder to decode audio graphs they generate.

My thesis introduces a declarative grammar for data sonification, ERIE, as a programmatic toolkit and abstraction that developers can use to express a wide range of sonification designs. ERIE supports various common encoding channels (time, duration, pitch, loudness, tapping, panning, and reverb), speech descriptions, audio sampling, and composition methods (repeat, sequence, and overlay), making it a good basis for use in the development of future sonification software.

2.4 Declarative Grammars for Data Visualization

Declarative programming is a programming paradigm where a programmer provides an abstract specification (or spec) describing the intended outcome and a compiler executes to generate the outcome. In this paradigm, declarative grammar defines rules for how to write a program. Declarative grammars have helped visualization authors to avoid complex programming (e.g., [26, 96, 147–150]). For example, a Vega-Lite [96] specification uses JavaScript object notation (JSON) to encode chart size, data source and transformation, visual encodings, multiple views, and user interactions using predefined primitives. Some declarative grammars target specific use-cases by leveraging more general-purpose grammars. For example, Gemini’s animated transition grammar formalizes chart animation entities [149] based on starting and ending visualizations specified using Vega [147].

Declarative grammars add value by providing internal representations and compilers for user applications. For example, many end-user tools like visualization recommender(s) [148, 151, 152] and editor(s) [153] use Vega-Lite [96] to represent the visualization design specification. In responsive visualization settings, Hoffswell et al. [99] provide a design editor using Vega-Lite [96]. Declarative grammars can also be useful when directly manipulating the targeted physical space is challenging like audio environments for sonification [28]. For example, some sonification toolkits (e.g., [135, 154]) adopt visual programming languages to allow for visually and interactively authoring data sonification, and those visual programming languages are backed by some kind of declarative expressions.

Therefore, declarative grammars are an important initial step to supporting

the development of end-user systems that behave consistently. Declarative grammars can be particularly useful for settings like multi-context visualization design where authors should be able to practice various skills required for different user contexts by freeing the authors from low-level technologies and letting them focus more on high-level designs.

2.4.1 Representing Visualization Design Knowledge as Constraints

Constraint logic programming (e.g., Prolog [155], Planning Domain Definition Language [156], Higher-order logic programming [157]) allows for expressing domain knowledge in a form that computer systems can efficiently and systematically operate. Representing design knowledge as constraints have benefited intelligent design tools. For example, layout design recommenders like DesignScape [158] and Scout [159] express design criteria and user preferences as constraints in order to suggest feasible and effective design alternatives.

Prior work on visualization framework and recommenders also proposed expressing visualization design knowledge as constraints. For example, Kindlmann and Scheidegger [89] suggest a formal framework that can help us to algebraically compute the effectiveness of a visualization design, such as charts that do not exhibit any hallucinating data pattern. Moritz et al. [148] propose Draco that expresses effectiveness criteria for visualization design [82, 160, 161] as declarative constraints using Answer Set Programming (ASP) [162, 163]. ASP represents knowledge as facts, rules for inference, and constraints. By assigning a weight term, or cost, to the violation of each constraint, Draco enables exploring a complex space of competing constraints. To represent diverse responsive design strategies, my responsive visualization work (Chapter 4 to 6) adopts constraint-based representations using ASP.

In Chapter 8, I propose a generalized approach to multi-context visualization that represents design tradeoffs as a set of declarative constraints so that the expression method can function as a platform for future discovery of novel user contexts.

2.5 Visualization Design Recommendation

Prior work proposed various recommendation methods for visualization design with respect to applicability and effectiveness. By applicability, I mean whether a visualization design is feasible and faithful to a given data specification. Mackinlay [160] introduces the *expressiveness* criterion that a visualization must express all and only the facts. For example, if a visualization design implies a pattern that the underlying data do not have, it is not an expressive design (e.g., a bar chart showing a descending pattern over a non-ordinal dimension). Kindlmann and Scheidegger [89] propose a set of algebraically defined principles for robust visualization designs that are not *hallucinating*, *confusing*, and *misleading*. Mackinlay [160] also proposes the *effectiveness* criterion that a visualization design should pursue accurate and prompt perception. Empirical studies [82, 161] find position encoding channels (e.g., x and y) are generally more effective than other channels like color and shape.

Central to building visualization design recommenders is to identify tasks that visualizations need to support. For example, visualization recommenders for data exploration aim to suggest visualization designs that can facilitate reasoning about analytic insights. They tend to employ statistical calculations to characterize properties of a visualization thought to relate to the insights a user can draw from it. Often these ‘insights’ are intended to capture how well a user can perform analytic tasks, such as recognizing trends or identifying and comparing data points. Tang

et al. [22] suggest detecting ‘top-k insights’ from data using statistical significance testing (e.g., low p -value of a linear regression coefficient for slope insight). Similarly, Foresight [20], DataSite [21], and Voder [83] use statistics calculated on the data, such as correlation coefficient and interquartile range, and recommend visualization types predicted to better support extracting such information.

While the above recommender approaches for analytic visualization consider the quality of individual design, those for sequenced visualizations (e.g., data storytelling, exploratory data analysis) consider the relationship between two views. GraphScape [84] offers a view similarity model that assigns costs to visualization pairs that are intended to approximate the cognitive cost of transitioning from one view to another in a visualization sequence. GraphScape applies an a priori cost model in which data transformation (e.g., binning, modifying scales) is always less costly than changes in encoding. Hence, filtering data has a lower cost than transposing axes. Dziban [164] extends GraphScape to suggest a view that is ‘anchored’ to the previous view for an exploratory data analysis process.

Both individual design quality and relationship between designs are important criteria for automated design recommendation of multi-context visualization. Multi-context design recommenders must consider search spaces to generate well-formed designs for each context and evaluation methods to ensure the consistency of insights across those versions. For example, while MobileVisFixer [93] employs a set of loss functions that optimizes for common mobile visualization criteria (e.g., minimum screen overflow, maximum font size, minimum overlap between elements, and minimum unused white space), these measures are not applicable for desktop versions. Therefore, my work on task-oriented insight loss measures aims to es-

timate the difference in how two responsive versions support visualization tasks, given a recommendation pipeline that generates well-formed mobile designs given a desktop design.

Mixed-initiative approaches

Visualization creation is often an iterative authoring process [14] where authors repeatedly revise their designs to convey their specific intention. Yet, a challenge in design iteration is the amount of manual effort for possible design options, which lead authors to design fixation. Authoring tools with automated design exploration can encourage authors to create charts for multiple user contexts by reducing the human effort involved in such prototyping.

Prior work on visualization tools for various domains has applied a mixed-initiative approach where users can make manual changes or apply system suggestions. For example, Voyager [151, 152] recommends different analytic visualizations for exploratory data analysis and let users drill down one of them. LADV [165] suggests dashboard designs based on an author's sketch and allows for making custom edits. ViA [166] supports repeatedly updating an author's intents and preferences to obtain better output from an AI generation model. Likewise, the mixed-initiative approach is widely used in machine learning-assisted data analytics [167], business intelligence [114], dashboard design [165, 168], and infographics [169].

Chapter 3

Characterizing Tradeoffs in Responsive Visualization

In practice, a few simple responsive strategies are well known, such as proportional rescaling [170] or responsive layout specification [171]. However, many design challenges or trade-offs that arise in transitioning visualization designs for smaller screens remain difficult for authors to address. For example, simple rescaling may cause overplotting of marks or a highly dense view, making it difficult to select marks on small touch screens. Removing interactions reduces the content of a visualization in ways that might threaten its ability to convey the same message to mobile readers as the original did.

Recent work [99] takes steps toward better support for authoring responsive visualization through a prototype visualization authoring system that enables authors to propagate design edits across different screen size versions of a visualization. However, the design space of responsive visualization strategies itself may be large, making it tedious to manually try out changes one by one. A deeper understanding of the design space of responsive visualization techniques—including a detailed characterization of what elements authors tend to add, remove, or change,

how they do so, and what trade-offs motivate their choices—is a first step toward formalizing responsive visualization design knowledge to further support authors through automated design recommendations.

Toward this goal, we first contribute a comprehensive summary of design strategies that authors currently use when creating smartphone versions of desktop designs. By comparing the desktop and smartphone views of 378 public facing visualizations, we identify 76 design patterns for responsive visualizations. Our analysis captures responsive visualization strategies (from desktop to smartphone) in terms of *Targets*, representing what is changed (data, encoding, interaction, narrative, references/layout) and *Actions*, representing how the targets are changed (e.g., increase bin size, aggregate, reduce width, externalize annotations). Readers can explore our design strategies with illustration and descriptions on our online gallery¹.

Then, we characterize key trade-offs in responsive visualization authoring. We propose that the overarching design challenge in responsive visualization is a density-message trade-off where authors seek to balance goals of maintaining graphical density with those of preserving the message or intended takeaways of their work. We observe that strategies addressing graphical density, layout, and interaction complexity often result in “message loss”, where “message” captures a viewer’s ability to recognize certain comparisons or relationships. We identify different forms of message loss, including loss of information, interaction, discoverability, concurrency of elements, and graphical perception. We conclude by discussing the implications of our characterization of design patterns and key trade-offs for responsive visualization tool design.

¹<https://mucollective.github.io/responsive-vis-gallery/>

3.1 Responsive Visualization Design Patterns

Based on our qualitative analysis of desktop and smartphone versions of 378 visualizations intended for communication, we characterize design strategies for responsive visualization and categorize them in terms of *Targets* and *Actions*.

3.1.1 Methods

3.1.1.1 Sample collection

We collected a sample of 378 *pairs of* desktop and corresponding smartphone visualizations (756 total visualizations) from the media (e.g., news outlets), data-driven reports from global organizations, and blog posts about responsive visualization. We first collected pairs of visualizations from 104 data-driven news articles containing visualizations from the *New York Times* (NYT) and the *Wall Street Journal* (WSJ)'s yearly galleries of data visualization articles (2016-2017) [172–175]. We included all articles with abstract data visualizations that map numerical and/or categorical data to visual variables and excluded illustration and photography-based articles. From this set, we obtained 280 pairs of visualizations. To this set, we added visualizations from 57 additional articles and visualization projects from international organizations (OECD, UNESCO), visualization authors' blog posts (e.g., Bremer [170]), *Mobile-Vis* gallery [171], and *Scientific American*, which provided both desktop and smartphone views (98 more pairs of visualizations). Figure 3.1 illustrates the properties of our sample. We provide the full list in Appendix A and in our interactive gallery. The size and diversity of sources in our sample suggest that it should offer a reasonable, albeit not comprehensive, snapshot of the design space for communication-oriented responsive visualization design.

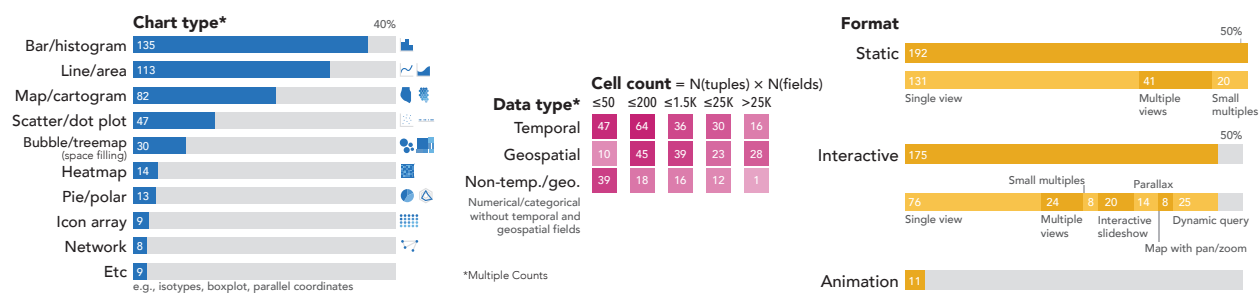


Figure 3.1: Properties of our visualization sample. We reconstructed cell count by multiplying the number of tuples (records) and the number of fields (columns) and grouped cell sizes by k-means clustering ($k = 5$).

3.1.1.2 Analysis

To characterize design strategies, two authors and an external coder iteratively coded differences between the desktop and smartphone visualizations in each pair using methods from grounded theory [176]. We started with open-coding [177] to build up a large set of descriptions of differences between desktop and smartphone versions of visualizations (e.g., add highlighting, remove an interaction feature). We then made several additional coding passes, grouping observations made from different pairs into single, recurrent strategies and returning often to examine the sample visualizations to confirm that a strategy was in fact the same. This process resulted in 76 design patterns or strategies. We observed that each of these strategies could be further distinguished by the *Target* of the change, representing what type of visual or design element was changed (e.g., annotations, data, encodings) and the *Action* describing the form of the change (e.g., removing, highlighting, increasing). Finally, we developed higher level groupings of Targets and Actions shared across strategies, respectively. This analysis distinguished five categories of Targets (Data, Encoding, Interaction, Narrative, and References/Labels) and five categories of Actions (e.g., Recompose, Rescale, Transpose, Reposition, and Compensate). We tabulated counts of different strategies observed across our sample and their co-

occurrence in Section 3.1.3.3.

3.1.1.3 Preliminary Survey of Authors

To supplement our analysis of examples, we surveyed 19 visualization authors with experience in designing responsive visualization (average 4.8 years), who we solicited through a posting on social media. We asked about their typical process of designing a responsive visualization (i.e., starting from a desktop view, from a smartphone view, or designing both simultaneously) and how many times they consider smartphone views in their design process (less than 10% of the time, less than half the time, about half the time, more than half the time, more than 90% of the time). Next, we asked them to rank seven design guidelines for responsive visualization. The guidelines were informed by prior work on responsive visualization and our initial analysis, like ‘maintaining the main takeaways’ and ‘maintaining the information density.’ We included open-ended questions to elicit any “rules of thumb” they used and difficulties they faced when authoring visualizations for mobile screens.

Eleven authors (58%) described creating smartphone visualizations after designing desktop views, similar to the findings of a recent interview study [99] with five authors. As a result, we default to describing design strategies as transformations of a desktop view in presenting our analysis. Yet, the different Action strategies (Section 3.1.3.1) we identify are invertible, so this direction is primarily a communication mechanism rather than a property of our analysis. When asked to rank different possible guidelines for responsive visualization, authors ranked “maintaining takeaways,” “maintaining information,” and “changing the design to acknowledge greater interaction difficulty on a smartphone” as most important. More than half (10, 53%) of the authors described strategies they used and/or concerns they

had in adjusting information density for smaller screens (e.g., “*Step by step information reveal rather than showing everything at once*” (P15), “*Creating a similar experience without overwhelming the user*” (P18)). Such statements informed our identification of important trade-offs in responsive design. Full survey questions and responses are provided in Appendix B.

3.1.2 Examples

We provide three examples of responsive visualization to introduce the reader to key design patterns.

3.1.2.1 *Bond Yield* - Data, Annotation, and Size

*Bond Yield*² illustrates strategies of information removal from a desktop to a smartphone view, and consequent changes in emphasis. The area mark on the left of the desktop view (Dt) in Figure 3.2 expresses observed world GDP growth from 2010 to 2015. The smartphone view (Sp) omits the grey area mark as well as two line marks representing five-year forecasts of GDP growth rate. In the desktop view, the omitted part had served to show that GDP growth rate projections were higher before the actual growth rate plummeted. The authors retained lines in the smartphone view that show a more recent decrease in the International Monetary Fund’s GDP growth rate forecasts. Data records for the years 2010 and 2011 have been removed (**remove records**), resulting in further changes to axes, annotations, and emphases. The scales of the x -axis (years) and y -axis (forecasted GDP growth) are consequently altered. The annotation and emphasis (in red and boldface) for the forecast of 2010

²<https://www.wsj.com/graphics/how-bond-yields-got-this-low/>

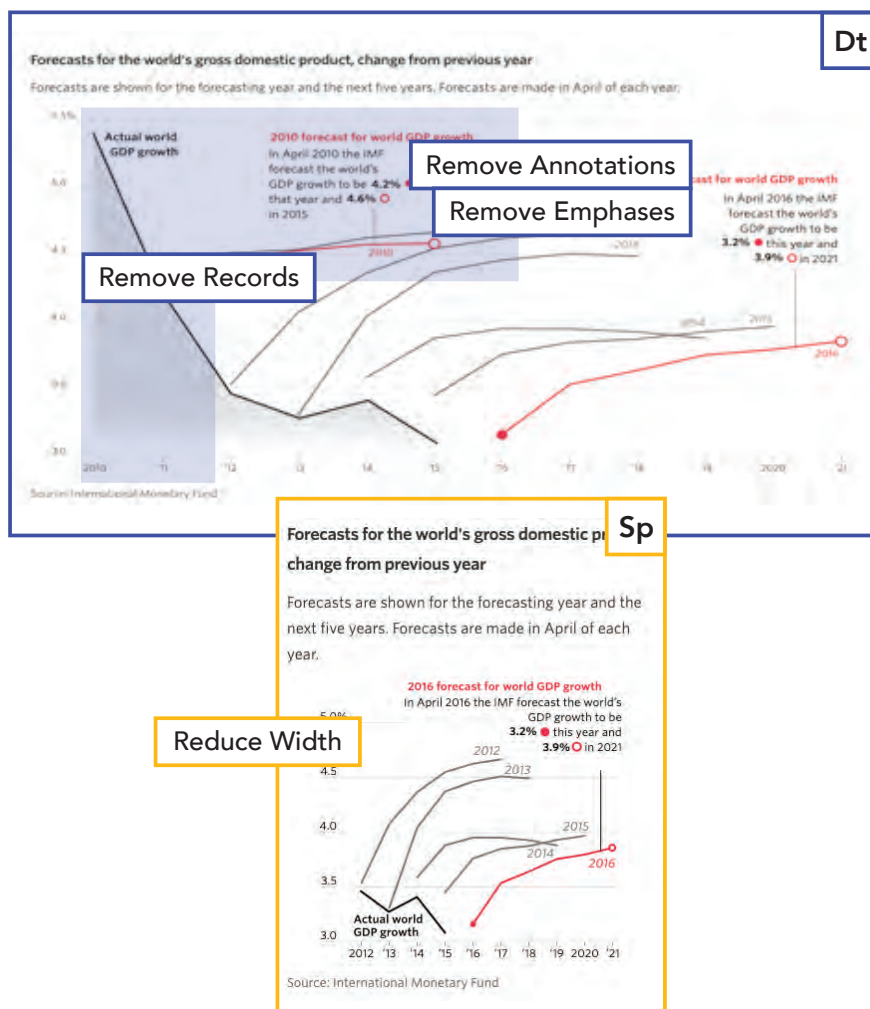


Figure 3.2: Screenshots of *Bond Yield's* desktop and smartphone view pair that illustrates *remove records*, *remove annotations*, *remove emphases*, and *reduce width*. Blue highlights indicate parts of the desktop view that are removed in smartphone.

observed GDP growth have been omitted from the smartphone view (**remove emphases** and **remove annotations**). Additionally, the relative width of the smartphone view is slightly reduced (**reduce width**), compared to its height relative to the desktop view.

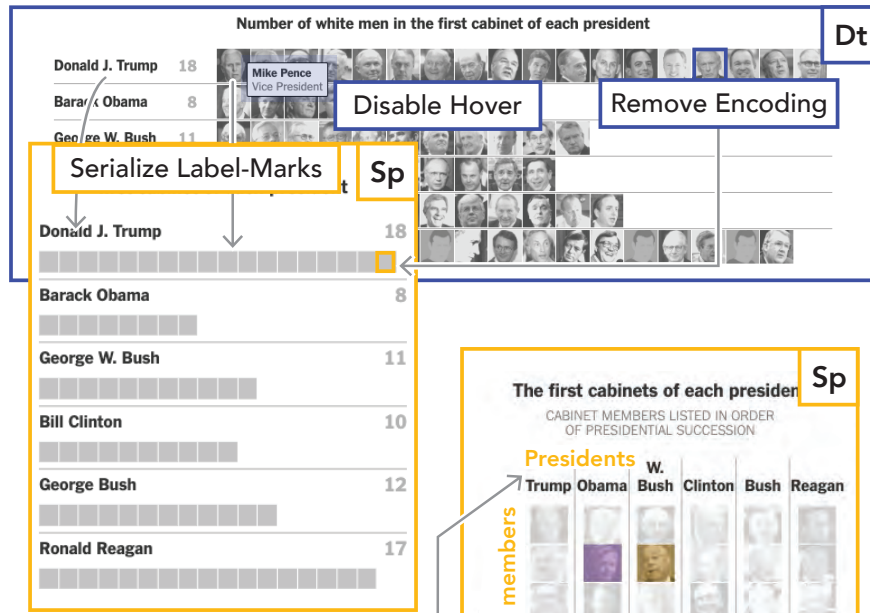
Two interrelated intentions may be behind these changes. First, the authors may have wanted to avoid an overly dense display caused by placing two long annotations close to each other. Second, they may have intended to support a more glanceable reading of the visualization by reducing the number of key points.

3.1.2.2 *U.S. Cabinet* - Encoding and Tooltips

*U.S. Cabinet*³ compares race and gender ratios in recent U.S. Cabinets, and demonstrates changes to visual encodings and interactions. In Figure 3.3, the desktop views of both the *upper* and *lower* visualizations share several similarities. They use the same encoding: a mapping of images of cabinet members' faces as bars. When the viewer hovers over each image in these desktop views, a tooltip appears and shows that member's name and role. However, the *upper* and *lower* visualizations exhibit different responsive transformations in terms of encodings and tooltips. First, in the *upper* visualization (Figure 3.3A), the authors omitted the images of faces and tooltips from the smartphone view (**remove encoding**—a nominal variable and **disable hover interactions**, respectively). Moreover, the labels and marks are serialized (**serialize label-marks**). In the *lower* visualization (Figure 3.3B), the axes are transposed from y (presidents) \times x (Cabinet members) to y (Cabinet members) \times x (presidents) (**transpose axes**). Also, the images of faces and the tooltip are pre-

³<https://nyti.ms/2jSp3WT>

(A) Upper



(B) Lower

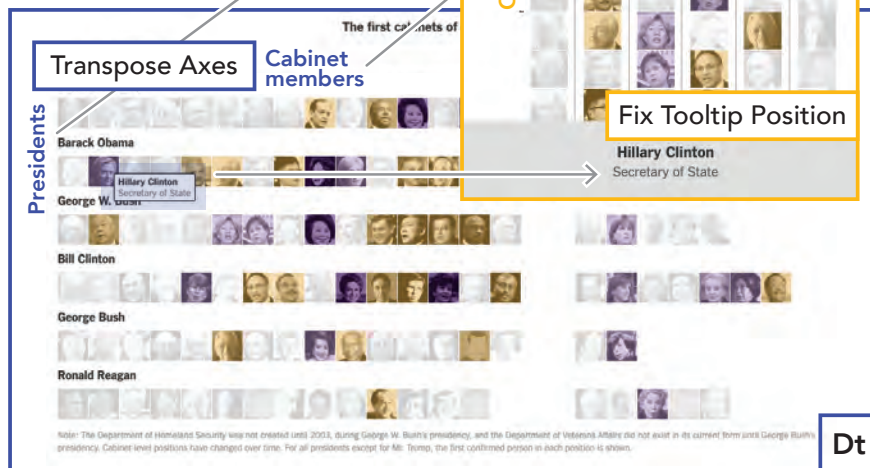


Figure 3.3: Screenshots of *U.S. Cabinet's* desktop and smartphone view pair that demonstrates *disable hover interaction*, *remove encoding*, *serialize label-marks*, *transpose axes*, and *fix tooltip position*.

served. However, in the smartphone view, the position of tooltips is fixed to the bottom of the screen (**fix tooltip position**), while on the desktop view, the tooltip is shown close to the corresponding image of a face (i.e., where it is triggered).

A rationale behind these decisions may be the role of each visualization in the article's narrative. The *upper* visualization shows one aspect of the data (white males), while the *lower* one provides a more comprehensive view including more variables (gender and race). Instead of maintaining the same design in both the *upper* and *lower* views, which might result in high visual density, the authors of *U.S. Cabinet* may have decided to simplify the *upper* visualization while transposing the axes to fit the *lower* visualization to the portrait aspect ratio.

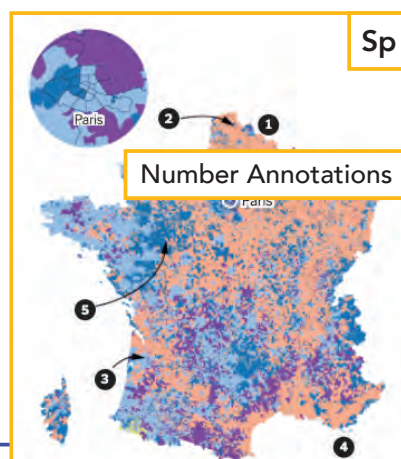
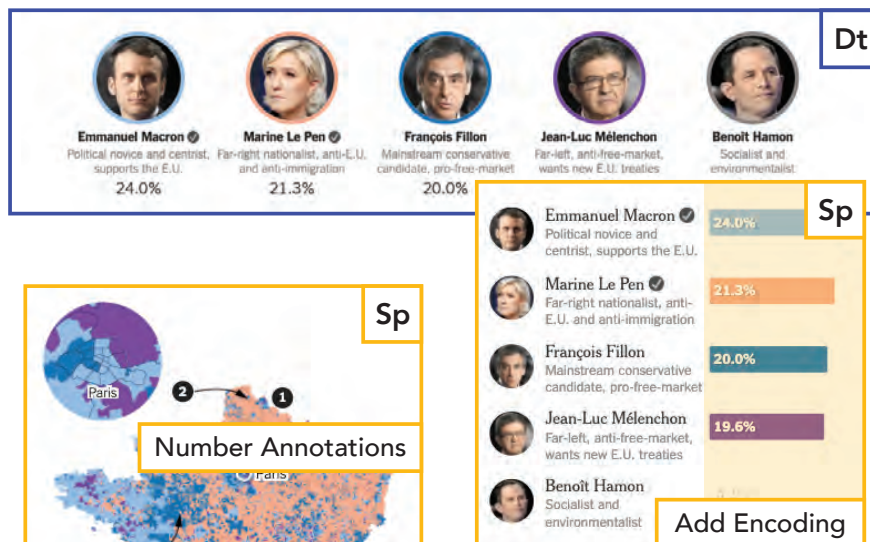
3.1.2.3 *French Election* - Addition and Compensation

*French Election*⁴, a map-based static visualization of results of the French 2017 presidential election, illustrates strategies of adding an encoding and compensating problems caused by another strategy. The *upper* visualization in the desktop view of Figure 3.4A uses a color encoding for the borders around the images of the candidate's faces (a nominal variable), but does not visually encode numerical data (the total vote shares of candidates), instead providing them as text. However, the smartphone version encodes the total vote shares of candidates using x position, resulting in a new bar graph (**add encoding**-a continuous variable). A possible intention behind this decision might be to ensure that the viewer perceives the values on smartphone.

The choropleth map in the *lower* visualization (Figure 3.4B) shows the distri-

⁴<https://nyti.ms/2pbI1uD>

(A) Upper



(B) Lower

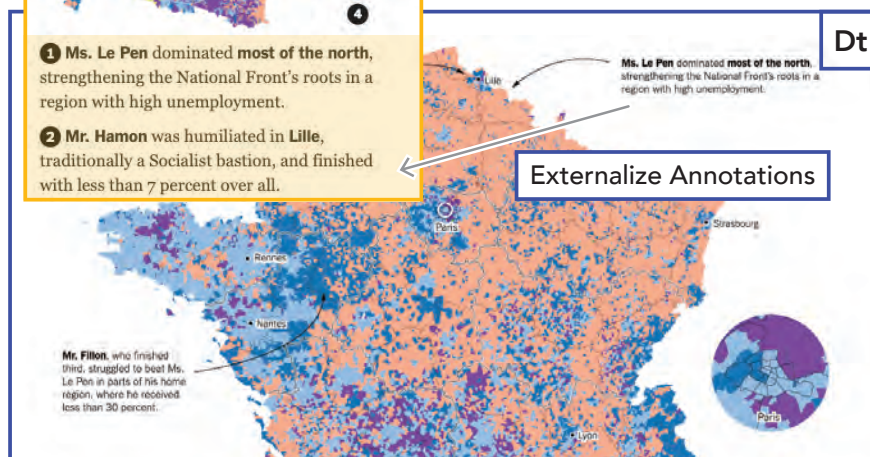


Figure 3.4: Screenshots of *French Election's* desktop and smartphone view pair that demonstrates *add encoding*, *externalize annotations*, and *number annotations*. Yellow highlights indicate parts that are added or repositioned in smartphone.

bution of the winners across France. Because of the discrepancy in population density between urban and rural areas, the predominant color on the map suggests a ranking that conflicts with the election outcome (i.e., the pink candidate loses the election). The authors rely on annotations to prevent misunderstandings in the desktop view. However, showing these annotations on smartphone at similar positions is unlikely to fit on the screen, so the authors moved the annotations out of the choropleth (**externalize annotations**). Presumably, to help readers locate the annotations to the map without background knowledge in French geography, the authors placed numbers in the original positions as a compensation method (**number annotations**).

3.1.3 Design Patterns

Our characterization of design patterns for responsive visualization distinguishes two dimensions of design decisions: (1) the *Target* (capturing what is changed from desktop to smartphone), and (2) the *Action* (capturing how the target is changed from desktop to smartphone). An overview of these two dimensions is shown in Figure 3.5, and a sample of design patterns is illustrated in Figure 3.6. On our explorable online gallery⁵, we provide a design guide in the form of pictograms and descriptions of the entire set of patterns. In the rest of this paper, we refer to visualization example articles in our interactive gallery for referenced strategies as **E#**⁶.

⁵<https://mucollective.github.io/responsive-vis-gallery/>

⁶This is reference to each example ‘article’ that often has multiple sample visualizations, and the numbers are not consecutive.

Dim. 1 Target	Dim. 2 Action	Input State	→	Output State
Data	Recompose	Input State	→	Output State
Record	Remove	Exist		Not exist
Field	Add	Not exist		Exist
Level	Replace	A		B
Encoding	Aggregate	Atomic		Aggregated
Interaction	Rescale	Bigger	↔	Smaller
Feature	Transpose	Input State	→	Output State
Trigger	Serialize	Parallel		Serial
Feedback	Parallelize	Serial		Parallel
Narrative	Axis-Transpose	X-Y		Y-X
Sequencing	Reposition	Input State	→	Output State
Annotations	Externalize	In the area of		Outside of
Emphases	Internalize	Outside of		In the area of
Text	Fix	Flexible		At Fixed
References/Layout	Fluid	At Fixed		Flexible
Labels	Relocate	at A		at B
References	Compensate	Input State	→	Output State
Layout	Toggle	-		w/ Toggle
Size	Number	-		w/ Numbers

Figure 3.5: The dimensions of design patterns for responsive visualization.

3.1.3.1 Targets: what elements are changed

The Target dimension consists of five types of entities that can be transformed in creating smartphone designs: data, encodings, interaction, narrative, and references and layout. The *data* category includes records (or rows or tuples), fields (or columns), and levels of hierarchy (or nesting). Transformations applied to data visualized in a desktop view typically result in visible changes in a smartphone visualization: changing the number of *records*, for example, can change the number of marks (e.g., line marks omitted in *Bond Yield*), as can changes to *levels of hierarchy* (e.g., changing from showing daily measurements to monthly), while changes

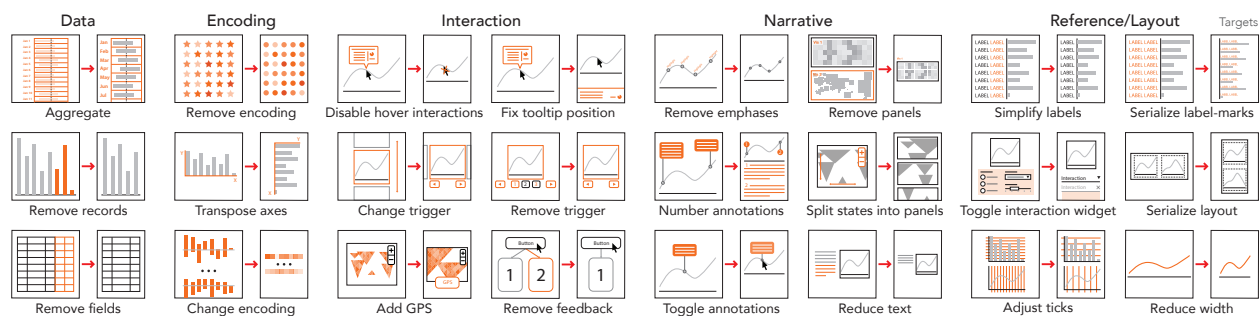


Figure 3.6: Examples of design patterns for responsive visualization, grouped by the target of the change (columns). The left and right side for each pattern denote desktop and smartphone views, respectively. Orange is used to highlight those elements that change from desktop to smartphone.

to what data *fields* are shown typically result in encoding changes (e.g., detail/image encoding removed in *U.S. Cabinet*). Changes to an *encoding* include switching a visual channel for showing a field (E139-size to length). The removal of an encoding often results from either removing a data field from the desktop source data (E1-a nominal variable on texture, E209-a continuous variable on hue, E236-continuous variables on position) or eliminating a redundant encoding from the desktop view (*Bond Yield*-area under line, E15-hue).

The *interaction* category describes targets related to a supported interaction in the desktop view, including a feature, trigger, or feedback mechanism. An interaction *trigger(s)* refers to how a viewer provides input to interact and *feedback* refers to the outcome of the interaction conveyed to the viewer. The *feature* subcategory refers to composites of interactions that realize a given functionality. For example, if a search feature receives user input via a text input box and an option list on desktop, and the text input box is removed on smartphone, then this is a change in trigger (E150). In our sample, we observed authors detached button triggers for zooming (when dragging interaction is available) (E14) and replaced a list of buttons with an option box (E139). However, if the search interaction functionality is

disabled on smartphone, we refer to it as removing a feature (E153). Authors in our sample omitted various features including sorting (E114), filtering (E150), and map browsing (E226). As illustrated in the *U.S. Cabinet* example, authors can further remove interaction features (tooltip) after removing a data field (detail).

The *narrative* category concerns sequence of information and authors' explicit messaging (annotation, emphases, text), inspired by prior work in narrative visualization [77, 78, 178, 179]. *Sequencing* deals with the existence of and methods for transitioning between multiple panels and states in a visualization. Panels refer to multiple views existing concurrently (e.g., in a poster style layout [78]), and states refer to multiple views sequenced or manipulated within the same panel (e.g., interactive slideshow). Changes made to sequencing can involve interactions when the sequencing method relies on related interactions (e.g., themed/numbered tabs-*split states into panels*). Sequencing has to do with how viewers “move” from one element to another (e.g., a fixed order by position or interactive tabs, or a random order by panning or zooming) whereas layout is more about how elements are placed on a page (e.g., horizontally versus vertically). *Annotations* and *emphases* are often associated with important data points or ranges. We use *text* to refer to sentences or paragraphs that appear along with a visualization and summarize it (E21-adding a summary sentence). Finally, the *reference and layout* category includes labels and references to help the viewer read the encodings and how visualizations and surrounding elements like text are laid out in a display.

3.1.3.2 Action: how targets are changed

Actions transform responsive visualization Targets in a desktop view for a smartphone view. Hoffswell et al. [99] described five high-level action categories (resize,

reposition, add, modify, remove). We extend their understanding of an *Action* dimension by defining action subcategories as functions with input and output states. We adopt this framing because it makes it possible to conceive of the inverses of the functions in a mobile-first design principle. For instance, the inverse of *externalize* is *internalize*, and the inverse of *relocate* is itself (i.e., un-relocating a target is another relocation of it).

The Action dimension consists of five categories: recompose, rescale, transpose, reposition, and compensate. The *recompose* category involves actions that change the existence of a target, including remove, add, replace, and aggregate. *Remove* actions refer to removing a target in the desktop for the smartphone design. For example, a *remove fields* pattern describes the removal of data fields, which leads to a concurrent removal of an encoding (e.g., *U.S. Cabinet-images* of faces). Authors in our sample often omitted hovering interactions for short labels (E19, E126) or tooltips (E226, E128) with corresponding hover highlight removed or maintained. Complex interaction features are often removed, formulated as the pattern *Disable X*, where *X* can be one of various interactions, such as hypothesis (E13), search (E153, E222, E227), and filter (E2, E138, E150). An *add* action inserts a target on smartphone that does not exist on the desktop view. Small targets, such as a call-out line (E267), or legend (E222), are occasionally added for smartphone views. Sometimes, more elaborate targets, such as summary text for fast reading (E21), a location finder for simplified interaction (E2), and context views for reduced focus views (E126, E202) are added. Remove and add actions are invertible. We observed a few instances of *replace* actions, referring to strategies that substitute a target in desktop with another target in smartphone. For example, *change measurements* refers to a transformation of data values to encode (e.g., from mapping raw values to mapping

ranks; E18 in Figure 1.2). An *aggregate* pattern reduces lower level values in a given data set to higher level aggregates using various aggregate functions (e.g., sum-E45, mean, count).

Rescale actions change a target from a bigger state to a smaller state or vice versa. For example, a *reduce width* pattern reduces the width of a visualization relative to the height, resulting in a narrower aspect ratio. A *simplify labels* pattern shortens labels through a predefined mapping (E46-1980 to '80, E8-January to J) while an *elaborate labels* pattern refers to detailing labels (E19) when the context for short labels is not concurrently visible.

Transpose actions change the orientation of targets. *Serialize* means placing two or more parallelly arranged elements on desktop in a vertically serial order on smartphone. Two or more panels, a pair of a visualization and a passage of text, or an interaction widget and a visualization are often serialized (*serialize layout*). It was one of the most frequent strategies in our sample. Within a visualization, labels and marks were frequently serialized (*serialize label-marks*, E43, E59). As the inverse of *serialize*, *parallelize* refers to placing two or more serially arranged elements on desktop in a horizontally parallel order on smartphone. This was often applied to legends (E1) and labels (E41) in our sample. An *axis-transpose* action exchanges *x*- and *y*-axes in charts with position encoding channels (*transpose axes*, *U.S. Cabinet*, E138) or a systematic layout of an interaction widget (*transpose interaction widget*, E141).

The *reposition* category refers to altering the position of targets, including *fluid*, *externalize-internalize*, and *relocate*. When labels, legends, and annotations are placed close to corresponding visual marks, they are often *externalized* from vi-

sualization to reduce visual density (*externalize labels*, E267; *legends*, E116; *annotations*, E262, E268). To the contrary, when small targets are placed outside of a visualization, they may be *internalized* (incorporated in the visualization) for effective use of space (*incorporate labels*, E6, E207; *internalize legends*, E116, E158). *Fix* and *fluid* actions refer to constraints on the arrangement of targets. For example, a tooltip for details-on-demand usually appears close to the corresponding data point when hovered on desktop. When a tooltip is big and likely to hide the chart on smartphone, the tooltip is often fixed at a particular position on screen (frequently at the bottom; *fix tooltip position*, E1, E129, E204). Similarly, a text message that interactively appears on desktop may be consistently displayed on a smartphone view (*unhide text*, E222). Authors can fix sequencing by giving a strict viewing order, for example *splitting explorable states into static panels* (E125, E132). In contrast, a *fluid* action refers to when elements are arranged in a fixed grid for on desktop, but that grid no longer exists in smartphone. For example, small multiples (E6, E16) and icon arrays (E19) are often rearranged following the screen width on smartphone (*fluid small multiples* and *fluid layout*, respectively). Authors at times *relocate* targets by *relocating annotations* (E20, E119) or *moving marks* like non-contiguous territories in a map (E159) to vacant areas in a chart.

Finally, the *compensate* category involves techniques used to compensate for the loss of information. When it is difficult to arrange labels (E133) or legends (E209) due to limited screen space, authors can prevent losing the information by *toggling* them. For example, it is possible to toggle an axis (i.e., a data field) in a parallel coordinate plot (E201) with multiple axes. Another compensation technique is *numbering*, which places numbers at the original positions of externalized targets on desktop (*French Election*).

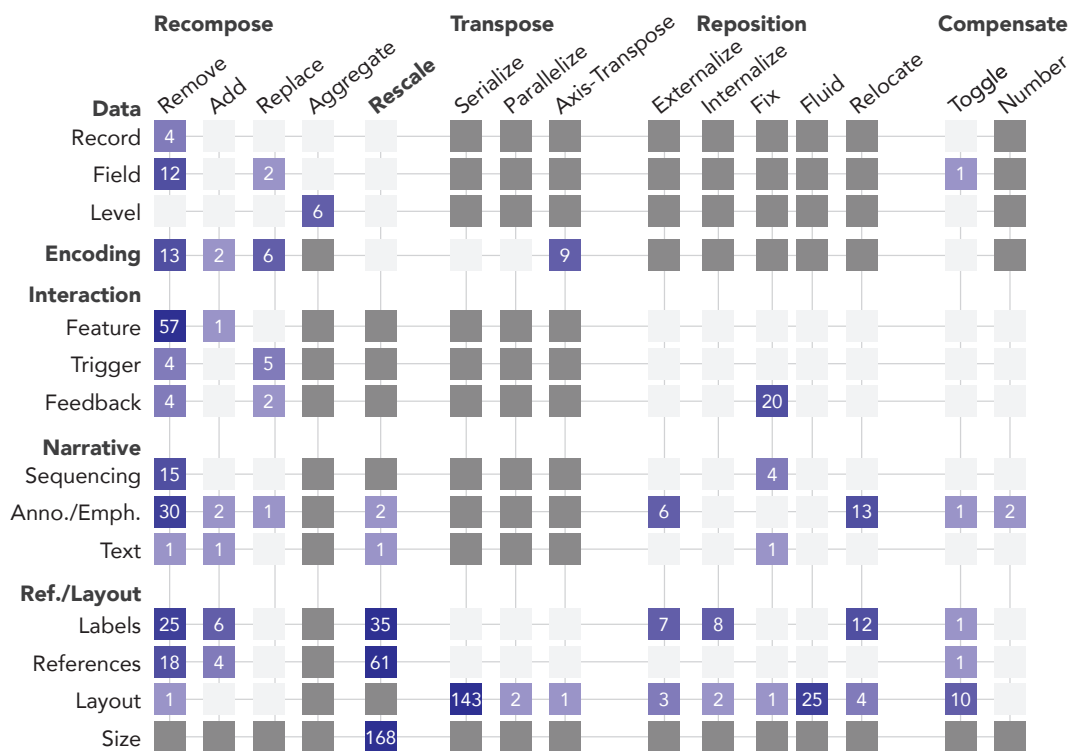


Figure 3.7: The distribution of responsive visualization strategies observed in our sample. Each strategy includes a Target (rows) and Action (columns), each of which we further categorized (bold labels). The gray-shaded cells denote combinations of target and action that we have not observed, and the dark gray cells indicate impossible combinations by definition. ‘Ref.’ stands for ‘references.’

3.1.3.3 Distribution of Responsive Visualization Strategies

As summarized in Figure 3.7, authors applied responsive transformations most frequently to Reference/Layout targets, followed by Interaction and Narrative targets. Transforming Data and Encoding targets was less frequent, though multiple authors employed strategies like removing data fields and removing, transposing, or replacing encodings. Overall, *rescale* and *remove* actions were most commonly used, with reducing size being most common, followed by *transpose* actions specifically involving serializing layout. We suspect that the automatability of these frequent strategies plays a role in their popularity. Authors may try to avoid substantive changes

between desktop and smartphone views, which require greater effort to make. That authors are exploring various types of more complex transformations, even if not in great frequency, suggests that alternatives to canonical simple responsive transformations can be preferable.

The matrix format of Figure 3.7 is used for layout purposes and does not indicate that every combination of target and action is possible. Combinations of targets and actions that we did not observe may suggest new responsive visualization design techniques to explore. For instance, authors could add a sequencing method (e.g., from small multiples to an interactive slideshow) or could fix labels if a desktop view has an extensive table format (e.g., freezing head columns). However, some combinations of target and action are not possible by definition. For example, authors cannot externalize data records or parallelize data fields because *reposition* and *transpose* actions are spatially defined while *data* targets are not. To the contrary, authors cannot aggregate interaction or layout because aggregation is a data-specific action although it can initiate downstream changes in other targets like labels.

3.2 Trade-offs in Responsive Visualization

Several insights from our analysis (Section 3.1) suggest that responsive visualization design is characterized by a set of trade-offs among competing goals. First, different authors use different strategies to resolve seemingly similar problems, implying that a single “best” solution may not exist for many situations [180, 181]. Second, the *compensate* actions that we observed imply trade-offs by suggesting that design strategies aimed at addressing one problem may result in other problems that need

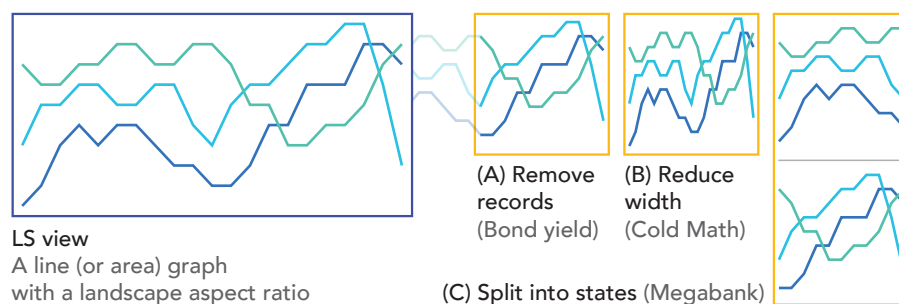


Figure 3.8: Motivating cases for trade-off analysis. From desktop line or area charts with a wide landscape aspect ratio, (A) *Bond Yield* removes records, (B) *Cold Math* reduces width, and (C) *Megabank* split states into panels for smartphone views.

to be addressed.

Our survey of responsive visualization authors (Section 3.1.1.3) indicates that authors may see maintaining the message of their visualization while adjusting information density (i.e., the amount of information per pixel) for smartphone devices as a central challenge in their work. We identify a set of trade-offs in responsive design related to an overarching trade-off between information density and preserving a visualization’s intended messages, which we conceptualize as a viewer’s ability to recognize certain comparisons or relationships in data. We describe three forms of information density problems that arise in transitioning designs from desktop to smartphone views, and five distinct types of losses describing ways in which intended messages or takeaways can be lost in attempts to address these problems.

3.2.1 Methods

In analyzing strategies, we reflected on what problems seemed likely to have led to the use of the strategy. For example, in Section 3.1.2, *Bond Yield*, *Cold Math* (E13), and *Megabank* (E132) are line or area charts with a wide landscape aspect ratio. The author of *Bond Yield* may have wanted to address graphical density by removing

records, while the author of *Cold Math* may have allowed changes to graphical perception to cope with the layout problem. The author of *Megabank* may have compromised interactive sequencing to address interaction complexity.

We first identified and compared desktop visualizations in our sample that shared similar design properties (e.g., high cardinality, a wide aspect ratio, multiple views, interaction features) and noted differences in applying design strategies in these cases. Through discussion and iterative coding passes, we taxonomized problems the authors may have wanted to address. Similarly, we sought to code ways in which visualization messages are changed or lost in smartphone views by applying those design strategies. Where possible, we drew on existing literature on visualization perception and interaction to motivate losses and density problems to enhance our understanding. We also noted how authors may have attempted to compensate for such changes in messages. We captured our evolving understanding of trade-offs during this coding process in an affinity graph mapping design problems, strategies, changes in message, and compensation, and iterated on this graph several times, returning often to our sample.

We summarize our results below but provide detailed characterizations of specific combinations of design choices manifesting trade-offs in our interactive gallery, each in terms of the underlying design problem prompting various design strategies and the downstream consequences of applying them.

3.2.2 Density Challenges

Graphical density poses challenges for authors when maintaining a large number of objects (e.g., marks, labels, annotations) in smartphone views results in higher

information density. A high information density may make it difficult to identify or perceive differences between data points (e.g., overplotting; c.f., [182, 183]).

Layout challenges occur when it is difficult to maintain the arrangement of bigger objects, such as individual visualizations, legends, or interaction widgets on a smartphone display. For example, fixed position elements, such as an overview and an interaction widget, may consume a larger proportion of the screen space on smartphone. Proportional rescaling of a visualization may overflow a single scroll height on smartphone, diminish the perceptibility of differences between values on a vertical scale, or decrease the impact of the visualization on the viewer's impressions by reducing its relative size.

Interaction complexity challenges occur when an interaction feature is not feasible on smartphone because it requires immediate rendering of numerous graphical objects (computing power) and/or more precise manipulation than is possible on smartphone (e.g., due to the fat-finger problem [102]).

3.2.3 Forms of Message Loss

Loss of information stems from the fact that one of the easiest ways to reduce graphical density or interaction complexity is to omit some information (e.g., *remove fields*, *remove annotations*). However, removing data, encodings, panels, or annotations may reduce the viewer's ability to get certain intended takeaways of a visualization, and make certain comparisons. This problem can happen when we remove explicit interpretations of data provided via text, overview views, distributional information, or information in the form of records or fields. For example, when authors aggregate data to adjust graphical density for a smaller screen, the viewer can no

longer make inferences about the distribution of aggregated variables unless the author takes specific steps to encode distribution through summary marks (e.g., error bars), changes encodings, or adds details-on-demand. When a fixed overview visualization is removed on the smartphone view, viewers may take a longer time to explore the visualization [184, 185].

Loss of interaction refers to loss of information that is available through interacting with a visualization (e.g., sorting data in a visualization by the viewer's criteria of interest). For example, when it is difficult to render a feature immediately on smartphone browsers, authors may remove it, or split states that a viewer previously reached through interaction into static panels. However, such changes may result in loss of other states that users can find by interacting with the desktop view.

Reduced discoverability refers to how using toggles and tabs to maintain information at a more appropriate density for a smaller screen can reduce viewers' abilities to find the information.

Reduced concurrency of elements results from how the reduced screen space on smartphone devices often leads authors to choose to serialize elements, such that serialized elements are no longer visible within a single scroll height of smartphone display. This can hamper comparisons across visualizations. Transposing x - and y -axes in a two dimensional view to better fit the portrait layout of smartphone can also lead to a visualization that is too long to fit within a single scroll height, hampering the viewer's ability to compare different values and assess high-level trends.

Changes in graphical perception can result from responsive strategies like dis-

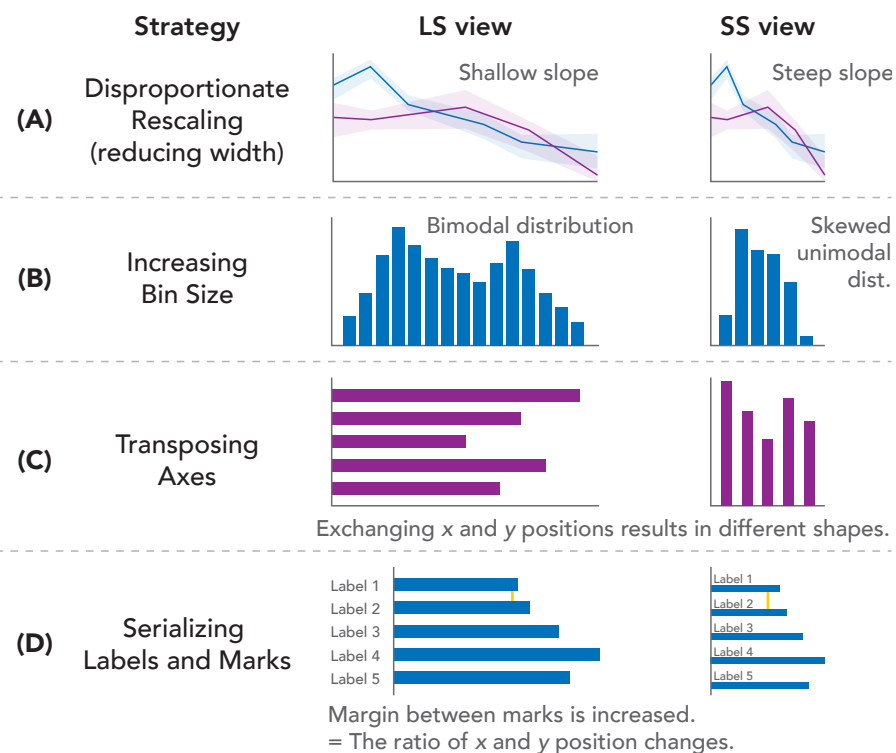


Figure 3.9: Examples of how transforming visualizations to fit narrower screen dimensions can change graphical perception.

proportionate rescaling, increasing bin size in aggregated views, transposing axes, or serializing labels and marks, as illustrated in Figure 3.9. Previous graphical perception studies [15, 186] have focused on slope perception; however, the graphical perception of other distributional information, such as dispersion or uncertainty may be affected by an aspect ratio change if they are encoded by position, length, or size channels. More generally, changing an encoding channel (e.g., position to color value, E213), transposing axes (e.g., *U.S. Cabinet*), or changing the range of a mapping (e.g., *serialize label-marks*) for a smartphone view is likely to prompt different impressions among readers relative to graphical perception in the desktop view.

3.2.4 Complexity of the Density-Message Trade-off

One reason navigating the trade-off between preserving information density and preserving message may be challenging is that what constitutes a message is often nebulous and rarely rigorously defined by an author. Instead, authors may experiment with different alternatives, relying on gisting and intuition to know when a change in design has hampered their goals. To evaluate message preservation between responsive alternatives, authors need to be sensitive to changes in visual attributes. For instance, a trend estimated on the same data (e.g., by regression) is invariant between responsive alternatives, while the visually implied trends may look different if those alternatives have changed aspect ratios, aggregation levels, or encodings.

The relationship between information and message preservation is also not always a direct mapping. In some cases, removing information or interactivity may strengthen a message, if that information was not critical to it. For example, if being able to detect an anomaly is an intended goal for viewers, then removing records that are not anomalies and do not significantly change the distribution should not affect their ability. If, however, the author prioritizes trend recognition, they could aggregate data in a way that preserves the slope of a best fit line or other properties like clusters, while obtaining a more appropriate degree of graphical density. To grapple with density-message trade-offs, authors must therefore think carefully about what they want to convey as takeaways and reason about the relative importance of different takeaways. In doing so, authors need to compare their ranking of different takeaways to what they perceive as changes in emphasis on those takeaways under different responsive alternatives. These complexities lead us to moti-

vate formalizing a notion of messages in visualization to take steps toward providing automated support for exploring design alternatives.

3.3 Discussion

Our characterization of design strategies, patterns, and trade-offs in responsive transformation of communicative visualization informs visualization research and practice in several ways. The design space implied by our results, which is captured by our design gallery, can help authors of responsive visualizations explore a larger space of design strategies. This more comprehensive coverage of the design space is useful to authors, who currently must rely on resources that describe strategies at a high level [99] or that provide technical documentation on a few, often more common strategies (e.g., responsive layout). Our results can also inform the design strategies that manual authoring tools (e.g., [99]) or machine learning-based approaches (e.g., [93]) for responsive design support. While this design space may not capture all responsive visualization strategies, we intentionally sought a relatively diverse sample spanning communication-oriented visualizations from the media, organizational reporting, and designerly interactive visualizations available online. Future work might extend the taxonomy we describe with more strategies as design innovations occur. Additionally, while we described our strategies from a default perspective of transitioning a desktop view for smaller screens, the strategies we describe can be understood from the opposite, smartphone to desktop direction.

Many design problems are characterized by the negotiation of trade-offs. By outlining key trade-offs in responsive visualization in terms of what types of information are “lost” by certain design strategies concerning information density, our

work aims to deepen understanding of the unique challenges. Our initial trade-off analysis provides only a first step to this larger goal. As people consume increasing amounts of information in multi-device environments, research effort around how to ensure that a set of visualizations capture the same “takeaways” despite design differences will become more important.

Moreover, the question of when a visualization preserves a message is integral to the process of designing visualizations for communication more broadly, as authors implicitly consider message preservation whenever they try out alternative designs. Currently, these judgments remain mostly subjective. A rich space for future research is to develop automated algorithms for predicting when two visualizations deliver the same “message,” operationalized as how well they support various tasks. Such attempts might use linear programming from human judgments as in GraphScape [84], or use deep learning models (e.g., [187]) or graphical statistical inference models (e.g., [188]) trained with human judgments. Formal approaches to capturing a visualization’s message may be useful in visualization design applications beyond responsive visualization, like simplification of content for different audiences.

3.3.1 A Roadmap for Responsive Visualization Recommendation

Our analysis naturally motivates the development of recommender systems for responsive visualization that leverage the primary density-message trade-off we identified. Given that strategies aimed at adjusting information density can lead to information loss in views for other screen sizes, it is important for authors to carefully consider which responsive view in the space of possible views achieves appropriate density while maintaining intended impressions or messages. Authors may fixate

on a design [10, 14], such as an desktop view that they have already thoroughly considered. They may also stick with a design presumably to avoid time-consuming design iterations for responsive alternatives [99]. That the same design specification does not lead to the same responsive transformation implies responsive visualization is a wicked problem where no single optimal solution may always exist [180, 181]. A lack of guidelines, combined with the reasons above, motivate tools that can help authors explore, perceive, and reason about the space of possible responsive alternatives given an original view.

A recommender approach requires formulating responsive visualization design as a search problem from a source view to target views (e.g., from desktop to smartphone views). We assume a scenario in which given a source view, a recommendation system populates a set of possible target designs ranked by how well they address density-message trade-offs.

An automated recommender approach entails several requirements that future research might consider. The first step to such an automated approach is to **encode responsive design strategies** to allow for a wider search space for alternative responsive views. Declarative grammars for visualization recommendation would be useful to generate a search space by formalizing design knowledge regarding responsive visualization as well as conventional guidelines (e.g., well-formedness [148], effectiveness [160]). In particular, a constraint-based approach that formalizes design knowledge as constraints (e.g., Draco [148]) can encode our design patterns as constraints (e.g., aggregating data with high cardinality, fixing tooltip position for smaller screens). Our representation of Actions as invertible functions with input and output states would be a useful schema in decomposing and formalizing

design patterns (e.g., *increase bin size* as from: `bin(size=15, field=A, ...)` → to: `bin(size=25, ...)`). Machine learning based approaches could also encode design strategies as multiple classification problems (e.g., a model that predicts whether each design strategy is applicable to a given source visualization). Chapter 5 will introduce a declarative grammar for expressing responsive transformations, and Chapter 6 will describe a recommendation search space written as a set of constraints.

Considering the complexity of density-message trade-offs (Section 3.2.4), future work should pursue ways to **operationalize message preservation** so that authors can better compare the relative importance of different messages under different alternatives. While it is not always easy to define visualization messages because of their implicitness, subjectivity, and domain specificity [189], prior approaches to insight-based visualization recommenders [20–22, 83, 190] have estimated visualization messages or insights based on analytic tasks (e.g., [191, 192]) such as estimating correlation and mean, finding data ranges, strength of trend (regression coefficients) from data. Research on how visualizations communicate messages or narratives may also be informative about the kinds of goals authors often have when designing charts [77, 78]. The next chapter (Chapter 4) will provide an approach to formalize changes to *task-oriented insights* under responsive transformations.

3.4 Conclusion

In this chapter, I first identified 76 design strategies for responsive visualization by analyzing 378 expert-created cases, implying the needs for a systematic representation for these strategies. From this design strategy analysis, I observed that authors

often applied those techniques to achieve an appropriate level of graphical density and interaction complexity for different device types, caused changes to embedded messages, and tried to compensate loss of information by applying other techniques. Based on this observation, I characterized design tradeoffs between preserving messages across responsive views and adjusting the graphical density of each view, calling for methods for reasoning about these tradeoffs. I use the findings of this chapter as a basis for building blocks of an interactive authoring system for responsive visualization. Chapter 4 will propose a set of loss measures approximating the differences in how two responsive views support readers' ability to obtain insights. Chapter 5 will introduce a declarative grammar for systematically expressing responsive transformation strategies. Chapter 6 will consolidate these building blocks into a mixed-initiative authoring tool where authors can apply designs generated by an automated recommender while retaining their ability to make custom designs.

Chapter 4

Formalizing Changes to Task-oriented Insights under Responsive Transformations

Visualization authors often transform their designs to accommodate different audience, styles, or display types. For example, authors might simplify charts for audiences with different graphical literacy, altering information conveyed in the new design [193]. Authors of responsive visualizations create multiple designs for different screen sizes and interactivity [99]. However, transforming visualizations may invoke trade-offs between desirable design criteria. For instance, in Figure 4.1, proportionate rescaling (a) makes it harder to compare values along the y -axis due to the reduced absolute height, while increasing the relative height (b) or transposing (c) can distort the shape of the represented distribution. Increasing the bin size (d) reduces possible comparisons and a viewer's ability to see distributional detail.

Unfortunately, design trade-offs make it difficult to reason about preserving takeaways or “insights” under visualization design transformations. Authors may need to iteratively try out different combinations of strategies like those in Figure 4.1 and compare them with the original design [99]. In the previous chapter (Section 3.2), we identified trade-offs in designing responsive visualization where authors try to

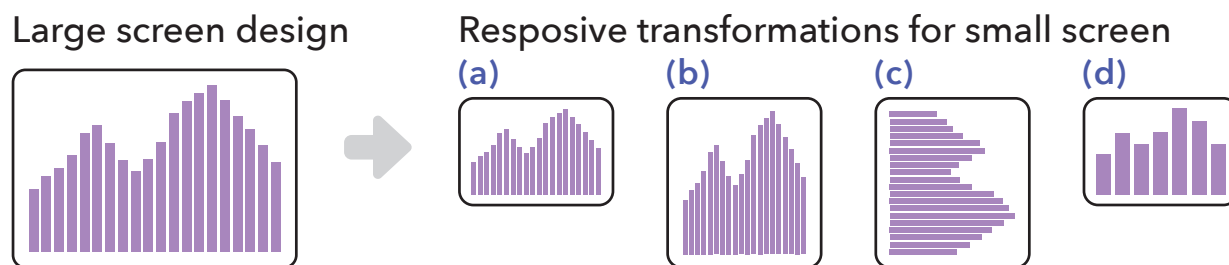


Figure 4.1: Example responsive transformations for small screen generated from a large screen design: (a) proportionate rescaling, (b) disproportionate rescaling, (c) transposing axes, and (d) increasing bin size.

find a balance between maintaining graphical density (i.e., an appropriate number of marks per unit screen size) and the preservation of a user’s ability to arrive at certain insights. For example, authors need to decide either to preserve distributional characteristics with higher visual density by rescaling proportionately (a) or to adjust visual density while losing distributional details by changing the bin size (d).

We contribute an automated approach to approximating the amount of change to task-oriented insights—insights that viewers are likely to be able to obtain from a visualization by performing visual tasks—under design transformation. We define measures that approximate a visualization’s support for three low-level visual analytic tasks discussed in the literature [191, 192]: the viewer’s ability to *identify* a datum, to *compare* pairs of data points, and to perceive a multivariate *trend*. We demonstrate the use of our measures for choosing between alternative transformations of a source desktop visualization in a responsive design context. We provide a prototype automated design recommender for responsive visualization that enumerates responsive design transformations based on an input source view and reasons about changes in task-oriented insights from the source design to each transformed design. Our recommender supports scatterplots, bar charts, line graphs,

and heatmaps with position, color, size, and shape encoding channels.

We train and test different machine learning (ML) models based on our measures to evaluate their utility for automatically ranking small screen visualization design alternatives given a desktop view. We achieve up to 84% accuracy (via a random forest model) in ranking a set of responsive transformations across a set of six source desktop views spanning different encoding channels. Models trained with our measures outperform two baseline models based on simple heuristics related to chart size changes (59%) and transposing of axes (63%). We discuss implications of our work for future research, including recommender-driven responsive visualization authoring and generalizing our approach to further visualization design domains such as simplification and style transfer.

4.1 Related Work

Prior approaches have focused on either capturing visualization insights or preserving visual consistencies between multiple views rather than insight consistency between views, keeping above decisions made manual. For instance, Cui et al. [21], Srinivasan et al. [83], Demiralp et al. [20], Tang et al. [22] characterize visualization insights using statistical measures like linear regression coefficients and p-values. Nevertheless, they are not intended for comparing multiple views, so these measures are invariant under cases like responsive visualization where the underlying data do not change. Furthermore, because these measures are defined in a top-down way, their applicability is limited when data have a pattern that is not pre-defined in their schema.

On the other hand, GraphScape [84] and Dziban [164] consider design consistency between a pair of views. The space of transformations covered by GraphScape [84] does not include view size transformations, so it cannot assign costs to changes in aspect ratio that are common in responsive visualization authoring. Although Dziban [164] extends GraphScape to suggest a view that is ‘anchored’ to the previous view for an exploratory data analysis process, it also assumes different subsets of data between the previous and current views and focuses more on similar chart encodings than on preserving task-oriented insights. Therefore, our work introduces a set of low-level measures that enable approximating changes to insights between a pair of responsive versions.

4.2 Problem Formulation

We propose formulating responsive visualization as a search problem from an input source view to transformed target views, following the characterization proposed in the last chapter. Consider a recommender that takes a source desktop view as input and returns a ranked set of targets as illustrated in Figure 4.2. The first step in creating such a recommender is to define a search space that can enumerate well-formed responsive targets. To generate useful target views from a source (large screen) visualization, a search space should cover common transformation strategies in responsive visualization, such as rescaling, aggregating, binning, and transposing [37, 99].

After enumerating target views, a responsive visualization recommender need to evaluate how well each target preserves certain information or “insights.” While the term insight can be overloaded [194], a relatively robust way to define insights comes from typologies for describing visualization judgments or patterns [191, 192].

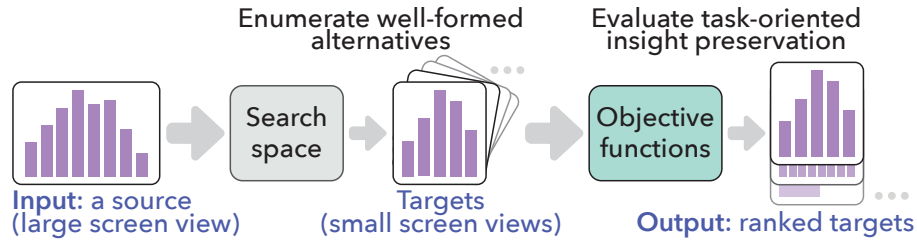


Figure 4.2: A pipeline for a responsive visualization recommender

These typologies suggest defining insights around common low-level visual analysis tasks like identifying and comparing data. In an automated design recommendation scenario, these task-oriented insights can be approximated by objective functions (i.e., loss measures) that capture support for common tasks, applied to both the source and target view. Finally, the recommender returns the set of target designs based on how well they minimize these loss measures. We formalize this problem and motivate and define three loss measures that we call *task-oriented insight preservation measures*. In Section 4.4, we describe a prototype visualization recommender in which we implemented the approach.

4.2.1 Notation

We define a visualization (or a view), V , as a three tuple

$$V = [D_V, C_V, E_V], \quad (4.1)$$

where D_V is the data used in V , C_V is a visualization specification (defining encodings, chart size, mark type, etc.), and E_V is a set of rendered values that we compute our measures on. For example, suppose a bivariate dataset with GDP and GNI fields (i.e., $D_V = \{x_1, x_2, \dots, x_n\}$, where $x_i = (x_i.GDP, x_i.GNI)$). C_V maps GDP and GNI to x and y positions of point marks, respectively, producing a scatterplot. The corre-

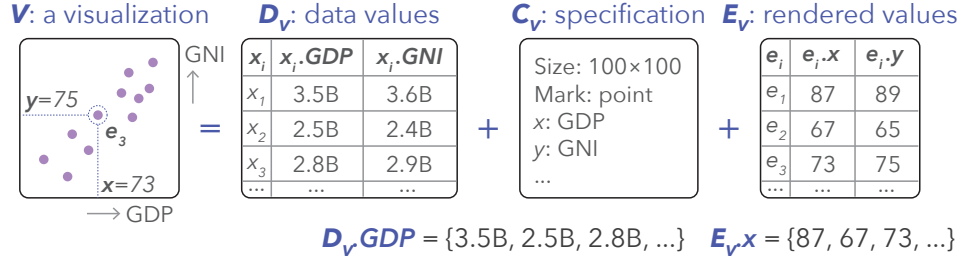


Figure 4.3: Our notation for a visualization. Rendered values are defined in the space implied by the visual variable (e.g., pixel space for position or size, color space for color).

sponding set of rendered values is a set of Cartesian coordinates on the XY-plane (i.e., $E_V = \{e_1, e_2, \dots, e_n\}$, where $e_i = (e_i.x, e_i.y)$ is the tuple of rendered values for x_i). Similarly, for a dataset containing a field *CO2* (emission) that is mapped to *color*, $e_i.color$ would correspond to the rendered value of $x_i.CO2$. For brevity, we define $D_V.field$ as a vector of *field* values and $E_V.channel$ as a vector of rendered values in *channel*. Our notation is also illustrated in Figure 4.3

Given a source view \mathbb{S} and a transformation (or target) \mathbb{T} , we represent the loss of insight type M from \mathbb{S} to \mathbb{T} as below:

$$Loss(\mathbb{S} \rightarrow \mathbb{T}; M) \tag{4.2}$$

For example, $Loss(\mathbb{S} \rightarrow \mathbb{T}; Trend)$ indicates trend loss from \mathbb{S} to \mathbb{T} .

4.3 Task-oriented Insight Preservation Measures

High level criteria for preserving task-oriented insights of a visualization include preserving datum-level information, maintaining comparability of data points, and preserving the aggregate features [192]. We use these distinct classes of information to define task-oriented insight loss measures for approximating how well a respon-

sive transformation preserves support for low-level tasks of identifying data, comparing data, and identifying trend. Our goal is to define a small set of measures that capture important types of low-level tasks a designer might wish to preserve in responsive transformation. Each measure should be distinct (i.e., mostly independent of the others) and should improve accuracy when combined with the others (such as through regression or ML modeling) to predict human judgments about how visualization transformations rank. Together, the measures should outperform reasonable baseline approaches based on simple heuristics. While chosen to cover three important classes of low-level analytic task, the measures we describe are not meant to be exhaustive, as there are many ways one could approximate support for task-oriented insights.

4.3.1 Identification Loss

Responsive visualization strategies often alter the number of visual attributes of marks that viewers can identify (affecting a low-level identification task [191, 192]). As illustrated in Figure 4.4, when the number of bin buckets of a histogram is decreased in a mobile view (a), each bar encodes more information on average than in the desktop view, such that some information about the distribution is lost. Similarly, strategies to adjust graphical density, like aggregating distributions (b) and filtering certain data (c), also reduce the number of identifiable attributes. We use *identification loss* to refer to changes to the identifiability of rendered values between a source view and a target.

Information theory, and in particular Shannon *Entropy* (entropy, hereafter) captures the information in a signal by measuring the minimum number of bits needed to encode it [195]. Given a random variable X , entropy is defined as $H(X) =$

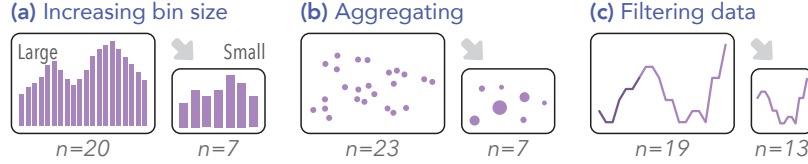


Figure 4.4: Responsive transformations that may cause identification loss.

$-\sum_{x \in X} P(x) \log_2 P(x)$. Applying this to visualization, suppose that for a source visualization \mathbb{S} , a vector for data field f , $D_{\mathbb{S}}f = \{x_1f, \dots, x_n f\}$, is mapped to an encoding channel c . The corresponding rendered values $E_{\mathbb{S}.c} = \{e_{1.c}, \dots, e_{n.c}\}$ compose a random variable $U_{\mathbb{S}.c}$ that takes the set of unique values of $E_{\mathbb{S}.c}$ as its outcome space, where the probability of $U_{\mathbb{S}.c}$ taking x is defined as the relative frequency of x in $E_{\mathbb{S}.c}$, formalized as

$$P(U_{\mathbb{S}.c} = x) = \text{Count}_i(e_{i.c} = x)/n \quad (4.3)$$

$$H(E_{\mathbb{S}.c}) = - \sum_x P(U_{\mathbb{S}.c} = x) \log_2 P(U_{\mathbb{S}.c} = x) \quad (4.4)$$

We can similarly compute the probabilities of rendered values, $P(U_{\mathbb{T}.c})$, and the entropy of an encoding channel, $H(E_{\mathbb{T}.c})$, for a target view \mathbb{T} . Finally, we can calculate the identification loss for the channel as the absolute difference in entropy (i.e., $|H(E_{\mathbb{S}.c}) - H(E_{\mathbb{T}.c})|$), where 0 difference is the identity. The final identification loss from \mathbb{S} to \mathbb{T} is the sum of absolute differences in entropy for each encoding channel c between the two views:

$$\text{Loss}(\mathbb{S} \rightarrow \mathbb{T}; \text{Identification}) = \sum_c |H(E_{\mathbb{S}.c}) - H(E_{\mathbb{T}.c})|, \quad (4.5)$$

4.3.2 Comparison Loss

Responsive transformations like resizing or scaling a view or aggregating data can alter the number of possible data comparisons that a user can make and how perceptually difficult they are (affecting a low-level comparison task [191, 192]). For instance, in Figure 4.5, resizing (a) diminishes the magnitude of difference between two highlighted data points in the smartphone design. In a mobile design with aggregation (b), viewers are no longer able to make each comparison that is available in the desktop view. This motivates estimating how similarly viewers are able to discriminate between pairs of points in a target view compared to the source view, which we refer to as *comparison loss*.

Empirical visualization studies (e.g., [82, 196]) often operationalizes accuracy as the viewer’s ability to perceive relationships between pairs of values. While simpler scalar statistics like a sum or mean might suffice under some transformations, a method that preserves the distribution of distances will be more robust to transformations that change the number of data points or scales (e.g., log-scale). We operationalize comparison loss as the difference in pairwise discriminability, measured using Earth Mover’s Distance (EMD), between the source and a target in each encod-

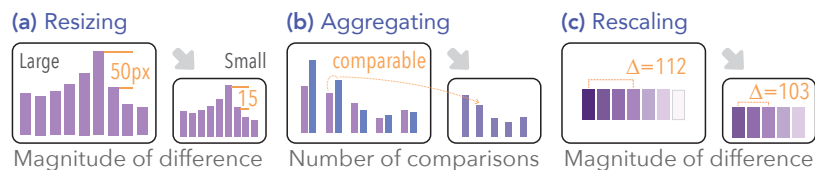


Figure 4.5: Responsive transformations that may cause comparison loss.

ing channel used in a visualization:

$$\text{Loss}(\mathbb{S} \rightarrow \mathbb{T}; \text{Comparison}) = \sum_c \text{EMD}(B_{\mathbb{S}.c}, B_{\mathbb{T}.c'}), \quad (4.6)$$

where $B_{\mathbb{S}.c}$ and $B_{\mathbb{T}.c'}$ are the *discriminability distributions* of the source and target views in encoding channel c and c' , respectively, that encode the same data field.

Given a source visualization \mathbb{S} , we define the discriminability distribution $B_{\mathbb{S}.c}$, of an encoding channel c for a view \mathbb{S} , as the set of distances between each pair of rendered values ($E_{\mathbb{S}.c}$) of \mathbb{S} in terms of c . This is formalized as

$$B_{\mathbb{S}.c} = \{d_c(e_{i.c}, e_{j.c}) : e_{i.c}, e_{j.c} \in E_{\mathbb{S}.c}\}, \quad (4.7)$$

where $d_c(\cdot, \cdot)$ is a distance metric for the encoding channel c .

4.3.2.1 Distance metrics

Ideally, comparison loss should account for differences in how well visual channels support perception of numerical values. Informed by visual perception models, we select several distance metrics intended to provide a rough proxy of the perceptual difference between two visual signals. While visual variables can have interaction effects [196–198], for simplicity in demonstrating our approach, we limit our use of perceptual distance metrics to encoding channel specific measures. However, as the state-of-the-art in predicting effects of visual variable interactions develops, our approach could be amended to consider combinations.

For position channels, we use the absolute difference between two position values (in pixel space), as human vision is highly accurate in discriminating posi-

tions according to Stevens' power law [18, 19] and empirical studies [182, 199]:

$$d_{position}(e_i.position, e_j.position) = |e_i.position - e_j.position| \quad (4.8)$$

We measure distance in a size channel using the absolute difference between two size values (in pixel) raised to the estimated Stevens' exponent of 0.7 [18, 19]:

$$d_{size}(e_i.size, e_j.size) = |e_i.size - e_j.size|^{0.7} \quad (4.9)$$

We calculate the Euclidean distance in CIELAB 2002 [200], a perceptual color space:

$$d_{color}(e_i.color, e_j.color) = \sqrt{(e_i.L - e_j.L)^2 + (e_i.a - e_j.a)^2 + (e_i.b - e_j.b)^2}, \quad (4.10)$$

where L , a , and b represent L^* , a^* , and b^* in CIELAB space.

Lastly, for shape encodings, we employ a perceptual kernel [201], a (symmetric) matrix of pairwise distances between visual attributes. The i, j -th element in the perceptual kernel for shape is the empirical probability of discriminating shape i from shape j based on an online crowdsourced experiment in which workers completed a triplet discrimination task where they chose the most dissimilar shape out of three shapes. Formally, our shape distance metric can be stated as:

$$d_{shape}(e_i.shape, e_j.shape) = P(e_i.shape \text{ is discriminated from } e_j.shape) \quad (4.11)$$

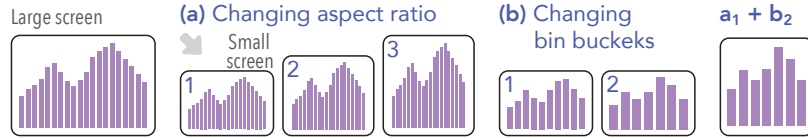


Figure 4.6: Responsive transformations motivating trend loss.

4.3.2.2 Comparing discriminability distributions

To quantify the discrepancy between the discriminability distributions of encoding channel c and c' (mapping the same field) for the source \mathbb{S} and target \mathbb{T} (i.e., $B_{\mathbb{S}.c}$ and $B_{\mathbb{T}.c'}$, respectively), we compute Earth Mover’s Distance [202] (EMD or Wasserstein distance). We use EMD, which measures the minimum cost to transform a distribution to another distribution, because it is non-parametric, symmetric, and unbounded. An EMD of 0 is the identity, and the greater the EMD is, the more different the two distributions are. Thus, the comparison loss between the source view \mathbb{S} and a target view \mathbb{T} is the sum of the EMD between their discriminability distributions in each encoding channel, formalized in Equation 4.6.

4.3.3 Trend Loss

Responsive transformations like disproportionate rescaling and changes to binning may impact the implied relationship (or trend) between two or more variables represented in a target view compared to the source view (affecting low-level trend identification [192]). As shown in Figure 4.6, different aspect ratios can alter the magnitude of the slope of a trend, and modifying bin size affect the amount of distributional information available. We use *trend loss* to refer to changes in the implied trend from the source to a target.

To capture representative data patterns while avoiding influences of noise, our

trend loss first estimates trend models between the source and target views using LOESS. We then compare the area (or volume) of the estimated trends because it is more sensitive to details that simpler methods (e.g., the difference between regression coefficients) might ignore. We define trend models for the quantitative encoding channels in our scope (position, color and size):

- $e_y \sim e_x$: a 2D trend of y on x as appears in a simple scatterplot, line chart, or bar graph.
- $e_{color} \sim e_x + e_y$: a 3D trend of color on x and y like a heatmap or a scatterplot with a continuous color channel
- $e_{size} \sim e_x + e_y$: a 3D trend of size on x and y (e.g., a scatterplot with a continuous size encoding)

After calculating trend models for a source and target, we can define trend loss as the sum of the relative area between curves (or volume between surfaces) of the estimated trends in each trend model (m). This is formalized as:

$$Loss(\mathbb{S} \rightarrow \mathbb{T}; \text{Trend}) = \sum_m A(LOESS(m_{\mathbb{S}}), LOESS(m_{\mathbb{T}})) \quad (4.12)$$

where A stands for the relative area between curves (ABC) between the source and target trends ($m_{\mathbb{S}}$ and $m_{\mathbb{T}}$), normalized by dividing by the area under the curve of the source trend for a 2D model. For a 3D model, A is the relative volume between surfaces (VBS), which is the VBS of the source and target trends divided by the volume under the surface of the source trend.

We estimate the trend models using LOESS regression [203] as it is non-parametric. We use uniform weights and bandwidth of 0.5 [203]. LOESS regression re-

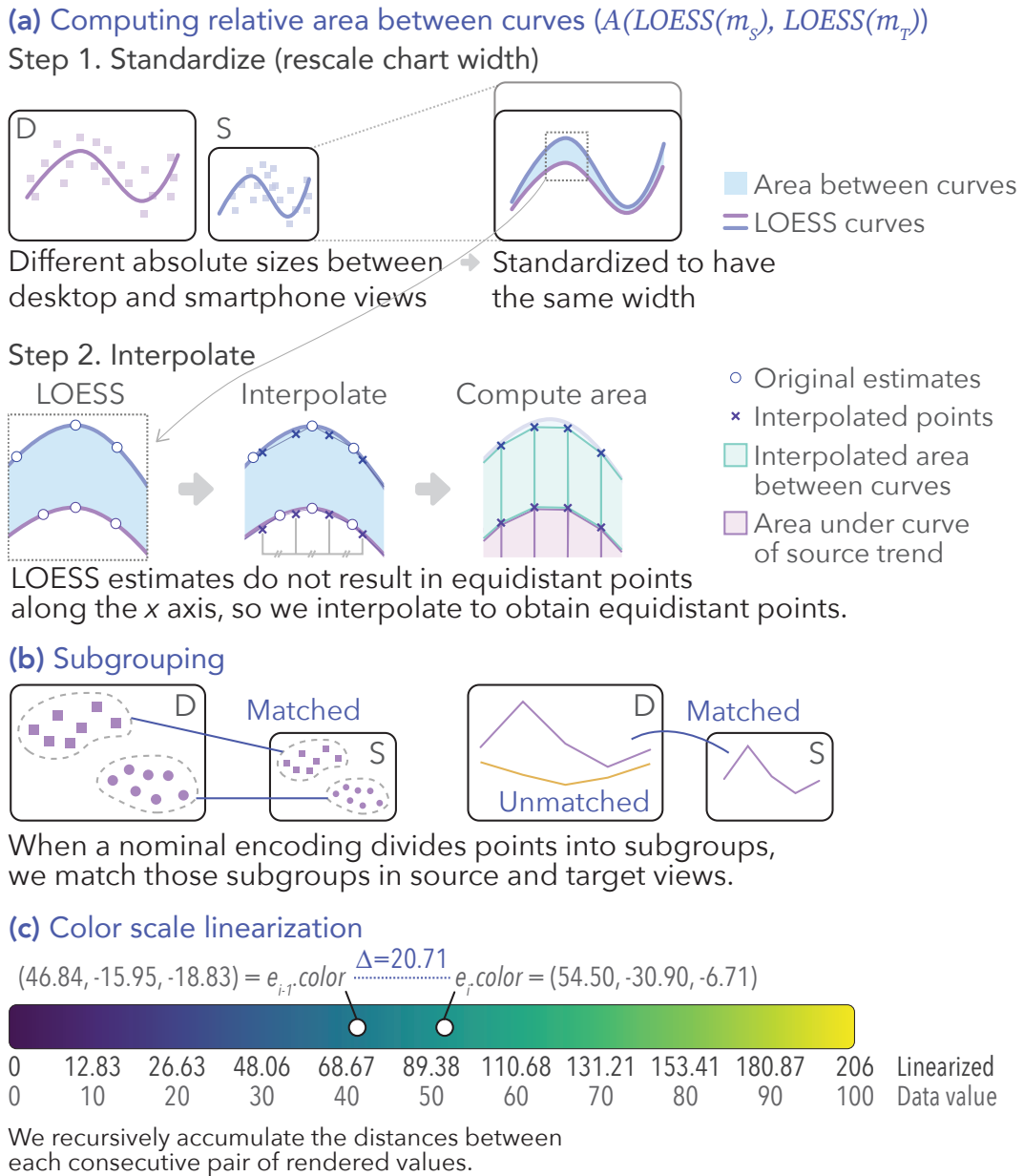


Figure 4.7: Components of computing trend loss. (a) Calculating area between curves by standardizing chart size and interpolating break points. (b) Dividing and matching subgroups. (c) Linearizing color scale. D indicates desktop view, and S denotes smartphone view.

turns an estimate at each observed value of the independent variable(s) (as an array of coordinates): an estimated curve for a 2D model and an estimated surface for a 3D model. Thus, when source and target views have different chart sizes or different sets of rendered values for the independent variable(s), it is difficult to directly compare the LOESS estimations. As shown in Figure 4.7a, we first standardize the chart sizes of two views by rescaling an estimated LOESS curve or surface in a target view to have the same chart width with the source. Then, we interpolate the LOESS curve to have equal distances between two consecutive coordinates for a 2D model (Figure 4.7b). We interpolate on 300 breakpoints in a 2D model by default, where one breakpoint corresponds to one to three pixels in many Web-based visualizations. For a 3D model, we interpolate 300×300 breakpoints from a LOESS surface in a similar way. Given these interpolations for the LOESS curves (or surfaces) in the source and target, we obtain the ABC (or VBS) segment at each breakpoint.

4.3.3.1 Subgroups

When a nominal variable encoded by color or shape divides the data set into subgroups, viewers might naturally consider each subgroup's trend independently. To distinguish trends implied by subgroups, we first identify and match subgroups which occur in both the source and target views by looking at their nominal data values, as depicted in Figure 4.7b. Then, we compute the relative ABC (or VBS) of each subgroup and combine them by taking their average.

4.3.3.2 Color scale linearization

Although a continuous color scale encodes a unidimensional vector, color is often modeled on a multi-dimensional space (e.g., RGB, CIELAB), which makes it com-

plex to estimate a LOESS surface. Similar to how common color schemes such as *viridis* or *magma* are designed to be perceptually uniform by keeping equi-distance in a perceptual color space between two consecutive color points [204], we can make use of the Euclidean distance between rendered color values in CIELAB to linearize a 3D color scheme. Specifically, we recursively accumulate the distances between each consecutive pair of rendered values to create a unidimensional vector. In Figure 4.7c, we show how the linear value of i -th color point is computed from that of $i - 1$ -th point; we take the calculated value of the $i - 1$ -th point and add to it the distance between the $i - 1$ -th and i -th points. The first color point is assigned as zero.

4.4 Prototype Responsive Visualization Recommender

To implement our task-oriented insight preservation measures, we developed a prototype responsive visualization recommender that enumerates and evaluates responsive designs (or targets). As shown in Figure 4.8, given an input source (desktop) view, our recommender first converts it to a partial specification, and then generates a search space of target (smartphone) views based on the partial specification. We adopt the desktop-first approach that visualization authors have described using [37, 99]. Finally, the recommender computes our measures between the source view and each target to rank those targets using an ML model trained on human-labeled rankings.

4.4.1 Enumerating Target Views

To enumerate target views, we need a formal grammar for representing visualization specifications and formulating a search space. We use Answer Set Program-

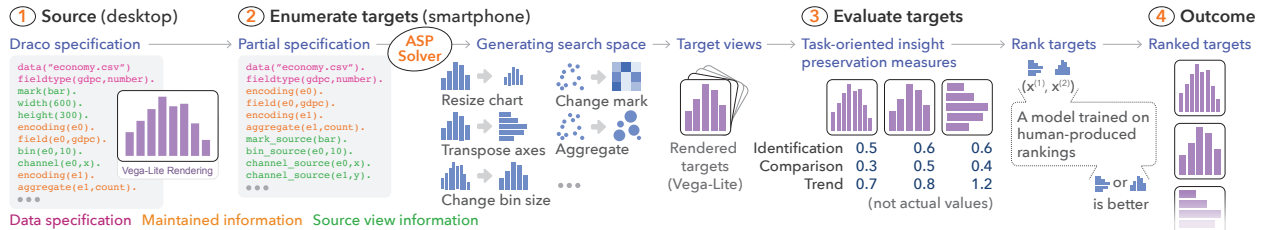


Figure 4.8: Prototype pipeline. (1) The full specification of an input source view in ASP. (2) Enumerating targets by extracting a partial specification of the source view and generating a search space using an ASP solver. (3) Evaluating targets by computing our loss measures and ranking them using a model trained on human-produced rankings. (4) Ranked targets.

ming (ASP) [162], particularly by modifying Draco [148]. ASP is a declarative programming language for complex search problems (e.g., satisfiability problems) that encodes knowledge as facts, rules, and constraints. Rules generate further facts, and constraints prevent certain combinations of facts. Formalized in ASP, for example, Draco has a rule that if an encoding is binned, then it is discrete, and a constraint that disallows logarithmic scale on a discrete encoding [148]. A constraint solver then solves an ASP program (the partial specification of a source view and our search space), returning stable sets of non-conflicting facts (enumerated target views with different transformation strategies). We use Clingo [205, 206] as our solver.

4.4.1.1 Converting to a partial specification

Our recommender converts the full specification of an input source view to a partial specification to allow applying responsive transformation strategies. We maintain the data specification (data file, data field definitions, and the number of rows) and encoding information (e.g., count aggregation, association of data field) that are not changed under transformation. We indicate the rest of the specification (mark type, chart size, and encoding channels) as information about the source view to con-

strain responsive transformation strategies (e.g., constraining possible mark type replacement, allowing for swapping position encodings for axis-transpose).

4.4.1.2 Generating a search space

Our goal in generating a search space is to produce a set of reasonable targets that a responsive visualization author might consider given a source view. We generate a search space by automatically applying responsive visualization transformations observed in Section 3.1 to a source visualization. Our prototype implements rescaling, aggregation, binning, transposing, and select changes to marks and encodings. For rescaling, we fix the width of target views and vary heights, in the range from the height resulting from proportionate rescaling to the height that forms the inverse aspect ratio with an increment of 50 px. For example, if the source view has a width of 600 px and a height of 300 px (an aspect ratio of 2:1) and the width of target views is fixed at 300 px, then the height varies from 150 px (2:1) to 600 (1:2) by 50 px (i.e., 150, 200, ..., 550, 600 px). Given a disaggregated source view, we generate alternatives by applying binning (max bin counts of 25, 15, and 5) and aggregation (count, mean, median, sum) as graphical density adjustment strategies. We also generate alternatives by transposing axes (i.e., swapping x and y position channels). Finally, in line with the observation of prior work that responsive visualization authors occasionally substituted mark types when adding an encoding channel for aggregation, we allow a mark type change in scatterplots from a point mark to a rectangle (heatmap). We formulate these strategies in ASP format and add them to Draco [148].

4.4.2 Evaluating and Ranking Targets

To evaluate enumerated targets, we calculate our loss measures on rendered values after rendering source and target views using Vega-Lite [96]. Then, we obtain rendered values, E_V , of a visualization V by gleaning Vega [147] states (a set of raw rendered values [207]). We implemented the loss measures in Python using SciPy [208]’s entropy and EMD functions. To compute LOESS regression, we use the LOESS package [209]. Finally, to rank the enumerated targets, we combine the computed loss values by training ML models, which we detail in Section 4.5. We use ML models for ranking instead of formalizing them in ASP because our measures are not declarative (not rule-based).

4.4.3 Examples

We introduce two example cases of transformations generated by our prototype and describe how our measures distinguish target views.

4.4.3.1 Case 1: Simple scatterplot

In the source scatterplot (Figure 4.9a), each point mark represents a country, and x and y positions encode Gini coefficients and annual growth rate of GDP per capita of different countries, respectively. The first example transformation (Ta1) is simple resizing. The second target view (Ta2) is transposed from the source view while keeping the size. The third and fourth target views (Ta3 and Ta4) are resized, binned in x and y scales, and aggregated by count, so the size of each dot represents the number of data points in the corresponding bin bucket. In the fifth target (Ta5), the mark type is changed from point to rectangle in addition to resizing, binning, and

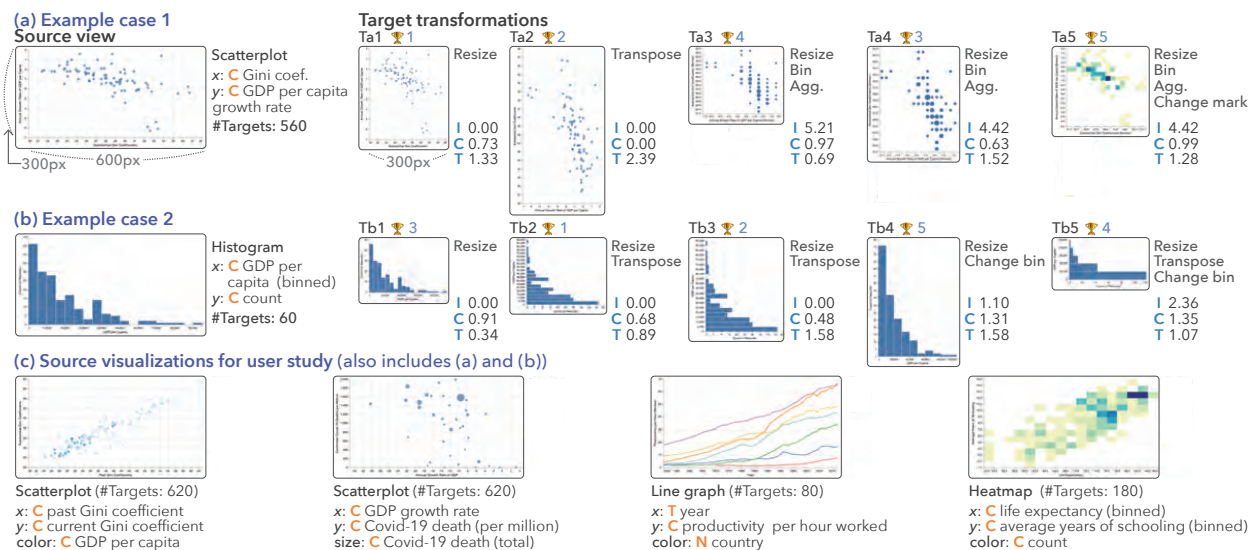


Figure 4.9: (a, b) Example target transformations enumerated by our prototype responsive visualization recommender (total size of search space per source given as #Targets). 🏆 indicates rankings of each five targets per source view predicted by our best model (see Section 4.5.3). (c) Source visualizations for our user study (also includes a and b). Sources views have width of 600px and height of 300px. The width of targets is fixed as 300px. Data sets are from *Our World in Data* [210–212]. C, Continuous, N, Nominal, T, Temporal, I, Identification loss, C, Comparison loss, and T, Trend loss.

aggregating, and the color of each rectangle encodes the number of data points in that cell.

Because Ta1 and Ta2 perfectly preserve the number of identifiable rendered values, identification loss is zero. Ta4 and Ta5 have more identifiable points than Ta3 (due to their smaller bin size), so they have smaller identification loss. While Ta1 has disaggregated values, Ta4 better preserves the distances between points in terms of position encoding, so it has smaller comparison loss. Compared to the source view, the implied trend given x and y positions in Ta1 has a more similar slope and hence smaller trend loss than Ta2, whereas Ta2 preserves the differences in the position encodings, resulting in zero comparison loss. Similarly, Ta3 has a smaller trend loss than Ta4 because Ta3 better preserves the visual shape of the distribution in the source view.

4.4.3.2 Case 2: Histogram

The source histogram in Figure 4.9b shows the distribution of GDP per capita of different countries. There are 23 bins along the x axis and each bar height (y position) represents the number of countries in the corresponding bin. The first target view (Tb1) is resized. The second and third target views (Tb2 and Tb3) are transposed with different resizing. In the fourth and fifth target views (Tb4 and Tb5) bin sizes are changed from (23 to 10 and 5, respectively), with Tb5 transposed.

As Tb1, Tb2, and Tb3 have no changes in binning, they have zero identification loss, whereas Tb4 and Tb5 has greater identification loss proportional to their bin sizes. While Tb1, Tb2, and Tb3 have the same binning, Tb3 has the most similar differences between bar heights and bar intervals in pixel space, so it has the smallest

comparison loss among them. Transposing axes (Tb3) better preserves the resolution for comparison (i.e., chart height and width), often resulting in the smaller comparison loss than other similarly transformed targets. Tb5 has smaller trend loss than Tb4 as it shows a similar aspect ratio to the source view, though inverted, as implied by x and y positions.

4.5 Model Training and Evaluation

A responsive visualization recommender should combine loss measures to rank a set of targets by how well they preserve task-oriented insights. For our prototype recommender, we train machine learning models to efficiently combine our loss measures and rank enumerated targets. We describe training data collection, model specification, and results.

4.5.1 Model Description

Prior approaches to ranking visualization designs (e.g., Draco-Learn [148], Deep-Eye [213]) utilize ML methods that convert the ranking problem to a pairwise ordering problem, such as RankSVM (Support Vector Machine) [214] and the learning-to-rank model [215]; we adopt a similar approach. A model, f , takes as input a pair of objects, $\mathbf{x} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$, and returns their orders (i.e., either $\mathbf{x}^{(1)}$ or $\mathbf{x}^{(2)}$ ranks higher).

$$f(g(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})) = \begin{cases} 1, & \text{if } \mathbf{x}^{(1)} \text{ appears higher in the ranking} \\ -1, & \text{if } \mathbf{x}^{(2)} \text{ appears higher in the ranking} \end{cases}, \quad (4.13)$$

where $g(\cdot, \cdot)$ is a mapping function that combines the features from a pair of objects. We consider vector difference and concatenation for g . Our models take two target

Table 4.1: The set of features for our ML models by each chart type. These features are either concatenated or differentiated for each pair of targets. Aggregated features are the sum of the corresponding Disaggregated features. Pink, bold-bordered circles represent required features, and yellow, light-bordered circles optional encoding-specific features.

Features		Chart types			
Aggregated	Disaggregated	Scatterplot	Bar graph	Line chart	Heatmap
Insight type	Enc./Model				
	Identification	x	○ Required	○	○
		y	○	○	○
		size	○ Optional		
		color	○	○	○
Comparison		shape	○	○	
		x	○	○	○
		y	○	○	○
		size	○		
		color	○	○	○
Trend		shape	○	○	
		y~x	○	○	○
		size~x+y	○		
		color~x+y	○		○

views representing transformations of the same source view and return the one with higher predicted ranking, as depicted in Figure 4.8.3.

4.5.1.1 Features

We define the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ where each row corresponds to a pair of target visualizations and columns represent the features (converted by g). We use our proposed loss measures as the features (Table 4.1). Aggregated features (A) refer to our three loss measures: identification, comparison, and trend loss, as described in Equations 4.5, 4.6, and 4.12 (Section 4.3). Disaggregated features (D) refer to the components of the aggregated features (e.g., the EMD value in each encoding channels for comparison loss). We standardized all features.

4.5.1.2 Model training

We train SVM with a linear kernel, K-nearest neighborhood (KNN) with $k = 1, 10$, logistic regression, decision tree (DT), and a Multilayer Perceptron (MLP) with four

layers and 128 perceptrons per layer, similar to other recent applications of ML in data visualization (e.g., Hu et al. [216], Luo et al. [213]). We also train ensemble models of DTs: random forest (RF) with 50, and 100 estimators, Adaptive Boosting (AB), and gradient boosting (GB). Given the moderate number of observations (1,067) in our data set, we use leave-one-out (LOO) as a cross validation iterator to obtain robust training results. We used Scikit-Learn [217] for training.

4.5.1.3 Baselines

In addition to the natural baseline of 50% (random), we include two simple heuristic-based baselines to evaluate the performance of our models. The first baseline (B1) includes the changes in chart width and height between a target and its source, capturing an intuition about maintaining size and aspect ratio. The second baseline (B2) is whether x and y axes are transposed, capturing an intuition that, of the strategies in our search space, transposing is the most drastic change.

4.5.2 Labeling

We obtained training and test data consisting of ranked target views for a set of source views using a Web-based task completed by nine visualization experts. As shown in Figure 4.10a, each labeler was assigned one out of three trial sets and performed 36 trials, with each trial asking them to rank five target transformations (small screen) given a source visualization (large screen).

4.5.2.1 Task materials

To create instances for labeling, we selected six desktop visualizations (source views) as shown in Figure 4.9c. Our goal was to include different chart types, multiple en-

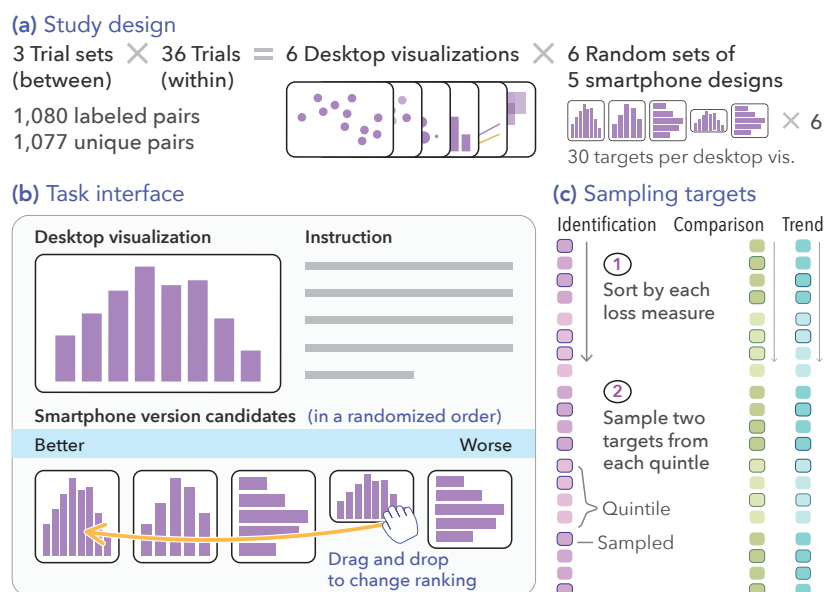


Figure 4.10: (a) Study design. (b) Task interface. (c) Quintile sampling of targets for task materials.

coding channels for identification and comparison losses, and different types of examples for trend loss (e.g., 2D/3D models, subgroups, color scale linearization). Our prototype generates 60 to 620 target transformations (2,120 in total) for these six source views. We generated three sets of 30 target views per source view for labeling, using quintile sampling per preservation (loss) measure, to ensure relatively diverse sets of targets. After sorting targets in terms of each of our three measures, we sampled two targets from each quintile of the top 100 targets per measure, as depicted in Figure 4.10c. We took the top 100 targets after inspecting the best ranked views per measure for each source view, to avoid labeling examples that might be obviously inferior. Because identification loss is measured using entropy and is primarily affected by how data are binned, certain source views had fewer than five unique discrete values within top 100 targets. In this case, we proportionately sampled each discrete value.

After sampling 30 targets for a source view in a trial set, we randomly divided them into six trials (but fixed these trials between labeler in the same trial set), so we had 1,080 pairs (1,077 unique pairs¹) labeled by three people each, as outlined in Equation 4.14. We randomly assigned each trial set to labeler (Figure 4.10a).

$$\begin{array}{c}
 \begin{array}{ccc}
 \text{\# of trials} & & \text{\# of pairs in each trial} \\
 \downarrow & & \downarrow \\
 6 & \times & \binom{5}{2} \\
 \uparrow & & \uparrow \\
 \text{\# of source views} & & \text{\# of trial sets}
 \end{array} \\
 6 \times \binom{5}{2} \times 6 \times 3 = 1080
 \end{array}
 \tag{4.14}$$

4.5.2.2 Labelers

All five authors, who have considerable background in visualization design and evaluation, and an additional convenience sample of four visualization experts (representing postdoctoral researchers and graduate students in visualization) participated in labeling. All labelers worked independently.

4.5.2.3 Labeling task

Each labeler was asked to imagine that they were a visualization designer for a responsive visualization project, tasked with ranking a set of small screen design alternatives created by transforming the source. Their goal was to consider what would be an appropriate small screen design that would also preserve insights or takeaways conveyed in the desktop version as much as possible.

The study interface is shown in Figure 4.10b. Each labeler completed 36 trials (6 desktop visualizations \times 6 sets of 5 smartphone design candidates). In each trial,

¹Each source view had 180 pairs, but the histogram source view with 60 transformations has 177 unique pairs.

the desktop visualization and five smartphone design candidates were shown, and labeler ranked the candidates by dragging and dropping them into an order. Trial order was randomized.

4.5.2.4 Aggregating labels

From the task, we collected human-judged rankings of 1,080 pairs each of which was ranked by three labelers. To produce our training data set, we aggregated the three labels obtained from the three labelers of each pair into a single label representing the majority opinion, such that that for the i -th pair $\mathbf{x}_i = (\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)})$, the label y_i is 1 if $\mathbf{x}_i^{(1)}$ is more likely to appear higher than $\mathbf{x}_i^{(2)}$, and -1 otherwise.

$$y_i = \begin{cases} 1, & \text{if } \mathbf{x}_i^{(1)} \text{ more often appears higher than } \mathbf{x}_i^{(2)} \\ -1, & \text{otherwise} \end{cases} \quad (4.15)$$

To avoid a biased distribution of training data as well as minimize the ordering effect within each pair, we randomized the order of pairs so that half of the pairs are labeled as 1 and the other half as -1 , which naturally sets the baseline training accuracy of 50%.

4.5.3 Results

All the experimental materials, and files used for analysis are included in the supplementary materials, available at <https://osf.io/jcvbx>.

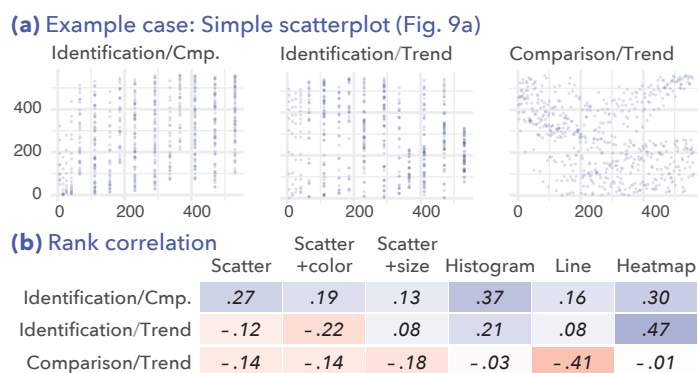


Figure 4.11: (a) Joint distributions of rankings of target views in each pair of aggregated features (the source visualization is shown in Figure 4.9a). (b) Kendall rank correlation coefficients for targets of our source views in Figure 4.9. Cmp (comparison).

4.5.3.1 Rank correlation between loss measures

To ensure that our loss measures capture different information about transformations, we compute and inspect rank correlations between each pair of aggregated, and each pair of disaggregated measures. If two different loss measures produce highly similar rankings of target views, then one of them might be redundant. Our measures tend to be orthogonal to each other (see Figure 4.11), with Kendall rank correlation coefficients [218] between -0.41 and 0.47 . The same pattern is observed for the disaggregated measures with overall correlation coefficients mostly between -0.5 and 0.5 (see Section C.1).

When the chart type of a source view allows a few, limited responsive transformation strategies due to its own design constraints (e.g., line chart, heatmap), the correlation between measures appear slightly higher than the other chart types. For example, it is often impossible to add a new encoding channel through aggregation or binning in a line chart. This makes the line chart more sensitive to chart size changes, resulting in relatively higher negative rank correlation between comparison and trend loss. Similarly, different binning levels in a heatmap can affect both

one’s ability to identify data points in different encoding channels and to recognize a trend implied by x and y on color channels (i.e., $e_{color} e_x + e_y$), leading to a slightly higher positive rank correlation between identification and trend loss.

4.5.3.2 Monotonicity

Ranking problems through pairwise comparison assume that the partial rankings used as input are consistent with the full ranking (monotonicity of rankings) [214, 219]. In other words, we need to ensure that the partial pairwise rankings that we calculate based on aggregated expert labels can yield a monotonic full ranking. Comparison sorting algorithms can be used to determine whether a monotonic full ranking can be obtained from pairwise rankings, as a comparison sort will only result in a monotonic ranking if the principles of transitivity ($a > b \wedge b > c \implies a > c$) and connexity ($\forall a$ and $b, a \leq b \vee b \leq a$) hold.

To confirm whether our expert labels satisfy the monotonicity assumption, we first sort the five target views in each of our 108 trials, using the ten aggregated pairwise rankings as a comparison function. Next, we check whether each consecutive pair in the reproduced ordering conflicts with the aggregated expert labels, because if a pair in the reproduced ordering is not aligned with the aggregated label, that trial violates the monotonicity assumption. 102 out of 108 trials (94.44%) in our data set had fully monotonic orderings. Of the six orderings which are not fully monotonic, five are partially monotonic with only one misaligned pair each (out of the ten ordered pairs). The other non-monotonic ordering (a trial with line chart as the source view) had multiple conflicts; we dropped this ordering from our training data, resulting in 1,070 training pairs (1,067 unique training pairs).

Table 4.2: (a) Prediction accuracy of our models, averaged over LOO cross validation. Other performance measures (AUC score and F1-score) appeared similarly to accuracy. e stands for the number of estimators of each random forest model. (b) Average importance of Disaggregated features ($g = \text{difference}$) measured by impurity-based importance from training a random forest model ($e = 50$) 10 times.

(a) Prediction accuracy												(b) Feature importances		
Models	Features	Disaggregated		Aggregated		Disagg. + Agg.		B1 (chart size change)		B2 (axes-transpose)		Features	Importance*	
		concatenate	difference	concatenate	difference	concatenate	difference	concatenate	difference	concatenate	difference			
	Mappings													
1-Nearest Neighborhood		78.73	77.88	71.60	67.48	77.79	77.13	51.08	49.02	62.32	59.70	Ident.	y .159	
K-Nearest Neighborhood		76.48	77.79	72.73	72.26	76.01	77.32	55.11	50.42	62.61	62.32	x .153		
Logistic Regression		78.63	78.07	75.91	76.38	78.63	77.98	53.70	54.08	62.42	62.42	color	.067	
SVM Linear		78.35	78.82	75.35	76.01	78.44	78.35	55.01	59.04	62.89	63.17	size	.026	
Decision Tree		76.19	75.91	70.20	67.95	76.19	74.70	50.05	51.08	63.17	63.17	Cmp.	y .141	
Random Forest (e=50)		83.04	82.38	76.66	74.70	82.19	82.57	50.89	50.61	61.20	62.89	x .114		
Random Forest (e=100)		84.07	82.29	77.41	74.32	82.76	82.38	51.55	50.98	62.04	63.07	color	.092	
Adaptive Boosting		81.16	79.01	72.73	73.01	79.94	78.73	56.42	53.05	62.42	63.17	size	.038	
Gradient Boosting		81.82	80.88	77.32	75.26	82.94	80.04	53.51	53.05	63.17	63.17	Trend	y-x .136	
Multilayer Perceptron		81.72	82.10	77.23	76.29	80.97	81.07	54.92	53.61	61.95	62.04	color-x+y	.055	
												size-x+y	.020	

4.5.3.3 Training results

Model performance

Overall, our models with disaggregated (D) and aggregated features (A) achieved prediction accuracy greater than 75% (Table 4.2a), showing the utility of our measures in ranking responsive design transformations. Ensemble models (RF, AB, and GB) with D features resulted in the highest overall accuracy (above 81%) because they iterate over multiple different models and we have a relatively small number of features. In particular, RF with D and 100 estimators showed highest accuracy of 84%. Our neural network model (MLP) also provided comparable performance to the ensemble models. Full training results are tabulated in Section C.2.

Models with D features in general obtained higher accuracy than A features, and combining them (D + A) did not provide significant gain in accuracy. Although they had only three features, our models with A features showed reasonable accuracy of up to 77.4% ($g = \text{concatenate}$) and 76.4% ($g = \text{difference}$). For mapping functions, concatenation performed slightly better than difference for our best per-

forming models (RF). A features performed much higher with concatenation than difference possibly due to the very small number of features.

Our models all outperformed those with both baselines features (B1 and B2), indicating that our loss measures capture information that simple heuristics, such as changes in chart size or axes transposition, are unable to capture. When we trained the best performing model (RF) with the features of only a single loss criterion (e.g., only trend), accuracy ranged from 52.6% to 79.7%, implying that our measures are more useful when combined than when used individually. As hypothetical upper bounds for accuracy, training and testing the model on the same data set resulted in accuracy from 84% (KNN) to 100% (RF).

Feature importance

To understand how the different loss measures function in our models, we inspected the importance of each disaggregated feature (mapping function $g = \text{difference}$) using the impurity-based importance measure (average information gain) by training a random forest model with 50 estimators (average over 10 training iterations). As shown in Table 4.2b, features related to position encodings ($x, y, e_x \sim e_y$) in general seem to have higher importance, which makes sense given their ubiquity in our sample.

Predicted rankings of example cases

Using the best prediction model (RF with 100 estimators), we predicted the rankings of example cases described in Section 4.4.3. Transformations from the simple scatterplot example (Figure 4.9a) are ranked as: Ta1 (resizing), Ta2 (transposing axes), Ta4, Ta3 (binning, resizing, aggregation), and Ta5 (binning, resizing, aggre-

gation, mark type change). Ta1 appears higher in ranking than Ta2 because Ta2 has higher trend loss, while Ta1 slightly sacrifices comparison loss. Ta4 is ranked in a higher position than Ta5 because Ta5 has higher comparison and trend loss. Responsive transformations from the histogram example (Figure 4.9b) are ranked as: Tb2 (transposing axes, resizing), Tb3 (transposing axes, resizing), Tb1 (resizing), Tb5 (resizing, changing bin size), and Tb4 (resizing, changing bin size). The transposed views (Tb2 and Tb3) are ranked higher than Tb1 probably because the model has more emphasis on comparison loss as the feature importance (Table 4.2b) shows. The ordering between Tb5 and Tb4 can be backed by the smaller trend loss of Tb5 while the difference in comparison loss between them appears subtle.

4.6 Discussion and Future Work

4.6.1 Extending and Validating Our Preservation Measures

We devised a small set of measures for three common low-level tasks in visualization use and found that they can be used to build reasonably well-performing ML models for ranking small screen design alternatives given a large screen source view. Our measures are not strongly correlated, and removing some of the measures results in lower predictive accuracy. However, there are other forms of prominent task-oriented insights that could extend our approach if approximated well, such as clustering data points or identifying outliers. As our measures lose information by processing rendered values, future work could estimate task-oriented insights with different methods, such as extracting and directly comparing image features from rendered visualizations.

There are also opportunities to strengthen and extend our measures through human subject studies. These include more formative research with mixed methods to understand heuristics and other strategies that visualization authors and users employ to reason about how well a design transformation preserves important takeaways. In addition, future work could conduct perceptual experiments that more precisely estimate human baselines for identification, comparison, and trend losses. We also used simple approximations of perceptual differences in position, size, and color channels which could be improved through new experiments specifically designed to understand how perception is affected on smaller screen sizes, adding to work like examining task performance on smaller screens by different chart types [43] and comparing task performance between small and large screens [53, 220]. A limitation is that our experiment was conducted on desktop devices. Future work could test on mobile devices, as well as explore mobile-first design contexts, as our measures are designed to be symmetric.

4.6.2 Responsive Visualization Authoring Tools

Our work demonstrates how task-oriented insight preservation can be used to rank design alternatives in responsive visualization. To do so, we formulated and evaluated our insight preservation measures on a search space representing common responsive visualization design strategies and mark-encoding combinations. However, our work should be extended in several important ways to support a responsive visualization authoring use case.

First, while more drastic encoding changes than those supported by our generator are rare in practice (as we observed in Section 3.1), this might be because responsive visualization authoring is currently a tedious process and authors sat-

isfice by exploring smaller regions of the design space. There are many strategies that could be added to a search space like the one we defined, and used to evaluate our measures as well as to learn more about how authors react when confronted with more diverse sets of design alternatives. For example, while we mainly consider single-view, static visualizations, many communicative visualizations employ multiple views and interactivity [77, 78, 178]. Ideally a responsive visualization recommender should be able to formulate related strategies (e.g., rearranging the layout of multiple views, omitting an interaction feature, editing non-data ink like legends). Recommenders may need to consider further conditions such as consistency constraints for multiple views [221], effectiveness of visualization sequence [84], semantics of composite visualizations [222], and effectiveness of interactive graphical encoding [223]. As indicated in Section 3.2, loss measures should be able to address concurrency of information because rearranging multiples views (e.g., serializing) can make it difficult to recognize data points at the same time on small screen devices. In addition, they should also account for loss of information that can only be obtained via user interaction (e.g., trend implied by filtered marks).

We envision our measures, and similar measures motivated to capture other task-oriented insights, being surfaced for an author to specify preferences on in a semi-automated responsive visualization design context. Because what our measures capture is relatively interpretable, authors may find it useful to customize them for certain design tasks, such as prioritizing one measure or changing how information is combined to capture identification, comparison, or trend loss. This is a strength of our approach relative to using a more “black-box” approach where model predictions might be difficult to explain. In Chapter 6, we demonstrate how our measures are applied to an interactive authoring tool with additional loss mea-

asures for approximating changes to text elements and graphical density. We also extend our search space with more strategies backed by a declarative syntax presented in Chapter 5.

4.6.2.1 Extending ML-based approaches

The human labelers in our experiment, including the authors, seemed to at times use strategies or heuristics such as preferring non-transposed views in their rankings or trying to minimize changes to aspect ratio for some chart types. However, models with our loss measures as features perform better than heuristic approaches like detecting axes-transpose and chart size changes, implying that task-oriented insights may be the right level at which to model rankings. As an extension, future work might learn pre-defined costs for different transformation strategies to reduce the time complexity of evaluating task-oriented insights preservation, similar to the approach adopted by Draco-Learn [148] which obtained costs for constraint violation. Learning such pre-defined costs may also enable better understanding how each responsive design strategy contributes to changes in task-oriented insights. An alternative approach could be to use our loss measures as cost functions and optimize different strategies to reduce them as MobileVisFixer [93] fixes a non-mobile-friendly visualizations for a mobile screen by minimizing heuristic-based costs. As recent deep learning models [93, 224] have performed well in visualization ranking problems, future work may further elaborate on those models. In doing so, one could combine our measures with image features (e.g., ScatterNet [224]) or chart parameters (e.g., aspect ratio, orientation [93]).

As noted in Section 4.5.3.2, there were a few partial and not fully monotonic orderings in our data set. A better model might ignore this assumption and try to iden-

tify highly recommendable transformations or classify them into multiple ordinal classes, yet this might come up with lower interpretability about recommendations due to a lack of explicit ordinal relationship between transformations.

4.7 Conclusion

In multi-context visualization authoring, maintaining consistency in insights across different contexts is challenging but necessary to ensure that readers obtain the same takeaways from different versions of the same visualization. To enable assessing changes to how a pair of views support readers' ability to obtain visualization insights in a responsive visualization setting, I proposed a set of task-oriented loss measures by capturing visualization insights as information that readers obtain by performing analytic tasks like identification, pairwise comparison, and trend recognition. I formulated these measures to be flexible, symmetric, and low-level so that they can be applied to various visualization designs using common encoding channels. I tested these measures by using them as objective functions for an automated recommender prototype for responsive visualizations, achieving 80% of accuracy in reproducing expert-labeled design rankings. In Chapter 6, I apply these measures to the design recommender for a mixed-initiative authoring tool. Before that, the next chapter will introduce another building block for the authoring tool—a systematic expression that allows for encoding a larger set of responsive design strategies than that used for the prototype in this chapter.

4.8 Acknowledgement

We thank NSF (#1907941) and Adobe for funding this work and anonymous labelers.

Chapter 5

CICERO: a Declarative Grammar for Responsive Visualization

Prior findings on responsive visualization design practices [37, 99] indicate that authors often start from a source view and then apply responsive transformations to produce a set of target views optimized for different screen types. However, this approach can be tedious as authors must manually explore, apply, and evaluate different responsive strategies one by one. For example, suppose that an author has created responsive views by crafting an artboard and/or specification per responsive view. When the author tries to revise one of the responsive views, they may need to transfer or reuse the revision strategies to other views by adjusting them to be applicable for the other views. In addition, they may have difficulty in expressing changes that occur across a design specification (e.g., example cases in Figure 5.1 and Figure 5.2). Authoring painpoints like these suggest a need for more intelligent authoring tools, such as semi- or fully automated recommenders that support exploring and reasoning about responsive design strategies.

A key step toward such intelligent responsive visualization authoring tools is a concise, declarative grammar that can express a diverse set of transformation

strategies. While declarative visualization grammars like Vega [147] and Vega-Lite [96] are well suited to developing more sophisticated visualization authoring tools, they are not necessarily well suited to representing visualization transformations; Hoffswell et al. [99] observe that different edit properties for text and marks in Vega-Lite [96] make it complicated to create the specifications for multiple versions of a visualization despite its high expressiveness. Indeed, many responsive visualization strategies identified in Chapter 3 can be written in Vega-Lite with high complexity. For instance, serializing labels and marks using Vega-Lite (i.e., placing them in a vertical order) requires layout adjustment keywords (Figure 5.1a1, line 7, 19–22), while parallelizing them (i.e., arraying them horizontally) does not require layout modifications in Vega-Lite (Figure 5.1b1, line 14). Whereas Vega-Lite requires authors to create separate specifications for each responsive view that interleave complex layout changes throughout the specifications, a declarative grammar for responsive transformations can express the same strategies in a simpler way as shown in Figure 5.1 (a2) and (b2). Such an approach can help visualization authors easily and quickly compose responsive design specifications and can help developers to more effectively develop authoring tools for responsive visualization.

To this end, we present CICERO: a flexible, expressive, and reusable declarative grammar for specifying responsive visualization transformations. The flexible *specifier* syntax of CICERO enables querying visualization elements using their role (e.g., mark, axis labels, title), underlying data, and attributes of visualization elements, independent of the structure of a source view specification. CICERO provides a compact set of *action* predicates (`add`, `duplicate`, `remove`, `replace`, `swap`, `modify`, `reposition`, and `transpose`) that can encode a diverse range of transformation techniques. Moreover, CICERO supports extracting and reusing generalizable transfor-

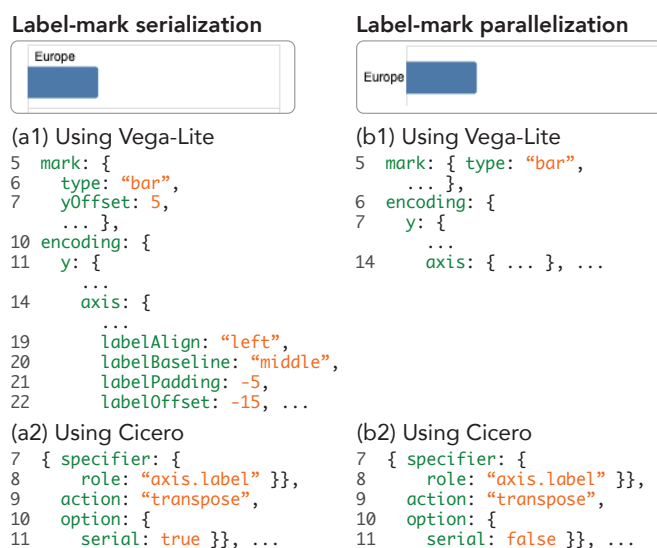


Figure 5.1: Design specifications for label-mark serialization using (a1) Vega-Lite and (a2) CICERO and parallelization using (b1) Vega-Lite and (b2) CICERO.

mations strategies across multiple responsive specifications. For example, the expressions (a2) and (b2) in Figure 5.1 can be reused on other visualizations with bar-like marks and axis labels.

To demonstrate the utility of CICERO, we develop a CICERO compiler for an extended version of Vega-Lite that we adapted to support annotations and other narrative devices and reproduce 13 real-world examples in CICERO (Section 5.5). We provide a set of principles for developing our CICERO compiler in terms of desirable properties of the association of visualization elements, preferable default behavior, and how to manage conflicts between transformations (Section 5.4). As CICERO is agnostic to the underlying structure of a source visualization, it can be leveraged in different visualization authoring tools. To demonstrate the feasibility of CICERO in such authoring tools, we describe how CICERO applies to a prototype recommender we developed for responsive transformations as a proof of concept and envision an approach to mixed-initiative authoring tools (Section 5.6.1). Future work can im-

plement a CICERO compiler for other declarative grammars like the original Vega-Lite [96] or ggplot2 [26] and other recommender approaches (e.g., [93]).

5.1 Background: Responsive Web and Visualization

Responsive design has been well-studied for the Web more generally [225, 226], but such techniques are not directly applicable to responsive visualization design because they are intended for Web layouts and based on limited knowledge of visualization design. For example, CSS media queries [227] express breakpoints for each responsive version of the contents. CSS specifications under a media query of `@media screen and max-width 600px` are shown only on a screen-side application (e.g., Web browser) with width ≤ 600 px. Similarly, CSS specifications under a media query of `@media speech` are used by speech synthesizers like a screen reader. However, using CSS alone cannot enable specification of many responsive transformations specific to visualization, such as transposing axis (requiring changes to scale functions), unfixing tooltip positions, changing mark types (requiring dynamic positioning), and transforming data (requiring custom JavaScript functions).

In practice, designers create responsive visualizations with multiple tools in an iterative manner. D3.js [25] is a highly expressive JavaScript (JS) library for SVG- or Canvas-based visualizations. According to prior work on visualization authoring practices [228, 229], designers often use D3.js (or equivalent tools) with ai2html [27], which renders Adobe Illustrator vector images (.ai files) to HTML. Designers first draw a visualization using D3.js [25], then load and edit the SVG graphic of the visualization as responsive ‘artboards’ in Adobe Illustrator [229]. Authors can also define responsive condition parameters for interactive visualizations using D3.js

(e.g., swapping scale functions for x and y positions for mobile screen). R3S.js [109] offers programming interfaces for such parameterization by extending D3.js [25]. However, it is not fully declarative, so authors need to imperatively define each transformation, which requires programming expertise. For example, to reposition a tooltip, which is a common responsive transformation strategy [37], R3S.js requires the use of custom CSS rules and/or JS functions.

For simple charts and quick edits, authors can utilize responsive properties of existing tools like Vega, Google Chart, and Microsoft Power BI. While Vega [147] and Vega-Lite [96] support some ‘sensible’ defaults, such as fitting the number of axis labels to the chart size, users need to have fully defined specifications for each of the responsive views. Google Chart [230] offers several default settings for mobile views such as truncating labels with an ellipsis (...). Power BI [113] provides defaults for responsiveness (e.g., making a visualization scrollable, rearranging legends, removing axis, etc.) [231]. While these tools can simplify the design process, their limited expressiveness may prevent authors from specifying intended responsive transformations, limiting their ability to convey insights.

Lastly, commercial tools like ZingChart and DataWrapper allow for responsive settings. ZingChart [97] provides ‘media rules’ through which a designer can declare a screen size condition for a visualization element (e.g., label: ‘October 4’ for screen size greater than 500px and ‘Oct. 4’ for screen size smaller than 500px). However, those media rules are dependent on the chart type—for example, transposing a scatterplot and a bar chart requires changes to data structure and the chart type, respectively—which limits the expressiveness and flexibility for responsive transformations. DataWrapper [24], an authoring tool for communicative visualizations,

allows authors to choose whether and how to show a visualization element for mobile screens (e.g., showing a table as a stack of cards [232], or numbering annotations [233]). However, it is not available in the form of a declarative grammar which limits how easily it can be extended or applied to future authoring tools, such as a mixed-initiative authoring tool.

5.2 Design Guidelines for a Responsive Visualization

Grammar

We derive three central design guidelines for a responsive visualization grammar based on prior work [97–99, 109, 228, 229] and previous chapters.

(D1) Be expressive. A responsive visualization grammar should be able to express a diverse set of responsive design strategies spanning different visualization elements. One approach is to characterize a responsive transformation strategy as a tuple of the visualization element(s) to change and a transformation action [37, 99]. Selecting visualization element(s) should support varying levels of customization for responsive transformations because transformations can include both systematic changes (e.g., externalizing all text annotations or shortening axis labels) and individual changes (e.g., externalizing a subset of annotations or highlighting a particular mark) [37]. A grammar needs to express responsive transformations as a concise set of ‘actions’ describing how visualization elements are changed [37, 99]. To be expressive, our grammar provides (1) a query syntax for selecting visualization elements both systematically and individually and (2) consistent, high-level action predicates that can encode a diverse set of responsive design strategies.

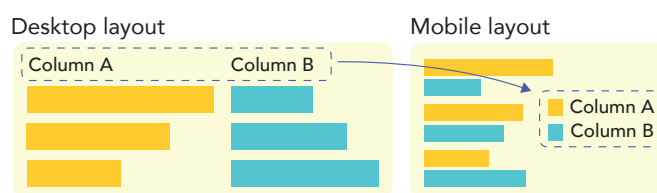


Figure 5.2: Responsive transformation from axis labels to a legend accompanied by a layout change for smaller display.

(D2) Be flexible. A responsive visualization grammar should offer flexibility in how an author can specify the behavior of an entity under a responsive transformation, independent of how the entity is expressed in the specification (or structure) of the source visualization. For example, suppose a visualization that has a nominal color encoding that maps dog, cat, and fox to red, blue, and green. Then, to select red marks, some authors can specify simply “red marks” (using attribute) while others can make the same selection by specifying “marks for dog” (using data). Furthermore, responsive transformations can occur across different visualization elements. For instance, as illustrated in Figure 5.2, one can change the layout by moving a column encoding to the row (partial view transpose) to accommodate a portrait aspect ratio. Following the previous transformation, the column labels can be replaced with a legend if there is a redundant mark property encoding. To be flexible, our responsive visualization grammar supports multiple expressions for specifying visualization elements that can be independent of the structure of a visualization.

(D3) Be reusable. A responsive visualization grammar should enable authors to easily (i.e., without making big changes) reapply generalizable responsive transformations across different visualizations. While reuse is straightforward for visualizations sharing the same properties, many responsive designs utilize generic trans-

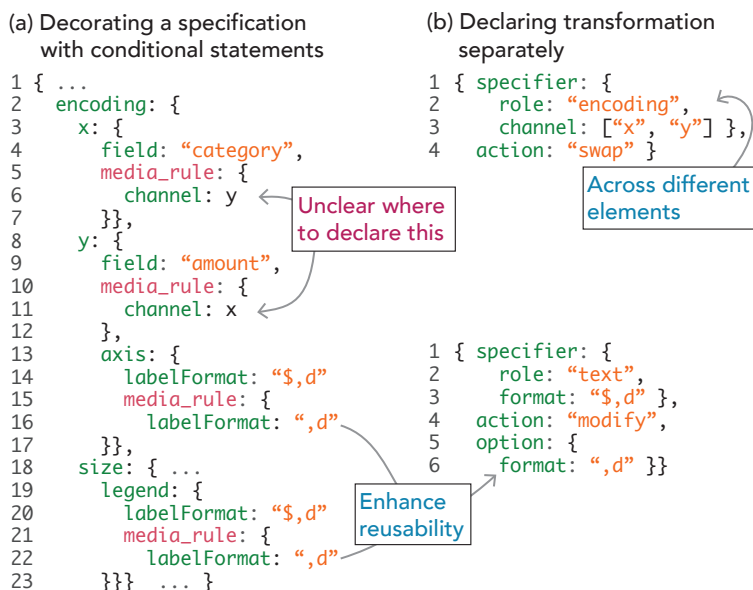


Figure 5.3: Two possible approaches to specifying responsive transformations. (a) Decorating a specification with conditional statements. (b) Separately defining responsive transformations.

transformations that are independent of the specific chart design, data, or base visualization (e.g., transposing the layout, numbering annotations, using a fixed tooltip position). Moreover, authors might want to repeat techniques only for certain features of a visualization (e.g., removing a data field regardless of chart type). To be reusable, our responsive visualization grammar represents each responsive transformation in a form that helps users to easily extract and apply transformations to other views.

With these guidelines in mind, there are several possible approaches for specifying responsive transformations, such as: (1) decorating a complete visualization specification and (2) separately defining responsive transformations. The first approach uses conditional keywords (e.g., `media_rule` in ZingChart [97]) to express transformations. For example, in Figure 5.3a, the `media_rule` keywords for the `x` (line 5–7) and `y` (line 10–12) encodings describe the changes for each encoding when viewed

in a media format (e.g., a ‘swap’ action). The `media_rule` keywords for the y axis (line 15–17) and the `size` legend (line 21–22) describe the same change to the label format for both types of elements. For the same set of transformations, the second approach in Figure 5.3b directly declares that the two position channels should be swapped and concisely describes changes to the label format for all text elements. While we choose to use the JSON format, other formats could be used to extend our approach; for example, Altair [234] is a Python wrapper for Vega-Lite [96] that leverages object-method chains rather than Vega-Lite’s JSON format.

While the first approach simplifies the learning process by extending an existing grammar, it can sometimes be tedious and unclear how to specify responsive transformations that apply to multiple elements. In particular, this approach often requires a single responsive change (e.g., transposing an axis) to be interleaved across multiple parts of the specification (Figure 5.3a, Line 5–7 and 10–12). In contrast, the second approach can enhance the reusability (**D3**: reusable) of a transformation specification by separating the desired responsive changes from the original visualization design. Furthermore, this approach can support more generalizable transformations that are independent of the original visualization structure (**D2**: flexible; e.g., changing all text formats directly). Therefore, in this work we opt for the second approach.

5.3 Responsive Visualization Grammar

We introduce CICERO, a declarative grammar designed to concisely express responsive transformations. Paired with a declarative specification for a source visualization, CICERO provides a concise syntax for describing responsive changes inde-

pendent of the structure of the original visualization specification. A single CICERO specification defines how to *transform* an initial visualization design to a new design, thereby encoding the responsive transformations required to convert a visualization into a responsive version for a particular format. A CICERO specification consists of a metadata object (`metadata`, line 2–4 of Figure 5.4a) and a list of transformation rules (`transformations`, line 5–78 of Figure 5.4a). The `metadata` object contains meta-information about the context for the target view (i.e., the intended environment, including information like the media type and screen size). The responsive strategies are encoded as separate rules in the list of `transformations`. We use a ‘list’ structure to enhance the reusability of the grammar by ensuring that each `rule` modularly describes a single responsive change to the source view (**D3**: reusable). The formal specification of the CICERO grammar is shown in Figure 5.5 and Section D.1.

The core components of a `rule` object include the `specifier` (which elements to change), an `action` (how to change the elements), and the `option` (what properties to change). The `specifier` queries the source visualization to identify the set of existing visualization elements to be transformed, and supports flexibly referencing visualization elements with varying levels of scope (**D2**: flexible). Then, the `action` and `option` provide high-level direction and detailed information about the change to be made to the selected elements, respectively, together encoding a wide range of transformations to elements selected by the `specifier` (**D1**: expressive). For example, the rule object in line 6–9 of Figure 5.4a states that the compiler should ‘modify’ (`action`) the ‘mark’ (`specifier`)’s ‘color’ to be ‘red’ (`option`).

In Section 5.5, we provide a complete walk-through of the “Bond Yields” ex-

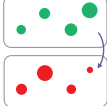
(a) Cicero specification overview

```

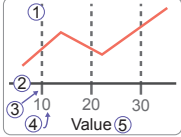
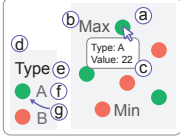
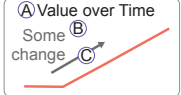
1 { name: "exampleSpec",
2   metadata: {
3     condition: "small",
4     aspectRatio: "portrait" },
5   transformations: [
6     { specifier: { role: "mark" },
7       action: "modify",
8       option: {
9         color: { value: "red" }}}],
78  ]}

```

a rule describing "modify the color of the marks to red"



(b) Cicero role expressions


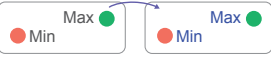
data	data sets
(data.)transform	transformations on raw data (e.g., filtering)
view	view/layout
(view.)row	the row elements of a view
(view.)column	the column elements of a view
(view.)facet	the facets of a multiple-view chart
(view.)axis	the axes of a view
(view.)hAxis	the horizontal axes of a view
(view.)vAxis	the vertical axes of a view
[axis].grid ①	the grid lines of axes
[axis].domain ②	the domain lines of axes
[axis].tick ③	the tick lines of axes
[axis].label ④	the labels of axes
[axis].title ⑤	the titles of axes
(view.)layer	axis/hAxis/vAxis
(view.)layer.transform	the layers of a view
(view.layer.mark) ⑩	transformations on data for layers
(view.layer.mark.label) ⑪	the marks of layers
(view.layer.mark.tooltip) ⑫	the text labels attached to marks
(view.layer.legend) ⑬	the legends of layers
(view.layer.legend.title) ⑭	the tooltips attached to marks
(view.layer.legend.label) ⑮	the titles of legends
(view.layer.legend.mark) ⑯	the labels of legends
(view.)title ⑰	the marks of legends
(view.)annotation ⑱	the title of a view
(view.)emphasis ⑲	the non-data text annotations
	the non-data informational marks

(c) Example transformations

```

1 { comment: "modify axis labels' color to blue
2   and axis domains' color to red",
3   specifier: { role: "axis" },
4   action: "modify",
5   option: {
6     label: { color: { value: "blue" } },
7     domain: { color: { value: "red" } }}}
8 { comment: "modify mark labels' color to blue",
9   specifier: { role: "mark" },
10  action: "modify",
11  option: {
12    role: "label",
13    color: { value: "blue" }}}

```

(c) Example transformations (continued)

```

14 { comment: "externalize annotations",
15   specifier: {
16     role: "annotation" },
17   action: "reposition",
18   option: { external: true }}
19
20 { comment: "transpose axes",
21   specifier: { role: "view" },
22   action: "transpose" }}
23
24 { comment: "transpose axes (equivalent)",
25   specifier: { role: "layer" },
26   action: "swap",
27   option: {
28     channel: ["x", "y"]}}
29
30 { comment: "serialize label-marks",
31   specifier: { role: "mark.label" },
32   action: "transpose",
33   option: { serial: true }}
34
35 { comment: "add values of 50 and 60 to axis",
36   specifier: { role: "axis" },
37   action: "add",
38   option: { values: [50, 60] }}
39
40 { comment: "duplicate an arrow mark (non-data)",
41   specifier: {
42     role: "emphasis",
43     id: "arrow" },
44   action: "duplicate",
45   option: { x: 50, y: 15 }}
46
47 { comment: "remove marks with a color channel",
48   specifier: {
49     role: "mark",
50     channel: "color" },
51   action: "remove" }}
52
53 { comment: "remove the color channel of marks",
54   specifier: {
55     role: "mark" },
56   action: "remove",
57   option: {
58     channel: "color" }}
59
60 { comment: "convert color channel to size channel",
61   specifier: {
62     role: "mark" },
63   action: "replace",
64   option: {
65     channel: { from: "color", to: "size" }}}
66
67 { comment: "replace axis label with color legend",
68   specifier: {
69     role: "axis.label", field: "plan" },
70   action: "replace",
71   option: {
72     to: {
73       role: "legend",
74       channel: "color" }}}
75
76
77 { comment: "exchange color and size channels",
78   specifier: { role: "mark" },
79   action: "swap",
80   option: {
81     channel: ["color", "size"]}}

```






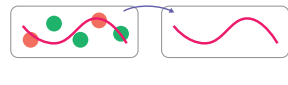
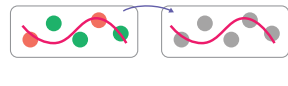


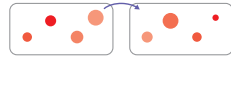











Figure 5.4: Examples and roles in the CICERO grammar. (a) An overview of a CICERO specification with a rule describing “modify the color of the marks to red”. (b) role expressions used in CICERO. (c) Example transformations referred to in Section 5.3.

```

CiceroSpec := Name?, Metadata?, Transformations
Name := <String>
Metadata := Condition?, MediaType?, AspectRatio?, ...?
Condition := xsmall | small | medium | large | xlarge | ...
MediaType := screen | paper | ...
AspectRatio := portrait | landscape | <Number> | ...
Transformations := <Rule>[]
Rule := Specifier, Action, Option?

Specifier := Role, Mark?, Index?, Id?, ..... Structure query
           Data?, Field?, Values?, Datatype?, ..... Data tquery
           Channel?, Operation?, Interaction?, ..... Attribute query
           $OtherAttributes?*

Action := modify | reposition | transpose | add
         | duplicate | remove | replace | swap

Option := Specifier* | To?, From?
To := Specifier*
From := Specifier*

Notation
"a := b, c": a is defined as a tuple of b and c, "a?": a is an optional argument, "...": extensible arguments, "<Abc>": data type,
"a ~ b, c": possible names for a are b and c, "a|b|c": either one of a, b, or c, "<A, B>[]": a list of a tuple of A and B,
"{}": key-value map (e.g., JavaScript Object, Python Dict), "<Number>": either a number or a string of a number with its unit (e.g., 350, "350px").

Note
*$OtherAttributes include encoding channels, role values, and other appearance-related properties (e.g., font styles, stroke styles, etc.).
*An option and its to and from properties share the same structure as a specifier but with different semantics (see Section 4.2).
#Possible role names are listed in Figure 5b.

```

Figure 5.5: The formal specification of CICERO. Section D.1 provides more detailed description.

ample; twelve additional examples are available in the online gallery¹. We chose properties and values for the `specifier`, `action`, and `option` in a principled fashion based on these example use cases (Section 5.5) and prior work [37, 99]. As a CICERO specification is independent of the structure of the source visualization, CICERO’s properties and values can be extended in the future as needed.

5.3.1 Specifier: Selecting elements to transform

A `specifier` indicates which elements to transform on the target visualization. A `specifier` should only express existing element(s) from the target view, which the compiler then uses to identify the corresponding element(s) and extract relevant

¹<https://see-mike-out.github.io/cicero-supplemental/>

properties. Authors tend to apply responsive transformations to groups of element(s) sharing the same role, such as axis labels, mark tooltips, or legend marks, as characterized in prior work [37, 99]. In addition, authors may want to include transformations specific to some data features (e.g., mark labels for specific data points, the axis corresponding to a particular data field) and/or the visual attributes of the visualization element(s) (e.g., red-colored bars). To express visualization elements using different characteristics, one can declare a specifier by *structure*, *data*, and *attribute* queries.

5.3.1.1 Structure query

Many declarative visualization grammars like ggplot2 [26], Vega [147], and Zing-Chart [97] define roles for visualization elements (e.g., marks, axes, or legends). Structure queries identify elements based on this role, and provide additional flexibility for selecting and grouping elements in different ways, regardless of how they are defined in the original visualization specification (D2: flexible). Keywords for structure queries include `role`, `mark`, `index`, and `id`.

The `role` keyword specifies the role of a visualization element (see Figure 5.4b). The `role` can be cascaded to specify subordinate elements like `"mark.label"` for labels associated with the visualization marks or `"legend.mark"` for legend marks. For brevity, cascaded role keywords can be shortened when its parent role is clear (e.g., `"layer.mark"` as `"mark"`; `"view.row"` as `"row"`, possible short forms are indicated as gray-colored and parenthesized in Figure 5.4b). The `mark` keyword specifies the type of mark, which is useful when there are multiple mark types in a visualization. One can include the `index` keyword to indicate the specific element to select from a group of related elements (e.g., `{role: "title", index: 1}` selects the second title element). To

indicate the first and last element, one can use "first" or "last" for the index value. Using "even" and "odd" can express every other (even and odd) element, respectively. The `id` keyword selects informational marks (`emphasis`) by their defined names or identifiers (e.g., line 43 in Figure 5.4).

5.3.1.2 Data query

A data query can reference a subset of `data`, a data `field`, the `datatype` of a variable, and `values` for elements to support varying level of customization in selecting visualization elements (**D1**: expressive). For example, the specifier `{role: "mark", data: {price: 30}}` selects all marks encoding a price value of 30. Likewise, the specifier `{role: "legend", datatype: "nominal"}` selects legends for nominal data variables; `{role: "axis", field: "price"}` expresses axes for the `price` field. The `values` keyword expresses a subset of values for a reference element that is tied to a certain data field like axis and legend. For instance, the specifier `{role: "axis.label", values: [30, 50]}` indicates the labels of an axis that encode value of 30 or 50. Similar to the `index` keyword for a structural query, one can use "even" and "odd" to specify every other (even and odd) value element. In order to support more complex data queries, we also provide a set of logical (NOT, AND, OR), arithmetic (`=`, `≠`, `≤`, `≥`, `≤`, `≥`), and string operations (`regex pattern`, `startsWith`, `includes`, `endsWith`) that can be composed to further select and filter elements based on properties of the data (**D2**: flexible).

5.3.1.3 Attribute query

An attribute query references visualization elements based on their properties or attributes. The primary attribute query keywords for identifying properties of visualization elements are: `channel`, `operation`, and `interaction`. The `channel` keyword

indicates whether the element has a certain encoding channel. For instance, the specifiers `{role: "layer", channel: "color"}` and `{role: "legend", channel: "color"}` indicate layers and legends with a color encoding channel, respectively. The `operation` keyword captures the type of data transformation operations applied to the elements (e.g., filter, aggregate), and the `interaction` keyword expresses the type of interaction features (e.g., zoom, interactive filter). CICERO also supports the use of style and position attribute keywords such as color, font size, orient, relative positions etc. (see `$OtherAttributes` in Figure 5.5). For marks, those style attributes can be used to indicate mark properties (e.g., static color value or color encoding channel). For example, `{role: "mark", color: "red"}` indicates red-colored marks.

5.3.2 Action & Option: Applying transformations

The `action` indicates how to change the elements queried by a specifier. We designed CICERO to provide a concise set of action predicates that can encode a large range of transformations (**D1**: expressive). The actions currently supported by CICERO are: `modify`, `reposition`, `transpose`, `add`, `duplicate`, `remove`, `replace`, and `swap`, chosen based on prior work [37, 99]. Our aim was to support a minimal set of action predicates from the prior work [37, 99]. For example, reposition actions in Kim et al. [37] can be efficiently expressed with using a single ‘reposition’ action and various option properties (e.g., `externalize` \rightarrow `reposition` + `{external: true}` and `fix` \rightarrow `reposition` + `{fix: true}`). The ‘modify’ action can also express these changes to positions, yet having a single ‘reposition’ keyword is likely simpler for authors to remember. This smaller set of action predicates does not sacrifice much expressiveness, as shown in our diverse set of examples in Figure 5.4, Section 5.5, and the

online gallery².

The `option` object in a rule further details the change indicated by the `action`. While the core structure of an `option` object is the same as a specifier, the structure and keywords vary based on the type of action. Keywords used in an `option` object refer to the properties or subordinate elements of the elements that were identified by the `specifier` (e.g., axis labels are subordinate elements of an axis), so a compiler should interpret an `option` object with regard to the `specifier`.

For example, one can use the `role` keyword to specify subordinate elements in an `option` object. An `option {role: "label"}` means legend labels if the specifier is `{role: "legend"}` or mark labels if the specifier is `{role: "mark"}`. When an `option` does not include the `role` keyword, then the properties in the `option` indicate those of the element identified by the `specifier`. For example, in line 8–9 of Figure 5.4a, `"color"` refers to the color of the `"mark"` (the `specifier` in line 6), while the `color` keyword in line 13 of Figure 5.4c expresses the color property of the marks' (`specifier`) labels (`option`). Finally, when `role` values are used as a keyword in the `option`, they indicate the subordinate elements of the element specified by the `specifier`. For instance, in Figure 5.4c, line 5–6 mean the color of the axes' (`specifier`) labels and domain lines (`option`), respectively. The entire transformation rule (line 1–6) states that the compiler should specify all the axes in the chart, and modify the labels' color to be blue and the domains' to be red.

²<https://see-mike-out.github.io/cicero-supplemental/>

5.3.2.1 Modify

A `modify` action changes the properties of an element to specific values, with an associated `option` object for expressing attributes of the elements selected by the `specifier`. For instance, one can modify the color of mark labels using the rule in line 8–13 of Figure 5.4. To make relative changes, including adding or multiplying an attribute value by some value, one can use `by` and `prod` operators, respectively. For instance, a user can express modifying the size of the specified marks by subtracting 30 using the `by` keyword:

```
{specifier: {role: "mark"}, action: "modify", option: {size: {by: -30}}}.
```

5.3.2.2 Reposition

A `reposition` action is a special type of the `modify` action designed to more intuitively support common transformations related to position properties like absolute positions (`x`, `y`), relative positions (`dx`, `dy`), externalization (`external`, `internal`), etc. For example, externalizing text annotations can be expressed as line 14–18 in Figure 5.4c. If a user wants to change the style and position properties together, then a `modify` action is recommended.

5.3.2.3 Transpose

A `transpose` action expresses the relative position of a pair of elements, the relationship of which is defined a priori, like two positional axes (`x` and `y`), labels associated with an axis or marks. A `transpose` action helps simplify expressions for relational properties. For example, the rule in line 20–22 (Figure 5.4) transposes the entire visualization. The equivalent is to `swap` the `x` and `y` position channels in `layers`, which CICERO expresses as in line 24–28. To serialize labels to their marks, one can use the

rule in line 30–33 with a `serial` keyword in the `option`. This behavior is the same as adjusting the label positions (relative x and y values) and mark offsets.

5.3.2.4 Add

An `add` action adds new elements in a visualization. Since the `specifier` only expresses existing elements (Section 5.3.1), the newly added elements are provided in an `option` object. For example, to express “add values of 50 and 60 to axis”, one can use the rule in line 35–38 in Figure 5.4c. When the existing axis selected by the `specifier` (line 36) has ticks and labels for each axis value, then the rule should result in adding ticks and labels for those values specified in the `option` (line 38).

5.3.2.5 Duplicate

A `duplicate` action copies the element identified by the `specifier`. If provided, an `option` indicates the properties for the duplicated element to change after duplication (e.g., repositioning the duplicated element in line 40–45 of Figure 5.4c). In this case, the `option` acts as a shortcut for a second `modify` transformation to update the newly added element.

5.3.2.6 Remove

A `remove` action removes elements identified by the `specifier` when no `option` is provided; when included, the `option` specifies the properties or subordinate elements that should be removed from the elements identified by the `specifier`. For instance, line 47–51 of Figure 5.4c removes all marks that include a `color channel` (no `option` is provided); to instead remove the color channel of these marks requires an `option` to be expressed (line 53–58).

5.3.2.7 Replace

A `replace` action expresses changes to the function of an entity while retaining its attributes. Sometimes, a visualization author may wish to change the role of an element such as changing from axis labels to legends (Figure 5.2) or changing an encoding channel of the marks to use increased screen space efficiently. There are two types of `replace` actions: replacing a property with another within an element and replacing the role of an element with another. For the first case, users can use the `from` and `to` keyword to indicate the original property and the replacing property. For instance, converting a channel from color to size can be expressed as the rule in line 60–65 (Figure 5.4c). Second, authors often change the role of elements across the visualization structure, which requires an `option` to not be subordinate to the specifier. In that case, users can use a `to` keyword to indicate that this rule is changing the structural property. For instance, one can replace an axis for the field plan with a legend for the color channel (which is meaningful only when the color channel encodes the same field) by having a rule shown in line 67–74.

5.3.2.8 Swap

A `swap` action exchanges two entities (roles and encoding channels) while retaining their properties, which shortens two `replace` actions and helps avoid potential conflicts. While a `swap` action has the same option structure with a `replace` action, it can also use an array to indicate properties to be swapped. For instance, to exchange the color and size channels, one can have a `swap` action and an array-based `option` as shown in line 77–81 (Figure 5.4c).

5.3.3 Reusability of CICERO Expressions

Responsive transformation strategies differ in how well they generalize across visualizations. Sets of public-facing Web visualizations often appear together in a data-driven article and may share datasets, chart types, and style schemes, thereby facilitating transformation reuse. For example, the data filtering rule in line 5–10 of Figure 5.13 can be reused for other charts sharing the same dataset because it references the data fields (`year`, `forecasted_year`) directly. However, this rule cannot necessarily be reused on charts with different datasets. On the other hand, authors can reuse the partial axes transpose rule in Figure 5.6 for charts with a similar format regardless of the underlying dataset as the transformation is declared independently. The flexible specifier syntax of CICERO is designed to allow authors to express more reusable transformations. For instance, the transformation for adding axis values in line 35–38 of Figure 5.4c can be reused on neighboring charts to provide better consistency. Alternatively, one can express the same rule as

```
{specifier: {role: "vAxis"}, action: "add", option: {index: "odd"}}
```

to make the rule more generalizable by not making direct reference to the underlying data scheme. Expression reusability is a core attribute of CICERO that naturally supports sophisticated visualization authoring tools, such as recommender systems, which we discuss further in Section 5.6.1.



Figure 5.6: An example CICERO rule describing partial transpose. The bars are grouped by columns in the left view (before) and by rows in the right view (after). The entire set of transformations for this case (Aid Budget) can be found in the Section F.1.

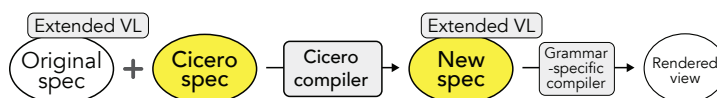


Figure 5.7: The pipeline for our CICERO compiler, developed for our extended version of Vega-Lite.

5.4 Principles for CICERO Compiler

To demonstrate the feasibility of CICERO and our proposed approach, we developed a compiler for our extended implementation of Vega-Lite. In the process, we identified ten principles we considered when implementing our CICERO compiler. As outlined in Figure 5.7, our prototype CICERO compiler takes as input a CICERO specification and a visualization design specification written in our extended Vega-Lite. Then, the CICERO compiler returns a transformed design specification in our extended Vega-Lite, which is eventually rendered by the compiler of our extended Vega-Lite. For each `transformation` rule, our compiler first selects an element(s) indicated by the `specifier`. If the element(s) exists, then the compiler applies the changes specified by the `action` and `option`. While developing the prototype compiler and deriving the principles below, we examined examples from prior work [37, 99] and considered how our compiler should deal with downstream effects to associated elements, the default behavior of a rendering grammar, and conflicting transformation rules. Future work can leverage our principles as useful semantics of the CICERO grammar when implementing custom CICERO compilers for other declarative visualization grammars. Appendix E details compiler APIs.

Extension to Vega-Lite

Our extended version of Vega-Lite provides a set of workarounds for public-facing visualization technique, such as text-wrapping and supplemental text (captions),

that are currently not supported in Vega-Lite [96], but were needed for our examples (e.g., external annotations). We use this extension to demonstrate the capabilities of CICERO for real-world use cases. The key differences from the current Vega-Lite are that our extension (1) uses trellis plot-based layouts [235] (rows and columns) instead of x and y encodings, (2) has many shortcuts to design techniques (e.g., wrapping text, map visualizations, interactive filters) for which Vega-Lite currently requires further specifications, and (3) supports richer communicative functionalities such as defining supplemental text elements like multiple subtitles or captions, creating graphical emphases that are not bound to data, allowing different formats of labels in the same axis, and so forth. The formal specification and description of our extended Vega-Lite are in Section D.2.

5.4.1 Associated elements

Visualization elements can have associations between them, which should inform how our CICERO compiler selects and handles the elements. For example, axis labels are dependent on the range of visualized data encoded by the x and y positions; hence, axis labels are associated with the ranges of visualized data values (line 15–21 of Figure 5.13). When a subset of data is omitted under a responsive transformation, then text annotations attached to the corresponding marks should be omitted as well (line 5–10 of Figure 5.13).

We describe two principles involving associated elements. First, our CICERO compiler **detects associated elements depending on how a user has defined the original design (P1)**. In the previous example (Figure 5.13), the two longer labels are declared as text elements of the line marks (i.e., tied to the marks in the same layer; `{type: "on-mark", field: "forecasted_year", items: [...], ...}`). Thus, filtering out a

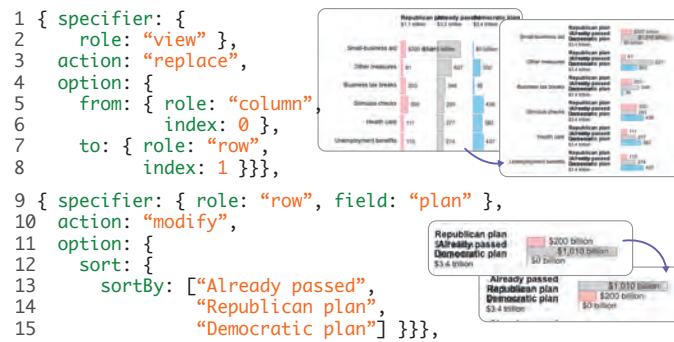


Figure 5.8: An example case (Aid Budget) for a downstream effect to the layout of elements (moving a column axis to a row axis; line 1–8) and applying a rule (reordering a nominal y axis) to the previously transformed view (line 9–15).

subset of data subsequently removes the corresponding marks and their associated labels. On the other hand, if the user has declared the text elements directly (without anchoring to certain data points), then the compiler should interpret them as independent elements that are not subordinate to any other element(s) or data.

Second, a transformation affecting the layout of a series of elements, such as adding, removing, or repositioning, has a downstream effect on the layout of their associated elements (P2), but not the static style. We do not allow downstream changes to style because the layout of one element and the static style of another are not meaningfully related whereas the relative layout between different elements does have a meaningful relationship. In the previous example, filtering out data points should not impact any independent, non-data annotations but should remove any associated text element(s). Similarly, converting a field from the column to the row of the chart (partial transpose) should move the axis labels (defined as `{type: "on-axis", field: "plan", items: [...], ...}`; i.e., tied to the axis of the plan field) for the field accordingly (see line 1–8 in Figure 5.8), but should not have side effects to the other properties—like the font weight or font size—of those elements.

5.4.2 Default behaviors

Declarative grammars often have default behaviors to make it easier to create a visualization. For example, Vega-Lite automatically generates legends and axis labels as a user declares color/size and position encoding channels. In compiling a CICERO specification, we were able to relatively easily reason about default behaviors regarding removing, modifying, and externalizing actions (e.g., “modify only what is specified” as a general software quality guideline or “externalize annotations at the bottom of the chart unless specified otherwise” based on our examples). However, adding a new element and internalizing an element can complicate the compile process, particularly when a user has underspecified the behavior. For example, when a user adds a new text annotation in the chart without specifying its position, then it is unclear how our CICERO compiler should behave. To guide such complex situations, we used a set of high-level default behaviors for our CICERO compiler.

First, when adding a new element to a series of elements, its appearance should **mimic the existing elements in the series (P3)**. For example, line 7–9 in Figure 5.9 adds new values for a vertical axis, resulting in newly added grid lines and labels. Then, they should look similar to the existing grid lines and labels without further specifying their appearance. Our CICERO compiler performs this addition by including those values in line 9 to the axis label and grid component in the specification (i.e., `{type: "on-axis", values: [100, 200, 300], ...}` → `{..., values: [50, 100, 150, 200, 250, 300], ...}`).

Second, our CICERO compiler **considers the appearance of elements in a similar role for new elements that are not part of an existing series of elements (P4)**. For example, when adding labels to a y axis that has no existing labels, although



Figure 5.9: An example use case (Disaster Cost) for our CICERO compiler’s default behavior for replacing the title as an internal annotation (line 1–6) and for introducing newly added elements (axis labels and grid lines; line 7–9).

they are not in the same series, it is more sensible to set their appearance similar to the labels on the x axis rather than the default presets of the rendering grammar. The similarity of the role between two series can be determined by whether they can be specified as the same `role` keyword (e.g., `{role: "axis.label"}` can specify both `{role: "hAxis.label"}` and `{role: "vAxis.label"}` if they both exist). Then, our compiler reuses the appearance attributes of the similar series of elements.

Third, when there are multiple series of existing elements, our CICERO compiler selects the one with the most similar structure (P5). As shown in Figure 5.10, for instance, when adding a new label to an axis that has two groups of existing labels in different styles, our CICERO compiler reasons about which of the two groups is most similar to the new label. We use the number of subelements (e.g., text segments) and the format of elements to find the most similar series of elements. For our approach, the compiler first identifies the number of newly added text segments (two). The one starting with “Jan. 19 ...” has two segments with different styles, and the “Feb. 29” one has a single segment. Then, by comparing the numbers of segments, the compiler matches the two-segment one (“Jan. 19 ...”) with the new labels.

Lastly, we consider the case where the position and style of a newly added or

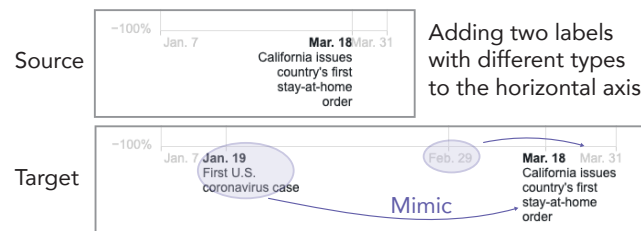


Figure 5.10: An example use case (Covid Spending 1) for our CICERO compiler’s treatment of multiple series of existing elements. In this case, our CICERO compiler adds new axis labels by mimicking the most similar type of the existing axis labels according to the number of subelements (text lines).

repositioned element cannot be fully determined because there is no existing series with a similar role. In this case, the compiler should leverage the following default behavior if not specified otherwise: as an overarching principle, **use the default options of the rendering grammar’s compiler (P6)** for newly added elements because users are expected to have some basic knowledge about how the rendering grammar behaves. For example, our extended Vega-Lite implementation does not automatically generate a legend for a new color scale, so our CICERO compiler for this extension similarly does not introduce a legend when adding a new color encoding. On the other hand, Vega-Lite’s default is to include a legend, so a CICERO compiler for Vega-Lite *should* add a legend. We had the following default behaviours for cases where the rendering grammar has no relevant default options based on our observations of common responsive design principles:

- Place (new) externalized annotations below the chart (see a4 in Figure 5.14).
- Place (new) internalized data annotations (or mark labels) at the center or the bottom of the associated data mark (see c4 in Figure 5.14).
- Place (new) internalized non-data annotations at the center of the largest contiguous empty space in the chart (see line 1–6 in Figure 5.9).

5.4.3 Conflict management

CICERO's list-based specification explicitly indicates the order of declared transformation rules. However, there are some cases where the order of rules may impact how the CICERO compiler interprets a given specification. Our compiler solves conflicts using the following methods, some of which are inspired by relevant CSS principles [236] that similarly deal with managing conflicts between ordered rule items. First, it may be confusing to select visualization element(s) in a `specifier` when other rules in the specification also transform the same element, which differs from general CSS use cases. For example, suppose there is a rule to transpose the x and y positions. This rule also results in swapping the horizontal and vertical axes as they are associated with the x and y position encoding channels. If a user wants to make some design changes in an axis that is the horizontal axis after transposing but is the vertical axis before transposing, defining a specifier for this rule might be confusing. A simple approach defaults to always specifying what is in the original view specification or what will appear in the transformed view. However, the former may not be useful for cases like making further changes to a newly added element, and the latter might make it difficult to compose a specification by requiring users to imagine the outcome status. As an overarching principle, our compiler **applies the current rule to a view that has been transformed by the previous rules (P7)** (e.g., line 1–8 and line 9–15 in Figure 5.8). This approach also implies that the compiler **applies the last declared rule (P8)** when there are two rules making changes to the same element for the same property, which is also a common practice with CSS specifications. Our compiler handles this principle by updating the target view specification for each transformation rule.

<pre> (a) More specific 1 { specifier: { 2 role: "mark", 3 datum: { 4 cat: "Apparel" }, 5 action: "modify", 6 option: { 7 color: { 8 value: "red"}}}} </pre>	<pre> (b) Less specific 1 { specifier: { 2 role: "mark" } 3 action: "modify", 4 option: { 5 color: { 6 value: "gray"}}}} </pre>
--	---

Figure 5.11: Rules to change the color of marks (a) by specifying the mark for the “Apparel” category and (b) by generally changing the color of all marks (independent of the data).

Next, our compiler assigns **higher priority to a more specific rule than a more generic rule for the same element (P9)** (note: not the same specifier)³. Here, the more attributes a `specifier` has, the more specific the rule is, inspired by CSS principles [237]. For example, suppose a user wants to change the color of a mark for the “Apparel” category (rule (a) in Figure 5.11) as well as changing the color of all bars (rule (b)). Here, the mark for “Apparel” is affected by both rules. Therefore, we recommend that generic color changes to other bars should not be applied to the mark for the “Apparel” category (i.e., rule (a) has higher priority than (b)). If a user does not want to apply a specific change (e.g., the custom color for the “Apparel” mark), then the user should omit the rule from the CICERO specification. Lastly, to enhance the degree of freedom in indicating the priority of rules, CICERO provides an `important` property for the same specifier, inspired by the `!important` keyword in CSS [237]. **Rules with the `important` property set to `true` have higher priorities than others (P10)** (i.e., compiled at the end). For example, a rule that changes the color of every axis label with `{important: true}` overrides another following rule that recolors a specific axis label⁴.

³See the ‘Justice Kennedy’ case (desktop to mobile) in the online gallery via <https://see-mike-out.github.io/cicero-supplemental/kennedy-desktop.html>.

⁴See the ‘Disaster Cost’ case (desktop to mobile) in the Section F.2.

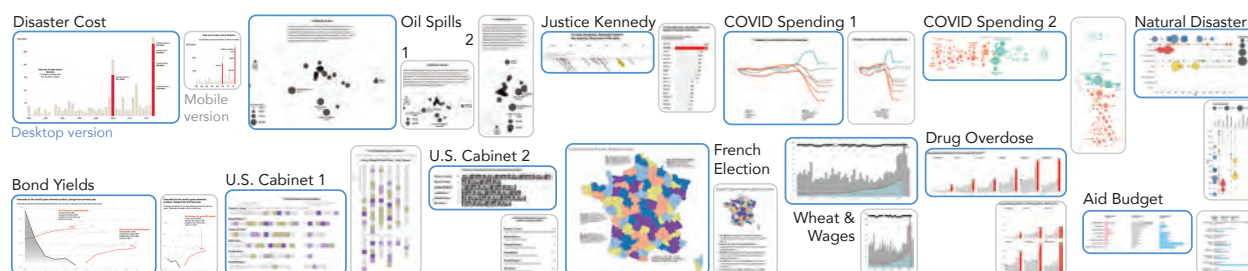


Figure 5.12: Thirteen responsive visualization use cases reproduced using CICERO. The blue- and gray-bordered views are the desktop and mobile versions, respectively. The mobile versions of the Oil Spills case are from (1) the original article and (2) the version suggested by Hoffswell et al. [99]. Full size images are included in the online gallery (<https://see-mike-out.github.io/cicero-supplemental/>).

5.5 Reproducing Real-World Examples

To demonstrate the expressiveness, flexibility, and reusability of CICERO and illustrate the above principles of our CICERO compiler, we present an in-depth walk-through of a mobile-to-desktop example (Bond Yields) using our extended version of Vega-Lite as the rendering grammar. We have twelve additional real-world inspired walk-through specifications that show the responsive changes step-by-step and two other detailed textual walk-throughs in the Appendix F that exhibit a variety of other transformations to visualization elements (i.e., data, marks, axes, title, labels, annotations, informational marks/emphasis, interaction, etc.) for both desktop-to-mobile and mobile-to-desktop transformations. The total of 13 example use cases includes three from Hoffswell et al. [99], four cases from Kim et al. [37], three recent responsive visualization cases (in our extended Vega-Lite), and two additional cases from the Vega-Lite example gallery that were not originally responsive but demonstrate the generalizability of our CICERO specifications to refine complex source views (in Vega-Lite). All 13 cases are listed in Figure 5.12 and provided in the online gallery (<https://see-mike-out.github.io/cicero-supplemental/>).

5.5.1 A Walk-through Example: Bond Yields

The Bond Yields example⁵ visualizes changes to both the actual and forecasted GDP growth rates over time. In the desktop version (Figure 5.13), the x position encodes the year from 2010 to 2021, and the y position indicates the GDP growth rate from 3.0 to 5.5. The area mark and black line mark represent the actual GDP growth rate from 2010 to 2015. The red and gray lines represent the five-year forecast of GDP growth rate for each year from 2010 to 2016; for example, the leftmost red line shows the estimated GDP growth rates for 2011 to 2015, as forecast in 2010. Transformations to produce the mobile version include (1) reducing the chart size, (2) removing the data points and labels for the forecast year of 2010 and 2011, (3) omitting the area mark, (4) truncating the y axis, and (5) repositioning an annotation. The CICERO spec is shown in Figure 5.13.

First, to resize the chart for a mobile phone, one can apply a `modify` action to the entire `view` (line 2–4). The `option` object indicates the `size` of 365 (width) \times 450 (height) to ensure that the chart fits a smartphone without requiring horizontal scrolling. Alternatively, one can use `{width: 365, height: 450}` in the `option`. Then, line 5–10 filters out (`remove`) the specified data points to simplify the view by reducing the information density. The `data` keyword in the `specifier` means `<year \leq 2011 (for the actual GDP growth rate) OR forecast year \leq 2011 (for the forecast)>`. Filtering out the data points removes (1) the two simple line marks for the forecast year of 2010 and 2011, (2) the data annotation for forecast year 2010, and (3) the corresponding parts of the area and black line mark for the actual GDP growth rates because each

⁵<https://www.wsj.com/graphics/how-bond-yields-got-this-low/>

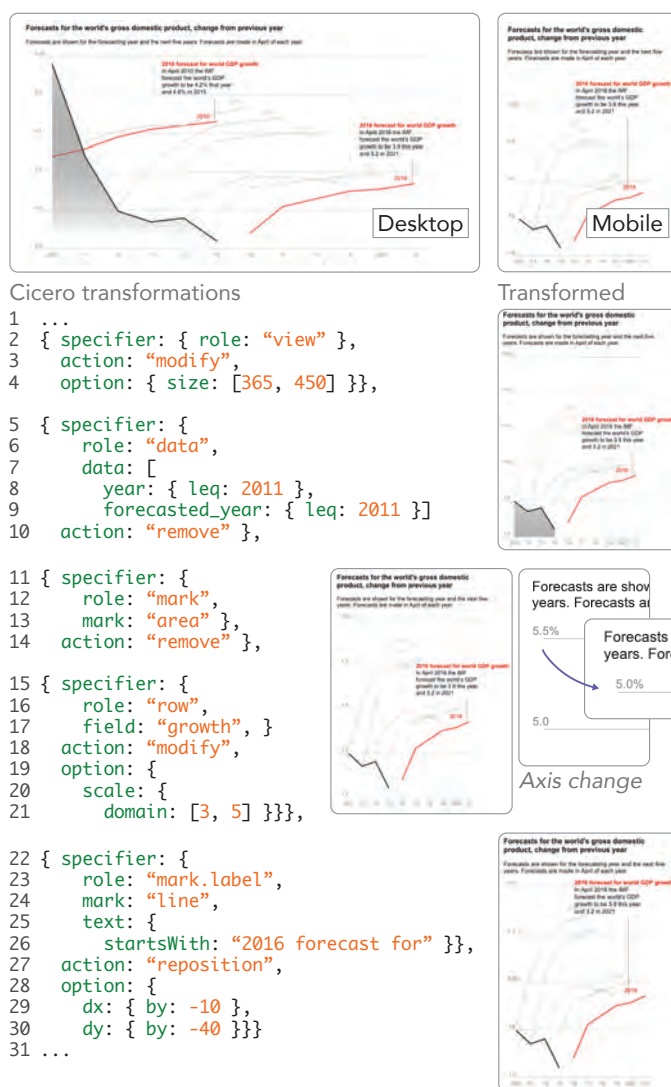


Figure 5.13: A walk-through example case of Bond Yields from a desktop version (top left) to a mobile version (top right). Starting with the desktop version, we first resize the chart to fit to a mobile screen (line 2–4), remove a subset of data for earlier years (line 5–10), remove the area mark (line 11–14), update grid lines by rescaling the domain of the y position channel (line 15–21), and reposition the annotation (line 22–30).

of these elements is associated with the filtered data (P2). This association is determined by the original visualization structure; if the annotations were declared as non-data elements, then the annotation for the 2010 forecast would remain (P1).

The `remove` transformation in line 11–14 omits the area mark specified by the

`mark` keyword. After filtering the earlier data, there is wasted space along the y-axis that unnecessarily compresses the data. To address this issue, the rule in line 15–21 changes the scale domain of the row field (`growth`) to `[3,5]`, resulting in the removal of the axis label and grid line for 5.5; the remaining elements automatically adjust to fill the newly vacated space (**P6**).

Lastly, the `reposition` rule in line 22–30 moves the mark label. Because there are many text elements associated with data marks (e.g., year names for each line), a specific `text` query is needed to select the label to move. For this rule, one can use the `startsWith` operator (line 25–26) to select elements with text starting with the specified string. Then, the `option` object changes the relative horizontal and vertical position (`dx` and `dy`, respectively) using the `by` operator which adds the specified value to the original value (i.e., moving the element by 10px left and by 40px upward).

5.6 Potential Applications for CICERO

Declarative grammars are particularly valuable for their utility in applications like visualization recommenders and authoring tools. In particular, they can function as a common representation method for different intelligent tools with similar purposes [238]. Visualization systems often use their own “internal representation” methods for their specific purposes [238]. Suppose we have two recommender models for different parts of a visualization (e.g., one for chart types and the other for annotations and emphases) that use heterogeneous representation methods. If they are translated to CICERO, then their recommender outcomes could be effectively combined to a user-side application. In this section, we describe how we used CI-

CICERO to represent a design space of responsive transformations in a prototype design recommender for responsive visualization as a proof of concept. We further discuss how CICERO might support mixed-initiative authoring tools.

5.6.1 Responsive Visualization Recommender

As a case study for potential applications for CICERO, we developed a recommender prototype for responsive visualization transformations using Answer Set Programming (ASP), which represents knowledge in terms of facts, rules, and constraints [162]. Our recommender takes a source visualization specification expressed in our extended version of Vega-Lite along with configuration preferences (e.g., intended screen size, strategies that a user wants to avoid, and a subset of data that can be omitted) which could hypothetically be provided by a user. Our recommender is intended to provide a diverse set of recommendations rather than showing several “optimal” visualization with slight differences. We encoded a set of common responsive visualization strategies motivated by prior work [37, 99] in ASP. Given the inputs and encoded strategies, Clingo [206], an ASP solver, generates a search space of responsive transformation strategy sets (corresponding to responsive visualization designs). To rank these strategy sets, we encoded heuristic-based costs that apply to individual strategies, and normalize and aggregate these costs to rank strategy sets representing design alternatives. We implemented three types of costs that apply to individual strategies: “popularity” costs based on the frequency of the strategy in prior analyses of professionally-designed responsive visualizations [37, 99]; density costs, where strategies that reduce information density are assigned lower cost than those that do not in a desktop-first pipeline, and vice versa in a mobile-first pipeline; and message preservation costs, where strategies (e.g., axis transpose, dis-

proportional rescaling) are assigned costs based on the extent to which prior work proposes that they affect the implied “message” of a visualization (c.f. Chapter 4).

In this pipeline, each recommended strategy set in the ASP format (e.g., `do(transpose_axes).`) are translated to a CICERO spec (e.g., `{specifier: {role: "view"}, action: "transpose"}`). While inference engines or models (e.g., ASP, ML classification, etc.) often employ their own abstract expressions for computational purposes, systems need to translate such abstract expressions (e.g., to JavaScript, Python, etc.) before utilizing them. For instance, ASP can efficiently perform propositional logic problems, but the ASP format cannot be directly used to perform actual tasks without translation. In the context of responsive transformation, directly using ASP codes to transform a visualization design specification (i.e., running JavaScript codes for each ASP code) is likely to complicate the translation with the lack of modularization. For example, whenever a recommender adds a new transformation strategy, the system has to look at every detail of different use cases, and doing so may not be consistent with the existing transformation strategies. This inconsistency in turn makes it more difficult to debug and extend the recommender. Instead, if we can translate those abstract transformations to systematic expressions like the CICERO grammar, then implementing recommenders for responsive visualization only needs to focus on generating a search space by modularizing the translation process. This process is similar to how Draco translates ASP expressions to Vega-Lite [96] and then renders a visualization [148].

Below, we illustrate example recommendations (Figure 5.14a) using our walk-through example (Section 5.5). We provide further details on our prototype recommender implementation, and describe example recommendation cases below and

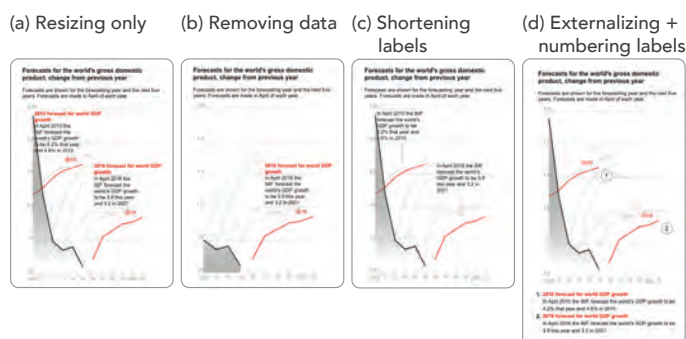


Figure 5.14: Selected examples among top seven recommendations for Bond Yields case from desktop to mobile. The original design is shown in Figure 5.13.

in the online gallery (<https://see-mike-out.github.io/cicero-supplemental/>). We emphasize, however, that our goal in developing the prototype recommender is to demonstrate the feasibility of using CICERO in such an approach, rather than to argue for the specific implementation of the cost model we used. In other words, our recommender should be interpreted as a proof of concept of our approach, rather than as an ideal recommender.

5.6.1.1 Example: Bond Yields

To generate candidate mobile views for the Bond Yields case, we include in the configuration the target size of a mobile view and a subset of data that can be omitted (referring to the original design). The first recommendation (Figure 5.14a) is simply resized to the target size. For this change, our ASP recommender returns `do(set_width,365).` and `do(set_height,450).`, and these abstract descriptions are translated to corresponding CICERO rules:

```
{specifier: {role: "view"}, action: "modify", option: {"width": 365, "height": 450}}.
```

In the second recommendation (b), the suggested omission is applied, similar to the original mobile view except for the remaining area mark and axis value for 5.5%. Our ASP engine expresses the transformation in an abstract way (`do(add_filter,f0).`,

where `f0` is a pointer to the user-suggested data filter statement), and then it is converted to a proper CICERO rule:

```
{specifier: {role: "data", data: [...]}, action: "remove"}.
```

The data annotations for the forecast years of 2010 and 2016 are shortened by removing the first line (the red text) in the third recommendation (c). For this change, our recommender converts an ASP rule, `do(remove_text_line,t2,0)`. where `t2` is a pointer to the annotations (or mark labels), to a CICERO rule:

```
{specifier: {role: "mark.label", field: "forecasted_year", index: 2},
  action: "remove", option: {items: {index: 0}}}
```

The fourth recommendation (d) externalizes the same data annotations below the chart with numbering for reference to the data marks. For this transformation, ASP rules, `do(externalize,t2)`. and `do(numbering,t2)`., are translated to a CICERO rule:

```
{specifier: {role: "mark.label", field: "forecasted_year", index: 2},
  action: "modify", option: {external: true, number: true}}
```

If the ASP rules were not compiled into our modularized CICERO grammar, the required changes to the original visualization specification would need to directly dissect many different parts of the specification, such as data, annotations, and axes. By modularizing this computation, CICERO can provide a more systematic representation of this change, which makes it easier to extend and debug our recommender.

5.6.1.2 Generalizability for Recommenders

CICERO can enhance modularization of responsive visualization tools by connecting tool-specific expressions and visualization grammars. For example, our recommender prototype uses ASP [162] to encode expressions with the Clingo solver [206]), and the CICERO compiler connects recommendations expressed in ASP to visualiza-

```

1 // A0&2: decrease X axis range
2 { specifier: { role : "view" },
3   action: "modify",
4   option: { width: 375 }}
5
6 // A9: decrease font size
7 { specifier: { role : "text" },
8   action: "modify",
9   option: { fontSize: { prod: 0.8 }}}

```

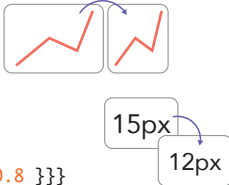


Figure 5.15: Expressing transformation strategies of MobileVisFixer [93] in CICERO. Line 2–4: decreasing the range of the x axis by reducing the width of the chart. Line 7–9: decreasing the font size using `prod` keyword.

tions in our extended Vega-Lite. Future work might start to leverage CICERO with machine learning-based recommenders. For instance, CICERO can express reusable transformation rules in MobileVisFixer [93] that translates non-responsively designed visualizations to mobile views. As shown in line 2–4 of Figure 5.15, CICERO expresses ‘reducing the range of x axis’ by expressing the change to the chart width (e.g., 375 pixel for mobile screens). Using the `prod` keyword in line 9, one can express reducing the font size of all the text elements relatively. In Appendix G, we provide a list of reusable CICERO expressions for MobileVisFixer [93] rules of which the meanings are clearly defined.

5.6.2 Mixed-initiative Authoring Tools

Users of visualization authoring tools may prefer different levels of customization and automation [239]. Tools like Microsoft Power BI [113], which automates design recommendations by converting a source visualizations using a set of default strategies, allow quick visualization creation, but can limit design expressiveness. In contrast, while the prototype proposed by Hoffswell et al. [99] and DataWrapper [24] do not have automated recommendation features, they enable more customization in making responsive designs.

Mixed-initiative authoring tools can provide a balance of automation and cus-

tomization capabilities, by allowing authors the ability to make manual responsive transformations or accept recommender-suggested transformations. Mixed-initiative authoring has been applied in exploratory data analysis settings (e.g., Voyager [151] and Dziban [164]) and dashboard design (e.g., LADV [165]). While our prototype recommender takes as input a representation of users' preferences, a next-generation authoring tool might aim to reason about responsive transformations that the user makes so as to recommend further or alternative transformations. For example, imagine creating the Bond Yields case (Section 5.5.1) without filtering out the subset of data. After resizing, the target visualization might look dense (Figure 5.14a1) although it maintains more takeaways compared to the actual design. Then, a user might decide to externalize the annotations instead of removing data. Following this manual change, a mixed-initiative authoring tool might suggest numbering the externalized annotations to support finding data references.

A mixed-initiative approach stands to reduce computational complexity by looking at the current state of edits rather than reasoning over a larger space of transformation combinations. Within a mixed-initiative authoring pipeline for responsive visualization, CICERO can be used to represent both system-recommended transformation strategies and user-driven manual edits, which can make such systems easier and more efficient to handle different sources of transformations (system and user). In addition, when an author updates the source visualization, CICERO can be used to reapply previous rules that are generalized to the updated chart (i.e., rules with the specifiers that can make queries from the updated chart). In Chapter 6, we introduce a mixed-initiative authoring tool for responsive visualization that is backed by CICERO.

5.7 Limitations

While CICERO and the CICERO compiler for our extended version of Vega-Lite can reproduce real-world use cases that represent a diverse set of transformations, future work should apply CICERO and future CICERO compilers to a bigger set of use cases to improve them and further extend the expressiveness of the grammar. For example, future work might focus on expressing complex user interactions (e.g., pan+zoom for a 3D visualization) with `specifiers`, inspired by declarative grammars for interactive visualizations (e.g., `trigger`, `signal`, and `event streams` in Vega [147, 240]), to better facilitate the application of such technologies to Web contexts where they have largely been underutilized [37, 99]. Another interesting future direction could be expressions for bounded dynamic behavior—the sizes or arrangement of elements dynamically change up to a certain limit, such as `max-width` and `flex-wrap` in CSS—in `options`. As it is a Web browser that implements CSS specifications, additional expressions for bounded dynamic behavior will be useful only if a rendering grammar supports such behavior. Furthermore, new design and evaluation studies for intelligent responsive authoring tools with CICERO might be useful to extend both CICERO and prior approaches in responsive visualization tooling [24, 37, 93, 97, 99, 109, 112].

Next, to demonstrate the full potential of CICERO in Web-based communicative visualizations, we chose to implement an extended version of Vega-Lite that can more easily express common techniques for narrative visualizations, such as externalizing annotations and applying word wrap to text labels. These capabilities are not straightforward to implement in Vega-Lite [99], so the resulting capabilities of a CICERO compiler for Vega-Lite may likewise be limited in what can be

expressed in rendered visualizations. As such grammars continue to develop, the corresponding compiler can be refined to support additional responsive functionalities. Furthermore, future work might need to apply these techniques to a larger class of declarative systems, such as extensions based on `ggplot2` [26] or `Vega` [147], to efficiently implement the corresponding CICERO compilers with a better understanding of their capabilities.

Finally, a CICERO specification defines a set of transformations to create a single responsive version and itself is not intended for direct rendering. As multiple responsive versions are necessary for different device types, an authoring system could bundle multiple CICERO specifications as a family using the `metadata` object in the specifications to decide when to apply each of them.

5.8 Conclusion

Authors often create responsive visualizations by transforming a semi-finalized design into another version. Motivated by this common practice, I proposed CICERO, a declarative grammar for responsive visualization design transformations. By decoupling expressions for design transformations from those for design specifications, CICERO disambiguates where to indicate transformations for multiple visualization elements and makes it easier to reuse the same transformation strategies. CICERO can express a variety of responsive visualization design techniques as demonstrated by reproducing diverse real-world cases. In the next chapter, I use CICERO as a core design representation to streamline automated design suggestions and custom human edits so that authors can seamlessly switch between automated and manual design modes.

Chapter 6

DUPO: a Mixed-initiative Authoring Tool for Responsive Visualization

As noted earlier, authoring responsive versions can be tedious, often requiring additional design iterations. A common approach is to (semi-) finalize a larger design (e.g., for desktop) and then transform it for smaller screens (e.g., tablets, smartphones), or vice versa. This process often involves making substantial design transformations beyond simple resizing [37, 99]. Authors may try to simplify this process by falling back on design strategies they have used in the past, even if they are not optimal [10]. Even authors who are well versed in a variety of responsive techniques typically face a time-consuming process of manually testing and evaluating designs one by one.

Consequently, some recent work proposes automated approaches to responsive design (e.g., Wu et al. [93] and Chapter 4) or visualization retargeting [50, 85]. Nevertheless, in many circumstances authors may want to more flexibly switch between agency and automation, such as newsrooms and public data reporting. While the ability to automate certain edits, like repositioning legends and rescaling size encodings, may be widely useful, authors may still need to manually edit the vi-

sualization in certain ways that are hard to effectively automate, such as rewriting text annotations to be more concise. A mixed-initiative approach provides flexibility by allowing users to accept automated recommendations from a system and/or make their own manual edits, and has been applied in a number of visualization systems [152, 164–166] and design tools [158, 159, 167].

We designed and implemented DUPO, a mixed-initiative authoring tool where users can create responsive communicative visualizations both through custom edits and automated assistance with flexible artboard management. The goal of DUPO is to support authors in exploring and reasoning about different responsive versions by making it quicker and easier to prototype design alternatives. DUPO implements a two-step design recommendation pipeline (i.e., *Exploration* and *Alteration*) to support design exploration with reduced redundancy. Users can first retrieve and explore more significant responsive design suggestions after or while creating a version of a visualization, such as changes to the layout, encoding choices, and data transformations like (dis)aggregation and filtering (*Exploration*). For each design recommendation, users can request further suggestions with more subtle changes to axis labels, legend position, etc. (*Alteration*). Users can also manually change the responsive versions at any time while using DUPO. The graphical user interface of DUPO supports flexible edit propagation not only across multiple artboards for manual edits [99] but also between users' custom edits and the system's automated suggestions.

We evaluated DUPO with six experienced responsive visualization authors who brought their own prior visualization designs to our study in order to explore new responsive versions using DUPO. Our participants overall believed DUPO could ben-

efit their day-to-day responsive visualization tasks by enabling rapid and high-fidelity prototyping of possible responsive design alternatives. Participants were able to seamlessly make custom edits to the automated designs, demonstrating that DUPO can support progressive mixed-initiative authoring. We observed that DUPO supported participants in exploring different suggestions and reasoning about them as inspiration for future designs. We finally discuss research opportunities to apply our mixed-initiative approach to other responsive visualization methods, such as template-, programming-, and graphic-based creation, and outline next steps for future software for responsive visualization within adaptive systems.

6.1 Usage Scenarios

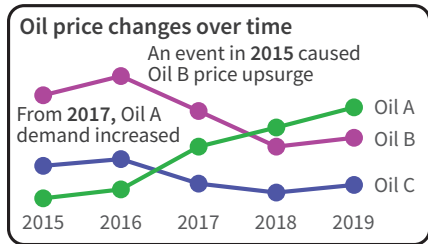
To motivate our design considerations for mixed-initiative responsive visualization authoring tools, we propose two usage scenarios in which a visualization author uses a mixed-initiative tool, based on prior formative findings [37, 99] and recommender approaches [93].

Desktop-first, novice author

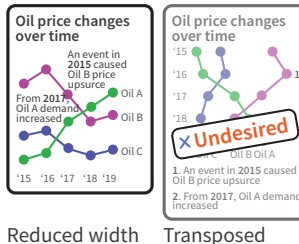
Riley (Figure 6.1 **A**) is a novice visualization author with limited experience in creating responsive designs. Riley uses a desktop device during the development process, and thus decides to start by designing a desktop visualization. After roughly finishing the desktop design of a line chart about oil price changes over time **A1**, Riley explores mobile versions suggested by the mixed-initiative tool **A2**. When examining the recommendations, Riley notices that some of the suggestions violate a design guideline in Riley’s organization that discourages putting a temporal field on a vertical axis in a line chart. To filter that case out, Riley marks it as “undesired,”

A Riley (Desktop-first, novice author)

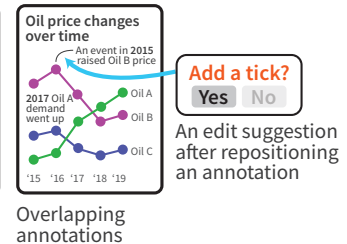
A1 Semi-finalized desktop version



A2 Explore mobile designs

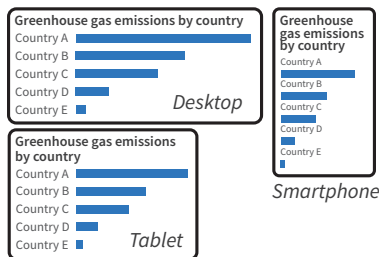


A3 Quick edit suggestion

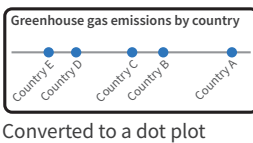


B Frankie (Simultaneous editing, experienced author)

B1 Initial responsive versions



B2 Explore desktop designs



B3 Propagate manual edits

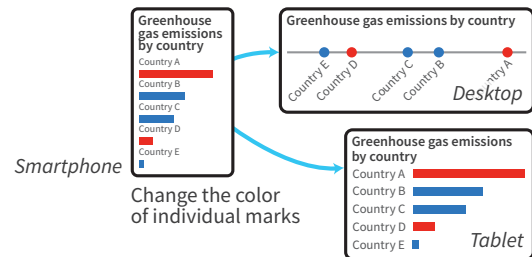


Figure 6.1: Two proposed usage scenarios for a mixed-initiative responsive visualization design tool. **A** A novice author creating responsive designs in a desktop-first manner. **B** An experienced author editing responsive artboards simultaneously.

so that the system does not recommend similar cases next time Riley retrieves design recommendations in this scenario. Riley thinks two versions, one with the reduced width and the other converted to a heatmap, seem useful. Riley selects both of them to discuss with their team. Riley realizes that some annotations are overlapping for these designs and decides to fix the position and width of annotations to avoid overlap and make them more readable **A3**. By doing so, one of the annotations is repositioned further away from the data point that it is indicating, so the tool suggests adding a tick between that distant annotation and the corresponding data mark. Riley accepts the quick edit suggestion.

Simultaneous editing, experienced author

Frankie (Figure 6.1 **B**) is an experienced visualization designer. Frankie prefers to create responsive designs simultaneously by managing multiple artboards to-

gether, which makes inspecting and comparing takeaways between versions easier. Frankie starts with artboards for desktop, tablet, and mobile, which are mandated screen breakpoints in Frankie's organization. After creating a horizontal bar chart about greenhouse gas emissions by country **B1**, Frankie realizes that the desktop version looks too wide and does not use the large screen space efficiently. Thus, Frankie retrieves some design recommendations using the mixed-initiative tool for the desktop version to reduce the time it takes to prototype different ideas **B2**. Frankie finds a one-dimensional dot plot that spreads the quantities (previously encoded by the bar length) along the same horizontal axis, which better uses the landscape aspect ratio of desktop devices without distorting the original takeaway about the rank of those countries. Frankie selects the dot plot version to further evaluate alongside the original bar chart. Frankie now wants to highlight two countries to indicate some important points **B3**. Thus, Frankie changes the color of the marks for those countries (in the mobile version) to be contrasting. This edit is propagated to the desktop and tablet versions.

6.2 Design Considerations

To motivate the design of our mixed-initiative responsive visualization authoring tool, DUPO, we derive the following four design considerations inspired by prior work and our usage scenarios (Section 6.1).

Support design exploration (DC1) A benefit of automation is quickly enumerating and ranking recommendations from a large design space so that users can efficiently explore many different options and avoid design fixation. To not overwhelm users with subtle differences, a recommender should minimize redundancy among its

suggestions [152, 158]. To support reasoning about the critical trade-off between preserving visualization messages and maintaining graphical density [37, 38], a recommender and its interface should employ methods capable of evaluating how well designs preserve important patterns or insights.

Support user control (DC2) Responsive visualization authors need to retain their agency by being able to control and understand the automation process [99]. In particular, users should be able to fine-tune the systems' recommendation methods (e.g., conveying which suggestions are less useful to guide future recommendation strategies). To support authors in making well-informed decisions about recommended alternatives, a mixed-initiative system should provide easy-to-interpret descriptions of what each suggestion is about and why it is suggested.

Support progressive authoring (DC3) Mixed-initiative systems should support an iterative design process that facilitates seamlessly switching between customization and automation [151, 152, 164–166]. To enable progressive authoring, users should be able to easily retrieve recommendations on demand, and recommended visualizations should be customizable as needed. In addition, a progressive authoring interface should support multiple ways to revisit and compare prior designs.

Support flexible artboard management (DC4) Users may have different preferences in creating responsive visualizations (e.g., desktop-first, simultaneous editing). Hoffswell et al. [99] provide four design guidelines for flexible artboard management. Users should be able to quickly preview and edit multiple artboards simultaneously as well as customize a specific artboard and propagate edits between designs. To support progressive authoring (DC3), such flexibility should also enable propagation between custom edits and automated design suggestions.

6.3 Recommendation Pipeline

To facilitate **design exploration (DC1)**, DUPO provides design suggestions using three pipelines: *Exploration*, *Alteration*, and *Augmentation*, as shown in Figure 6.2. For the main recommendation workflow, DUPO provides design suggestions in two steps: *Exploration* and *Alteration*. The *Exploration* pipeline (Figure 6.2 **C**) suggests design alternatives with high-level changes mainly to mark types, layout, data transformations, and encodings. The *Alteration* pipeline **D** suggests design alternatives with subtle, low-level changes to text elements (e.g., annotations, titles, and labels), tooltips, references, etc. The differentiation into these two steps aims to reduce redundancy during **design exploration (DC1)**, and enable fine-grained refinement of preferred recommendations (**DC2**).

DUPO also provides a simplified *Augmentation* pipeline **E** that recommends a single responsive transformation as a possible next step in response to manual user edits. These recommendations (also known as *quick edits*) encode commonly co-occurring design patterns from the prior responsive visualization literature [37, 99].

6.3.1 Exploration: High-level Transformations

6.3.1.1 Step 1: Input

The *Exploration* pipeline takes a source design and set of user constraints as input **C2** in order to provide **user control (DC2)** over the outcome of the recommendation pipeline.

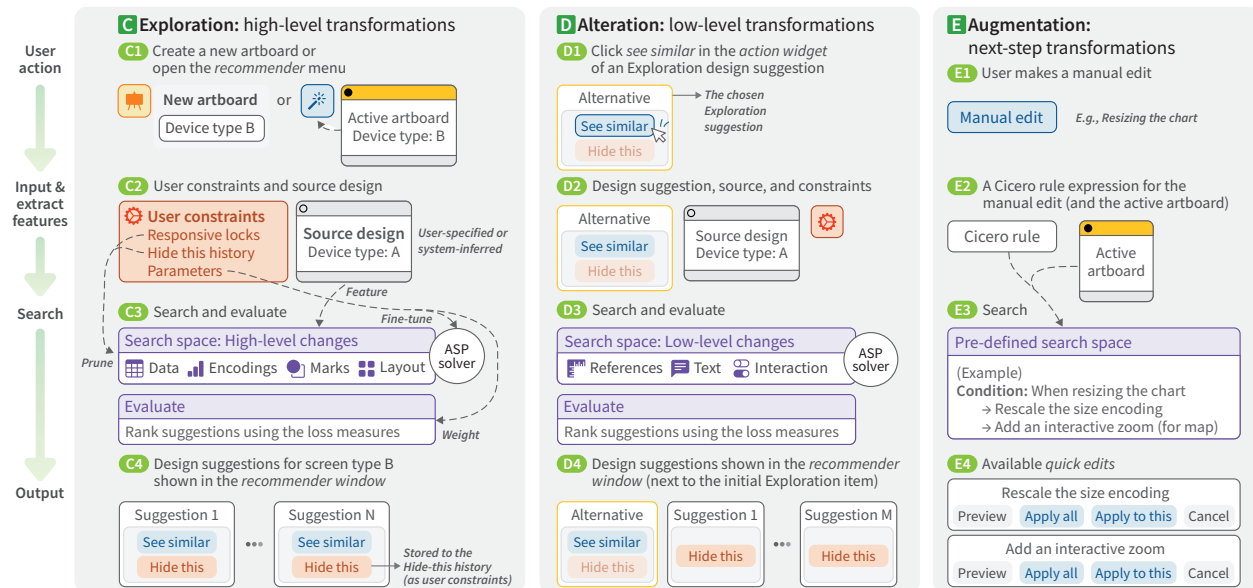


Figure 6.2: DUPO’s three recommendation pipelines: *Exploration*, *Alteration*, and *Augmentation*. Initiated by a user action, each pipeline gathers the input data, runs a search, and returns the output suggestions. **C** The user can run the *Exploration* pipeline to see high-level transformations (e.g., to data, marks, encodings, and layout) by pressing the *recommender button* from the *toolbar* or creating a new artboard **C1**. Then, DUPO takes the source design and user constraints as input **C2**. DUPO generates alternatives with high-level changes and ranks them **C3**. These design alternatives are presented in the *recommender window* **C4**. If the user clicks the *hide this* button of an alternative, DUPO stores the alternative design in the *hide this history* as a user constraint for next time the user runs the *Exploration* pipeline. **D** The user can run the *Alteration* pipeline to see low-level transformations (e.g., to references, text, interactions) by selecting *see similar* for an alternative design **D1**. DUPO takes the alternative of interest, the user constraints, and the source design as input **D2**. DUPO then populates alternatives for low-level changes and evaluates them **D3**. The *Alteration* suggestions then appear next to the initial *Exploration* design **D4**. **E** The *Augmentation* pipeline is initiated after a user makes a manual edit **E1**, at which point DUPO takes its CICERO rule as input **E2** and searches a pre-defined search space **E3** to suggest *quick edits* that are commonly applied with the manual edit **E4**.

Step 1a: Source design

DUPO uses the source design to determine inapplicable transformations while maintaining the underlying data characteristics (e.g., data type), aesthetic choices (e.g., encoding scales), and text contents. For example, if the source design is a pie chart, then transformation strategies for a bar chart (e.g., transposing) are omitted from the search space unless there is a change to the mark type. During the evaluation stage, ranking measures compare each suggested outcome with the source design in terms of how well they preserve the design information. The source design also helps maintain consistency, which reflects common responsive visualization authoring practices where authors apply transformations to an initial version to produce other versions, as described in the usage scenario (Figure 6.1 **A**).

Step 1b: User constraints


The *Exploration* pipeline leverages three types of user constraints as input: (1) *responsive locks* that provide explicit preferences about what elements cannot be updated; (2) *hidden recommendations* that provide implicit feedback about what transformations are undesirable; and (3) *recommender parameters* that provide abstract feedback on the outputs and weights applied by the recommender. First, the user can apply two types of *responsive locks*: *element-locks* and *position-locks*. An *element-lock* allows the user to specify elements they want to prevent from being removed and/or restyled. For example, if the user *element-locks* a legend, then DUPO ensures that all alternatives will include a legend, but still allows for repositioning the legend within the design. To prevent repositioning of the element, the user can specify a *position-lock*. We distinguish between an *element-lock* and a *position-lock* to enhance the degree of freedom for controlling the recommender given that authors do not always modify and reposition elements together [37]. Using these *responsive locks*,

DUPO prunes suggestions with the locked elements from the search space.

Second, when viewing recommendations in DUPO, the user can choose to *hide this* suggestion (i.e., mark it as undesired) to implicitly update the behavior of the recommender. DUPO then records the unique transformations of that hidden suggestion (i.e., the transformations that do not appear in any other recommendations at the time) and removes them from the search space for subsequent rounds of recommendation.

Third, the user can provide recommender parameters including the maximum number of suggestions (at each time of request), the weights for the ranking measures, and how “drastic” the transformations are. Here, non-drastic transformations are those that only apply to resizing and transposing; drastic transformations include those that update encodings and layout beyond transposing. For example, if the maximum number of suggestions is five and the drastic parameter is 0.6, then three suggestions will include drastic changes.

6.3.1.2 Step 2: Search space generation

When the *Exploration* pipeline is initiated, DUPO generates a search space  that combines predefined responsive transformation techniques with the user constraints outlined above. Technically, the search space consists of facts describing the user input, rules encoding the responsive transformations, and hard user-specified constraints, expressed using the ASP grammar [205]. DUPO uses Clingo, an ASP solver [206], to compute the search space for a set of suggestions. These suggestions are translated into CICERO specifications (Chapter 5) so that the CICERO compiler can convert the design accordingly.

These pre-defined techniques include high-level transformation rules for mark type, layout (rows and columns; small multiples), data transformations (e.g., filtering or aggregation), and encoding channels **C3**. When required for well-formedness, these techniques include changes to text elements (e.g., repositioning annotations when transposing the axes). Transformation strategies for the *Exploration* search space are motivated by several goals: minimize changes between responsive designs, avoid overplotting, fit to the new aspect ratio, and maintain graphical density (i.e., the white space and overlapping area). For each rationale, we encode transformation strategies observed in prior work [99] and previous chapters (Chapter 3 and Chapter 5). For example, DUPO suggests aggregation when the outcome device size is smaller than the source to avoid overplotting. We document these rationales and strategies in Section H.1.

Transformation strategies encoded in our search space are applied based on the device and responsive authoring direction (e.g., mobile-first, desktop-first). Chart resizing and transposing axes, for example, are the default strategies across those conditions. While DUPO suggests further *Exploration* suggestions, it only suggests those default transformations for tablets in a desktop-first direction unless requested by the user. DUPO suggests moving annotations out of the chart for desktop-first whereas it suggests moving titles into the chart for mobile-first.

6.3.1.3 Step 3: Ranking measures

After generating design alternatives from the search space and before returning them to the user, DUPO evaluates them using several ranking measures in comparison with their source design in terms of “message” and density **C3**. The ranking measures are motivated by a trade-off between preserving takeaways and adjusting

density in responsive visualizations (Chapter 4). We use the task-oriented loss measures for *identification*, *comparison*, and *trend* proposed in Chapter 4 to approximate changes to a visualization’s support for those tasks as a proxy for “message.” We added a *text* loss that estimates changes to the text elements (whether removed or changed), given their importance in communicative visualizations. For density, we included an *overplotting ratio* (as a proportion of the overplotted area when elements are overlapping) and an *occupation ratio* (as a proportion of the non-white space out of the entire chart area), inspired by the clutter reduction method proposed by Ellis et al. [241].

Provided as a way to express users’ different priorities, these six measures (four insights and two density) are combined into a single scalar value as a weighted sum. By testing them on example use cases, we heuristically determined the initial weights in a way that prioritizes trend insight. Users can also **manually fine-tune the weights** in the interface (DC2). We provide additional details in Section H.3.

6.3.2 Alteration: Low-level Transformations

The user can request *Alteration* suggestions for each *Exploration* suggestion (Figure 6.2 D1). The *Alteration* pipeline suggests low-level changes to text elements (e.g., title, annotations), references, tooltips, etc. that make relatively subtle modifications to the design. Given that some *Exploration* suggestions may already include low-level changes in order to ensure well-formedness, the *Alteration* pipeline takes the *Exploration* suggestion as input to prune duplicate transformations from the search space, in addition to leveraging the source design and user constraints D2. The search space encodes a variety of the low-level changes documented by prior work [37, 99] D3. For example, *Alteration* from phone to desktop suggests adding

labels for a quantitative axis, internalizing a short title, and adding a tooltip. Users can request *Alteration* multiple times for the same *Exploration* suggestion.

6.3.3 Augmentation: Next-step Transformations

Prior work [37] indicates that some responsive strategies are commonly applied together, so DUPO suggests *quick edits* that recommend next-step transformations for a manual user edit. As shown in Figure 6.2 E, whenever a user makes a manual edit E1, DUPO passes the CICERO rule representing the edit E2 to a pre-defined search space E3, and then suggests applicable *quick edits* in the user interface E4. To identify relevant suggestions, the *Augmentation* search space considers the direction of the edit and the intended device type. For instance, DUPO suggests a *quick edit* for fixing the tooltip position at the bottom when a tooltip is *added* (direction) for a *phone* version (device type). DUPO supports a variety of *quick edits* including adding or removing tick lines after repositioning annotations and moving the title text into the chart area when it is added for desktop or tablet versions, for example. We outline the entire set of *quick edit* rules in Section H.2.

6.4 User Interface

Based on our design considerations (Section 6.2), we implement DUPO, a mixed-initiative authoring tool for responsive visualization. DUPO supports designing communicative visualizations with common visual encodings, annotations, and interactivity (including zoom+pan, tooltips, a brush-based context view, and an interactive legend for selecting marks on hover). The following sections describe the key features of DUPO, as illustrated by the walkthrough in Figure 6.3; we provide

a user manual with a video demo detailing all of the functionality of DUPO and a walkthrough video online¹.

6.4.1 Overview

WALKTHROUGH: *Kris starts using DUPO by manually drafting an initial bar chart for a smartphone screen in a mobile-first manner* **F1**.

As shown in Figure 6.4, DUPO has two key components: an *editing interface* **G** for manually editing the visualization through pre-defined marks (with layers) and drag-and-drop interactions, and a *recommender window* **K** to present the responsive recommendations for the current version. The *editing interface* of DUPO consists of a *toolbar* **G1**, *editing panel* **G2**, *navigation bar* **G4**, and *artboard area* **G5**.

As the main control menu of DUPO, the *toolbar* offers functionalities for manual edits (e.g., updating properties of the *mark* or *layout*) and system features (e.g., *preview* and *export*). A user can access the menus by using the *toolbar*, pressing a keyboard shortcut, or double-clicking a chart element, which opens an *editing panel* specific to the chosen menu. As shown in Figure 6.4 **G**, for example, when the *mark* menu (**🔗**) is selected, the *editing panel* offers options for customizing the mark type and details (e.g., use a point on line marks) and mark property encoding channels (e.g., fill color or mark size). Following the drag-and-drop variable assignment in popular tools like Tableau Software, the user can drag a field from the floating *data widget* **G3** and drop it on the field form (or shelf). DUPO also enables limited direct manipulation for repositioning annotations directly from the *artboard area*.

¹<https://see-mike-out.github.io/dupo-supplementary/>

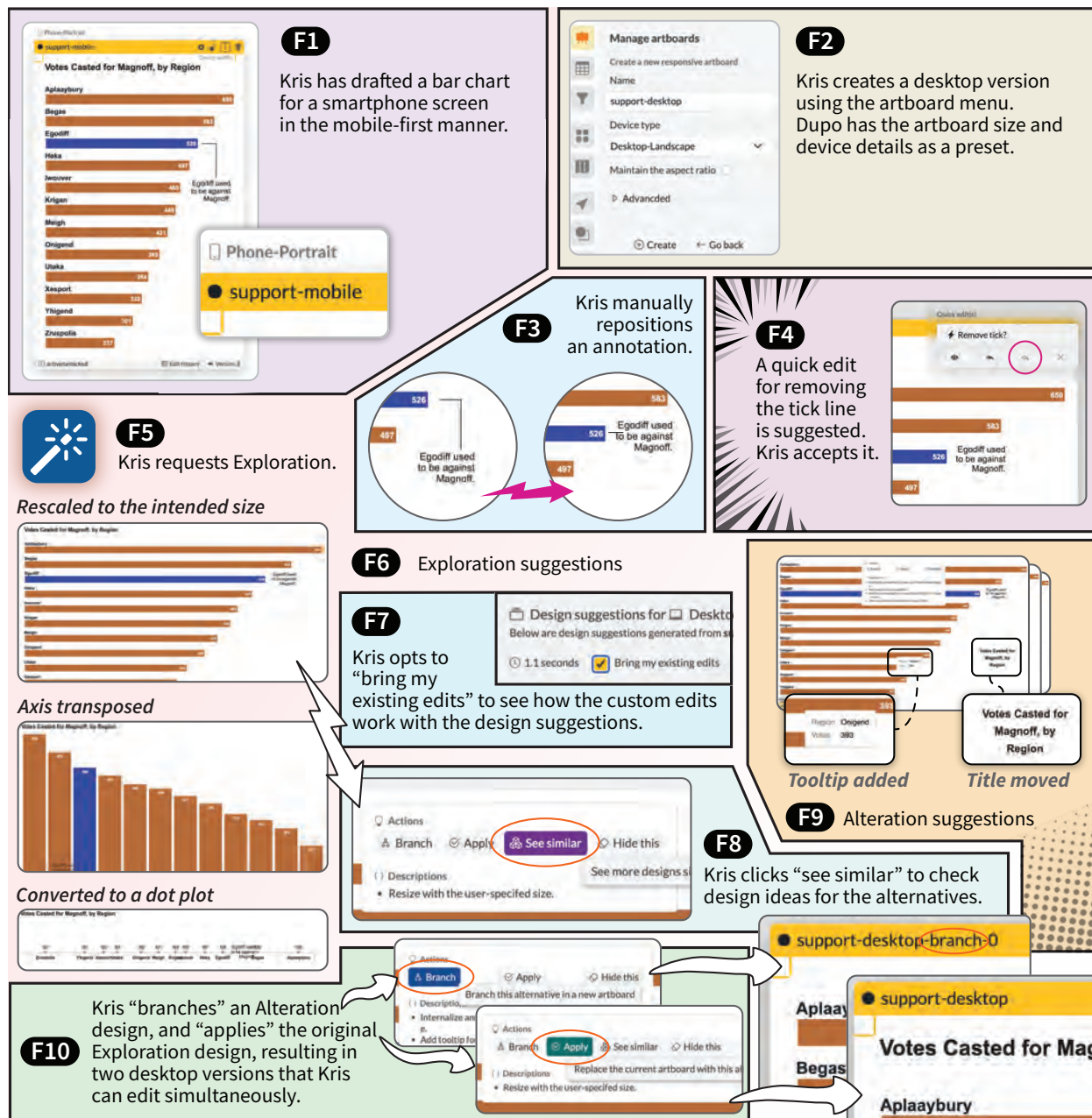


Figure 6.3: A walkthrough example for creating a responsive visualization using DUPO in a mobile-first manner. Kris starts by drafting an initial bar chart for a mobile phone, and then uses DUPO’s three recommendation pipelines to refine the design.

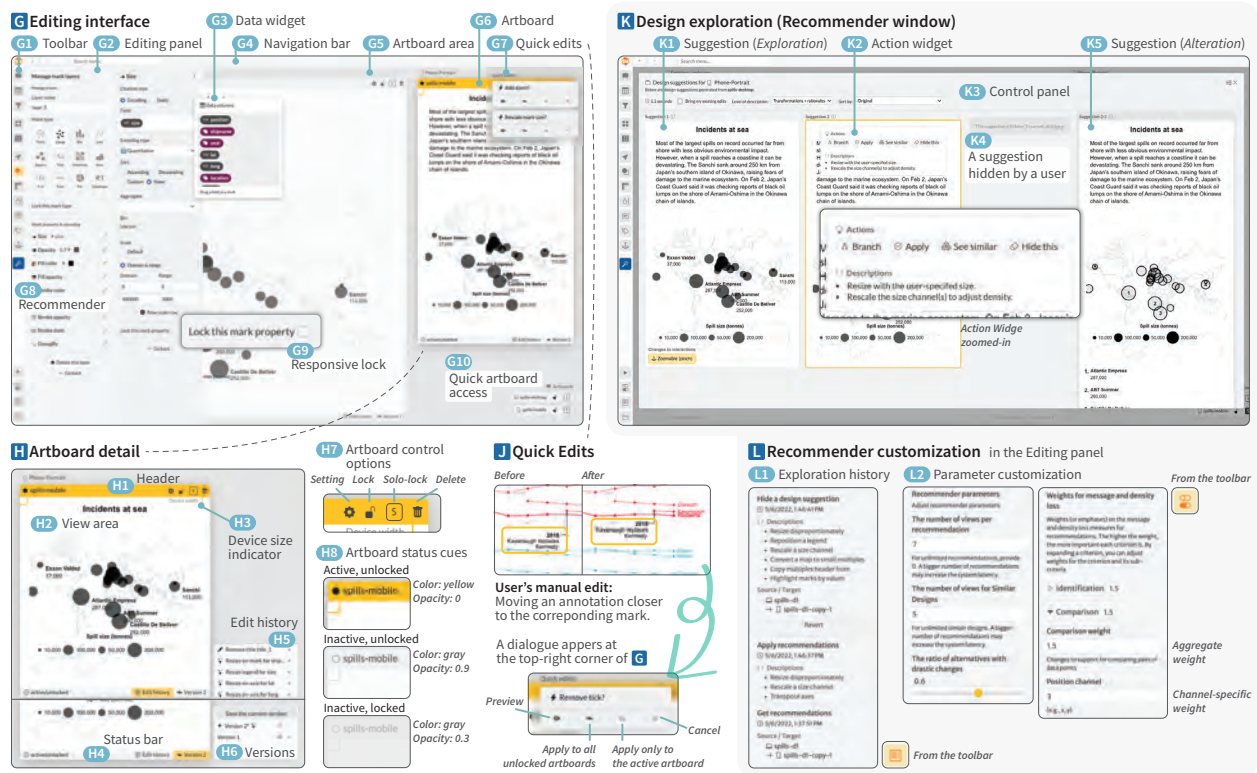


Figure 6.4: An overview of DUPO's interface. **G** By selecting the *mark* menu (🔍) from the *toolbar* **G1**, the user can edit marks (e.g., mark type and encodings) in the *editing panel* **G2**. To set an encoding channel, the user drags a data field from the *data widget* **G3** and drops it on the desired field. In the *navigation bar* **G4**, the user can undo or redo by clicking the arrow icons and search the system menus. The *artboard area* **G5** displays responsive *artboards* **G6**. The user can load suggestions using the *recommender button* **G8**, and set *responsive locks* on elements that the recommender must keep **G9**. At the bottom right corner, the user can *quickly access particular artboards* **G10**. **H** Each artboard consists of *header* **H1**, *view area* **H2**, and *status bar* **H4**. The *header* provides options to manage its *status* and *settings* **H7** and indicates the *activeness* and *lock status* **H8**. In the *view area*, the *device size indicator* **H3** helps the user to check if the content overflows the intended space. From the *status bar*, the user can access the *edit history* **H5** and *versions* **H6**. **J** Some manual edits by the user prompt DUPO to recommend associated *quick edits*, appearing on the top right corner of the *editing interface* **G7**. **K** The user has requested *design suggestions* **K1** generated for a smartphone version. From the *action widget* **K2**, the user can *branch* each suggestion as a new artboard, *apply* it to the current artboard, request further *Alteration* suggestions, *hide* a suggestion **K4**, and read the rationales for each suggestion. The *control panel* **K3** allows users to *apply their custom edits* to the suggestions and toggle the depth of descriptions in the *action widget*. **L** The user can review the *history* (📄) of their use of the recommender **L1** and customize the recommender parameters **L2** (⚙️) from the *editing panel* **G2**.

The *navigation bar* offers undo and redo options and a search form for quickly accessing relevant menus. At the bottom right corner, the user can access each artboard from the artboard list (*quick artboard access*) **G10**, which scrolls to the selected artboard in the *artboard area*.

6.4.2 Artboard Management

WALKTHROUGH: *Kris creates a desktop version using the artboard menu by selecting the “Desktop-Landscape” preset, which has a default artboard size and relevant device details already defined **F2**.*

The *artboard area* (Figure 6.4 **G5**) displays responsive artboards, and allows a user to **see and edit multiple artboards simultaneously (DC4)**, inspired by Hoffswell et al. [99]. An *artboard interface* **H** consists of a *header* **H1**, *view area* **H2**, and *status bar* **H4**. The *header* includes the name of the artboard and indicates its status **H8**, as well as providing buttons for artboard settings (e.g., artboard size and targeted device size), lock, solo-lock, and deletion **H7**. The *view area* **H2** of an artboard displays the responsive design. On the top and left offsets of the *view area*, *device size indicators* **H3** (i.e., the yellow ticks), analogous to rulers in graphical software, help the user to check whether the content overflows the screen of the intended device.

The *status bar* **H4** of an artboard provides access to the *edit history* and *version list* to allow authors to revisit previous designs (**DC3**). If a user clicks the *edit history* **H5** button, then they can see the list of manual and automated edits. To help users distinguish the source of edits for better overall **control (DC2)**, DUPO uses different icons: a pencil icon (✎) for manual edits, a cursor icon (↙) for direct manipulation, a star icon (☆) for edits from design suggestions, and a lightning icon (⚡)

for augmented quick edits. A user can undo each edit by clicking the cancel button (X). In the *version list* H6, a triangle icon (▶) indicates the current version, and the star icon (☆) denotes a version suggested by DUPO. When a user wants to compare the current version with a previous one, they can preview a version with the eye icon (👁). If the user wants to revert to a previous version, then they can “check out” a version with the check-mark icon (✓). The user can save the current artboard at any time as a new version in the *version list* to provide more **user control** over the state of the responsive versions (DC2).

Given that DUPO supports **edit propagation across multiple artboards** (DC4; c.f. [99]), it is important to enable the user to **control and see related status information** (DC2). Each artboard maintains an *activeness* status to indicate whether it is the source of changes for edit propagation (“from”) and a *lock* status to denote whether it is receiving propagated edits (“to”). An *active* artboard (which applies to only one artboard at a time) refers to the artboard that a user is making changes to. Each custom edit made to the active artboard is propagated only to the other *unlocked* artboards. The user can activate an unlocked artboard by clicking it, indicated by the yellow *header* color and in the *status bar*. The lock button in the artboard *header* H7 toggles and shows the lock status (locked: 🔒, unlocked: 🔓). To edit a single artboard without edit propagation, the user can click *solo-lock* to lock all the other artboards (S). The *status bar* H4, opacity, and header color H8 of an artboard cues the *lock* and *activeness* status at a glance.

6.4.3 Edit Augmentation (*Quick Edits*)

WALKTHROUGH: *Kris manually moves an annotation closer to the corresponding data mark for the desktop version* F3. DUPO then suggests a quick edit to remove the tick line

connecting them as their relationship looks clearer **F4**, which Kris accepts.

After a user makes a manual edit, DUPO suggests *quick edits* that are commonly combined with the user's edit for other responsive visualization use cases [37] (*Augmentation*), as shown in Figure 6.4 **J**. Applicable *quick edits* appear on the top right corner of the *artboard area* **G7**, with options to preview a *quick edit*, apply it to the current artboard or all the unlocked artboards, and cancel it.

6.4.4 Responsive Design Recommendations

As described in Section 6.3, DUPO distinguishes the primary recommendation workflow into a two-step approach: high-level changes to mark types, layout, data transformations, and encodings (*Exploration*) and low-level transformations regarding references, annotations, text elements, and more (*Alteration*). Responsive recommendations for the current *active* visualization are shown in the *recommender window* **K**.

6.4.4.1 Design Exploration

WALKTHROUGH: *To continue refining the design, Kris clicks the recommender button to request new Exploration suggestions for the desktop version **F5–F6**. Before exploring them in the recommender window, Kris first applies the previous edits **F7** to all the suggestions.*

After (semi-) finalizing an artboard for a certain screen type (i.e., after indicating a dataset, a layout, and a mark layer), the user can explore design alternatives by creating an artboard for a different screen type or by clicking the *recommender button* (***** in Figure 6.4 **G8**) in the *toolbar*. DUPO then opens the *recommender window* **K** consist-

ing of the *control panel* **K3** and design suggestions. The *recommender window* initially shows design suggestions for the current active artboard that consist of combinations of changes made to high-level visualization elements like visual encodings, layout, and data (i.e., *Exploration* suggestions). The order of the suggestions implicitly encodes DUPO's estimate of the effectiveness of the designs, as described in Section 6.3.1.3. To support **progressive authoring (DC3)**, DUPO lets the user apply the edits from the current active artboard (if applicable) to the design suggestions using the *control panel*. To do so, DUPO applies the user's manual edits (as Cicero rules) to the automated transformation rules so that those manual edits are displayed.

6.4.4.2 Interaction with Design Suggestions

WALKTHROUGH: *Kris finds an appealing recommendation for a resized bar chart, but is not completely satisfied with it. An action widget shows up as Kris hovers over the suggestion. Kris reads the description of the design changes, and clicks see similar to explore other Alteration examples **F8**. For example, one suggestion involves moving the title into the chart and adding a tooltip **F9**. After exploring design suggestions, Kris decides to branch an Alteration design as a separate version and apply the initial Exploration suggestion to the active artboard **F10**.*

When the user hovers on a suggestion, an *action widget* (Figure 6.4 **K2**) allows them to *load similar* design suggestions, *apply* the suggestion to the current artboard, *branch* the suggestion as a new artboard, and *hide the suggestion*. To help users understand what transformations were applied to the design and why, the *action widget* shows the list of the responsive transformations (e.g., resizing, encoding changes) applied to the suggestion. If the user clicks the *see similar* button in the *action widget*, DUPO shows new alternatives **K5** that include low-level changes to the text placement or

style (i.e., *Alteration* suggestions).

The *branch* and *apply* options allow the user to further refine **designs they explored (DC1)** and provide different ways to revisit previous designs to support **progressive authoring (DC3)**. If the user chooses to *branch* a suggestion from the *action widget* **K2**, a new artboard is created with the chosen design, thereby preserving the current artboard and allowing the user to compare and iterate on multiple versions of the design. If the user chooses to *apply* a suggestion, DUPO saves the existing design in the version history **H6** and updates the artboard with the chosen design. The user can then make manual edits to **progressively** refine the design **(DC3)**. When the user clicks *hide this*, DUPO immediately removes the suggestion **K4** and records the recommendation as one not to suggest again. The user can cancel the *hide this* behavior immediately or revert it from the *exploration history* **L1**, in order to better **control** the recommender behavior **(DC2)**.

6.4.5 User Controls

DUPO enables the user to **control the recommender (DC2)** in various ways, including applying *responsive locks*, reviewing the *exploration history*, adjusting *recommender parameters*, and *quick sorting*. The user can indicate visualization elements (e.g., marks or layout) that they want to prevent the recommender from changing (Figure 6.4 **G9**) within the *editing interface*. The user can review their interactions with the recommender in the *exploration history* menu (**☰**); this menu shows the user what suggestions they have applied, branched, or hidden, and allows the user to revert *hide this* decisions **L1**. The *preference* menu (**⚙**) allows the user to adjust recommender parameters **L2**. As a simplified fine-tuning method, the user can choose a sorting criterion as either identification, comparison, trend recognition, text content, or graphical

density from the *control panel* **K3**. In the *control panel*, the user can also adjust the level of detail in the descriptions (i.e., transformations only or transformations with rationales).

6.4.6 General System Features

DUPO supports common GUI-based user interaction for graphical software, such as color pickers and sliders. To preview responsive versions together, the user can use the *device preview* menu (▶) that rescales visualizations using the pixel per inch (PPI) value of each artboard as a way to quickly verify the output of their responsive views. DUPO lets the user *export* their artboards as an HTML file with media queries (⤴) that ensure that each artboard appears only for the specified browser size and/or aspect ratio (if provided). In addition, DUPO offers several general system *preference* options, such as switching device size indicators on and off, and changing the default artboard presets.

6.4.7 System Design Refinement

We refined the interface based on feedback from a pilot study with four visualization designers (see Appendix I for details). For example, DUPO originally had two ways to add a responsive artboard: using the *artboard* menu to create a blank artboard and duplicating an existing artboard with a new size. Users could specify the source design for the duplicated artboard as the original one, which was intended to enable users to specify source designs flexibly. However, our pilot participants were confused about having multiple options for the seemingly same task. To consolidate these options, by default we duplicate the source design (the earliest created artboard) when the user creates a new artboard in the *artboard* menu, rather than

requiring a separate pipeline to produce this behavior. We also provide an option to change the source design for the recommendation to retain the same flexibility. There were also originally separate buttons for high-level and low-level design suggestions; in the revised system, we moved the button for low-level suggestions to the *action widget* (Figure 6.4 [K2](#)).

6.4.8 Implementation Details

DUPO renders visualizations using Vega-Lite [96] with our own custom extensions for common Web-based communicative visualization techniques (e.g., text wrapping and positioning annotations relative to corresponding data marks). To support **progressive authoring (DC3)**, it is important to streamline human edits and automated suggestions so that users can revisit both manual and automated edits. To do so, DUPO uses the CICERO grammar to express both human and automated responsive design transformations. Each CICERO rule is composed of a *specifier* (what to change; e.g., axis, layout), *action* (type of change; e.g., add, modify), and *option* (how to change; e.g., font size, mark type). Whenever there are new CICERO-expressed edits, DUPO compiles it with other existing rules to update the visualization in each artboard. DUPO uses Answer Set Programming (ASP) [162, 205] with the Clingo solver [206] to search the design suggestions, and translates the resulting ASP models (set of transformations) to CICERO rules. DUPO's interface and recommender are backed by Svelte [242] and FastAPI [243].

6.5 User Study

Motivated by our design considerations (Section 6.2), we evaluated DUPO with six expert responsive visualization authors to answer the following research questions: *How well can authors explore alternative design ideas in DUPO (RQ1)? How do authors combine manual edits and automated design suggestions (RQ2)? How do authors use DUPO’s recommender control options while interacting with the action widget (RQ3)?* We also consider general feasibility and usefulness questions: *Are DUPO’s design suggestions and final outcomes deemed reasonable by authors (RQ4)? Do authors see DUPO as a feasible tool for their day-to-day responsive visualization tasks (RQ5)? What do authors see as the remaining challenges to address to improve DUPO’s usefulness (RQ6)?* For greater ecological validity, our study asked participants to create responsive versions for prior visualizations they had created. Evaluating DUPO with participants’ own use cases was also useful for helping us identify small improvements to DUPO to better support certain design situations while keeping the overall two-stage recommendation approach and other key features consistent across participants.

6.5.1 Methods

6.5.1.1 Participants and use cases

We recruited six graphics reporters (E1–E6) who regularly published responsive visualizations on media outlets, snowballing via social media (Twitter, Mastodon) and Slack (News Nerderly, Data Visualization Society). We asked participants to provide candidate use cases from their own work. We chose one of these examples that fit the scope of DUPO (in terms of supported visual encodings and interactivity) and exhibited variations in visual encoding combinations and layout. As shown in Fig-

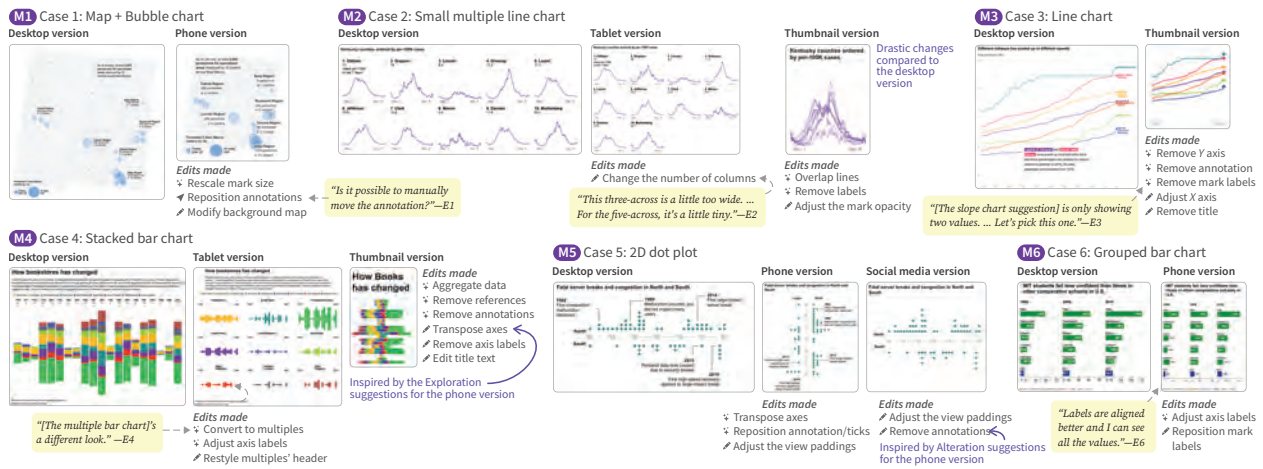


Figure 6.5: Selective design outcomes from the user study with six expert responsive visualization authors. Participants provided their own visualizations for use in the study. For anonymization purposes, we replaced participants' data sets while maintaining the cardinality and data type, edited the text content but preserved annotation length, and altered the color schemes. A star icon (☆) indicates an edit from the recommendations, a pencil icon (✎) denotes a manual edit made by a participant, and a cursor icon (☞) represents a manual edit using direct manipulation.

ure 6.5, participants contributed a map (E1), small multiples of line charts (E2), a layered and annotated line chart (E3), a stacked bar chart (E4), a 2D dot plot with many annotations (E5), and a grouped bar chart (E6). For interactivity that DUPO does not support (E3: parallax; E4: view toggling), we took a single representative view (E3: conclusion chart; E4: default view).

6.5.1.2 Study procedure

Before each session, we asked two background questions about the design process behind each participant's contributed visualization and their preferred authoring tools. During each remote-control Zoom session, the participant first went through a step-by-step guided training (creating a bar chart for desktop and exploring design suggestions for a phone) to get familiar with DUPO. Then, participants spent 30 minutes completing the main study task: to create responsive versions for their

desktop design (which we replicated in DUPO) while thinking aloud. We asked them to create the phone version first and then work on tablet, thumbnail, and print versions if time allowed. While DUPO supports different directions of responsive design (e.g., mobile-first, simultaneous editing), we decided to fix the desktop-first direction for two main reasons: (1) all participants reported desktop-first as their typical design direction when asked prior to the study, and (2) our goal was to observe how they use DUPO’s design suggestions along with manual editing, rather than how DUPO affects the direction of creation.

After the main task, participants self-evaluated their work and then completed a 15-minute interview. For self-evaluation, participants rated their satisfaction on a scale from 1 (Very unsatisfied) to 5 (Very satisfied): *Given the limited design time you had, how satisfied are you with this draft?* We followed up the self-evaluation by asking about any further edits they had wanted to make if given more time. In the structured interview, we asked participants’ about their overall reactions to seeing the design suggestions, comparison between DUPO and their day-to-day tasks, and DUPO’s features that they would want for their usual tools. We then asked about the usefulness of the *Exploration* suggestions for the phone version by showing them the suggestions again. Lastly, we asked about suggestions for general system improvement. Each participant was compensated with USD 60 (gift card). Detailed study protocols are included in Section J.1.

6.5.2 Results

We analyzed the think-aloud and interview transcripts using top-down thematic analysis [244] given the research questions outlined above. We analyzed the think-

aloud transcripts, system logs, and screen recordings² together to identify design exploration patterns. We find that DUPO supports exploring a range of responsive transformations and creating satisfactory designs, yet our participants suggested improvement for direct manipulations and different ways to show *Alteration* suggestions.

6.5.2.1 Support for design exploration (RQ1)

During the study sessions, participants made two to four responsive versions across different screen sizes that they were pleased with over a relatively short amount of time (about 30min), thereby demonstrating that DUPO supports creating multiple outcomes. We observed participants reasoning about different design suggestions in various ways. We tagged the task recordings with observations of how participants appeared to be reasoning about the designs as they interacted with the *recommender window*. First, we observed that participants reviewed *most of the suggestions*, rather than just one or two that initially appealed to them, suggesting that the recommendation sets were compelling.

Second, participants frequently compared two or more suggestions, explicitly reflecting on which one was better for their purposes. For example, comparing suggestions for a stacked bar chart with small multiples for a tablet version, E4 noted that “[*The multiple bar chart*]’s a different look. It’s an emphasis on the [*regions*] as opposed to the overall [*volume*] of the [*independent book stores*] like you get here [*with the stacked bar chart*].” E4 then chose the small multiples (Figure 6.5 M4, Tablet). Participants also confirmed the need for authoring agency over small changes by consid-

²Refer to Section J.2 for details about the log data and screen recording analyses.

ering possible manual edits to the suggestions, for example, by asking “*Is it possible to manually move the annotation?*” (E1; M1, Phone) or by saying “*I’m gonna take out all the annotations for . . . social [media]*” (E5; M5, Social).

Overall, participants seemed relatively confident in the design choices they made, which they articulated by summarizing their decisions. For instance, E3 first summarized that a slope chart suggestion for his thumbnail version “*is only showing two values*” and then said, “*let’s pick this one [with more detail]*” because he had already noted that he wanted to keep the details (M3, Thumbnail).

In terms of fixation on well-known designs, participants in general made a phone version that seemed similar to their original phone versions. This fixation is not necessarily surprising given that they had already gone through rigorous design iterations in a newsroom-like setting. E2 mentioned, “*It’s hard to disconnect from my original decisions I made.*” However, participants considered designs with more drastic changes for the versions that they had previously paid less attention to (e.g., tablet, thumbnail). For example, given a small multiple line chart, E2 overlaid those lines for a thumbnail (Figure 6.5 M2). With a grouped bar chart for desktop, E6 chose a heatmap for a thumbnail because it seemed to “*make people curious*” about the content, though she still wanted further aesthetic changes. We observed design suggestions for a previous version sometimes influencing participants’ designs for later versions. For example, E4 manually transposed the axes for the thumbnail (M4), inspired by similar *Exploration* suggestions for the phone version. Likewise, E5 removed annotations for the social version (M5), pointing out *Alteration* (see similar) suggestions for the phone version.

6.5.2.2 Mixed-initiative usage patterns (RQ2)

Given that participants immediately saw design suggestions upon creating a new responsive artboard, they tended to choose a design suggestion first before making manual edits. Participants generally spent less time using the recommender ($m=3.0\text{min}$, $s=1.9$) than making manual edits ($m=5.3$, $s=5.8$) per version. Yet, they took relatively less time to make minimal or already brainstormed manual edits, such as E2's column adjustment for small multiples (Figure 6.5 M2, Tablet) and E5's social version inspired by the suggestions for the phone (M5, Social).

DUPO was used to automate changes to overall sizes (scaling the chart size, text size, etc.), layout (transposing), and encoding, while participants tended to manually edit text elements (annotations and titles) and reference objects, which are essential in narrative visualization. For example, E4 changed the title text for text alignment and justification. After making an edit, participants tended to closely examine every element to ensure that all important information was preserved, which helped them identify what they might need to edit next. After moving annotations in the phone version, for example, E5 realized that he needed to further adjust the spacing. E6 requested recommendations after making some manual edits, and then branched an *Alteration* suggestion. Then, E6 made more edits to the two candidate artboards simultaneously via edit propagation, before finalizing one of them.

Participants tended to make manual edits by testing out a variety of different values for the visual encoding (e.g., by repeatedly changing the x or y value to position an annotation appropriately). When we filtered out manual edits (written as CICERO rules) dealing with the same element (i.e., the same specifier and option properties) from the log of participants' manual edits, the average number of manual

edits reduced from 85.3 ($s=82.1$) to 34.8 ($s=20.5$). In contrast, participants applied an average of 28.3 ($s=12.7$) edits by the recommender. We note that each recommender rule changed multiple properties, and participants explored multiple design suggestions with at most fifteen rules. Overall, DUPO enabled participants to focus on edits that needed careful visual inspection, while expediting high-level responsive transformations.

6.5.2.3 Users' control and interpretability (RQ3)

Participants interacted with features in the *recommender window*, and often read the descriptions provided in the *action widget*. For example, E6 actively used the *hide this* features, and she valued *hide this* because “*it removes all the noise that you don't need, and you can focus on the limited options that you will get.*” Using the *bring my existing edits* feature, E6 said, “[*it*] is a learning experience . . . in terms of how I can do better [*because I could*] compare what I did earlier [*manually*] and some new suggestions.” Meanwhile, E4 wished for more control in selecting a source design by asking, “*Can I make a thumbnail based on the mobile version?*” instead of the desktop version. E4 also wanted to use the *hide this* feature independently for each version.

6.5.2.4 Design satisfaction (RQ4)

Across all sessions, participants created and rated their satisfaction for 17 designs. In general, participants were satisfied (seven “very satisfied” and seven “satisfied”) with the versions they created. Some responded “just okay” to their last versions because they did not have enough time to complete an initial version (e.g., E3, Print) or they wished to fine-tune the designs (e.g., E2, E4, Thumbnail).

Participants in general found our design suggestions to be reasonable and real-

istic potential improvements to their design process. E4 said, “[DUPO] *literally made 90% of the chart I would have made by hand on my own for a mobile [phone].*” E2 mentioned that his team “*actually did mock up a version that was like this [heatmap].*” Even though participants sometimes thought some design suggestions were less visually appealing due to drastic changes from the desktop versions or less preferred label arrangement, all participants seemed able to understand why those examples were suggested. For example, E3 did not like transposing the line chart, yet he acknowledged that “*in some cases, it can also be helpful.*” E6 said some axis-transpose options were not aligned with her intention with the original bar grouping.

When we asked participants to rank the *Exploration* suggestions for the phone version during the follow-up interview, they exhibited several criteria, such as similarity with the source design and well-formedness. For example, E2 ranked the two suggestions with encoding changes (to a heatmap) in the third and fifth place, saying that they were “*missing the point [of] . . . what the desktop one was doing.*” Considering the well-formedness of the design, E6 said, “*Labels are aligned better and I can see all the values*” as she ranked the design suggestion she chose for the phone version M6 in first place. Participants found some *Exploration* suggestions to be subtle in a way that we did not expect previously. While comparing aggregated and non-aggregated *Exploration* alternatives, for example, E1 was not able to figure out the difference at first. Once we clarified the aggregation, E1 ranked the aggregated version in the first place, saying, “*that’s actually really smart.*”

6.5.2.5 Feasibility and remaining challenges (RQ5 & RQ6)

Comparing to their day-to-day tools, participants believed DUPO could help reduce the time required to prototype design ideas by providing high-fidelity designs. For

example, E1 noted that “I [had to] create three [responsive] versions. It takes up a lot of time. Even though you can copy and paste, there’s still a lot of manual work going on.” E4 said that, “It wasn’t just a little pencil sketch or a crappy chart that needed work. Basically, it did all that work.” Pointing out that “mobile is like a whole different world to think about,” E2 said DUPO could help to “get into that mindset much easier.” E6 mentioned that being able to have different designs for responsive versions, which her tool (Flourish) did not support, would allow her team to consider further encodings like treemaps for desktop versions with simplified mobile versions.

However, those familiar with Adobe Illustrator or similar tools pointed out the needs for extended direct manipulation. For example, E5’s visualization was annotation-heavy, so positioning the annotation and adjusting the spacing accordingly were important tasks. E5 found it inconvenient to make these changes in DUPO due to the limited direct manipulation. Participants also needed additional support for inspecting *Alteration* suggestions as they often included more subtle changes. E3 proposed suggesting “edits” rather than “designs,” and letting users test out different edits to help them make sense of what is being changed.

6.6 Discussion

We implemented DUPO, a mixed-initiative tool for creating responsive visualizations, to support design exploration, users’ agency over recommendation procedures, progressive authoring, and flexible artboard management. DUPO provides two-step design suggestions (*Exploration* and *Alteration*), edit *Augmentation* (*quick edits*), various user control options (e.g., the *action widget*), and edit propagation across artboards as well as between manual and automated designs. We evaluated DUPO

with six expert data journalists using their own visualization cases, observing that DUPO supports exploring different design ideas, helping users to come up with satisfactory outcomes. Below, we discuss implications to other responsive visualization design approaches and future challenges for mixed-initiative authoring with adaptive systems.

6.6.1 Next Step: Adaptive Recommendation

Compared to analytic domains like exploratory data analysis, responsive design involves a high degree of creativity in order to craft and effectively communicate a narrative across different end-user devices. Many organizations that require responsive design may develop their own individual preferences or design patterns that are necessary to incorporate into the design process. In addition, we will discover new design patterns for responsive visualization as the area is evolving, demanding an extended search space. To make this process easier, responsive visualization tools can employ an adaptive recommender that updates its search space by learning design patterns from users. While DUPO offers implicit user constraints using *hide-this*, adaptive mixed-initiative systems can learn and share new responsive techniques across different users. This approach opens up several challenges for future work in addition to developing an adaptive mechanism. For example, such a system may need to determine whether to learn a design pattern of a user and whether one user's technique would be useful for other users. It is also necessary to ask how to support users to easily control the outcome of the recommenders with a growing search space.

6.6.2 Limitations and Future Work

DUPO currently considers common mark types, encodings, and interactions supported by the extended Vega-Lite (Section D.2). While we outlined high-level rationales for generating *Exploration* design suggestions, we do not claim that our search space is exhaustive. Thus, future work could extend DUPO with regard to encodings and design suggestions. Next, our user study only considered the desktop-first approach. A longer term user study could observe how mixed-initiative approaches impact responsive visualization authoring in different settings like collaborative creation or simultaneous editing and improve the system with adaptive learning. Lastly, we qualitatively evaluated DUPO by inviting expert users. Future work could compare future responsive visualization tools with DUPO as a benchmark.

6.7 Conclusion

In designing multi-context visualization, like responsive visualization, automating design recommendation is necessary to help authors to avoid design fixation because the multiplicity of user contexts amplifies the time and effort needed for design exploration and prototyping. To address this difficulty, DUPO, a mixed-initiative authoring tool for responsive visualization, offers an automated design recommender along with features for manual edits. DUPO is backed by CICERO (Chapter 5) and a custom extension of Vega-Lite [96] for design representation as well as the task-oriented insight loss measures (Chapter 4) for objective functions for its design recommender. In the user study, expert visualization authors found DUPO to be useful in achieving satisfactory responsive designs while reducing costs of exploring and prototyping design ideas and, making it an attractive tool for their day-to-day work.

Chapter 7

ERIE: a Declarative Grammar for Data Sonification

Data sonification plays an important role across various domains like data accessibility, scientific observation, data-driven art, and museum exhibitions [30]. For people with Blindness or Vision Impairment (BVI), sonification makes it possible to access data presented on screen. In science museums or digital news articles, data sonifications can support authoring more immersive data narratives by diversifying cues. These diverse purposes may require different sonification designs.

However, creating data sonification is often laborious because of limited software-wise support for auditory channels, compared to a robust set of expressive visualization toolkits (e.g., D3 [25], ggplot2 [26]). An ability to express diverse designs helps creators and developers to be less constrained in making their artifacts. Due to a lack of expressive tools for data sonification, however, many prior empirical works in accessible visualization rely on more hand-crafted methods (e.g., using Garage Band by Wang et al. [29]) or solution-specific approaches (e.g., Hoque et al. [70]). For example, Sonification Sandbox [66]’s authoring interface for data sonifications does not support expressing a sequence or overlay of multiple sonifications. Cre-

ators of artistic sonifications or data stories need to use additional audio processing software to combine those sonifications, which requires a different set of skills. Furthermore, those tools are not programmatically available, so it is hard to apply them to use cases with data updates or user interactions. While several R and JavaScript libraries support creating data sonifications (e.g., DataGoBoop [132], PlayItByR [133], Sonifier.JS [127]), they are tightly bound to the associated visualization's chart type (e.g., histogram, boxplot) or support few encoding channels (e.g., pitch only), limiting authors' potential to compose diverse data sonification designs.

To facilitate research and tool development for data sonification, we contribute ERIE, a declarative grammar for data sonification. We developed ERIE with the goal of supporting independence from visual graphs, expressiveness, data-driven expression, compatibility with standard audio libraries, and extensibility with respect to sound design and encodings. At high level, ERIE's syntax specifies a sonification design in terms of *tone* (the overall quality of a sound) and *encoding* (mappings from data variables to auditory features). ERIE supports various *tone* definitions: oscillator, FM (frequency modulation), and AM (amplitude modulation) synthesizer, classical instruments, periodic waveform, and audio sampling. Authors can specify various auditory *encoding* channels, such as time, duration, pitch, loudness, stereo panning, tapping (speed and count), and modulation index. Authors can also use ERIE to express a composition combining multiple sonifications via repetition, sequence, and overlay. Our open-sourced ERIE player for web environments supports rendering a specified sonification on web browsers using the standard Web Audio and Speech APIs. ERIE's queue compiler generates an *audio queue* (a scheduled list of sounds to be played), providing the potential for extending ERIE to other audio environments like C++ and R.

We demonstrate ERIE’s expressiveness by replicating accessibility and general-purpose sonification designs proposed by prior work (e.g., Audio Narrative [64], Chart Reader [72], and news articles [245]). We provide an interactive gallery with a variety of example sonification designs. We conclude by outlining necessary future work for ERIE, including technological hurdles, potential use cases, and blueprints for supporting interactivity and streaming data.

7.1 Background: an Overview of Auditory Channels

Different auditory channels, such as pitch or volume, are physicalized into a waveform. We first describe a few core concepts related to a sound wave: *frequency* and *amplitude*. The frequency of a sound wave refers to the number of wave cycles (e.g., a local peak to the next local peak) per second, and its unit is hertz (Hz). A sound with a higher frequency has shorter wave cycles, and people tend to perceive it as a higher pitch. The amplitude of a sound wave means the extent or depth of a waveform. A larger amplitude makes a louder sound.

Commonly used channels in prior work include pitch, loudness (or volume), tapping, timing, panning, timbre, speech, and modulation index [246]. *Pitch* refers to how sound frequency is perceived with an ordered scale (low to high; e.g., Do-C, Re-D, Mi-E). *Loudness* means how loud or intense a sound is, often physically measured using the unit of decibel. *Timing* is when a sound starts and stops being played; the time interval is termed *duration* (or length). *(Stereo) panning* refers to the unidimensional spatial (left to right) position of a sound by controlling the left and right volume of two-channel stereo audio. *Timbre* (or instrument, put more casually) means the quality of a sound, such as piano sound, synthesizer, bird sound,

etc. Modulation-based synthesizers (or synths), such as frequency modulation (FM) and amplitude modulation (AM), have two oscillators, a carrier for the main sound and a modulator that changes the carrier's waveform through some signal processing (simply put). A *modulation index* (MI) for such synths refers to the degree of modulation in signal processing. The frequencies of two oscillators generate the *harmonicity* between them.

An audio mapping of a non-categorical variable can have a positive or negative *polarity*. A positive polarity scale maps a higher data value to a higher audio value (e.g., high pitch, high volume), and a negative polarity scale maps a higher data value to a lower audio value. While a sonification designer should be able to specify the range of an audio scale, audio scales are capped by the physical space. For example, the common audible frequency spectrum is known to range from 20 Hz to 20,000 Hz [247].

7.2 Formative Study: Gaps in Sonification Development Practices

To motivate our design of ERIE with awareness of existing practices used in developing data sonification, we surveyed recently published data sonification tutorials and designs. To understand practices being shared among sonification developers, we collected nine online tutorials for coding sonifications by searching with keywords like “sonification tutorial,” “audio graph tutorial,” or “sonification code.” To see techniques beyond tutorials, we inspected 24 data sonifications with code or

detailed methodology descriptions from Data Sonification Archive¹ that were published from 2021 through 2023. This collection included tutorials and designs created by active sonification contributors like Systems Sound² and Loud Numbers³. We include the list of the sonification tutorials and designs we collected in Appendix K. We tagged sonification tutorials and designs in terms of software or libraries used, functionality of code written by the creators (e.g., scale functions, audio environment settings), and output formats (e.g., replicability of designs, file formats). Overall, this preliminary survey identified that **developers currently rely on ad-hoc approaches due to the lack of expressive sonification approaches.**

7.2.1 Findings

7.2.1.1 Converting to auditory values then connecting to music software

Most tutorials (7 out of 9) introduced *music programming libraries* like music21, Max, PyGame, Tone.js, sequenceR, Sonic Pi, and MIDIFile, and most (15 out of 24) sonification designs used them. These libraries take as input auditory values like pitch notes or frequencies, volumes, and time durations. That is, developers still need to define scale functions that convert data values to auditory values, requiring an understanding of physical properties of different auditory variables. For example, the “Sonification 101” tutorial⁴ describes how to map data points to notes with a four-step procedure. First, a developer normalizes the data point into a range from 0 to 1, then multiplies by a scalar to keep them in a certain range. Third, the developer

¹<https://sonification.design/>

²<https://www.system-sounds.com/>

³<https://www.loudnumbers.net/>

⁴<https://medium.com/@astromatrusso/sonification-101-how-to-convert-data-into-music-with-pythhon-71a6dd67751c>

specifies a list of notes to map data points to. Last, they write a for loop to convert each data point to the corresponding note from the list. On the other hand, a tutorial by Propolis⁵ introduces a linear scale function.

Then, developers need to connect those computed values to other music libraries by configuring custom instruments. To be able to create custom instruments using low-level libraries like MIDITime or Tone.js, the developer needs to have professional skills like how to import and control audio samples and what audio nodes to control to adjust different audio properties. For instance, common sonification encodings like gain, pitch, and distortion level are governed by different audio nodes. More experienced professional creators chose to use more advanced music software like Ableton Live, Supercollider, and Touch Designer that enable live performances or art installations.

7.2.1.2 Difficulty in Reusing Sonification Designs

Whether created programmatically or not, many existing sonification cases are available as multimedia files (audios or videos). This practice makes it harder to inspect how they were created in terms of data-to-music scales, instrument details, etc.. Even if a sonification's codes are available, it is often hard to reuse the custom code because developers have to manually inspect the code in terms of different variable names to locate where to make changes for their designs. For example, to change the domain, range, and transformation type (e.g., sqrt, log) of a certain scale, then they have to find the relevant lines and manually change them by writing something like a linear scale function (e.g., `aScaleFunction(x) {return min(1600, max((log(x)`

⁵<https://propolis.io/articles/making-animated-dataviz-sonification.html>

- $\log(30)$) / ($\log(500) - \log(30)$) * 1600, 200);}), which is not always straightforward, particularly for less experienced sonification developers. This difficulty in reusing custom code is also widely known among visualization practitioners [248].

7.3 Design Considerations

Leveraging prior empirical studies, sonification toolkits (Table 2.1), and development practices (Section 7.2), we developed the ERIE grammar and compiler as a toolkit for sonification developers with the following considerations in mind.

(C1) Be independent

Many existing sonification libraries that provide APIs are strongly tied to visual forms, such that they support sonifying a particular visualization instead of authoring a sonification. While this approach can make it easy to generate sounds, it prevents sonification creators from exploring the many alternative designs one might generate by directly expressing audio graphs. Furthermore, it ignores different tasks implied by similar visualization designs. For example, point marks can be a scatterplot for assessing correlation or a residual plot for judging model fit, potentially calling for different sonification designs. We designed the ERIE grammar to be independent of visual forms to maximize design possibilities.

(C2) Be expressive

To support independently creating various sonification designs, it must be possible to express different sound qualities, auditory channels, and combinations of multiple sonifications. Expressive toolkits enable researchers and developers to navigate a variety of design ideas. Thus, ERIE supports specifying different sound de-

signs (e.g., instrument types, discrete vs. continuous sounds) and different auditory channels for data encoding, and also allows for specifying sonification sequences and overlays.

(C3) Be data-driven

Sonification can be a useful tool for enhancing presentations of data in other modalities (e.g., visualization), in addition to standing on its own. Creating sonification often starts with implementing ad-hoc functions to convert data to audio properties as shown earlier. Under the assumption that ERIE's users may have limited understanding and skill with respect to acoustic engineering and audio programming, it makes more sense to be able to declare data-to-audio conversions with a few configuration terms. Consequently, we designed ERIE's syntax to express *data* instead of *sound* by leveraging the *grammar of graphics* [249] and its popular implementations [26, 96, 147], such as their scale expressions for encoding channels.

(C4) Be extensible

A toolkit may not be able to support all potential cases in advance, particularly when the design space is unlimited. ERIE allows for sampling audio files, configuring FM and AM synths, and defining periodic waves (combining multiple sine and cosine waves). Furthermore, ERIE provides a method to define and connect custom audio filters (e.g., distortion, biquad filters) that can have extra auditory encoding channels.

(C5) Be compatible with standards

The expressiveness and extensibility criteria are constrained by specific audio environments. As different display media affect the resolution of images, sound repre-

sentations are highly sensitive to audio environments, such as processing capacities and equipment. Thus, compatibility with the standards of a targeted environment is critical, similar to how we use SVG or Canvas for web visualizations. We consider two standards for sonification: (1) physical units and (2) rendering standards. First, ERIE’s queue compiler generates a scheduled list of sound items using standard auditory units (e.g., Hz and musical notes for pitch, the panning range from -1 to 1) so as to be used in other audio environments (e.g., external music software). Our ERIE player for web employs the Web Audio and Speech APIs to enable cross-browser experience.

7.4 ERIE Grammar

We formally describe the syntax of the ERIE grammar to show how ERIE is designed to be **expressive (C2)** and **data-driven (C3)**. At a high level, ERIE expresses a sonification design using a sound instrument (*tone*) and mappings from data to auditory values (*encoding channels*). After walking through an example case, we describe how ERIE expresses a data sonification design, including top-level specification, stream, data input and transform, tone, encoding, stream composition, and configuration. The formal definition of ERIE is provided in Figure 7.1. In describing ERIE, we distinguish *developers* who create sonifications from *listeners* who listen to sonifications. For details, refer to Appendix L and the online documentation⁶.

⁶<https://see-mike-out.github.io/erie-documentation/>

7.4.1 A Walkthrough Example

To help imagine how ERIE works in specifying a sonification design, we introduce a simple auditory histogram for a quantitative data variable, *miles per gallon* with a range from five to 50, from a ‘*cars.json*’ dataset [250]. In this sonification, *miles per gallon* is discretized into nine bins by five miles, and the bins are communicated by mapping them to time. The count (aggregation) of each bin is mapped to pitch.

To construct this example using ERIE, we first specify the data to sonify by providing its URL:

$$data = \{url = cars.json\}$$

Then, we need data *transforms* for binning and count aggregation. The below expression creates bins for the *miles per gallon* field using default binning options (*auto*). This operation defines two additional fields for the start and end point of each bin. The expression further assigns *miles-per-gallon-bin* to the name of bucket start points (*as*) and *miles-per-gallon-bin-end* to the name of end points (*end*).

$$bin = \{field = miles-per-gallon, auto = true, \\ as = miles-per-gallon-bin, end = miles-per-gallon-bin-end\}$$

For the count aggregation, the below expression specifies doing a *count* operation, and names the resulting field *count*. To count the values for each bucket, this expression sets a *group-by* field to the bin start point field (*miles-per-gallon-bin*) generated

Table 7.1: The results of data transforms in Section 7.4.1.

<i>miles-per-gallon-bin</i>	<i>miles-per-gallon-bin-end</i>	<i>count</i>
5	10	1
10	15	52
15	20	98
20	25	78
25	30	77
30	35	56
35	40	27
40	45	8
45	50	1

by the previous bin transform.

$$aggregate = \{op = count, as = count, group-by = miles-per-gallon-bin\}$$

To have the results of the bin transform feed-forward to the count aggregation, these two transforms are ordered as:

$$transform = [bin, aggregate]$$

Applying these transforms to the ‘cars.json’ data results in Table 7.1.

Second, we need to define how to sonify the specified data in terms of overall qualities (*tone*) and auditory mappings (*encoding*). We indicate that the sound should be segmented or discrete:

$$tone = \{continued = false\}$$

Then, we need three encoding channels: when to start each sound (*time*), when to

end it (*time2*), and its *pitch*. The *time* channel encodes the bin start points (*miles-per-gallon-bin*):

$$time = \{field = miles-per-gallon-bin, type = quantitative, scale = \{length = 4.5\}\}$$

The above expression also specifies that the *time* channel encodes a *quantitative* variable and that the total *length* of the auditory histogram is 4.5 seconds. We want to finish each bin's sound with respect to the bucket's endpoint. Because bins' start and end points are in the same unit and scale, we use an auxiliary *time2* channel:

$$time2 = \{field = miles-per-gallon-bin-end, type = quantitative\}$$

Note that this *time2* channel has no *scale* expression because it uses the same scale as the *time* channel. Next, we encode the *count* of each bin to a *pitch* channel in a way that a higher count is mapped to a higher pitch (*positive polarity*), using the below expression:

$$pitch = \{field = count, type = quantitative, \\ scale = \{domain = [0, 100], range = [220, 660], polarity = positive\}\}$$

This expression further specifies that this *pitch* channel maps a *domain* (from 0 to 100) to a pitch frequency *range* (from 220Hz–A4 note to 660Hz–A6 note). These three encoding channels are combined as:

$$encoding = \{time, time2, pitch\}$$

#	Type	Sound
1	Speech	Start playing.
		Start End Duration Timbre Pitch
		0 0.5 0.5 Sine 224.4
		Start End Duration Timbre Pitch
		0.5 1 0.5 Sine 448.8
		Start End Duration Timbre Pitch
		1 1.5 0.5 Sine 652.2
		Start End Duration Timbre Pitch
		1.5 2 0.5 Sine 563.2
2	Tone	Start End Duration Timbre Pitch
		2 2.5 0.5 Sine 558.8
		Start End Duration Timbre Pitch
		2.5 3 0.5 Sine 466.4
		Start End Duration Timbre Pitch
		3 3.5 0.5 Sine 338.8
		Start End Duration Timbre Pitch
		3.5 4 0.5 Sine 255.2
		Start End Duration Timbre Pitch
		4 4.5 0.5 Sine 224.4
3	Speech	Finished.

Table 7.2: The sonification output for an auditory histogram in Section 7.4.1. “#” indicates the playing order of each part. Units: seconds (start, end, duration) and Hz (pitch). “Sine” means a sinusoidal oscillator.

Lastly, the above expressions are combined into a *spec* as:

$$spec = \{data, transform, tone, encoding\}$$

This *spec* results in the sonification output shown in Table 7.2 (see the online gallery⁷ for the actual audio). The equally-sized bins are mapped to the start and end times, and the aggregated counts by each bin is mapped to the pitch frequencies.

⁷<https://see-mike-out.github.io/erie-editor/?ex=histogram-1>

1. Top-level Specification

```
Spec := {Title, Description, Auditory description
        (Stream | Overlay | Sequence), Design definition
        Dataset, Tick, Synth, Wave, Sampling, Config}
For stream composition and customization
```

2. Stream: a unit sonification design

```
Stream := {Data, Transform, Tone, Encoding, Config}
Overlay := [Stream] Playing multiple streams together
Sequence := [Stream | Overlay] Playing one stream after another
(Note: a nested sequence = a sequence)
```

3. Data and datasets: what to be sonified

```
Data := Name<String> | Url<UrlString> | Values<Array>
Dataset := [{Name<String>, (Url<UrlString> | Values<Array>)}]
Allows for registering datasets and using them in overlaid or sequenced streams
```

4. Transform: modifying data for sonification design purposes

```
Transform := [Aggregate | Bin | Density | Fold |
             Calculate | Filter | ... ]
```

5. Tone: overall audio quality ≈ mark or glyph

```
Tone := {ToneType,
         Continued<Boolean>, When an audio property changes
         Filter} Discrete: momentarily pause and resume
         Continuous: no pause and resume
ToneType := {default | sawtooth | triangle | square
            Musical | piano | pianoElec | violin | guitar | metal
            Drums | hihat | snare | highKick | lowKick | clap
            Noise | whiteNoise | pinkNoise | brownNoise
            * <String> Custom instruments
Filter := [FilterName<String>]
```

6. Encoding: mapping from data to audio

```
Encoding := [Channel: ChannelDef]
Channel := {time | time2 | duration | tapSpeed | tapCount | pitch | detune | pan
           | loudness | timbre | postReverb
           | modulationIndex | harmonicity | speechBefore
           | speechAfter | repeat | * <String>}
Custom channels via audio filters
ChannelDef := { ( {Field<string> [string] [Channel='repeat']},
                Dynamic channel EncType, Scale)
              | {Condition, Value<Any>} ),
              (Ramp) [Tone.Continued=True],
              Aggregate, Bin, Inline data transforms
              (TimeUnit, TimeLevel) [EncType='temporal'],
              (Speech<Boolean>, By) [Channel='repeat'],
              (Tick<TickItem|String>) [Channel='time']}
For specific encoding types and channels
EncType := 'quantitative' | 'ordinal' | 'nominal' | 'temporal'
Condition := [{Test<String>, Value<Any>}]
Ramp := 'linear' | 'exponential' | 'abrupt' How gradually to change audio properties
Aggregate := 'mean' | 'median' | ...
Bin := <Boolean> | {maxbins, nice, step, exact}
By := 'sequence' | 'overlay' | ['sequence' | 'overlay']
How to arrange repeated streams
TimeUnit := 'year' | 'month' | 'day' | ... Category (aggregate by)
TimeLevel := 'year' | 'month' | 'day' | ... Precision (aggregate up to)
```

7. Scale: customizing how a data variable is mapped to an auditory variable

```
Scale := {Description, Polarity, Domain, The set of the data variable to map
         (Range | MaxDistinct<Boolean> | Times<Number>
         Explicitly Maximum audible range By multiplying data by a factor
         | (Length<Number>) [Channel='time'])
         For a time channel, by providing the length of the stream
         (ScaleType, Zero<Boolean>) [EncType='quantitative'],
         (Timing) [Channel='time'], Whether to include zero in the domain
         (Band<Number>) [Channel='time' | tapSpeed | tapCount']}
Description := Boolean | DescriptionMarkUpString
Domain := [Any] Whether to and how to generate the description of the channel ≈ legend
Range := [Any]
Polarity := 'positive' | 'negative'
Higher data value... to higher audio value to lower audio value
ScaleType := 'linear' | 'log' | 'pow' | 'sqrt' | 'symlog'
The type of the scale function for a quantitative channel
Timing := 'absolute' | 'relative' | 'simultaneous'
For a discrete tone, at the time one after another all together
each audio point is played... specified by the mapping
```

8. Tick: a sound repeating every certain time interval ≈ axis

```
Tick := [TickItem] This list form allows different ticks for different streams.
... Referred to by Channel.Tick
TickItem := {Name<String>, Interval<Second>, OscType,
            Pitch<Hz>, Loudness<Gain>,
            PlayAtTime0<Boolean>} Whether to play the tick at time 0 (default = true)
OscType := 'sine' | 'sawtooth' | 'triangle' | 'square'
Gain := Number<[0,Infinity]>
```

9. Synth: defining a custom FM or AM synthesizer

```
Synth := [SynthItem] Referred to by Tone.ToneType
SynthItem := {Name<String>, SynthType,
             Envelope AttackTime<Second>, ReleaseTime<Second>,
             Carrier CarrierType<OscType>, CarrierPitch<Hz>, CarrierDetune<Detune>,
             Modulator ModulatorType<OscType>, ModulatorPitch<Hz>,
             ModulatorVolume<Gain>,
             Modulation (ModulationIndex<Number>) [SynthType='fm'],
             (Harmonicity<Number>) [SynthType='am']}
SynthType := 'fm' | 'am' Detune := Number<[-1200, 1200]>
```

10. (Periodic) Wave: defining the waveform of an oscillator by using cosine and sine terms

```
Wave := [WaveItem] Referred to by Tone.ToneType
WaveItem := {Name<String>, Real, Imag}
Real := [Number] Cosine terms Imag := [Number] Sine terms
```

11. Sampling: importing external audio files as instruments

```
Sampling := [SamplingItem]
SamplingItem := {Name<String>, Referred to by Tone.ToneType
                Sample}
Sample := Mono<UrlString> | Octave
Sound files... w/o pitch like drums w/ pitch like pinao or violin
Octave := {C0<UrlString>, ..., C7<UrlString>}
```

12. Config: specifying configuration options

```
Config := [Key<String>: Value<Any>]
E.g., the key "timeUnit" can have a value of "beat" or "second"
```

Notations

A := B	A is defined as B.	(A) [B=C]	A is available when B is C.	A < [B, C] >	A number type A with a range between B and C.
{A, B}	A tuple of A and B.	A < B >	An item of type A.	[A]	A list of type A.
A B	A or B.	* < A >	Anything of type A.	[A: B]	A dictionary with key of type A and value of type B.

Figure 7.1: The formal definition of Erie. For applicable elements, roughly analogous visualization elements are denoted by ≈ signs.

7.4.2 Top-Level Specification and Stream

We first define a simple, single data sonification specification in ERIE (a *spec*, hereafter) as a tuple of *stream*, *dataset*, *tick*, *synth*, *wave*, *sampling*, *title*, *description*, and *config*:

$$spec := \{stream, dataset, tick, synth, wave, sampling, title, description, config\}$$

The curly brackets { } indicate a tuple of elements.

A *stream* represents a unit sonification design, consisting of *data* (what to sonify), *transform* (operations to the data), *tone* (overall sound quality), and *encoding* (mappings from data to sound values):

$$stream := \{data, transform, tone, encoding\}$$

To pre-define and reuse elements in multiple *stream*, a developer can use different lists of named objects for *dataset*, *tick*, *synth* (synthesizers), *wave* (periodic wave), and *sampling* (using external audio files as a *tone*). A developer can specify a speech-based *title* and *description* that are played before the audio graph. The *config* of a *spec* configures a sonification design, such as the speed of speech (*speech rate*) and whether to skip playing the *title* text (*skip title*).

7.4.3 Data, dataset, and Transform

A sonification *stream* must have data to sonify and ERIE supports three methods to do so: providing the *URL* of a data file, providing an array of *values*, or providing the

name of a predefined dataset in the *dataset* object.

$$data := URL \mid values \mid name,$$

where the vertical bar sign $|$ denotes ‘or.’ A *dataset* object consists of the named definitions of data items using *URL* or *values*.

$$dataset := [\{name, URL\} \mid \{name, values\}]$$

The square brackets $[]$ denote a list of elements.

After pre-processing the data to sonify, a developer may need to perform additional, simple data transforms for sonification design purposes, such as the binning for the auditory histogram in the walkthrough. The developer can list transform definitions in a *transform* object. In the walkthrough, for example, the *bin* transform created new data variables for the start and end points of each bin, and the *count aggregate* reshaped the data with a new variable for the count of each bin.

7.4.4 Tone

To set the baseline sound of a sonification stream, a developer needs to specify the sound *tone*. A tone is roughly analogous to a mark or glyph in a visualization given that data values are mapped to its properties like pitch. ERIE expresses the *tone* of a stream using an instrument *type* (e.g., piano, FM or AM synth), an indicator of whether a sound is *continued*, and a set of audio *filters*.

$$tone := \{type, continued, filter\}$$

An instrument *type* can be specified by its name, such as ‘sawtooth’ (oscillator) or ‘violin,’ where default is a sinusoidal oscillator in our implementation. If a sound is *continued*, two sound points are connected without a pause. For more diverse audio expressions, the developer can provide audio *filters* like distortion or equalizer.

7.4.5 Encoding

The *encoding* of a stream defines how data variables are mapped to different auditory properties (e.g., pitch and loudness) of a *tone*. ERIE supports three classes of channels: dynamic, conditioned, and static. A *dynamic channel* encodes a data variable (or field) to the respective auditory property. It is defined in terms of a data *field* from the *stream*’s data, the data type of an encoding (*enc-type*), its *scale* details, its *ramping* method, and inline data transform options (*aggregate* and *bin*):

$$channel_d := \{field, enc-type, scale, ramp, aggregate, bin\}$$

The data type of encoding (*enc-type*) can be either *quantitative*, *ordinal*, *nominal*, or *temporal*, reflecting common data types. For a continuous tone, a *ramping* method specifies how to smoothly transition one auditory value to another. The transition can be abrupt (no-ramping), linear, and exponential.

A developer may need to emphasize certain data values by making them sound different instead of encoding every data value using a scale. In the walkthrough, suppose that the developer wants to indicate bins with more than 80 counts using a louder sound. Supporting such cases, a *conditioned channel* has a *condition* list for

special values and a *value* for the others.

$$\text{channel}_c := \{\text{condition}, \text{value}, \text{ramp}\}$$

The *condition* is a list of *test* conditions and desired *values*.

$$\text{condition} := [\{\text{test}, \text{value}\}],$$

where if a data value meets a *test* condition, then the specified *value* is assigned.

Then, the above example can be expressed as:

$$\text{loudness} = \{\text{value} = 0.5, \text{condition} = [\{\text{test} = (\text{datum.count} > 80), \text{value} = 1\}]\}$$

Lastly, a static channel only needs a *value* (i.e., $\text{channel}_s := \{\text{value}\}$).

7.4.5.1 Scale

The *scale* of a dynamic encoding channel essentially consists of the *domain* (data values to map) and *range* (audio values to be mapped) of the mapping. From the walkthrough, the domain of $[0, 100]$ and the range of $[220, 660]$ of the pitch channel compose a linear function $f(x) = (660 - 220) \times \frac{x}{100} + 220$. There are shortcuts for defining a *range*. When *max-distinct* is set to *true*, then the widest possible range is used (e.g., the lowest to highest human-audible pitch). The *times* multiplies each data value by itself to compute auditory values. To verbally describe the scale, a developer can provide *description* using a markup expression (see Section L.2.3), anal-

ogous to a legend in a visualization. A baseline *scale* is formally defined as:

$$scale := \{domain, (range \mid max-distinct \mid times), description\}$$

For a quantitative variable, the developer can further specify *scale-type* (e.g., square-root, log, and exponential), the inclusion of *zero point*, and *polarity*:

$$scale_q := \{ \dots, polarity, scale-type, zero \}$$

An ellipsis (...) denotes the baseline properties.

7.4.6 Composition

Combining multiple *streams* is necessary to create rich auditory data narratives (e.g., [64, 72]). For example, a stream for vote share can be repeated to provide statistics for different regions. Alternatively, two streams, one for vote shares and one for the number of elected officers in a certain region, can be sequenced to deliver more information about election results in the region. Streams for different polls can be overlaid to support synchronized comparison. ERIE supports expressing data-driven repetition and concatenation-based composition.

7.4.6.1 Data-driven repetition: Repeat channel

Data analysts commonly examine a measure conditional on one or more categorical variables. For instance, the developer may want to extend the walkthrough case by replicating the auditory histogram by the *origin* of manufacture (i.e., three histograms for U.S.A., Japan, and Europe). To support such cases, a *repeat* channel defines how to repeat a *stream* design by one or more data fields. From the previ-

#	Type	Sound
1	Speech	U.S.A., 3
2	Tone	[The histogram for origin U.S.A and 3 cylinders]
3	Speech	U.S.A., 4
4	Tone	[The histogram for origin U.S.A and 4 cylinders]
...
9	Speech	U.S.A., 8
10	Tone	[The histogram for origin U.S.A and 8 cylinders]
11	Speech	Japan, 3
12	Tone	[The histogram for origin Japan and 3 cylinders]
...
19	Speech	Japan, 8
20	Tone	[The histogram for origin Japan and 8 cylinders]
21	Speech	Europe, 3
22	Tone	[The histogram for origin Europe and 3 cylinders]
...
29	Speech	Europe, 8
30	Tone	[The histogram for origin Europe and 8 cylinders]

Table 7.3: The sonification stream order for the auditory histograms repeated by the *origin* and *cylinders* variables.

ous example, the developer can repeat the auditory histogram by the *origin* and the number of *cylinders* (values: 3, 4, 5, 6, and 8):

$$repeat = \{field = [origin, cylinders]\}$$

In this case, the repeat order is nested, such that the histograms for the cylinder values are played for each origin. A *repeat* channel has a *speech* property to indicate whether to speak out the value(s) for each repeated stream. If *speech* is set to *true* for this example, the repeated streams are played as shown in Table 7.3.

Suppose the developer now wants to simultaneously play (i.e., overlay) the auditory histograms for different cylinder values to reduce the playtime. To do so, the

#	Type	Sound
1	Speech	U.S.A.
2	Tone-Overlay	[The histograms for U.S.A. and cylinder values:] 3 4 5 6 8
3	Speech	Japan
4	Tone-Overlay	[The histograms for Japan and cylinder values:] 3 4 5 6 8
5	Speech	Europe
6	Tone-Overlay	[The histograms for Europe and cylinder values:] 3 4 5 6 8

Table 7.4: The sonification stream order for the auditory histograms sequenced by the *origin* field and overlaid by the *cylinders* field.

developer can use the *by* property in the *repeat* channel:

```
repeat = {field = [origin, cylinders], by = [sequence, overlay], speech = true}
```

This results in a sonification queue shown in Table 7.4.

7.4.6.2 Concatenation: Sequence and overlay

Two or more separate *streams* can be combined as a *sequence* (playing one after another) or an *overlay* (playing all together at the same time). To enable these multi-stream compositions, we extend the definition of a stream:

```
stream := {data, tone, encoding, title, description, config}
```

Consequently, a top-level spec is also redefined as:

```
spec := {(stream | overlay | sequence), transform,  
dataset, tick, synth, wave, sampling, title, description, config}
```

These extensions allow for specifying the title, description, and configuration of each sub-*stream* as well as global data transforms. The *config* object in a sub-*stream* overrides the top-level *config*. The *transform* object defined in a *stream* of a *sequence* is applied after the top-level (global) *transform* object.

Then, an *overlay* is formalized as a list of streams, and a *sequence* is defined as an ordered list of streams and overlays:

$$\begin{aligned} \textit{overlay} &:= [\textit{stream}] \\ \textit{sequence} &:= [\textit{stream} \mid \textit{overlay}] \end{aligned}$$

Note that a nested sequence, $[\textit{sequence}, \textit{sequence}]$, is also a *sequence*.

7.4.7 Configuration

A *config* object specifies overall controls for the sonification. For example, when a sonification consists of multiple streams that use the same auditory encodings and scales, the developer can skip playing the scale descriptions for the non-first streams. When a sonification needs more musical representation, a developer can change the *time-unit* from seconds (default) to beats. For background, when BPM is 100, one beat corresponds to 0.6 seconds ($= 60/100$). In this case, the developer can specify the tempo (beat per minute, or BPM) and whether to round raw beats to integer beats (e.g., 3.224 to 3). When the time unit of sonification is set to beats, then other time-related units are also accordingly converted. For instance, the unit for a *tap-speed* channel becomes taps per beat.

7.5 ERIE Compiler and Player for Web

A family of compilers and renderers for declarative grammar produces the output as expressed in a design spec. For ERIE, a *queue compiler* compiles a spec to an *audio queue* representing a schedule of sounds to be played in terms of their physical values. Then, a *player* renders this audio queue into actual sounds. We separate the queue compiler from the player to allow listeners to control when to play or pause a sonification and to support developing players for different audio environments, such as CSound [251]. We implemented and open-sourced a spec API, a queue compiler, and a player for a web environment⁸ using web standard APIs in JavaScript (C5: Compatibility with standards).

7.5.1 Supported Presets

Compilers and renderers of declarative grammar often provide default presets. ERIE compiler and player offer the following presets.

Data and data transform

ERIE's compiler supports multidimensional data in a relational table form (e.g., CSV, JSON). Since we assume that a developer has done fundamental data processing and transforms (e.g., fitting a regression model), our compiler supports a minimum set of data transform types that include aggregation, binning, kernel density estimation, folding (columns to rows; e.g., `[{A: 1, B: 2}]` → `[{key: "A", value: 1}, {key: "B", value: 2}]`), filtering, and calculation.

⁸<https://github.com/see-mike-out/erie-web>

Instrument types

Our web player supports musical instruments (classical piano, electronic piano, violin, guitar, metal guitar, clap, hi-hat, high-kick, low-kick), noises (white, pink, and brown), simple oscillators (sine, sawtooth, triangle, and square forms), configurable FM and AM synths, and periodic waves.

Audio filters

Our web player offers preset filters such as a dynamic compressor, a distortion filter, an envelope node, and various types of biquad filters. These filters can be chained in the *tone* of a *stream*.

Encoding channels

Our queue compiler handles *time*, *time2*, *duration*, *tap-speed*, *tap-count*, *pitch*, *detune*, *pan*, *loudness*, *timbre*, *post-reverb*, *modulation index*, *harmonicity*, *repeat*, *speech-before*, and *speech-after* channels. Different audio filters can have extra encoding channels. For example, a lowpass biquad filter attenuates frequencies above a certain cutoff, and it can have a *biquad-frequency* channel to set the cutoff.

Scale descriptions

ERIE's queue compiler generates a description of each scale to give an overview of the sonification. A scale description functions as an auditory legend in a sonification. For example, the scales of the *time* and *pitch* channels from the walkthrough is auditorily described as shown in Table 7.5.

#	Type	Sound										
4	Speech	The <i>miles-per-gallon</i> is mapped to time. The duration of the stream is 4.5 seconds.										
5	Speech	The <i>count</i> is mapped to pitch. The minimum domain value 0 is mapped to										
6	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>220</td> <td>1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	0	0.3	Sine	220	1
Start	Duration	Timbre	Pitch	Loudness								
0	0.3	Sine	220	1								
7	Speech	and the maximum domain value 100 is mapped to										
8	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>660</td> <td>1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	0	0.3	Sine	660	1
Start	Duration	Timbre	Pitch	Loudness								
0	0.3	Sine	660	1								

Table 7.5: The default scale description provided by *Erie* for the walkthrough case. These items are played before the sonification in Table 7.2 by default.

7.5.2 Spec API

We implemented ERIE syntax in JavaScript. For example, the spec of the walkthrough can be written as below.

```

1 // Create a spec object as a single stream.
2 let spec = new Stream();
3 // Assign the data URL to the spec.
4 spec.data("url", "cars.json");
5 // Add the bin transform
6 let bin = new Bin("miles-per-gallon");
7 bin
8   .as("miles-per-gallon-bin", "miles-per-gallon-bin-end")
9   .nice(true); // as/end names -> "auto" binnig
10 spec.transform.add(bin);
11 // Add the count aggregation
12 let aggregate = new Aggregate();
13 // setting operation and the new field name -> setting group-by
14 aggregate.add("count", "count")
15   .groupby(["miles-per-gallon"]);
16 spec.transform.add(aggregate);
17 // Set the tone of the stream.
18 spec.tone.continued(false);
19 // encodings
20 // Set the time channel for the "quantitative" field "miles-per-gallon-bin"
21 // Set the timing to absolute.
22 spec.encoding.time
23   .field("miles-per-gallon-bin", "quantitative")
24   .scale("timing", "absolute")
25   .scale("length", 4.5);
26 // Set the time2 channel for the field "miles-per-gallon-bin-end".

```

```

27 spec.encoding.time2.field("miles-per-gallon-bin-end");
28 // Set the pitch channel for the "quantitative" field "count".
29 spec.encoding.pitch.field("count", "quantitative")
30     .scale("domain", [0, 100])
31     .scale("range", [220, 660])
32     .scale("polarity", "positive");

```

This spec is equivalent to the following JSON object, which can be obtained via the `get` method of the spec API. This JSON syntax reuses some Vega-Lite [96] expressions, supporting cases where visualization and sonification need to be provided concurrently.

```

33 // results of spec.get()
34 { "data": { "url": "cars.json" },
35   "transform": [{
36     "bin": "miles-per-gallon",
37     "as": "miles-per-gallon-bin",
38     "end": "miles-per-gallon-bin-end",
39     "nice": true,
40   }], {
41     "aggregate": [{ "op": "count", "as": "count" }],
42     "groupby": ["miles-per-gallon-bin"] }],
43   "tone": { "continued": false },
44   "encoding": {
45     "time": {
46       "field": "miles-per-gallon-bin",
47       "type": "quantitative",
48       "scale": { "timing": "absolute", "length": 4.5 }},
49     "time2": { "field": "miles-per-gallon-bin-end" },
50     "pitch": {
51       "field": "count",
52       "type": "quantitative",
53       "scale": { "domain": [0, 100], "range": [220, 660] }}}}}

```

7.5.3 Queue Compiler

Given a spec, our queue compiler converts data values to auditory values. The outcome audio queue is an ordered list of sub-queues, and each sub-queue item can have one of these four types: *speech*, *tone-series*, *tone-speech-series*, and *tone-overlay*. A *speech* queue consists of natural language sentences that are played one after an-

other. A *tone-series* queue is a timed list of non-speech sounds, and a *tone-speech-series* queue is a timed list of sounds and speeches. Each sound in a sub-queue of these two types is expressed in terms of their actual auditory values (e.g., Hz for pitch). Lastly, a *tone-overlay* queue consists of multiple *tone-series* queues that are played simultaneously. An audio queue is not nested except *tone-overlay* queues, and a *sequence* spec is compiled to multiple flattened sub-queues.

To compile a spec into an audio queue, a developer can run `compileAudioGraph` function, which asynchronously compiles the spec to an audio queue:

```
54 let audioQueue = await compileAudioGraph(spec.get());
```

7.5.4 Player for Web

We developed an ERIE player for web environments using the standard Web Audio API [252] and Web Speech API [253]. The player offers several playing options: play from the beginning, pause, resume, stop, play from a sub-queue, and play from one sub-queue to another.

```
55 audioQueue.queue.play(); // Play from the beginning
56 audioQueue.queue.pause(); // Pause
57 audioQueue.queue.resume(); // Resume from where it was paused
58 audioQueue.queue.stop(); // Stop playing
59 audioQueue.queue.play(i); // Play from the i-th sub-queue
60 audioQueue.queue.play(i, j); // Play the i-th to (j-1)-th sub-queues.
```

7.5.5 Filter and Channel Extension

To achieve certain sound effects, a developer could use audio filters in addition to custom instruments (e.g., configured synth, sampling). Furthermore, those audio filters can encode data variables (e.g., the amount of distortion to express air quality). To widen such design possibilities, ERIE offers APIs for defining custom audio

filters that can have additional encoding channels (**C4: Extensibility**).

To describe the process of defining a custom filter, imagine that a developer wants to add an envelope filter with encodable `attack` and `release` times. `Attack` means the time duration from the zero volume at the beginning of a sound to the highest volume, and `release` refers to the time taken from the highest volume to the zero volume at the end of the sound [254]. The developer first needs to define the filter as a JavaScript `class` that can be chained from a sonification sound to an output audio device. This class should have `connect` and `disconnect` methods to enable the chaining, following the Web Audio API syntax [255]. Then, the developer needs to define an `encoder` function that assigns the `attack` value for each data value to the filter and a `finisher` function that assigns the `release` values to the filter. Refer to the online documentation⁹ for technical details.

7.6 Demonstration

To demonstrate ERIE grammar's **independence from visualization (C1)** and **expressiveness (C2)**, we walk through novel examples. We also replicated and extended prior sonifications to show the feasibility of our compiler and player for sonification development. In addition to the below examples, more use cases, such as a confidence interval, histogram, and sonification of COVID-19 death tolls, are available in our example gallery¹⁰.

⁹<https://see-mike-out.github.io/erie-documentation>

¹⁰<https://see-mike-out.github.io/erie-editor/>

7.6.1 Example Sonification Designs

We show three representative example cases to show how ERIE can be used.

7.6.1.1 Data sparsity

Given five data tables named A to E, suppose we want to identify and compare their sparsity (the portion of cells that are empty) using a tap-speed channel. We have a nominal variable, dataset `name`, and a quantitative variable, `sparsity`, and the data looks like:

```
1 let data = [
2   { "name": "A", "sparsity": 0.4 },
3   { "name": "B", "sparsity": 0.6 },
4   { "name": "C", "sparsity": 0.2 },
5   { "name": "D", "sparsity": 0 },
6   { "name": "E", "sparsity": 0.9 }];
```

Now, we want to map the `name` field to the `time` channel of a sonification and the `sparsity` to the `tapSpeed` channel, so that a sparse dataset with a higher sparsity value has slower tapping. First, we create a single-stream sonification spec object and set a description text.

```
7 let spec = new Stream();
8 spec.description("The sparsity of different datasets.");
```

Then, we assign the `data` to this spec.

```
9 spec.data("values", data);
```

With a default sine-wave oscillator, we need a discrete tone to represent separate data tables, which can be specified as:

```
10 spec.tone.type("default").continued(false);
```

Next, we set the `time` encoding channel as described earlier.

Q.	Type	Sound																		
1	Speech	To stop playing the sonification, press the X key.																		
2	Speech	The sparsity of different datasets.																		
3	Speech	This stream has the following sound mappings.																		
4	Speech	The category is mapped to time.																		
5	Speech	The sparsity is mapped to tap speed. The minimum value 0 is mapped to																		
6	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>$[\text{.19}, \text{.01}] \times 10$</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0-----2</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	0	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.01}] \times 10$						0-----2
Start	Dur.	Timbre	Pitch	Loud.	Tapping															
0	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.01}] \times 10$															
					0-----2															
7	Speech	and the maximum value 1 is mapped to.																		
8	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>No tapping</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	0	2	Sine	523.25 (C5)	1	No tapping						
Start	Dur.	Timbre	Pitch	Loud.	Tapping															
0	2	Sine	523.25 (C5)	1	No tapping															
9	Speech	Start playing.																		
10	Tone-Speech	<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Speech</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>-</td> <td>“A”</td> </tr> </tbody> </table>	Start	Dur.	Speech	0	-	“A”												
		Start	Dur.	Speech																
		0	-	“A”																
		<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>$[\text{.19}, \text{.17}] \times 6$</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0- - - - - -2</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.17}] \times 6$						0- - - - - -2
		Start	Dur.	Timbre	Pitch	Loud.	Tapping													
		After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.17}] \times 6$													
							0- - - - - -2													
		<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Speech</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>-</td> <td>“B”</td> </tr> </tbody> </table>	Start	Dur.	Speech	After prev.	-	“B”												
		Start	Dur.	Speech																
		After prev.	-	“B”																
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>$[\text{.19}, \text{.41}] \times 4$</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0- - - -2</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.41}] \times 4$						0- - - -2		
Start	Dur.	Timbre	Pitch	Loud.	Tapping															
After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.41}] \times 4$															
					0- - - -2															
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Speech</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>-</td> <td>“C”</td> </tr> </tbody> </table>	Start	Dur.	Speech	After prev.	-	“C”														
Start	Dur.	Speech																		
After prev.	-	“C”																		
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>$[\text{.19}, \text{.07}] \times 8$</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0----- -----2</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.07}] \times 8$						0----- -----2		
Start	Dur.	Timbre	Pitch	Loud.	Tapping															
After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.07}] \times 8$															
					0----- -----2															
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Speech</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>-</td> <td>“D”</td> </tr> </tbody> </table>	Start	Dur.	Speech	After prev.	-	“D”														
Start	Dur.	Speech																		
After prev.	-	“D”																		
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>$[\text{.19}, \text{.01}] \times 10$</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0----- -----2</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.01}] \times 10$						0----- -----2		
Start	Dur.	Timbre	Pitch	Loud.	Tapping															
After prev.	2	Sine	523.25 (C5)	1	$[\text{.19}, \text{.01}] \times 10$															
					0----- -----2															
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Speech</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>-</td> <td>“E”</td> </tr> </tbody> </table>	Start	Dur.	Speech	After prev.	-	“E”														
Start	Dur.	Speech																		
After prev.	-	“E”																		
<table border="1"> <thead> <tr> <th>Start</th> <th>Dur.</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Tapping</th> </tr> </thead> <tbody> <tr> <td>After prev.</td> <td>2</td> <td>Sine</td> <td>523.25 (C5)</td> <td>1</td> <td>$[\text{.91}, \text{.19}, \text{.91}]$</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0 # 2</td> </tr> </tbody> </table>	Start	Dur.	Timbre	Pitch	Loud.	Tapping	After prev.	2	Sine	523.25 (C5)	1	$[\text{.91}, \text{.19}, \text{.91}]$						0 # 2		
Start	Dur.	Timbre	Pitch	Loud.	Tapping															
After prev.	2	Sine	523.25 (C5)	1	$[\text{.91}, \text{.19}, \text{.91}]$															
					0 # 2															
11	Speech	Finished.																		

Table 7.6: The audio queue resulting from a sparsity sonification spec in Section 7.6.1.1. “Q” indicates the index of each sub-queue. “After prev.” means “play after the previous sound” within the same sub-queue. A tapping pattern, $[a, b] \times c$, means a tap sound for a seconds and a pause for b seconds are repeated c times (the last pause is omitted). A tapping pattern, $[a, b, c]$, means a pause for a seconds, a tap sound for b seconds, and a pause for c seconds.

```
11 spec.encoding.time.field("name", "nominal");
```

This `time` channel should use relative timing to allow for playing each data table name before the sound for the corresponding sparsity value.

```
12 spec.encoding.time.scale("timing", "relative");
```

We then specify the `tapSpeed` channel for the quantitative sparsity channel.

```
13 spec.encoding.tapSpeed.field("sparsity", "quantitative");
```

This `tapSpeed` channel has the domain of $[0, 1]$. We want to map this domain to the range of $[0, 5]$ (i.e., zero to five taps per second) for 2 seconds:

```
14 spec.encoding.tapSpeed.scale("domain", [0, 1])
15   .scale("range", [0, 5]).scale("band", 2);
```

Since a higher sparsity value should have a lower speed, we need negative `polarity`:

```
16 spec.encoding.tapSpeed.scale("polarity", "negative");
```

This results in a single tap sound for the sparsity value of 0.1. To play this sound in the middle of the time `band` (two seconds), we set the `singleTappingPosition` property as `middle`:

```
17 spec.encoding.tapSpeed
18   .scale("singleTappingPosition", "middle");
```

To support identifying these tapping sounds at different speeds, we need a `speechBefore` channel for the `name` channel.

```
19 spec.encoding.speechBefore.field("name", "nominal");
```

We do not need a scale description for this `speechBefore` channel in this case.

```
20 spec.encoding.speechBefore.scale("description", "skip");
```

Table 7.6 shows the audio queue compiled from this spec.

species	island	bodyMass
Adelie	Torgersen	3,750
Adelie	Biscoe	4,300
Adelie	Dream	2,900
Chinstrap	Dream	3,450
Gentoo	Biscoe	6,300

Table 7.7: A preview of the `penguins.json` dataset.

7.6.1.2 Kernel density estimation

In exploratory data analysis pipelines, examining the distributions of variables of interest is a common first step. It is important to observe the entirety of a distribution because some distributional information, such as multi-modality, are not captured by summary statistics like mean and variance. In addition to histograms, data analysts often estimate the probability density of a quantitative variable using a kernel smoothing function (i.e., kernel density estimation or KDE). In this example, we want to sonify a KDE of the `bodyMass` variable of the `penguins.json` data [250]. This sonification will encode the density by pitch and the variable's value by time and panning. Then, we repeat this sonification design for different `species` and `islands` of penguins.

The `penguins.json` dataset consists of `species`, `island`, and `bodyMass` fields. The nominal `species` and `island` fields form five combinations as shown in the first two columns of Table 7.7. The `bodyMass` field roughly ranges from 2,500 to 6,500.

First, we create a single-stream spec object, set the description, and assign the data.

```
1 let spec = new Stream();
2 spec.description("The kernel density estimation of body mass by species and
   island.");
3 spec.data("url", "penguins.json");
```

Next, we need to add a `density` transform for the KDE of the `bodyMass` field grouped by `species` and `island`.

```
4 let density = new Density();
5 density.field("bodyMass").extent([2500, 6500])
6   .groupby(["species", "island"]);
7 spec.transform.add(density);
```

This transform results in a new data table that has four columns: `value` (evenly distributed `bodyMass` values, e.g., 2500, 2550, ..., 6450, 6500), `density` (the density estimate of each `value` element), `species`, and `island`.

Third, we use a `continued` tone because we want to sonify continuous KDEs.

```
8 spec.tone.type("default").continued(true);
```

Given this `tone` design, we set the `time`, `pan`, and `pitch` channels. We map the `value` field to `time` and `pan` to give both temporal and spatial senses of sound progression.

```
9 spec.encoding.time.field("value", "quantitative");
10 spec.encoding.pan.field("value", "quantitative");
```

Then, we detail the `scale` of the `time` channel by setting the `length` of each repeated sound to three seconds and indicating the `title` of this `scale` in the scale description.

```
11 spec.encoding.time.scale("length", 3)
12   .scale("title", "Body Mass values");
```

Similarly, we set the `polarity` of the `pan` channel to `positive` and note the same scale `title`.

```
13 spec.encoding.pan.scale("polarity", "positive")
14   .scale("title", "Body Mass values");
```

We encode the `density` field to the `pitch` channel with `positive` polarity and a pitch range of `[0, 700]` (Hz).

```

15 spec.encoding.pitch.field("density", "quantitative")
16     .scale("polarity", "positive")
17     .scale("range", [0, 700])
18     .scale("title", "kernel density");

```

KD estimates usually have infinite decimals (e.g., 0.0124...), which makes it hard to understand when read out (e.g., in the scale description). To prevent reading all the decimals, we specify the read `format` of the density estimates so that they are only read up to the fourth decimal.

```

19 spec.encoding.pitch.format(".4");

```

ERIE uses format expressions supported by D3.js [25].

Now, we repeat this spec design by the `species` and `island` fields using a `repeat` channel.

```

20 spec.encoding.repeat
21   .field(["species", "island"], "nominal")
22   .speech(true).scale("description", "skip");

```

Table 7.8 illustrates the audio queue compiled from this spec. Sub-queue 4 to Sub-queue 8 are the scale descriptions for the `time`, `pan`, and `pitch` channels with audio legends. Sub-queues 10 to 24 represent the specified KDE sonification for each combination of the `species` and `island` values.

7.6.1.3 Model fit sequence

After fitting a linear regression model, a necessary task is to check the model fit by examining the residuals. Common methods for residual analysis include a residual plot (residual vs. independent variable) and a QQ plot (residual vs. normal quantile). For this task, we assume that we have already fitted a linear regression model

Q.	Type	Sound										
1	Speech	To stop playing the sonification, press the X key.										
2	Speech	Kernel density of Body Mass by Species and Island.										
3	Speech	This stream has the following sound mappings.										
4	Speech	The Body Mass value is mapped to time. The duration of the stream is 3 seconds.										
5	Speech	The Body Mass value is mapped to pan. The domains values from 2500 to 6500 are mapped to										
6	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Sine</td> <td>523.25</td> <td>-1 (leftmost)</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0	Sine	523.25	-1 (leftmost)	1
		Start	Timbre	Pitch	Pan	Loudness						
0	Sine	523.25	-1 (leftmost)	1								
<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0.6</td> <td>Sine</td> <td>523.25</td> <td>1 (rightmost)</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0.6	Sine	523.25	1 (rightmost)	1		
Start	Timbre	Pitch	Pan	Loudness								
0.6	Sine	523.25	1 (rightmost)	1								
7	Speech	The Kernel density is mapped to pitch. The domains values from 1.654e-30 to 0.0011 are mapped to										
8	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Sine</td> <td>0</td> <td>0 (center)</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0	Sine	0	0 (center)	1
		Start	Timbre	Pitch	Pan	Loudness						
0	Sine	0	0 (center)	1								
<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0.6</td> <td>Sine</td> <td>700</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0.6	Sine	700	0	1		
Start	Timbre	Pitch	Pan	Loudness								
0.6	Sine	700	0	1								
9	Speech	This sonification sequence consists of 5 parts.										
10	Speech	Stream 1. Adelie and Torgersen.										
11	Speech	Start playing.										
12	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Sine</td> <td>7.3928</td> <td>-1</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0	Sine	7.3928	-1	1
		Start	Timbre	Pitch	Pan	Loudness						
		0	Sine	7.3928	-1	1						
		<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0.015</td> <td>Sine</td> <td>9.0813</td> <td>-0.99</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0.015	Sine	9.0813	-0.99	1
		Start	Timbre	Pitch	Pan	Loudness						
0.015	Sine	9.0813	-0.99	1								
...												
<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>2.88</td> <td>Sine</td> <td>6.0194</td> <td>0.92</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	2.88	Sine	6.0194	0.92	1		
Start	Timbre	Pitch	Pan	Loudness								
2.88	Sine	6.0194	0.92	1								
<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>Sine</td> <td>1.4486</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	3	Sine	1.4486	1	1		
Start	Timbre	Pitch	Pan	Loudness								
3	Sine	1.4486	1	1								
...												
22	Speech	Stream 5. Gentoo and Biscoe.										
23	Speech	Start playing.										
24	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Sine</td> <td>0.0000</td> <td>-1</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	0	Sine	0.0000	-1	1
		Start	Timbre	Pitch	Pan	Loudness						
0	Sine	0.0000	-1	1								
<table border="1"> <thead> <tr> <th>Start</th> <th>Timbre</th> <th>Pitch</th> <th>Pan</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>Sine</td> <td>9.2425</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Start	Timbre	Pitch	Pan	Loudness	3	Sine	9.2425	1	1		
Start	Timbre	Pitch	Pan	Loudness								
3	Sine	9.2425	1	1								
25	Speech	Finished.										

Table 7.8: The audio queue resulting from a kernel density estimate sonification spec in Section 7.6.1.2. “Q” indicates the index of each sub-queue. The pitch values (range from 0 to 700) are low because they are representing the both-side tails of each estimated density distribution.

of Sepal Length on Petal Length (*Petal Length* \sim *Sepal Length*), and computed the residuals. For the residual plot, we use a `residuals` dataset with two fields: `sepalLength` (independent variable) and `residuals`. For the QQ plot, we use a `qq` dataset with two fields: `normalQuantile` and `residuals`¹¹. With these datasets, we want to create two sequenced sonifications for residuals and comparison to normal quantiles (i.e., recognizing their trends).

We first register the datasets.

```
1 let qqData = [...];
2 let qqDataset = new Dataset("qq");
3 qqDataset.set("values", qqData);
4 let residualData = [...];
5 let residualDataset = new Dataset("residuals");
6 residualDataset.set("values", residualData);
```

Second, we define a sonification for a residual plot. When errors of a model fit are unbiased, the residuals are evenly distributed along values of the independent variable and on both sides of the central line indicating 0 error. With this residual plot sonification, we want to capture the evenness of residual distribution by mapping the residuals to `modulation` index and `pan` channel. In this way, a larger residual is mapped to a more warped sound, and a negative residual is played on the left side and a positive residual is played on the right side. A good model fit will generate a sonification where the sound quickly (e.g., 150 sound points within 5 seconds) moves between different modulation index and pan values, making it harder to differentiate their degrees of warping and spatial positions. In contrast, a bad model fit will generate an audio graph where listeners can easily sense some groups of sounds sharing the same degree of warping on a certain spatial position.

¹¹Alternatively, these two datasets can be a single dataset. Here, we use two datasets for demonstration purposes.

We use a `time` channel for the `sepalLength` field.

To do so, we create a single stream with the `residuals` dataset.

```
7 let residualSpec = new Stream();
8 residualSpec.name("Residuals");
9 residualSpec.data.set(residualData);
```

For the tone, we use an FM synth, named `fm1`.

```
10 let synth = new SynthTone("fm1");
11 synth.type("FM");
12 residualSpec.tone.set(synth);
```

The residual sonification uses a `time` channel for the `sepalLength` values and `modulation` index and `pan` channels for the residuals that roughly range from -2.5 to 2.5 . This design is specified as below:

```
13 residualSpec.encoding.time
14   .field("sepalLength", "quantitative")
15   .scale("timing", "absolute").scale("length", 5)
16   .scale("band", 0.15).format(".4");
17 residualSpec.encoding.modulation
18   .field("residual", "quantitative")
19   .scale("domain", [-2.5, 0, 2.5])
20   .scale("range", [4, 0.001, 4]).format(".4");
21 residualSpec.encoding.pan
22   .field("residual", "quantitative")
23   .scale("domain", [-2.5, 0, 2.5])
24   .scale("range", [-1, 0, 1]).format(".4");
```

Next, we specify a QQ plot sonification. A good model fit should also exhibit normally distributed residuals. By plotting the quantiles of the residuals against the expected quantiles of a normal distribution (range from 0 to 1), we want to observe how much the residuals deviate from the expectation that they are normally distributed. A visual QQ plot shows the gap between the theoretical and observed distribution by plotting them in a Cartesian space, which is the same format used for a residual plot at high level. However, a sonification author may need to directly

encode the gap because overlaying the normal and residual distributions with different pitches or volumes may make it harder to decode the gap, indicating the need for specifying a sonification design **independently from visualization (C1)**. Thus, we compute the normalized residuals' `deviation` (within 0 to 1) from normal quantiles to directly encode the gap. This transform is done using the below `calculate` transforms, resulting in two additional fields: `normalized` and `deviation`.

```

25 let qqSpec = new Stream();
26 qqSpec.name("QQ plot");
27 qqSpec.data.set(residualData);
28 // normalize residuals using its min (-2.477468) and max (2.495122).
29 let normalized = new Calculate("(datum.residual + 2.477468)/(2.495122 + 2.4
    77468)", "normalized");
30 let deviation = new Calculate("datum.normalized - datum.normalQuantile", "
    deviation");
31 qqSpec.transform.add(normalized).add(deviation);

```

Then, we map the `normalQuantile` to `time`, the magnitude of the `residual` to `pitch`, and the `deviation` to `pan`. These mappings will produce sounds that are spatially centered when the deviation is smaller but are played from left or right when the signed deviation is bigger.

```

32 qqSpec.tone.continued(false);
33 qqSpec.encoding.time
34   .field("normalQuantile", "quantitative")
35   .scale("length", 4).scale("band", 0.2)
36   .scale("title", "Normal Quantile").format(".4");
37 qqSpec.encoding.pitch
38   .field("residual", "quantitative")
39   .scale("polarity", "positive")
40   .scale("title", "Residual").format(".4");
41 qqSpec.encoding.pan
42   .field("deviation", "quantitative")
43   .scale("domain", [-0.2, 0, 0.2])
44   .scale("range", [-1, 0, 1])
45   .scale("title", "Deviation from normal distribution")
46   .format(".4");

```

Lastly, we merge the residual and QQ streams (`residualSpec`, `qqSpec`) into a se-

quenced stream.

```
47 let modelFit = new Sequence(residualSpec, qqSpec);
48 modelFit.description("The residuals of a linear regression model of Sepal
    Length on Petal Length.");
```

This spec results in the sonification described in Table 7.9.

7.6.2 Replication of Prior Use Cases

We replicate several sonification use cases (e.g., applications and data stories) and extend their features to demonstrate how feasibly creators can use ERIE in development settings. We include the ERIE specs used for the below replications in our example gallery¹².

7.6.2.1 Audio Narrative

Audio Narrative [64] divides a temporal line chart into segments that represent different data patterns, such as increase, decrease, and no change, and offers a sonification and speech description for each segment. To show how ERIE can be used in such applications to generate sonifications, we created an example case that Audio Narrative could create by using ERIE for sonification and speech generation, as shown in Figure 7.2. We used a ‘*stocks.json*’ dataset [250] for this replication. We use two variables, *stock price* and *date*, from this dataset.

Suppose an Audio Narrative-like application already has a line chart segmented and relevant speech texts generated. The next task is to create sounds for those segments and speech texts. Using ERIE, the application can simply write a sonification

¹²<https://see-mike-out.github.io/erie-editor/>

Q.	Type	Sound																												
1	Speech	To stop playing the sonification, press the X key.																												
2	Speech	The residuals of a linear regression model of Sepal Length on Petal Length.																												
3	Speech	This sonification sequence consists of 2 parts.																												
4	Speech	Stream 1. Residual plot.																												
5	Speech	The first stream has the following sound mappings.																												
6	Speech	The Sepal Length is mapped to time. The duration of the stream is 5 seconds.																												
7	Speech	The residual is mapped to pan. Residual values are mapped as -2.5																												
8	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>-1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	pan	0	0.3	fm1	523.25	1	-1																
Start	Duration	Timbre	Pitch	Loudness	pan																									
0	0.3	fm1	523.25	1	-1																									
9	Speech	0 (zero)																												
10	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	pan	0	0.3	fm1	523.25	1	0																
Start	Duration	Timbre	Pitch	Loudness	pan																									
0	0.3	fm1	523.25	1	0																									
11	Speech	2.5																												
12	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	pan	0	0.3	fm1	523.25	1	1																
Start	Duration	Timbre	Pitch	Loudness	pan																									
0	0.3	fm1	523.25	1	1																									
13	Speech	The residual is mapped to modulation. Residual values are mapped as -2.5																												
14	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Modulation</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>4</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Modulation	0	0.3	fm1	523.25	1	4																
Start	Duration	Timbre	Pitch	Loudness	Modulation																									
0	0.3	fm1	523.25	1	4																									
15	Speech	0 (zero)																												
16	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Modulation</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>0.001</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Modulation	0	0.3	fm1	523.25	1	0.001																
Start	Duration	Timbre	Pitch	Loudness	Modulation																									
0	0.3	fm1	523.25	1	0.001																									
17	Speech	2.5																												
18	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Modulation</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>4</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Modulation	0	0.3	fm1	523.25	1	4																
Start	Duration	Timbre	Pitch	Loudness	Modulation																									
0	0.3	fm1	523.25	1	4																									
19	Speech	Start playing.																												
20	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loud.</th> <th>Pan</th> <th>Modulation</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.15</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>0.0841</td> <td>0.3372</td> </tr> <tr> <td colspan="7" style="text-align: center;">...</td> </tr> <tr> <td>4.85</td> <td>0.15</td> <td>fm1</td> <td>523.25</td> <td>1</td> <td>-0.4721</td> <td>1.8888</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loud.	Pan	Modulation	0	0.15	fm1	523.25	1	0.0841	0.3372	...							4.85	0.15	fm1	523.25	1	-0.4721	1.8888
		Start	Duration	Timbre	Pitch	Loud.	Pan	Modulation																						
		0	0.15	fm1	523.25	1	0.0841	0.3372																						
...																														
4.85	0.15	fm1	523.25	1	-0.4721	1.8888																								

Table 7.9: The audio queue resulting from a model fit sonification spec in Section 7.6.1.3. “Q” indicates the index of each sub-queue.

Table 7.9 continued.

Q.	Type	Sound																								
21	Speech	Stream 2. QQ plot.																								
22	Speech	The second stream has the following sound mappings.																								
23	Speech	The Normal Quantile is mapped to time. The duration of the stream is 4 seconds.																								
24	Speech	The Deviation from normal distribution is mapped to pan. Deviation from normal distribution values are mapped as -0.2																								
25	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>523.25</td> <td>1</td> <td>-1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Pan	0	0.3	Sine	523.25	1	-1												
Start	Duration	Timbre	Pitch	Loudness	Pan																					
0	0.3	Sine	523.25	1	-1																					
26	Speech	0 (zero)																								
27	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>523.25</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Pan	0	0.3	Sine	523.25	1	0												
Start	Duration	Timbre	Pitch	Loudness	Pan																					
0	0.3	Sine	523.25	1	0																					
28	Speech	0.2																								
29	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>523.25</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Pan	0	0.3	Sine	523.25	1	1												
Start	Duration	Timbre	Pitch	Loudness	Pan																					
0	0.3	Sine	523.25	1	1																					
30	Speech	The Residual is mapped to pitch. The minimum value -2.477 is mapped to																								
31	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>207.65</td> <td>1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	0	0.3	Sine	207.65	1														
Start	Duration	Timbre	Pitch	Loudness																						
0	0.3	Sine	207.65	1																						
32	Speech	and the maximum value 2.495 is mapped to																								
33	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.3</td> <td>Sine</td> <td>1600</td> <td>1</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	0	0.3	Sine	1600	1														
Start	Duration	Timbre	Pitch	Loudness																						
0	0.3	Sine	1600	1																						
34	Speech	Start playing.																								
35	Tone	<table border="1"> <thead> <tr> <th>Start</th> <th>Duration</th> <th>Timbre</th> <th>Pitch</th> <th>Loudness</th> <th>Pan</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.2</td> <td>Sine</td> <td>207.65</td> <td>1</td> <td>-0.0167</td> </tr> <tr> <td colspan="6">...</td> </tr> <tr> <td>3.8</td> <td>0.2</td> <td>Sine</td> <td>1600</td> <td>1</td> <td>0.0167</td> </tr> </tbody> </table>	Start	Duration	Timbre	Pitch	Loudness	Pan	0	0.2	Sine	207.65	1	-0.0167	...						3.8	0.2	Sine	1600	1	0.0167
		Start	Duration	Timbre	Pitch	Loudness	Pan																			
		0	0.2	Sine	207.65	1	-0.0167																			
...																										
3.8	0.2	Sine	1600	1	0.0167																					
36	Speech	Finished.																								

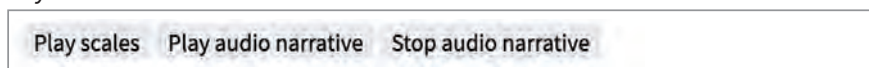
A. Auditory encoding customization

A listener can set an encoding channel to map the stock price variable and the scale range for that channel. It is currently set to pitch with a range of 200 Hz to 1,600 Hz.



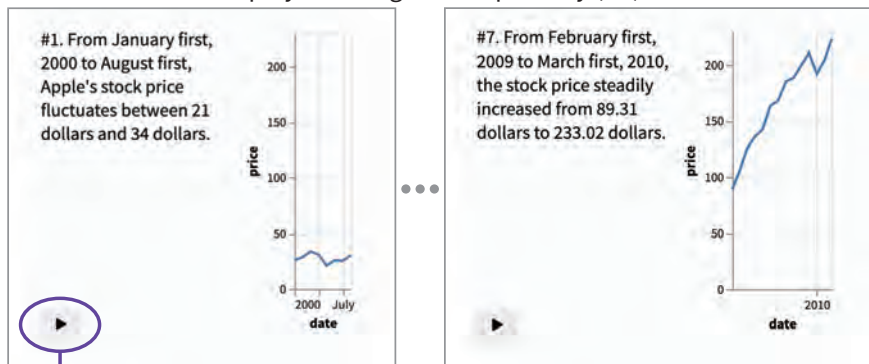
B. Player options

The listener can play and stop the scale description and the Audio Narrative sonification. These buttons are accessible with a screen reader and keyboard interaction.



C. Audio narrative segments

The listener can also play each segment separately (C1).



C1. Play button for a segment

Figure 7.2: Our replication and extension of Audio Narrative [64] using ERIE. In addition to the originally offered sequencing and speech description, we included options for using different encoding channels (A) and playing the scale description (B).

spec for each segment as below:

```

1 { "description": "...",
2   "data": [ /* Segment i data */ ],
3   "tone": { "continued": true },
4   "encoding": {
5     "time": { "field": "date", ... },
6     "pitch": { "field": "stock price", ... } } }

```

By setting a `description`, the application can play the speech for each segment. Having the above as a template, the application then merge the specs for all the segments as a `sequence`:

```

1 { "sequence": [{ /* Segment 1 stream */ }, ...
2               { /* Segment N stream */ }],
3   "config": {
4     "forceSequenceScaleConsistency": { "pitch": true },
5     "skipScaleSpeech": true }}

```

The `forceSequenceScaleConsistency` in the `config` forces the segment streams to use the same `pitch` scale. As sonifications can benefit from the user’s ability to personalize design choices [127], we extend this Audio Narrative case by allowing for using a loudness or pan channel to encode a variable and adjusting the scale range of those channels. Furthermore, we add an option that separately plays the scale descriptions of a sonification. ERIE supports this by using a `skipScaleSpeech` option in the `config`.

7.6.2.2 Chart Reader

Given a visualization, Chart Reader [72] enables hover interaction that reads out values and generates a sonification for the selected data mark(s). Furthermore, Chart Reader supports creating several “data insights” that allow a sonification author to draft more customized text messages, similar to the chart segments supported by Audio Narrative. We replicate this use case by reusing the above Audio Narrative

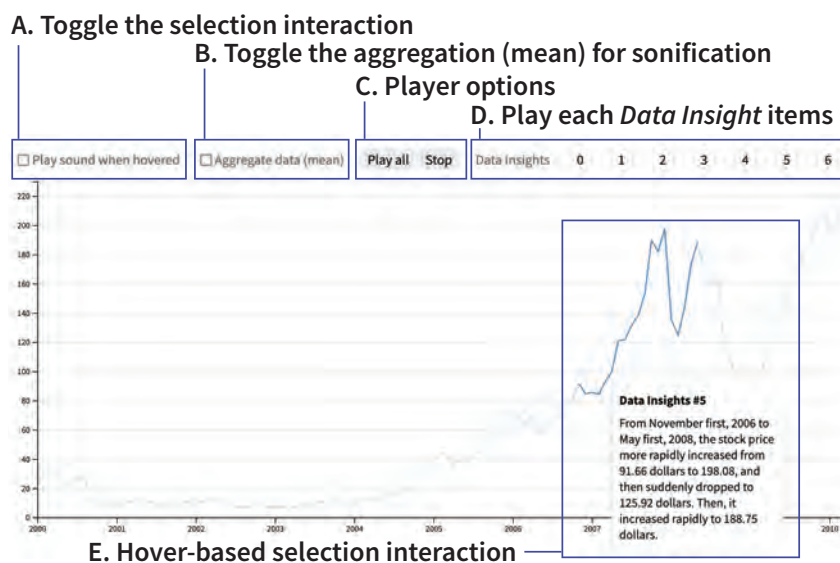


Figure 7.3: Our replication and extension of Chart Reader [72] using ERIE. We further included user options for toggling the hover/selection interaction (A) and aggregation (B).

replication, given their underlying structural similarity (segmentation of a chart with descriptive text), as shown in Figure 7.3.

In this case, the sonification and description text of a chart segment is played whenever the corresponding part in the chart is selected, or the button for the segment is triggered via a mouse or keyboard. This uses the same segment template spec as Audio Narrative replication, but they are not sequenced. We set the pitch scale's *domain* as the minimum and maximum values of the sonified variable so that the segments can share the same pitch scale even though they are not sequenced in the same specification. This technique is often used in data visualization cases as well. We further include several customization options for toggling the hover interaction and data aggregation. By reusing the above sequence, we also include an option to play all the 'data insight' segments.

7.6.2.3 Nine Rounds a Second

The *Nine Rounds a Second* article [245] covers the mass shooting case in Las Vegas in 2017 where the gunman was known to have had a rapid-fire gun. This article compares the Las Vegas case with the mass shooting case in Orlando in 2016 and the use of automatic weapons. In this article, a dot plot visualizes the shooting count over time to show how fast shots were fired. To make it even more realistic, the authors of this article included a sonification that mimics actual gun sounds.

We replicate this news article sonification by mapping the shooting time to a `time` channel and the three cases (Las Vegas, Orlando, and automatic weapon) to a `repeat` channel, as shown in Figure 7.4. We use an electronic drum's `clap` sound that ERIE's player supports as a preset because it sounds similar to a gunshot sound. The original article had an interaction that when the name of a case is selected, it plays only the relevant part. To support that, we use the `audioQueue.play(i, j)` method, so that the player only plays from the `i`-th sub-queue to `j`-th sub-queue. In this case, the first sub-queue is the name of a case, and the last sub-queue is the sonification sound (two sub-queues in total).

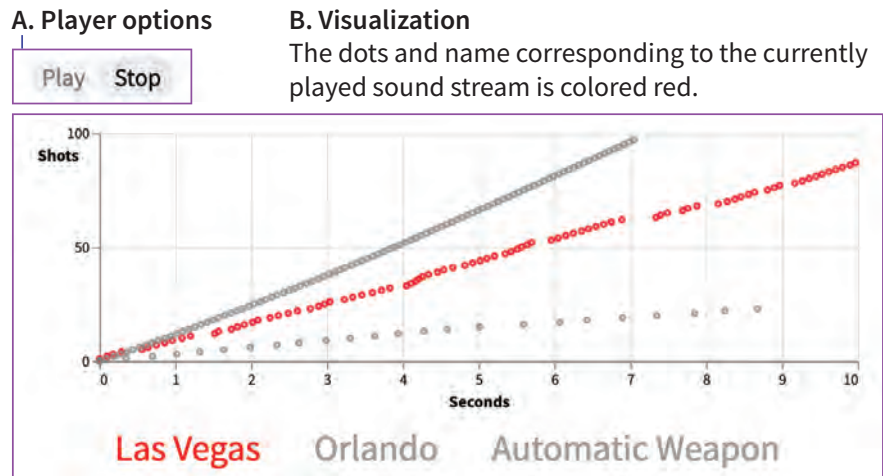


Figure 7.4: Our replication of the *Nine Rounds a Second* article [245] using ERIE.

7.7 Discussion

We contribute ERIE, a declarative grammar for data sonification, with five design goals: independence as a sonification grammar, expressiveness, data-driven syntax, compatibility with audio standards, and extensibility of functionalities. Below, we briefly discuss remaining technological challenges, and then we motivate future sonification research that could use ERIE.

7.7.1 Technological Hurdles

While developing ERIE, we faced two major technical hurdles in using the Web Audio and Speech APIs. First, there is no standard API that can capture (i.e., generating pure audio files from the source) the sound generated using those APIs. Instead, users need to use third-party audio capture applications or record sound as it is being played out of the device (which also records room noise and causes distortions due to audio feedback). Thus, we implemented a workaround Chrome exten-

sion¹³ using Chrome-specific APIs. Second, speech sounds generated using the Web Speech API cannot overlap which limits ERIE’s expressiveness, such as the potential to overlay different streams with speeches and tones. Thus, related technological extensions to those APIs could help express a more diverse set of audio graphs.

7.7.2 Potential Use Cases of ERIE

We expect our implementation of ERIE (compiler, player, and extension APIs) to facilitate various future work on data sonification research and tooling. In addition to the use cases like sonification for detecting model fit (which might be extended to properties like model convergence), sonification authoring applications, and popular media (Section 7.6), future sonification research could use ERIE to ask questions related to, for example, perceptual intuitiveness and effectiveness of different sonification strategies (e.g., [29, 69]). Given that sonification design specs expressed in ERIE can be parameterized as a declarative grammar, sonification researchers could use ERIE to more systematically generate different stimuli according to their experiment conditions. Such research will expand understanding around which audio graph formats are best suited for different tasks or auditorily pleasant, providing foundations for building intelligent tools like sonification recommenders. Furthermore, future sonification tools for data analysis or narrative authoring could use ERIE as their internal representation to maintain user-specified designs. To support sonification researchers and developers to test out ERIE, we provide an online editor for ERIE¹⁴.

¹³<https://github.com/see-mike-out/erie-chrome-ext>

¹⁴<https://see-mike-out.github.io/erie-editor/>

7.7.3 Future Work

ERIE is our first step of an expressive declarative grammar for data sonification. Future work should extend ERIE to support more dynamic use cases, such as interactivity, streaming data, and different audio environments.

7.7.3.1 Interactive sonification

Interactivity is often necessary for data analysis because one static data representation cannot provide a full picture. While it is possible to use ERIE in interactive user interfaces with customizability as we demonstrated (Section 7.6.2), ERIE could better support interactive data sonification with native expressions. A prerequisite to developing an interactive grammar for data sonification is some understanding of how a sonification listener would trigger a user interaction and receive its feedback using different modalities. For instance, various approaches to using a keyboard, speech recognition, tabletop screens, or mobile haptic screens for interactive sonification are fruitful topics like personalized sonifications for future research to explore (e.g., [71, 74]).

7.7.3.2 Expressing sonifications for streaming data

Sonification has been used for various real-time streamed data from traditional Geiger counters to audio graphs for physics [123]. While it is relatively simple for visualization to show existing data points and newly received data points, sonification-based tools may need to build a notion of “existing” given the transient characteristic of sound. For example, a visualization dashboard can express newly received data by adding corresponding visual marks, and the viewers can easily compare them with the existing visual marks. However, a sonification monitor may need to play

sounds for some past data points, announce the auditory scales, or use notifications for some signals, depending on the task that the listeners want to achieve. Thus, future work should ask how to indicate and contextualize newly arrived data points, what portion of existing data points should be played again if needed, and how to auditorily imply that a system is waiting on new data.

7.7.3.3 Intelligent authoring tools for data sonification

As a declarative grammar, ERIE can make it easier to create data sonifications by allowing developers to declare sonification designs with a few keywords rather than leaving them tedious jobs like inspecting online code and adjusting it to get ad-hoc solutions. To design effective data sonifications, developers still need to learn relevant knowledge from empirical studies, just as being able to use visualization grammars like D3.js [25], Vega-Lite [96], and ggplot2 [26] do not necessarily mean one can easily create effective visualizations. To support developers in authoring useful sonifications, future work could explore more intelligent approaches like automated design recommenders for different purposes like data analytics, data journalism, and data art.

7.7.3.4 Beyond web environments

Data sonification can also be useful for other environments like statistical programming and server-side applications. For example, ERIE player for R Studio (a popular integrated development environment for R) could benefit building tools for statistical sonifications like those described above. As R Studio is backed by Chromium (the same web engine for Chrome and Edge), ERIE's web player may need to be extended slightly to support this environment. To support server-side production of

data sonifications, direct generation of raw pulse-code modulation data (digital representation of analog signals) [256] would be useful.

7.7.4 Limitations

While our primary contribution is the ERIE grammar, a usable player could make it easier to learn the grammar and apply it to different applications. We provide an online player for sonifications backed by ERIE with baseline functionalities like playing a single queue and showing audio queue tables. As ERIE is an open-source project, extensions for more player controls (e.g., playing a single sound) could benefit sonification developers and users with respect to debugging and navigation. Next, intending ERIE as a low-level toolkit for sonification developers to use, we prioritized independence from visualization, expressiveness, and compatibility with audio programming standards. As ERIE is not a walk-up-and-use tool, future work could benefit from reflecting on use cases from longer term observations of developer communities.

7.8 Conclusion

While data sonification is an important data representation method for various domains, such as immersive data storytelling and accessible data visualization, current tooling offers compartmentalized support, requiring authors to switch between different tools. As a more comprehensive toolkit for authoring audio graphs, I presented ERIE, a declarative grammar for data sonification. ERIE abstracts sonification designs in terms of data, tone (overall sound quality), and encoding (mapping from data to auditory features) with extensibility through audio sampling and fil-

ters. To reify this approach, I developed a compiler and player for ERIE intended for web-based audio environments. The ERIE grammar and its compiler helped with exploring possible sonification designs in a more concise way, allowing for coming up with novel sonification designs. I also demonstrated how ERIE can support developing end-user applications with enhanced reusability and customizability.

Chapter 8

Discussion: Toward a Framework for Multi-context Data Visualization

While I have looked at a few distinct contexts—responsive visualization and data sonification—separately, a bigger challenge is to produce visualizations for arbitrary combinations of those user contexts because those context factors are often intertwined in real world. For example, responsive visualization considers different device types but not people with Blindness, so responsive visualization should also be accessible. To create accessible responsive visualization, authors need to consider their non-visual representations like alternative text and sonification are faithful to each device-specific version. In case where desktop and mobile versions share different encodings or different levels of aggregation, then multiple non-visual versions are likely to be necessary. Similarly, audience retargeting for colleagues with limited data literacy may need to consider several device types, considering situations where those colleagues use different devices to access visualizations. Failing to support combination of user context factors may result in exclusion of certain audience groups (e.g., inaccessible on mobile phones) as well as making authors time and effort in vein (e.g., no one likes visualizations shared on a collaboration platform). This can overwhelm visualization authors given current tools as they

have to switch between different tools and learn design requirements and technical skills for a broader range of user contexts. To effectively manage those contexts, authors also have to be able to explicitly define a set of user contexts they want to support—beyond screen size breakpoints.

To support visualization authors to deal with multiple user contexts, future tools need to be made easier and streamlined, ideally with fewer or a single tool. An important first step to streamlining tools is to devise a method to abstractly express visualization user contexts and tradeoffs in designing them. Tools connected by consistent underlying expressions will allow tool developers and authors to flexibly define user contexts they want to support instead of relying on those predefined by a tool. As previous chapters demonstrate, a consistent syntax for design representation is a cornerstone of more advanced tools like automated recommenders and design fixers for multi-context visualization.

In addition, future research will inform us with better evidence in perception of various data representations (e.g., sonification, tactilization, etc.), and we may observe nascent context types. Heterogeneous or ad-hoc approaches to encoding future advances in multi-context data visualization will only make it unnecessarily challenging to consolidate them into useful tools. Thus, we should be able to express different multi-context data visualization problems in a generalized and systematic way so as to flexibly apply relevant technologies, identify gaps in current practices, and guideline further research.

The goal of this chapter is to illustrate a blueprint for such a systematic, generalized method for expressing visualization user contexts and design tradeoffs. I first define a *multi-context data visualization design problem* as a problem by solving

which we can obtain an authoring tool or a design recommender that support creating visualizations for a certain set of user contexts. Solving this problem means finding constraints representing design tradeoffs for a set of user contexts with a cost model.

Inspired by the previous chapters, a promising approach to generalizing multi-context data visualization design is *accommodation with preservation*—accommodating designs for each context while preserving support for users’ ability to obtain insights. For example, CICERO and ERIE let us accommodate visualization designs for targeted user contexts (device types and sonification). The task-oriented insight measures inform what kinds of low-level tasks we need to preserve in creating multi-context visualizations.

This approach has several expected benefits for tool developers and researchers as well as visualization authors. Developers and researchers for multi-context visualization systems can easily come up with tools that reflect novel context combinations instead of building an entirely new tool. While I do not expect individual authors to define contexts from scratch, the flexibility in defining various context types and their relationships could make it easier for them to fine-tune a given tool. For example, through an interactive end-user interface, authors could toggle different device types or modalities and adjust weighting terms of existing visualization constraint models like Draco [148] or GraphScape [84], depending on their communication purposes. As this chapter presents a preliminary proposal about generic expressions for multi-context visualization, future work is warranted to develop and reify the idea and turn it into useful and usable applications.

8.1 Problem Formulation

A multi-context data visualization design problem (D) is composed of a context set (T), an accommodation constraint set (A), a preservation constraint set (P), and a cost model (C):

$$D = (T, A, P, C) \quad (8.1)$$

A context set (T) is a list of context names with their attributes, which specifies what kinds of user contexts a problem aims to support. An example context set for a responsive visualization problem could consist of a list of (*desktop, smartphone, tablet, printing, social-media*), where some versions share the same attributes. For instance, *social-media* and *smartphone* are both for mobile phone devices, while *desktop*, *smartphone*, and *tablet* contexts share a browser environment that affords interactivity.

An accommodation constraint set (A) is a collection of design constraints that define desirable qualities *per each context* in terms of feasibility, applicability, and effectiveness. For example, an accommodation constraint for a smartphone version given a responsive visualization problem could be the maximum screen width (soft). A sonification version under an accessible visualization problem could have an accommodation constraint for available auditory channels and their physically feasible ranges (hard).

A preservation constraint set (P) is a collection of design constraints that identify qualities to be preserved *across different contexts*. For instance, responsive visualization versions need to deliver the same visual insights as much as possible. In

some cases, a certain insight type may take higher priority than other types depending on communication goals (e.g., comparison over trend). There are cases where authors must use the same color scheme if they are provided via the same outlet (e.g., as a part of a news article).

Lastly, a cost model (C) is a function that assigns the cost for the violation of each soft constraint. For instance, violating a constraint for using the same mark types may have a smaller cost than violating an insight-loss constraint so that design candidates with smaller insight loss values are preferred over those sharing different mark types. For data sonifications, sequencing multiple simpler streams could be preferred to having too many auditory encoding channels. While the cost model is closely associated with soft constraints, the separation of a cost model from the constraints is to clarify that the same constraint can have different costs depending on a specific problem. For instance, a constraint for limiting data aggregation on large screen devices could have less cost for communicative visualizations than for analytic visualizations.

8.1.1 Defining a Context Set

A context set T includes the names of contexts like ‘desktop visualization’ or ‘mobile sonification.’ While a constraint can specify the list of contexts to control, it should also be able to indicate a subset of the contexts by their shared characteristics, such as contexts for mobile phone and contexts for sonification. An ability to define attributes of contexts can allow for expressing such characteristic-based subset. For instance, a responsive visualization and sonification problem can have a context set of (*desktop-visual*, *phone-visual*, *desktop-audio*, *phone-audio*). Then, we can specify

their attributes as below:

context-attribute(desktop-visual, desktop)
context-attribute(desktop-visual, visual)
context-attribute(phone-visual, phone)
context-attribute(phone-visual, visual)
context-attribute(desktop-audio, desktop)
context-attribute(desktop-audio, audio)
context-attribute(phone-audio, phone)
context-attribute(phone-audio, audio)

In this way, a constraint for indicating that sonification versions must use auditory channels can specify relevant contexts with their attributes (i.e., *audio*).

8.1.2 Defining Accommodation Constraints

An accommodation constraint articulates desired qualities per context. An accommodation constraint is defined in terms of *context*, *strength*, *target*, *direction*, and *value*:

$$\textit{Accommodate} = (\textit{context}, \textit{strength}, \textit{target}, \textit{direction}, \textit{value}, \textit{condition}) \quad (8.2)$$

The *context* of a constraint indicates one or more contexts from the context set, T , that the constraint applies to. The *strength* can be either *hard* or *soft*. A hard constraint cannot be violated, and violating a soft constraint imposes a cost. Example hard constraints include not exceeding a certain size limit for a smartphone visual-

ization and using auditory channels for a sonification. A soft constraint might limit the number of interaction features on a smartphone version. Next, the *target* refers to what visualization elements are affected by this constraint, such as task-oriented insight, layout, mark, etc. The *direction* and *value* specify how to satisfy the constraint. For numeric elements like the width of chart or the number of encoding channels, a *below* direction considers the *value* as the upper bound, and an *above* direction considers it as the lower bound. A *match* direction indicates that specified versions should try to (soft) or must (hard) have the exact value. An *include* direction indicates that the actual value of a design should try to or must include the constraint's value (as a range or a set).

For a closer look, consider the following examples. A responsive visualization problem can have an accommodation constraint for smartphone versions to have limited interactivity (e.g., up to 3 features). This can be written as below:

$$\begin{aligned} &Accommodate(context = smartphone, \\ &\quad strength = soft, \\ &\quad target = interactivity.count, \\ &\quad direction = below, \\ &\quad value = 3) \end{aligned}$$

Then, an application based on this constraint should assess whether the number of interaction features in a candidate design for a smartphone version exceeds the *value* (3), and if so, it should assign a certain cost. Suppose that the cost model indicates that the cost of violating this soft constraint is the difference between the specified value and the number of interaction features in a design candidate mul-

multiplied by 3. For instance, a candidate with four interaction features has a cost of 3 ($= (4 - 3) \times 3$), and another with five interaction features has a cost of 6 ($= (5 - 3) \times 3$). This constraint would not assign cost to candidates with three or less interaction features.

Similarly, an accessible visualization problem can specify to use auditory channels for an audio graph version as a hard constraint:

$$\begin{aligned} \text{Accommodate}(\text{context} = \text{attribute}(\text{audio}), \\ \text{strength} = \text{hard}, \\ \text{target} = \text{channels}, \\ \text{direction} = \text{include}, \\ \text{value} = [\text{time}, \text{pitch}, \text{volume}, \text{pan}]) \end{aligned}$$

This constraint specifies the *contexts* using the *attribute* shared by targeted contexts, which is *audio* graph. Under this constraint, a recommender or an authoring tool must disallow an audio graph candidate from having a channel not included in the specified set (e.g., distortion, color).

8.1.3 Defining Preservation Constraints

A preservation constraint states properties that must to be consistent across multiple contexts. A preservation constraint is defined in terms of *contexts*, *reference* (optional), *strength*, and *target*:

$$\text{Preserve} = (\text{context}, \text{reference}, \text{strength}, \text{target}, \text{condition}) \quad (8.3)$$

Preservation constraints do not need a direction or value because they should be evaluated with respect to the specified *reference* context or with each other. For example, DUPO used the task-oriented insight loss measures in reference to the earliest design. If an automated design recommender suggests a set of responsive versions, then it should use the loss measures to compare and minimize the differences between each pair of versions.

A preservation constraint for maintaining visualization messages for DUPO can be formulated as:

$$\begin{aligned} & \textit{Preserve}(\textit{strength} = \textit{soft}, \\ & \quad \textit{context} = [\textit{desktop}, \textit{smartphone}, \textit{tablet}, \textit{printing}, \textit{thumbnail}], \\ & \quad \textit{reference} = \textit{\$earliest}, \\ & \quad \textit{target} = \textit{insight}) \end{aligned}$$

This constraint allows systems like DUPO sets the *earliest* created version to be the reference version for minimizing changes to insights across other versions.

8.1.4 Solving a Multi-context Visualization Problem

Under this framework, solving a multi-context visualization problem that is defined by the context set and constraints means deriving a cost model and realizing operations for the contexts, constraints, and cost model. There are different approaches to obtaining a cost model. A cost model can be defined as a mathematical model like the task-oriented insights loss measures (Chapter 4) or a heuristics-based loss functions like MobileVisFixer [93]. Alternatively, a cost model could encode empirical study results like GraphScape [84] and Draco [148]. A cost model can also employ a

hybrid approach that combines different methods.

To be applicable, the contexts, constraints, and cost model need to be translated into an operable form like a machine learning model or logical constraints. On the one hand, a reinforcement learning-based recommender can have a policy that encodes different design strategies as actions. Then, the recommender combines the constraints and cost model as a reward function so that it can iteratively compare and assess design candidates for different versions in a way that minimizes the cost. For instance, if visual and auditory histograms are chosen (precondition or environment), their task-oriented insight preservation is assessed (interpreter), based on which the model attempts different encodings (actions). This process is iterated until the model reaches the minimal cost. On the other hand, logic programming approaches like Answer Set Programming [162] can convert the constraints into propositions, rules, and constraints that consider the contexts. It is worth noting that while my framework specifies constraints as desired qualities, logic programming constraints often take a form that indicates undesirable cases. For example, for a constraint that keeps a text size of 13 points or lower, a system should detect whether a design has a text size above 13.

8.2 Example Cases

To demonstrate the feasibility of this proposal, I illustrate how to define contexts and constraints for two example problems: accessible responsive visualization and audience retargeting on multiple devices.

8.2.1 Accessible Responsive Visualization

For communicative visualizations like those on news outlets, it is common to produce responsive visualizations that are accessible. To support authors who need to create accessible responsive visualizations on a daily basis, suppose that a developer wants to solve this problem and build a design recommender that takes datasets and some design preferences and produce a set of visualization versions for desired contexts. Assuming that an author wants to support desktop and smartphone devices for simplicity, they need the following contexts:

$$\text{Context} = (\text{desktop-visual}, \text{desktop-alt}, \text{desktop-audio}, \\ \text{smartphone-visual}, \text{smartphone-alt}, \text{smartphone-audio}),$$

such that each device can support a *visualization*, *alternative text*, and *audio* graph. While this problem can be further simplified as two visualizations for the device types, an alternative text, and a sonification, I separately specify those combinations in case where desktop and smartphone versions have significantly different visual representations, and they need alternative text and sonification reflecting that difference.

Then, we can specify accommodation and preservation constraints as shown in Table 8.1 and Table 8.2, respectively, with a heuristics-based cost model. Accommodation constraints attempt to specify requirements for the size of a visualization, the lengths of an alternative text and a sonification, and applicable encoding channels by different modalities. Preservation constraints specify that visualization versions should share the same level of support for task-oriented insights and sonifica-

tion versions should do the same in reference to the corresponding visualization versions.

The cost model defined here should not be understood as representing an optimal case, rather it is meant for demonstrating the concept. For example, using too many interaction features on a smartphone version (weighted by 2) is discouraged than slight height differences (weighted by 0.02). For sonification versions, using uncommon channels (other than pitch, volume, and pan) is discouraged with a weight of 3. Combined with some generic effectiveness constraints for visualizations, sonifications, and alternative texts, a resulting tool receives data characteristics and encoding channel preferences and returns a set of data representations for the specified contexts. Section 8.3.1 will detail this pipeline.

8.2.2 Audience Retargeting on Multiple Devices

In organizational settings, data analysts often need to repurpose their analytic visualizations into simpler formats [57, 258]. Typically, data analysts arrive at an insight to deliver by configuring their analytic visualization like those in dashboards [57]. Suppose an organization where the data analysts need to communicate with non-analyst colleagues on a collaboration platform like Slack or Microsoft Teams or give a presentation on a large screen like TV screens. To be aligned with such workflow, the visual analytic tool should support generating two versions for threaded conversation applications (e.g., Slack or Teams) on desktop and mobile devices and a presentation version on screens larger than typical desktop monitors, comprising

Table 8.1: The accommodation constraints for the accessible responsive visualization problem described in Section 8.2.1. \$Difference means the gap between the constraint’s value and a candidate’s value

Context	Strength	Target	Direction	Value	Description	Cost
[Any]-Visual	Hard	Representation	Match	Visual channels	Visualization versions must have visual encoding channels only.	-
Desktop-Visual	Soft	Width	Match	1,000 pixels	A desktop visualization should aim to have a width of 1,000 pixels.	\$Difference × 0.01
Desktop-Visual	Soft	Data.Transform	Match	No aggregation	A desktop visualization should aim to not aggregate data points.	Aggregate.Count × 1
Desktop-Visual	Hard	Text.Size	Above	12 points	A desktop visualization must have text sizes greater than 12 points.	-
Smartphone-Visual	Hard	Width	Below	430 pixels	A smartphone visualization must not have a width greater than 430 pixels.	-
Smartphone-Visual	Hard	Width	Above	300 pixels	A smartphone visualization must not have a width greater than 300 pixels.	-
Smartphone-Visual	Soft	Height	Match	600 pixels	A smartphone visualization should aim to have a height of 600 pixels.	\$Difference × 0.02
Smartphone-Visual	Hard	Text.Size	Above	10 points	A smartphone visualization must have text sizes greater than 10 points.	-
Smartphone-Visual	Hard	Label.Text.Size	Below	13 points	A smartphone visualization must have label text sizes smaller than 13 points.	-
Smartphone-Visual	Soft	Interactivity.Count	Below	3	A smartphone visualization should aim to have three or less interaction features.	\$Difference × 2
[Any]-Alt	Hard	Representation	Match	Descriptive text	Alternative text versions must have descriptive text.	-
[Any]-Alt	Soft	Overall.Length	Below	10 seconds	Alternative text versions must have overall description of fewer than ten seconds.	\$Difference × 2
[Any]-Alt	Hard	Exploration	Match	Interactive	Alternative text versions must support interactive exploration (e.g., Data Navigator [257]).	-
[Any]-Audio	Hard	Encoding.Channels	Match	Auditory channels	Sonification versions must have auditory encoding channels only.	-
[Any]-Audio	Hard	Encoding.Channel	Match	Time	Sonification versions must have a time channel.	-
[Any]-Audio	Soft	Encoding.Count	Below	3	Sonification versions should try to have three or less encoding channels.	\$Difference × 3
[Any]-Audio	Soft	Encoding.Channel	Match	[Pitch, Volume, Pan]	Sonification versions should try to have a pitch, volume, or pan channel.	3
[Any]-Audio	Soft	Length	Below	5 seconds	Sonification versions should try to last for five or less seconds.	\$Difference × 2

the following context set:

$$\text{Context} = (\text{analytic}, \text{conversation-desktop}, \text{conversation-smartphone}, \text{presentation}),$$

where the other versions should be consistent with the original *analytic* version.

Table 8.3 and Table 8.4 specify the accommodation and preservation constraints, respectively. At high level, accommodation constraints set the intended sizes and regulate the number of encoding channels to cap the complexity of the versions for conversation and presentation. Interactions other than filter and zoom are impractical with higher weight (5) on conversation versions. Preservation constraints attempt to state that the versions for communication should have the same level

Table 8.2: The preservation constraints for the accessible responsive visualization problem described in Section 8.2.1. *Loss should range from 0 to 1.

Context	Strength	Reference	Target	Description	Cost
All	Hard	None	Encoding.Fields	All versions must encode the same set of data fields.	
All	Hard	None	Encoding.Transform	All versions must have the same type of data transformation.	
[Any]-Visual	Soft	None	Task-oriented insights	Visualization versions should try to share the same level of support for tasks.	$\$Loss^* \times 2$
[Any]-Visual	Soft	None	Graphical density	Visualization versions should try to share the same level of graphical density.	$\$Loss^* \times 1.5$
[Any]-Visual	Soft	None	Mark types	Visualization versions should try to share the same mark types.	2
[Any]-Audio	Soft	None	Tone types	Sonification versions should try to share the same tone types.	2
[Any]-Visual	Soft	None	Encoding channels	Visualization versions should try to share the same encoding channels.	$\$Difference \times 0.2$
[Any]-Visual	Hard	None	Color schemes	Visualization versions must share the same color schemes.	-
Desktop-[Visual, Alt, Audio]	Soft	Desktop-Visual	Task-oriented insights	Alternative and sonification versions for desktop should try to share the same level of support for tasks in reference to the desktop visualization version.	$\$Loss^* \times 2$
Smartphone-[Visual, Alt Audio]	Soft	Smartphone-Visual	Task-oriented insights	Alternative and sonification versions for smartphone should try to share the same level of support for tasks in reference to the smartphone visualization version.	$\$Loss^* \times 2$
[Any]-Audio	Soft	None	Encoding.Channels	Sonification versions should try share the same encoding channels.	$\$Difference^* \times 3$
[Any]-Audio	Soft	None	Duration	Sonification versions should try to have the same duration.	$\$Difference^* \times 0.2$
[Any]-Audio	Soft	None	Speed	Sonification versions should try to have the speed (per data points).	$\$Difference^* \times 0.5$

of support for task-oriented insights. These constraints have higher costs than the previous responsive visualization problem because mismatched perceptions about the data may have more harm in organization settings.

Table 8.3: The accommodation constraints for the audience retargeting problem described in Section 8.2.2.

Context	Strength	Target	Direction	Value	Description	Cost
Not(Analytic)	Hard	Encoding.Count	Below	4	Non-analytic versions must have four or fewer encoding channels.	-
Conversation-Desktop	Soft	Width	Match	1,000 pixels	A conversation-desktop version should aim to have a width of 1,000 pixels.	$\$Difference \times 0.01$
Conversation-Desktop	Hard	Text.Size	Above	12 points	A conversation-desktop must have text sizes greater than 12 points.	-
Conversation-Smartphone	Hard	Width	Below	430 pixels	A conversation-smartphone must not have a width greater than 430 pixels.	-
Conversation-Smartphone	Hard	Width	Above	300 pixels	A conversation-smartphone must not have a width greater than 430 pixels.	-
Conversation-Smartphone	Hard	Width	Match	100vw	A conversation-smartphone's width must be 100% to the screen width (i.e., relative width).	-
Conversation-Smartphone	Soft	Height	Match	600 pixels	A conversation-smartphone should aim to have a height of 600 pixels.	$\$Difference \times 0.02$
Conversation-Smartphone	Hard	Aspect ratio	Match	Preserved	A conversation-smartphone must keep its aspect ratio when resized.	-
Conversation-Smartphone	Hard	Text.Size	Above	10 points	A conversation-smartphone must have text sizes greater than 10 points.	-
Conversation-Smartphone	Hard	Label.Text.Size	Below	13 points	A conversation-smartphone must have label text sizes smaller than 13 points.	-
Conversation-[Any]	Soft	Interactivity	Include	[Filter, Zoom]	Conversation versions can have filter or zoom interactions (only).	$\$Difference \times 5$
Conversation-[Any]	Soft	Interactivity.Count	Below	0	Conversation versions should aim to have filter or zoom interactions (preferably none).	$\$Difference \times 0.5$
Presentation	Hard	Width	Match	3,000 pixels	A presentation version must have a width of 3,000 pixels.	-
Presentation	Hard	Text.Size	Above	36 points	A presentation must have text sizes greater than 36 points.	-
Presentation	Hard	Interactivity.Count	Match	0	Conversation versions must not have any interaction.	-

Table 8.4: The preservation constraints for the audience retargeting problem described in Section 8.2.2. *Loss should range from 0 to 1.

Context	Strength	Reference	Target	Description	Cost
All	Soft	Analytic	Task-oriented insights	Non-analytic versions should try to share the same level of support for tasks in reference to the analytic version.	$\$Loss^* \times 7$
All	Hard	Analytic	Color schemes	All the versions must share the same color schemes in reference to the analytic version.	-
Conversation-[Any]	Soft	None	Graphical density	Conversation versions should try to share the same level of graphical density.	$\$Loss^* \times 1.5$
Conversation-[Any]	Soft	None	Mark types	Conversation versions should try to share the same mark types.	2
Conversation-[Any]	Soft	None	Encoding channels	Conversation versions should try to share the same encoding channels.	$\$Difference \times 0.2$

8.3 Potential Applications

To show the feasibility and potentials of this approach, I illustrate a prototype application using the accessible responsive visualization example (Section 8.2.1) and describe other potential application cases based on this approach.

8.3.1 A Prototype Recommender Builder for Accessible Responsive Visualization

To demonstrate the feasibility of the expression method for contexts and constraints introduced in Section 8.1, I implemented a prototype recommender builder for accessible responsive visualization that considers visualizations and sonifications on desktop and smartphone devices based on the example case described in Section 8.2.1. I simplified the context set and applied a heuristics-based cost model for the demonstration purposes. As overviewed in Figure 8.1, this prototype builder includes four steps: (a) problem definition, (b) templates, (c) data configuration, and (d) generating suggestions. The first two steps are for tool developers to build a recommender, and the outcome of them is a design recommender, which allows individual authors to complete the last two steps.

Developer: Problem definition

A developer first defines the problem using the proposed expressions using YAML, which is later parsed as Answer Set Programming predicates. For example, a ‘desktop-visual’ context is defined in YAML as:

```
1 - context: desktop_visual
2   attr: [desktop, visual]
```

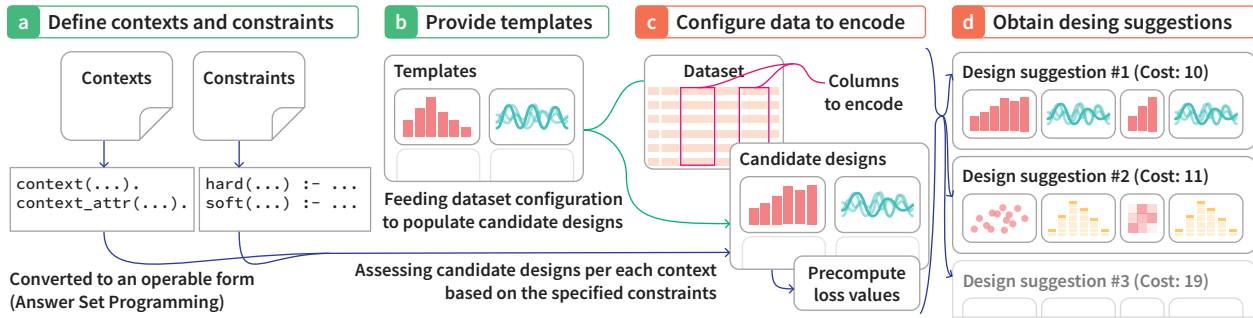


Figure 8.1: The pipeline of the prototype recommender builder for accessible responsive visualization. (a) A developer defines contexts and constraints that are translated to Answer Set Programming expressions. (b) The developer provides templates for visualization and sonification. (c) An author configures a dataset and columns to encode, which are used to populate candidate designs. At this stage, the builder computes the loss values between each pair of populated designs. (d) The author obtains design suggestions ranked by the cost caused by violating soft constraints. I assume that the cost model is provided by the developer while providing a room for designers to make minor adjustments.

and then converted to the below ASP expression:

```

1 context(desktop_visual).
2 context_attr(desktop_visual, desktop).
3 context_attr(desktop_visual, visual).

```

Constraints expressed using YAML are also parsed into ASP constraints. For example, the below accommodation constraint sets the representation type for visual versions to visualization.

```

1 - name: visualMustUseVisual
2   _class: accommodate
3   contexts:
4     attr: [visual]
5   strength: hard
6   target: representation
7   direction: match
8   value: visualization

```

Then, this is parsed into the following ASP expression:

```

1 :- representation(X,T,V), V != visualization, context_attr(X, visual),
2   view(X,T).

```

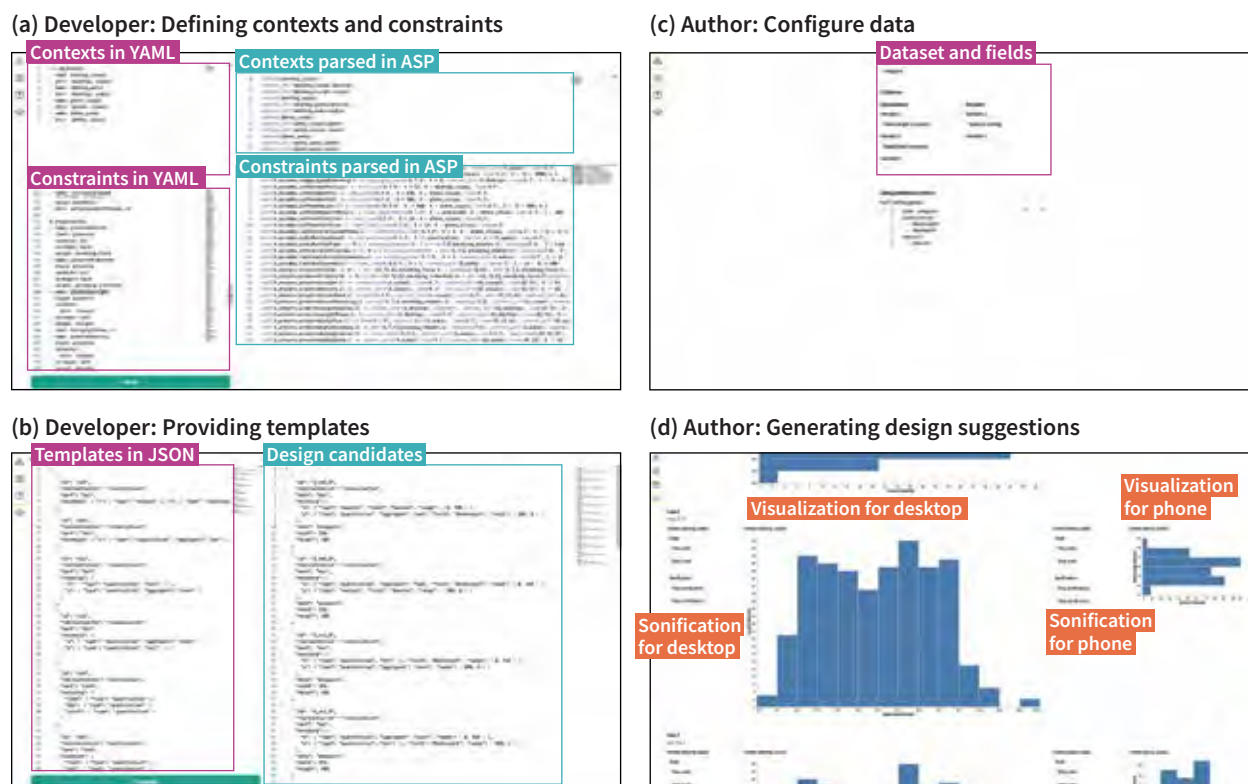



Figure 8.2: A prototype design recommender for accessible responsive visualization. (a) A developer defines the set of contexts and relevant constraints. (b) A developer provides a set of design templates. (c) An author specifies the dataset and data fields to encode. (d) An author receives design suggestions.

This ASP expression is interpreted as: “a version X for contexts with the attribute of visual (context_attr) that has a template T (view) must not have representation, V , that is not visualization.” Note that although an ASP constraint only deals with two contexts, the solver applies the same constraint to every eligible pair.

Similarly, a soft preservation constraint for insight loss across desktop versions can be first expressed using YAML as below:

```

1 - name: prserveinsightDesktop
2   _class: preserve
3   contexts:
4     attr: [desktop]
5   reference: desktop_visual
6   strength: soft
7   target: insight

```

```
8 cost: multiply($loss, 2)
```

Then, this definition is translated into an ASP expression as below:

```
1 cost(C) :- context_attr(X,desktop), view(X,T), X = desktop_visual,
2           context_attr(Y,desktop), view(Y,S), X != Y,
3           loss_value(T,S,insight,L), C = |L| * 2.
```

The interpretation of this ASP expression is: “two versions for different contexts, X and Y , with the attribute of desktop that use templates, T and S , respectively, should not have a loss_value, L , for insight between T and S . The origin of the loss value should always be desktop_visual. A violation of this constraint will have a cost of C which equals to L multiplied by 2.”

Developer: Templates

This prototype applies a template-based suggestion so as to compute loss values values between different designs in advance and hence reduce the processing time for the later design recommendation. In this way, whenever an author configures the dataset and data fields to encode, the prototype can compute loss values using the scale functions of each template’s encoding channels. For example, if a template employs a color channel, then the prototype generates a scale function to compute rendered color values which are in turn used to calculate loss values. Each template defines the size of a view (width and height for a visualization template and duration for a sonification template) and encoding channels to use. For instance, a visualization shown in Figure 8.2 (d) has a nominal x position channel and a quantitative y position channel with ‘sum’ aggregation. Then, the computed loss values are passed as an ASP expression to the recommender later using the above loss_value predicate. I used the task-oriented insight loss measures (Chapter 4) and density loss measures (Section H.3.2). This and the previous steps returns a recommender

for individual authors to use.

Author: Data configuration

To use the design recommender built by the previous steps, an author needs to provide a dataset and its columns to encode. By feeding the data fields and their types to each template, the prototype populates the candidate designs for the recommender to search over. For instance, in Figure 8.2 (c), the author specified a ‘penguins’ data set and two quantitative variable ‘BeakLength’ and a nominal variable ‘Species.’ The prototype feeds this data configuration to the templates, resulting in design candidates in Figure 8.2 (b).

Author: Generating suggestions

Lastly, an author receives design suggestions that are ranked by the costs caused by violating soft constraints. Expressed in ASP, each suggestion is a set consisting of a version per context. Each version is first translated to a Vega-Lite [96] specification if it is a visualization and to an ERIE specification if a sonification.

I included eight visualization and eight sonification templates. With the above data configuration, it populated 24 different templates because four sonification templates required two nominal variables. Combined with the above hard constraints, the recommender generated 256 suggestions. The suggestion set in Figure 8.2 (d) was one of the top candidates with a large histogram for desktop visualization and a transposed histogram with an increased bin size for smartphone visualization. The sonification designs for desktop and phone were pitch-based auditory histograms.

8.3.2 Future Application Cases

8.3.2.1 Accessible Statistics

Statistics and data analytics rely on perceptual data representations like residual plots after doing a regression or contingency tables after training a model. Multimodality is a key to make data work accessible for people with Blindness or Low Vision [28, 259–261]. Different representation modes like sonification, speech, tactile, and braille have different strengths and weaknesses. Speech descriptions are most widely applicable but of limited precision in expressing details in a data distribution. While sonifications can more precisely express those details, readers tend to perceive auditory encodings less accurately compared to tactile representations [19]. Tactilizations and braille can support more accurate data perception, but the devices to support them (e.g., braille printers, tactile printers, 3D printers) are too expensive for general readers to use on a daily basis. In addition, a team of Blind and sighted analysts would want to have visual and non-visual representations corresponding to each other so that each collaborator always has the same level of information. Thus, providing a wide range of available modalities is an important requirement for making statistics and data analytics accessible.

However, it can be too tedious to manually create each of those representations every time they plot data. Data analysis processes involve a lot of plots, and it can be difficult to just create a single chart using packages like `ggplot2` [26]. To support multimodal statistical data representations, systems like MAIDR [261] offer engineering pipelines using templates for certain visualization designs like bar chart, heatmap, box plot, scatterplot, and regression lines. While those charts are essential for data analytics, we often need to try designs like density plots, confidence envelopes, etc.

that are beyond those basic ones.

A potential future system for accessible statistics could encode different plot types as templates based on their primary tasks like comparison, correlation, etc. With the proposed generalization, the future accessible statistics system could define each modality and a context, have accommodate constraints for the representation types and available encoding channels, and set maintaining primary tasks to be supported as preservation constraints. Such a system will be a useful platform that can support multimodal data representation systems like MAIDR [261] or VoxLens [262] to be more flexible in choosing data representation designs.

8.3.2.2 Multi-modal Data Monitor

Training large scale models or observing astronomical phenomena often takes a long time span for days and weeks. It is often important for researchers to keep monitoring intermediate processes of those tasks so that they can avoid missing important observations. For example, model fitting might take longer than expected to reach a terminal condition like convergence, and being able to detect it could prevent the researchers from finding it out too late. Intermediate signals during an astronomical observation are important because it could indicate some errors in their signal processing methods (e.g., infrared light frequencies to chemicals) or phenomena that the researchers attempt to figure out.

However, it can be too tedious if researchers have to keep monitoring such streaming data on screen. Multimodal data representation with visualization and sonification can be useful for data monitor by allowing researchers to work on other tasks in parallel and come back to their data visualization when they detect a situa-

tion of interest from the sonification. In this scenario, the visualization should contain precise details, and sonification could be less precise but not annoying when ambiently played. I emphasize that the sonification monitor would be less useful if it only works as an alert because situations that need attention could not always be determined a priori.

A useful preservation constraint for multi-modal data monitor could be the calibration between visual and auditory signals so that once a scientist detects an auditory signal, they can easily spot the situation on the visualization. This calibration could be achieved, for example, via setting the same thresholds so that the monitor can spot when a value goes above or below a certain level using different volumes or pitch frequencies. Alternatively, a data monitor could provide intuitive visual and audio correspondence in order for scientists can have a better ambient sense about their data. Example strategies include finding natural timbre for a categorical variable (e.g., Hoque et al. [70]), mapping lightness/darkness of sound (via modulation and harmonicity) for the seriousness of a situation, and using spatial audio for the indication of geographical or topological directions (e.g., Ferres et al. [61]).

Chapter 9

Conclusion

In this dissertation, I have demonstrated abstractions and interactive systems for authoring data visualizations for diverse audiences coming from different user contexts. In particular, I focused on responsive visualization for different device types and data sonification for accessibility and immersive storytelling. Based on these projects, I proposed a more generalized approach to expressing multi-context visualization to support flexibly and explicitly defining user contexts in terms of accommodation and preservation constraints. After summarizing the contributions of this dissertation, I reflect them in light of the challenges that I identified in Section 1.1. Then, I discuss a couple of future research directions with respect to transferring the contributions to other design or practice domains and combining them to extend the generalized approach.

9.1 Summary of Contributions

I demonstrated a full-fledged approach to tooling for responsive visualization, ranging from characterizing key design tradeoffs to building a useful interactive authoring system. In Chapter 3, based on the preliminary design pattern analysis, I first

characterized a major design tradeoff between adjusting graphical density for each view and maintaining “messages” across versions. I identified three density challenges that motivate changes to visualization design for different device types: adequate graphical density, layout feasibility, and appropriate interaction complexity. I characterized five forms of message loss caused by design transformations to adjust density. They include loss of information, loss of interaction, reduced discoverability, reduced concurrency, and changes in graphical perception.

As an automated method for reasoning about changes to visualization “messages” or takeaways, I introduced a set of task-oriented insight loss measures in Chapter 4. I implemented those measures to support automatically reasoning about changes (or loss) to “messages” under responsive transformation by approximating messages as insights that readers arrive at by performing low-level tasks: identification, comparison, and trend. These formally defined loss measures take a pair of responsive versions for different device types and return estimated changes to support for the three low-level tasks. Under an automated recommender pipeline, these measures achieved up to 84% accuracy rate in reproducing expert-labeled rankings.

To express a vast range of possible responsive visualization transformations, I implemented the CICERO grammar and developed its compilers (Chapter 5). As a declarative grammar, CICERO expresses responsive transformations as an ordered list of rules, each consisting of a specifier for what to change, an action for how to change the specified element(s), and an option for detailing action keywords. Our CICERO compiler applies transformation rules to a design spec for a certain device type and produces another spec for a different device type. I demonstrated the ex-

pressiveness, flexibility, and reusability of the CICERO grammar by reproducing 13 real-world cases. In addition, the CICERO grammar and compilers feasibly supported an automated design recommender.

I designed and implemented DUPO, a mixed-initiative authoring tool for responsive visualization, to support automated design exploration with reduced manual effort in design prototyping while allowing authors to express their custom design intents (Chapter 6). DUPO recommends responsive designs and edits with three pipelines: *Exploration* for more noticeable changes, *Alteration* for more subtle changes, and *Augmentation* for next-step edits. In a user study with six expert graphics reporters, participants generally created satisfactory responsive visualization designs. They found DUPO's design suggestions reasonable and noted that DUPO helped them to reason about various designs with reduced manual efforts.

Lastly in Chapter 7, I presented ERIE, a declarative grammar for data sonification, to help developers to succinctly express their audio graph designs while avoiding engineering overheads like low-level audio programming. At a high level, ERIE expresses a sonification design in terms of data, tone, and encoding. ERIE supports expressing a variety of sonification designs by offering a large set of auditory encodings, multi-stream sonification designs, and custom extensions like sampling and filters. ERIE's compiler generates an audio queue as an intermediate step, which allows for building players for various audio environments. Built based on standard Web Audio and Speech APIs, ERIE's player for web supports cross-browser experiences. To demonstrate the expressiveness of ERIE, I provided various novel sonification designs for tasks like Q-Q plot, residual plot, and kernel density estimations. Furthermore, I replicated and extended prior sonification applications like Chart

Reader [72] and Audio Narrative [64] to show the feasibility of using ERIE in an end-user application development.

9.2 Reflections

Producing data visualization for multiple user contexts is essential but challenging due to the difficulties in design exploration, management of message consistency, and limited tool support. Avoiding design fixation by exploring various ideas requires time and effort on the part of individual authors to try out and assess them, and supporting multiple user contexts amplifies such costs for design exploration. At the same time, authors have to ensure that embedded takeaways are consistent across those multiple versions. Given these challenges, authors need to learn and use a variety of necessary technical stacks like data processing, visual design, and audio programming, while having to switching between tools, rather than being able to focus on reasoning about their designs per se. These production-wise difficulties result in data representations that are not suitable for intended contexts (e.g., overflowing contents, non-screen-readable contents), limiting readers from accessing their visualizations in certain contexts. In this dissertation, I attempted to demonstrate that appropriate abstraction of a design space and tradeoffs leads to useful interactive systems that facilitate visualization authors to deal with such challenges with reduced time and effort.

First, the abstractions of changes to visualization insights (Chapter 4) and design transformations (Chapter 5) for responsive visualization successfully supported implementing a mixed-initiative authoring system, DUPO (Chapter 6). I note that these abstractions stem from my characterization of a design space and tradeoffs

(Chapter 3). In addition, with a comprehensive survey in state-of-the-art tools (Table 2.1) and online tutorials (Section 7.2) for data sonification, I presented ERIE as an abstraction for data sonification design (Chapter 7). ERIE supported building various sonification applications by easily reusing expressions and allowing for personalization (Section 7.6.2).

Second, the task-oriented insight loss measures help automated responsive visualization tools to reason about how consistently responsive versions support task-oriented insights (Chapter 4). In particular, these measures are clearly defined with customizable weight terms, offering ways for authors to express their priorities. For example, DUPO allows for immediately sorting design suggestions based on these measures as well as changing the default weight terms (Section 6.4.4 and Section 6.4.5).

Lastly, the tools presented in this dissertation show potential for redirecting authors' attention from dealing with technical challenges to reasoning about designs. In the user study of DUPO (Section 6.5), expert authors were able to inspect designs (e.g., different layout structures, data aggregation) that they had not thought of previously, but with much reduced effort. Even when they adhered to their original designs, being able to explore designs helped them to be more confident with their designs. Similarly, ERIE can replace tedious work like writing scale functions to map data to sound or connecting different digital audio components with a few lines of design specification. Given that it is hard to reuse, customize, and share sonification programs with current practices (Section 7.2), ERIE will help data workers with limited skills to express their ideas and reuse and customize prior designs, freeing them from weary jobs like debugging and technical testing.

9.3 Future Research Directions

I have only looked at two context types, responsive visualization and sonification, in this dissertation. As noted in Section 2.1, there are other context types like re-targeting for audience groups with varying levels of expertise in domain-specific data analytics, style transfer due to design requirements of different publication venues, and spatial data experiences like exhibitions at a museum or classrooms. In addition, as our communication media, learning environments, and work formats change, we may observe different combinations and distinctions among user context types than those covered in this dissertation. I outline a couple of future research directions: (1) extending our approaches to different visualization design domains and (2) combining our approaches to support novel set of user contexts.

9.3.1 Extension to Other Design Domains

9.3.1.1 Task-oriented Insight Loss Measures to Other Design Transformation Domains

Our approach to task-oriented insight preservation is likely to be useful in visualization design domains beyond responsive visualization, like style transfer and visualization simplification, although other domains may need different design transformation techniques. Style transfer, for instance, may involve techniques like aggregation but is more likely to change visual attributes of marks or references. While our loss measures are designed to be low-level enough to apply relatively generically to visualizations, their precise formulation and strategy for combining them might warrant changes in other domains. For example, in visualization simplification, minimizing trend loss is likely to be more important than preserving identification

and comparison of individual data points. Style transfer often focuses on altering color schemes or mark types [75] which can result in different discriminability distributions, so it might put more emphasis on comparison loss.

9.3.1.2 CICERO with other visualization grammars

We built a CICERO compiler using a Vega-Lite extension for communicative visualizations as a design spec representation. To support other visualization domains like collaborative data analytics that need responsive design, future work could work on CICERO compilers for relevant design representation methods. For example, Mosaic [263] is a software architecture for rapid processing and interactive visualization of high-volume data, suitable for analytic dashboards. Thus, a CICERO compiler for Mosaic would be a useful extension to support responsive collaborative data analytics. As Mosaic's interoperation between data and charts rely on SQL expressions, CICERO's specifier (i.e., visualization element query statement) could be re-expressed to be compatible with a SQL syntax.

In addition, many data-driven articles, although not accompanied with lots of visualizations, require that interaction features (e.g., search, filter) be repositioned or reconfigured for different device types. For example, interactions requiring mouse or keyboard-based precise selection or appearing in a side-by-side panel may not be ideal for smartphone devices. Hence, pairing CICERO with declarative expressions for user interactions like Varv [264] could benefit data journalists authoring interaction-heavy data representations.

9.3.1.3 DUPO for diverse visualization authoring techniques

DUPO employs a graphic-based interface, similar to Adobe Illustrator and Canva, that allows for visually manipulating chart elements; this approach has several benefits with respect to implementing the recommender. First, our approach can regularize the representation of user input for the recommender and hence easily specify the roles of chart elements, such as using our extended Vega-Lite [96] for the visualizations and CICERO for responsive transformations. In contrast, users can have more freedom in representing their designs with programming-based approaches (e.g., writing code for D3.js [25]), which poses more complexity in interpreting different visualization roles. Second, DUPO can visually demonstrate design suggestions within its graphical interface, whereas programming-based approaches may require additional rendering processes. Third, DUPO supports flexibility in customizing responsive artboards without necessarily maintaining the same mark type, encoding channels, or text elements, compared to the template-based tools (e.g., Datawrapper [24] or Flourish [23]), as pointed out by E6.

Our user study participants for DUPO generally leverage a diverse set of visualization tools including template-based, programming-based, and graphic-based tools. To better support different authoring experiences, it is important to extend our mixed-initiative approach to different types of tools. As noted above, however, there are several challenges to do so for template-based and programming-based tools, illuminating future research opportunities. Using a template-based tool, for example, users choose a template of fixed responsive designs, and they can only use those designs. Instead, such tools could allow for combining each responsive version from different templates, so that a recommender can search designs more

flexibly and users can have more freedom with their designs, as noted by E2 and E6. Doing so would require asking research questions like how to search over a template design space in a way that supports preserving design information across responsive versions. For programming-based approaches, recommenders could employ an extraction model [75] or AI code suggestion [265] for responsive visualization. Such approaches will benefit from real-time bundlers (e.g., Rollup [266]) that renders the outcome as a user makes changes to the code, potentially enabling immediate visual demonstration of design recommendations expressed programmatically.

9.3.1.4 Toward data sensification

For ERIE, we intentionally reused expressions suggested by several implementations of the Grammar of Graphics [249] like Vega-Lite [96] and ggplot2 [26] to make it intuitive for typical data workers to use ERIE. This extension from visual grammar to auditory grammar shows a possibility for even further extension to general data sensification that includes tactilization and physicalization. Data tactilization encodes data variables to haptic variables like texture, density, and depth (at most 2-3 mm) on a two-dimensional space [267]. Data physicalization maps data variables to x , y , and z positions with other visual channels like color, tactile, and pattern¹.

Tactile printers and 3D printers are popular rendering methods for those tactile and physical representations. Yet, popular libraries for tactilization like TactileR [115] support a few chart types, and modeling for 3D printing is generally known to be challenging [268]. Therefore, a flexible, consistent formalization supporting diverse perceptual channels will help data sensification authors and learners can

¹See an online gallery (<http://dataphys.org/list/gallery/>) by Dragicevic and Jansen.

easily hop in different perceptual channels. Such flexibility will be extremely useful for enabling multimodal data experiences, such as exhibitions, data arts, hand-held devices (e.g., [74]), and virtual and augmented reality-based computing (e.g., [269]).

9.3.2 Combining Approaches

In Chapter 8, I illustrated a blueprint for a generalized approach to multi-context data visualization as a way to capture and support novel combinations of user context types. To make this generalized approach more promising for supporting visualization authors with limited technical skills, a number of technological building blocks are warranted to reify this approach. An important research direction for technological advance is to interweave software for individual context types. Below, I outline how future work can interlace our contributions to responsive visualization and data sonification so as to support diversified user context combinations in the future.

First, the task-oriented insight loss measures can be extended to cover auditory channels. As the definitions of our measures (except distance functions) are agnostic to specific encoding channels, computing loss values for auditory channels itself is not challenging. Yet, weighting the perceptual loss between a visual channel to auditory channel (e.g., accuracy/precision in recognizing position vs. pitch) needs more investigation. In addition, distance functions for the comparison loss could consider characteristics of auditory channels (e.g., applying Steven's power law for pitch perception).

Second, CICERO expressions and compiler could be extended to accommodate auditory channels with ERIE. For example, the `role` keywords should include sonifi-

cation-specific structural characteristics (e.g., tone vs. mark, sequence/overlay vs. row/column, and audio sampling). A CICERO compiler between visualization and sonification should also consider a set of “smart” default strategies to help authors and tool developers to avoid from a lot of tedious, routine transformations (e.g., a default tone and mark) in accordance with the principles suggested in Section 5.4.

Above extensions to task-oriented loss measures and CICERO could also benefit DUPO for authoring accessible responsive visualizations. Yet, representing sonification for authoring is another challenge for interface-wise support. For example, a potential DUPO extension could incorporate different sound representation methods like spectrogram (frequencies and velocity over time) and waveform. For easier debugging, ERIE’s audio queue could be presented as a table, linking its element to the corresponding parts in a sound representation and a visualization version.

Lastly, ‘task’ is an overarching keyword for the work presented in this dissertation. For example, the task-oriented loss measures approximate visualization messages by formalizing support for low-level tasks given that readers arrive at a certain insight by performing relevant tasks. These measures aim to maintain the same extent of task support across different responsive versions. In addition, we designed ERIE to be independent from visualization representations. The underlying intention was to support designing sonifications to be suitable for data tasks because tools that fix a sonification design to a visual form are likely to ignore the correspondence to reader’s tasks. Furthermore, prior work in visualization tools [20–22, 83, 148, 151, 152, 164] also incorporated user tasks in supporting user’s discovery of insights.

This emphasis on user tasks hints at a task-first approach to scalable support for multi-context data visualization. This approach can benefit building tools that

allow for explicitly expressing “messages” and takeaways to be consistent across different versions. For example, an authoring tool can take intended tasks as input and produce optimal representations. Similarly, a design fixer like Wu et al. [93] could capture and consider likely intents of a version in suggesting another version for a different context.

To enable this task-first approach for multi-context visualization, interesting future work includes a flexible expression for specifying tasks. While the task-oriented insight loss measures consider three low-level tasks types, an extended expression, for example, could indicate priorities for the relationship between particular data points or describe combinations of low-level tasks types to form higher-level insights. Systematically designed, such an expression could benefit setting preservation constraints for the generalized approach discussed in Chapter 8.

9.4 Concluding Remarks

Data visualization augments our abilities to explore data. However, existing research and tooling tend to focus on a single user context (e.g., expert users using desktops) and be implicit about user contexts. This lack of consideration for user contexts amplifies costs for using visualizations to communicate with readers in different viewing conditions. To lower this hurdle, therefore, I contribute abstractions and interactive systems that facilitate authors to accommodate their designs for various device types and for varying visual abilities and to preserve insights across responsive visualization designs. Through generalization and extension of my approaches to other user context types and their mixtures, I envision future visualization environments will better support users situated in different contexts.

References

- [1] J. H. Larkin and H. A. Simon, “Why a diagram is (sometimes) worth ten thousand words,” *Cognitive Science*, vol. 11, no. 1, pp. 65–100, 1987. DOI: 10.1016/S0364-0213(87)80026-5.
- [2] E. R. Tufte and P. Graves-Morris, *The Visual Display of Quantitative Information. Vol. 2*. Graphics Press Cheshire, 1983.
- [3] W. Willett, B. A. Aseniero, S. Carpendale, P. Dragicevic, Y. Jansen, L. Oehlberg, and P. Isenberg, “Perception! immersion! empowerment! superpowers as inspiration for visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 22–32, 2022. DOI: 10.1109/TVCG.2021.3114844.
- [4] J. W. Tukey and P. A. Tukey, “Computer graphics and exploratory data analysis: An introduction,” in *Proceedings of Annual Conference and Exposition: Computer Graphics*, 1985.
- [5] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala, “Graphical histories for visualization: Supporting analysis, communication, and evaluation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1189–1196, 2008. DOI: 10.1109/TVCG.2008.137.
- [6] Y. Fu and J. Stasko, “More than data stories: Broadening the role of visualization in contemporary journalism,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–20, 2023. DOI: 10.1109/TVCG.2023.3287585.
- [7] F. Yang, “Data visualization for health and risk communication,” in *The Handbook of Applied Communication Research*. John Wiley & Sons, Ltd, 2020, pp. 213–232. DOI: <https://doi.org/10.1002/9781119399926.ch13>.

- [8] S. Fedeli and K. E. Matsa, *Use of Mobile Devices for News Continues to Grow, Outpacing Desktops and Laptops*, <https://pewrsr.ch/2uvqS04>, 2018. (visited on 07/21/2018).
- [9] F. Elavsky, J. Mankoff, and A. Satyanarayan, “Increasing data equity through accessibility,” 2022, A response to the Office of Science and Technology Policy’s Request for Information on “Equitable Data Engagement and Accountability”. DOI: 10.48550/arXiv.2210.01902.
- [10] K. Duncker, “On problem-solving,” *Psychological Monographs*, vol. 58, pp. i-113, 1945, (L. S. Lees, Trans.) DOI: 10.1037/h0093599.
- [11] D. G. Jansson and S. M. Smith, “Design fixation,” *Design Studies*, vol. 12, no. 1, pp. 3-11, 1991. DOI: 10.1016/0142-694X(91)90003-F.
- [12] J. S. Linsey, I. Tseng, K. Fu, J. Cagan, K. L. Wood, and C. Schunn, “A study of design fixation, its mitigation and perception in engineering design faculty,” *J. Mech. Des.*, vol. 132, no. 4, p. 041003, 2010. DOI: 10.1115/1.4001110.
- [13] R. J. Youmans and T. Arciszewski, “Design fixation: Classifications and modern methods of prevention,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 28, no. 2, pp. 129-137, 2014. DOI: 10.1017/S0890060414000043.
- [14] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer, “Reflections on how designers design with data,” in *Proceedings of International Working Conference on Advanced Visual Interfaces*, ser. AVI ’14, ACM, 2014, pp. 17-24. DOI: 10.1145/2598153.2598175.
- [15] W. S. Cleveland and R. McGill, “Graphical perception and graphical methods for analyzing scientific data,” *Science*, vol. 229, no. 4716, pp. 828-833, 1985, ISSN: 0036-8075. DOI: 10.1126/science.229.4716.828.
- [16] W. S. Cleveland, “A model for studying display methods of statistical graphics,” *Journal of Computational and Graphical Statistics*, vol. 2, no. 4, pp. 323-343, 1993. DOI: 10.1080/10618600.1993.10474616.
- [17] J. Heer and M. Agrawala, “Multi-scale banking to 45 degrees,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 701-708, 2006. DOI: 10.1109/TVCG.2006.163.

- [18] S. S. Stevens, “On the psychophysical law,” *Psychological review*, 1957. DOI: 10.1037/h0046162.
- [19] S. S. Stevens, *Psychophysics: Introduction to its perceptual, neural and social prospects*. Routledge, 2017.
- [20] Ç. Demiralp, P. J. Haas, S. Parthasarathy, and T. Pedapati, “Foresight: Recommending visual insights,” *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1937–1940, 2017. DOI: 10.14778/3137765.3137813.
- [21] Z. Cui, S. K. Badam, M. A. Yalçın, and N. Elmqvist, “Datasite: Proactive visual data exploration with computation of insight-based recommendations,” *Information Visualization*, vol. 18, no. 2, pp. 251–267, 2019. DOI: 10.1177/1473871618806555.
- [22] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang, “Extracting top-k insights from multi-dimensional data,” in *Proceedings of ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’17, ACM, 2017, pp. 1509–1524. DOI: 10.1145/3035918.3035922.
- [23] *Flourish*, <http://flourish.studio/>.
- [24] Datawrapper GmbH, *Datawrapper*, <https://www.datawrapper.de/>, 2013.
- [25] M. Bostock, V. Ogievetsky, and J. Heer, “D³ data-driven documents,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011. DOI: 10.1109/TVCG.2011.185.
- [26] H. Wickham, “A layered grammar of graphics,” *Journal of Computational and Graphical Statistics*, vol. 19, no. 1, pp. 3–28, 2010. DOI: 10.1198/jcgs.2009.07098.
- [27] A. Tse, *Ai2html*, <http://ai2html.org/>, 2011.
- [28] S. C. S. Joyner, A. Riegelhuth, K. Garrity, Y.-S. Kim, and N. W. Kim, “Visualization accessibility in the wild: Challenges faced by visualization designers,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’22, Association for Computing Machinery, 2022. DOI: 10.1145/3491102.3517630.
- [29] R. Wang, C. Jung, and Y. Kim, “Seeing through sounds: Mapping auditory dimensions to data and charts for people with visual impairments,”

- Computer Graphics Forum*, vol. 41, no. 3, pp. 71–83, 2022. DOI: <https://doi.org/10.1111/cgf.14523>.
- [30] A. Supper, “Sublime frequencies: The construction of sublime listening experiences in the sonification of scientific data,” *Social Studies of Science*, vol. 44, no. 1, pp. 34–58, 2014. DOI: 10.1177/03086312713496875.
- [31] T. Hermann, “Taxonomy and definitions for sonification and auditory display,” ICAD ’08, pp. 1–8, 2008.
- [32] G. Kramer, B. Walker, T. Bonebright, P. Cook, J. H. Flowers, N. Miner, and J. Neuhoff, “Sonification report: Status of the field and research agenda,” 1997, Report prepared for the National Science Foundation.
- [33] C. Scaletti, “Sound synthesis algorithms for auditory data representations,” in *Auditory Display, Sonification: Audification, and Auditory INterfaces*, G. Kramer, Ed., 1994, pp. 223–251.
- [34] C. Jung, S. Mehta, A. Kulkarni, Y. Zhao, and Y.-S. Kim, “Communicating visualizations without visuals: Investigation of visualization alternative text for people with visual impairments,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 1095–1105, 2022. DOI: 10.1109/TVCG.2021.3114846.
- [35] S. J. Cantrell, B. N. Walker, and Ø. Moseng, “Highcharts sonification studio: An online, open-source, extensible, and accessible data sonification tool,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD ’21, ICAD, 2021, pp. 211–216. DOI: 10.21785/icad2021.005.
- [36] F. Dombois, O. Brodewolf, O. Friedli, I. Rennert, and T. Koenig, “Sonifyer: A concept, a software, a platform,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD ’08, ICAD, 2008.
- [37] H. Kim, D. Mortiz, and J. Hullman, “Design patterns and trade-offs in authoring communication-oriented responsive visualization,” *Computer Graphics Forum*, vol. 40, no. 3, pp. 459–470, 2021. DOI: 10.1111/cgf.14321.
- [38] H. Kim, R. Rossi, A. Sarma, D. Moritz, and J. Hullman, “An automated approach to reasoning about task-oriented insights in responsive visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 129–139, 2022. DOI: 10.1109/TVCG.2021.3114782.

- [39] H. Kim, R. Rossi, F. Du, E. Koh, S. Guo, J. Hullman, and J. Hoffswell, “Cicero: A declarative grammar for responsive visualization,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’22, ACM, 2022. DOI: 10.1145/3491102.3517455.
- [40] H. Kim, R. Rossi, J. Hullman, and J. Hoffswell, “Dupo: A mixed-initiative authoring tool for responsive visualization,” vol. 30, no. 1, pp. 934–943, 2024. DOI: 10.1109/TVCG.2023.3326583.
- [41] H. Kim, Y.-S. Kim, and J. Hullman, “Erie: A declarative grammar for data sonification,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’24, ACM, 2024. DOI: 10.1145/3613904.3642442.
- [42] M. Brehmer, B. Lee, P. Isenberg, and E. K. Choe, “Visualizing ranges over time on mobile phones: A task-based crowdsourced evaluation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 619–629, 2019. DOI: 10.1109/TVCG.2018.2865234.
- [43] M. Brehmer, B. Lee, P. Isenberg, and E. K. Choe, “A comparative evaluation of animation and small multiples for trend visualization on mobile phones,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 364–374, 2020. DOI: 10.1109/TVCG.2019.2934397.
- [44] B. Lee, R. Dachsel, P. Isenberg, and E. K. Choe, *Mobile Data Visualization*. CRC Press, 2021.
- [45] P. Isenberg, P. Dragicevic, W. Willett, A. Bezerianos, and J. Fekete, “Hybrid-image visualization for large viewing environments,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2346–2355, 2013. DOI: 10.1109/TVCG.2013.163.
- [46] K. Matković, H. Hauser, R. Sainitzer, and M. E. Gröller, “Process visualization with levels of detail,” in *IEEE Symposium on Information Visualization*, ser. InfoVis ’02, 2002, pp. 67–70. DOI: 10.1109/INFVIS.2002.1173149.
- [47] M. Rønne Jakobsen and K. Hornbæk, “Sizing up visualizations: Effects of display size in focus+context, overview+detail, and zooming interfaces,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’11, Association for Computing Machinery, 2011, pp. 1451–1460. DOI: 10.1145/1978942.1979156.

- [48] R. Rosenbaum and B. Hamann, “Progressive presentation of large hierarchies using treemaps,” in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. Coming, Eds., Springer Berlin Heidelberg, 2009, pp. 71–80. DOI: 10.1007/978-3-642-10520-3_7.
- [49] R. Rosenbaum, J. Zhi, and B. Hamann, “Progressive parallel coordinates,” in *IEEE Pacific Visualization Symposium*, ser. PacificVis ’12, IEEE, 2012, pp. 25–32. DOI: 10.1109/PacificVis.2012.6183570.
- [50] E. Di Giacomo, W. Didimo, G. Liotta, and F. Montecchiani, “Network visualization retargeting,” in *International Conference on Information, Intelligence, Systems and Applications*, ser. IISA ’15, IEEE, 2015, pp. 1–6. DOI: 10.1109/IISA.2015.7388095.
- [51] M. R. Jakobsen and K. Hornbæk, “Interactive visualizations on large and small displays: The interrelation of display size, information space, and scale,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2336–2345, 2013. DOI: 10.1109/TVCG.2013.170.
- [52] F. Vernier, N. Lesh, and C. Shen, “Visualization techniques for circular tabletop interfaces,” in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI ’02, Association for Computing Machinery, 2002, pp. 257–265. DOI: 10.1145/1556262.1556305.
- [53] T. Blascheck, L. Besançon, A. Bezerianos, B. Lee, and P. Isenberg, “Glanceable visualization: Studies of data comparison performance on smartwatches,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 630–640, 2019. DOI: 10.1109/TVCG.2018.2865142.
- [54] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono, “Research directions in data wrangling: Visualizations and transformations for usable and credible data,” *Information Visualization*, vol. 10, no. 4, pp. 271–288, 2011. DOI: 10.1177/1473871611415994.
- [55] E. Kandogan, A. Balakrishnan, E. M. Haber, and J. S. Pierce, “From data to insight: Work practices of analysts in the enterprise,” *IEEE Computer Graphics and Applications*, vol. 34, no. 5, pp. 42–50, 2014. DOI: 10.1109/MCG.2014.62.
- [56] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, “Enterprise data analysis and visualization: An interview study,” *IEEE Transactions on*

- Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2917–2926, 2012. DOI: 10.1109/TVCG.2012.219.
- [57] A. Crisan, B. Fiore-Gartland, and M. Tory, “Passing the data baton: A retrospective analysis on data science work and workers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1860–1870, 2021. DOI: 10.1109/TVCG.2020.3030340.
- [58] A. Mathisen, T. Horak, C. Klokmoose, K. Grønbaek, and N. Elmquist, “Insideinsights: Integrating data-driven reporting in collaborative visual analytics,” *Computer Graphics Forum*, vol. 38, no. 3, pp. 649–661, 2019. DOI: <https://doi.org/10.1111/cgf.13717>.
- [59] V. S. Morash, Y.-T. Siu, J. A. Miele, L. Hasty, and S. Landau, “Guiding novice web workers in making image descriptions using templates,” *ACM Transactions on Accessible Computing*, vol. 7, no. 4, 2015. DOI: 10.1145/2764916.
- [60] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmquist, “Visualizing for the non-visual: Enabling the visually impaired to use visualization,” *Computer Graphics Forum*, vol. 38, no. 3, pp. 249–260, 2019. DOI: 10.1111/cgf.13686.
- [61] L. Ferres, A. Parush, S. Roberts, and G. Lindgaard, “Helping people with visual impairments gain access to graphical information through natural language: The igrph system,” in *Computers Helping People with Special Needs*, Springer Berlin Heidelberg, 2006, pp. 1122–1130. DOI: 10.1007/11788713_163.
- [62] N. W. Kim, S. C. Joyner, A. Riegelhuth, and Y. Kim, “Accessible visualization: Design space, opportunities, and challenges,” *Computer Graphics Forum*, vol. 40, no. 3, pp. 173–188, 2021. DOI: 10.1111/cgf.14298.
- [63] T. Murillo-Morales and K. Miesenberger, “Audial: A natural language interface to make statistical charts accessible to blind persons,” in *Computers Helping People with Special Needs*, Springer International Publishing, 2020, pp. 373–384. DOI: 10.1007/978-3-030-58796-3_44.
- [64] A. Siu, G. S-H Kim, S. O’Modhrain, and S. Follmer, “Supporting accessible data visualization through audio data narratives,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’22, ACM, 2022. DOI: 10.1145/3491102.3517678.

- [65] B. N. Walker, "Magnitude estimation of conceptual data dimensions for use in sonification.," *Journal of experimental psychology: Applied*, vol. 8, no. 4, 2002.
- [66] B. N. Walker and J. T. Cothran, "Sonification sandbox: A graphical toolkit for auditory graphs," in *Proceedings of International Conference on Auditory Display*, ser. ICAD '03, ICAD, 2003, pp. 161–163.
- [67] B. N. Walker, M. T. Godfrey, J. E. Orlosky, C. Bruce, and J. Sanford, "Aquarium sonification: Soundscapes for accessible dynamic informal learning environments," in *Proceedings of International Conference on Auditory Display*, ser. ICAD '06, <http://www.icad.org/Proceedings/2006/WalkerGodfrey2006.pdf>, 2006, pp. 238–241.
- [68] B. N. Walker, "Consistency of magnitude estimations with conceptual data dimensions used for sonification," *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, vol. 21, no. 5, pp. 579–599, 2007.
- [69] B. N. Walker and L. M. Mauney, "Universal design of auditory graphs: A comparison of sonification mappings for visually impaired and sighted listeners," *ACM Transactions on Accessible Computing*, vol. 2, no. 3, 2010. DOI: 10.1145/1714458.1714459.
- [70] M. N. Hoque, M. Ehtesham-Ul-Haque, N. Elmqvist, and S. M. Billah, "Accessible data representation with natural sound," in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, Association for Computing Machinery, 2023. DOI: 10.1145/3544548.3581087.
- [71] M. Agarwal, F. Alfieri, S. Ali, J. Jorgensen, and L. Muralidharan, *Sonify*, <https://hcii.cmu.edu/mhci/capstone/2016/bloomberg/index.html>.
- [72] J. R. Thompson, J. J. Martinez, A. Sarikaya, E. Cutrell, and B. Lee, "Chart reader: Accessible visualization experiences designed with screen reader users," in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, Association for Computing Machinery, 2023. DOI: 10.1145/3544548.3581186.
- [73] J. Godfrey, *Brailler*, <https://github.com/ajrgodfrey/BrailleR>.
- [74] P. Chundury, Y. Reyazuddin, J. B. Jordan, J. Lazar, and N. Elmqvist, "Tactualplot: Spatializing data as sound using sensory substitution for

- touchscreen accessibility,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–11, 2023. DOI: 10.1109/TVCG.2023.3326937.
- [75] J. Harper and M. Agrawala, “Deconstructing and restyling d3 visualizations,” in *Proceedings of ACM symposium on User Interface Software and Technology*, ser. UIST ’14, ACM, 2014, pp. 253–262. DOI: 10.1145/2642918.2647411.
- [76] J. Harper and M. Agrawala, “Converting basic d3 charts into reusable style templates,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 3, pp. 1274–1286, 2017. DOI: 10.1109/TVCG.2017.2659744.
- [77] J. Hullman and N. Diakopoulos, “Visualization rhetoric: Framing effects in narrative visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2231–2240, 2011. DOI: 10.1109/TVCG.2011.255.
- [78] E. Segel and J. Heer, “Narrative visualization: Telling stories with data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1139–1148, 2010. DOI: 10.1109/TVCG.2010.179.
- [79] C. Conati and H. Maclaren, “Exploring the role of individual differences in information visualization,” in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI ’08, ACM, 2008. DOI: 10.1145/1385569.1385602.
- [80] C. Ziemkiewicz and R. Kosara, “Preconceptions and individual differences in understanding visual metaphors,” *Computer Graphics Forum*, vol. 28, no. 3, pp. 911–918, 2009. DOI: 10.1111/j.1467-8659.2009.01442.x.
- [81] J. B. Carroll, *Human cognitive abilities: A survey of factor-analytic studies*. Cambridge university press, 1993.
- [82] Y. Kim and J. Heer, “Assessing effects of task and data distribution on the effectiveness of visual encodings,” *Computer Graphics Forum*, vol. 37, no. 3, pp. 157–167, 2018. DOI: 10.1111/cgf.13409.
- [83] A. Srinivasan, S. M. Drucker, A. Endert, and J. Stasko, “Augmenting visualizations with interactive data facts to facilitate interpretation and communication,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 672–681, 2019. DOI: 10.1109/TVCG.2018.2865145.
- [84] Y. Kim, K. Wongsuphasawat, J. Hullman, and J. Heer, “Graphscape: A model for automated reasoning about visualization similarity and sequencing,” in

- Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI '17, ACM, 2017, pp. 2628–2638. DOI: 10.1145/3025453.3025866.
- [85] Y. Wu, X. Liu, S. Liu, and K. Ma, “ViSizer: A visualization resizing framework,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 2, pp. 278–290, 2013. DOI: 10.1109/TVCG.2012.114.
- [86] M. Wattenberg and D. Fisher, “A model of multi-scale perceptual organization in information graphics,” in *IEEE Symposium on Information Visualization*, ser. InfoVis '03, IEEE Computer Society, 2003. DOI: 10.1109/INFVIS.2003.1249005.
- [87] M. Wattenberg and D. Fisher, “Analyzing perceptual organization in information graphics,” *Information Visualization*, 2004. DOI: 10.1057/palgrave.ivs.9500070.
- [88] H. Jänicke and M. Chen, “A salience-based quality metric for visualization,” *Computer Graphics Forum*, 2010. DOI: 10.1111/j.1467-8659.2009.01667.x.
- [89] G. Kindlmann and C. Scheidegger, “An algebraic process for visualization design,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2181–2190, 2014. DOI: 10.1109/TVCG.2014.2346325.
- [90] Y. Wang, F. Han, L. Zhu, O. Deussen, and B. Chen, “Line graph or scatter plot? automatic selection of methods for visualizing trends in time series,” *IEEE Transactions on Visualization and Computer Graphics*, 2018. DOI: 10.1109/TVCG.2017.2653106.
- [91] R. Veras and C. Collins, “Discriminability tests for visualization effectiveness and scalability,” *IEEE Transactions on Visualization and Computer Graphics*, 2020. DOI: 10.1109/TVCG.2019.2934432.
- [92] N. Elmqvist and J. Fekete, “Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 439–454, 2010. DOI: 10.1109/TVCG.2009.84.
- [93] A. Wu, W. Tong, T. Dwyer, B. Lee, P. Isenberg, and H. Qu, “MobileVisFixer: Tailoring web visualizations for mobile phones leveraging an explainable reinforcement learning framework,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 464–474, 2021. DOI: 10.1109/TVCG.2020.3030423.

- [94] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko, “Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’18, ACM, 2018, pp. 1–13. DOI: 10.1145/3173574.3173697.
- [95] D. Ren, B. Lee, and M. Brehmer, “Charticulator: Interactive construction of bespoke chart layouts,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 789–799, 2019. DOI: 10.1109/TVCG.2018.2865158.
- [96] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, “Vega-lite: A grammar of interactive graphics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017. DOI: 10.1109/TVCG.2016.2599030.
- [97] ZingSoft, *Zingchart*, <https://www.zingchart.com/>, 2009.
- [98] K. Andrews, *Responsive visualisation*, https://mobilevis.github.io/assets/mobilevis2018_paper_4.pdf, ACM, 2018.
- [99] J. Hoffswell, W. Li, and Z. Liu, “Techniques for flexible responsive visualization design,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’20, ACM, 2020, pp. 1–13. DOI: 10.1145/3313831.3376777.
- [100] B. Hinderman, *Building Responsive Data Visualization for the Web*. John Wiley & Sons, 2015.
- [101] L. Chittaro, “Visualizing information on mobile devices,” *Computer*, vol. 39, no. 3, pp. 40–45, 2006. DOI: 10.1109/MC.2006.109.
- [102] B. Lee, P. Isenberg, N. H. Riche, and S. Carpendale, “Beyond mouse and keyboard: Expanding design considerations for information visualization interactions,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2689–2698, 2012. DOI: 10.1109/TVCG.2012.204.
- [103] F. Lehmann and M. Kipp, “How to hold your phone when tapping: A comparative study of performance, precision, and errors,” in *Proceedings of ACM International Conference on Interactive Surfaces and Spaces*, ser. ISS ’18, ACM, 2018, pp. 115–127. DOI: 10.1145/3279778.3279791.

- [104] J. Conradi, “Influence of letter size on word reading performance during walking,” in *Proceedings of International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI ’17, Association for Computing Machinery, 2017. DOI: 10.1145/3098279.3098554.
- [105] Google, *How People Use Their Devices: What Marketers Need to Know*, Last accessed Sep 9, 2017. <https://storage.googleapis.com/think/docs/twg-how-people-use-their-devices-2016.pdf>, 2016.
- [106] J. MacKay, *Screen time stats 2019: Here’s how much you use your phone during the workday*, <https://blog.rescuetime.com/screen-time-stats-2018/>, 2019.
- [107] K. A. Cook and J. J. Thomas, “Illuminating the path: The research and development agenda for visual analytics,” National Visualization and Analytic Center, Tech. Rep., 2005, <https://www.hSDL.org/?abstract&did485291>.
- [108] C. Körner, *Learning Responsive Data Visualization*. Packt Publishing, 2016.
- [109] J. Leclaire and A. Tabard, “R3s.js—towards responsive visualizations,” in *Workshop on Data Exploration for Interactive Surfaces*, ser. DEXIS ’15, 2015, pp. 16–19.
- [110] J. L. Kolodner and L. M. Wills, “Case-based creative design,” AAAI, Tech. Rep. SS-93-01, 1993, pp. 95–102.
- [111] J. L. Kolodner and L. M. Wills, “Powers of observation in creative design,” *Design Studies*, vol. 17, no. 4, pp. 385–416, 1996. DOI: 10.1016/S0142-694X(96)00021-X.
- [112] A. Wu, L. Xie, B. Lee, Y. Wang, W. Cui, and H. Qu, “Learning to automate chart layout configurations using crowdsourced paired comparison,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’21, ACM, 2021. DOI: 10.1145/3411764.3445179.
- [113] Microsoft, *Power bi*, <https://powerplatform.microsoft.com/en-us/>, 2011.
- [114] Tableau Software, *Tableau*, <https://www.tableau.com/>, 2003.
- [115] J. Seo, *Tactiler*, <https://github.com/jooyoungseo/tactileR>.

- [116] European data, *Data Visualisation Guide | What is accessibility?* Last accessed Mar 10, 2024. <https://data.europa.eu/apps/data-visualisation-guide/what-is-accessibility>, 2023.
- [117] *Who data design language | accessible*, Last accessed Mar 10, 2024. <https://apps.who.int/gho/data/design-language/principles/accessible/>, 2023.
- [118] WSC, *Web content accessibility guidelines (wcag) 2.1*, Last accessed Mar 10, 2024. <https://www.w3.org/TR/WCAG21>.
- [119] D. Fan, A. Fay Siu, H. Rao, G. S.-H. Kim, X. Vazquez, L. Greco, S. O’Modhrain, and S. Follmer, “The accessibility of data visualizations on the web for screen reader users: Practices and experiences during covid-19,” *ACM Transactions on Accessible Computing*, vol. 16, no. 1, 2023. DOI: 10.1145/3557899.
- [120] F. Elavsky, C. Bennett, and D. Moritz, “How accessible is my visualization? evaluating visualization accessibility with chartability,” *Computer Graphics Forum*, vol. 41, no. 3, pp. 57–70, 2022. DOI: 10.1111/cgf.14522.
- [121] A. Polli, “Atmospherics/Weather Works: A Spatialized Meteorological Data Sonification Project,” *Leonardo*, vol. 38, no. 1, pp. 31–36, 2005. DOI: 10.1162/leon.2005.38.1.31.
- [122] J. Dunn and M. A. Clark, “Life Music: The Sonification of Proteins,” *Leonardo*, vol. 32, no. 1, pp. 25–32, 1999. DOI: 10.1162/002409499552966.
- [123] P. Ghosh, *God particle signal is simulated as sound*, Last accessed Aug 2, 2023. <https://www.bbc.co.uk/news/10385675>, 2010.
- [124] S. Van Ransbeeck, *Sonification art*, Last accessed Aug 2, 2023 <https://sonificationart.wordpress.com/>.
- [125] B. J. Tomlinson, R. M. Winters, C. Latina, S. Bhat, M. Rane, and B. N. Walker, “Solar system sonification: Exploring earth and its neighbors through sound,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD ’17, 2017, pp. 128–134. DOI: doi.org/10.21785/icad2017.027.
- [126] S. Dini, L. A. Ludovico, S. Mascetti, and M. J. Valero Gisbert, “Translating color: Sonification as a method of sensory substitution within the museum,”

- in *Proceedings of International Web for All Conference*, ser. W4A '23, Association for Computing Machinery, 2023, pp. 162–163. DOI: 10.1145/3587281.3587706.
- [127] A. Sharif, *Sonifer.js*, Last accessed July 20, 2023. <https://github.com/athersharif/sonifier>, 2022.
- [128] A. Sharif, O. H. Wang, and A. T. Muongchan, ““what makes sonification user-friendly?” exploring usability and user-friendliness of sonified responses,” in *Proceedings of International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '22, Association for Computing Machinery, 2022. DOI: 10.1145/3517428.3550360.
- [129] Z. Kondak, T. A. Liang, B. Tomlinson, and B. N. Walker, “Web sonification sandbox-an easy-to-use web application for sonifying data and equations,” WAC '17, 2017.
- [130] M. Poret, J.-M. Celerier, D.-C. Myriam, and S. Catherine, “Proof of concept of a generic toolkit for sonification: The sonification cell in ossia score,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD '03, ICAD, 2023.
- [131] Apple Inc., *Audio graph — apple developer documentation*, https://developer.apple.com/documentation/accessibility/audio_graphs.
- [132] A. Seong and J. Seo, *Datagoboop*, Last accessed July 20, 2023. <https://github.com/akseong/datagoboop>, 2020.
- [133] E. Brown, *Play it by r*, Last accessed July 20, 2023. <http://playitbyr.org/gettingstarted.html>, 2011.
- [134] R. M. Candey, A. M. Schertenleib, and W. L. Diaz Merced, “Xsonify sonification tool for space physics,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD '06, ICAD, 2006.
- [135] S. Pauletto and A. Hunt, “A toolkit for interactive sonification,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD '04, ICAD, 2004.
- [136] S. Phillips and A. Cabrera, “Sonification workstation,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD '19, ICAD, 2019.

- [137] O. Ben-Tal, J. Berger, B. Cook, M. Daniels, G. Scavone, and P. Cook, “Sonart: The sonification application research toolbox,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD ’02, ICAD, 2002, pp. 151–153.
- [138] C. M. Wilson and S. K. Lodha, “Listen: A data sonification toolkit,” ICAD ’96, 1996.
- [139] S. K. Lodha, J. Beahan, T. Heppe, A. Joseph, and B. Zane-Ulman, “Muse: A musical data sonification toolkit,” in *Proceedings of International Conference on Auditory Display*, ser. ICAD ’97, ICAD, 1997.
- [140] S. Barrass, “Personify: A toolkit for perceptually meaningful sonification,” in *Proceedings of the Australian Computer Music Association Conference*, 1995.
- [141] J. Trayford, *Strauss*, <https://github.com/james-trayford/strauss>, 2021.
- [142] J. Hannam, *Starsound*, <https://www.jeffreyhannam.com/starsound>, 2014.
- [143] A. Cibils, *Soda: Sonification of data for learning analytics*, <https://github.com/AndreCI/Soda4LA>, 2020.
- [144] H. Zhao, C. Plaisant, B. Shneiderman, and J. Lazar, “Data sonification for users with visual impairment: A case study with georeferenced data,” *ACM Transactions on Computer-Human Interaction*, vol. 15, no. 1, 2008, ISSN: 1073-0516. DOI: 10.1145/1352782.1352786.
- [145] L. Ferres, G. Lindgaard, L. Sumegi, and B. Tsuji, “Evaluating a tool for improving accessibility to charts and graphs,” *ACM Transactions on Computer-Human Interaction*, vol. 20, no. 5, 2013. DOI: 10.1145/2533682.2533683.
- [146] *Tone.js*, Last accessed Sep 12, 2023. <https://tonejs.github.io/>.
- [147] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, “Reactive vega: A streaming dataflow architecture for declarative interactive visualization,” vol. 22, no. 1, pp. 659–668, 2016. DOI: 10.1109/TVCG.2015.2467091.
- [148] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer, “Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 438–448, 2018. DOI: 10.1109/TVCG.2018.2865240.

- [149] Y. Kim and J. Heer, “Gemini: A grammar and recommender system for animated transitions in statistical graphics,” *IEEE Transactions on Visualization and Computer Graphics*, 2021. [Online]. Available: <https://doi.org/10.1109/TVCG.2020.3030360>.
- [150] J. Hoffswell, A. Borning, and J. Heer, “Setcola: High-level constraints for graph layout,” in *Computer Graphics Forum*, 2018. DOI: 10.1111/cgf.13440.
- [151] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, “Voyager: Exploratory analysis via faceted browsing of visualization recommendations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 649–658, 2016. DOI: 10.1109/TVCG.2015.2467191.
- [152] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer, “Voyager 2: Augmenting visual analysis with partial view specifications,” in *Proc. CHI*, ser. CHI ’17, ACM, 2017, pp. 2648–2659. DOI: 10.1145/3025453.3025768.
- [153] A. Satyanarayan and J. Heer, “Lyra: An Interactive Visualization Design Environment,” *Computer Graphics Forum*, vol. 33, no. 3, pp. 351–360, 2014. DOI: 10.1111/cgf.12391.
- [154] D. Quick and P. Hudak, “Grammar-based automated music composition in haskell,” in *Proceedings of ACM SIGPLAN Workshop on Functional Art, Music, Modeling and Design*, ser. FARM ’13, ACM, 2013, pp. 59–70. DOI: 10.1145/2505341.2505345.
- [155] A. Colmerauer and P. Roussel, “The birth of prolog,” in *History of Programming Languages—II*. ACM, 1996, pp. 331–367. DOI: 10.1145/234286.1057820.
- [156] D. M. McDermott, “The 1998 ai planning systems competition,” *AI magazine*, vol. 21, no. 2, pp. 35–35, 2000.
- [157] D. A. Miller and G. Nadathur, “Higher-order logic programming,” in *Third International Conference on Logic Programming*, E. Shapiro, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 448–462.
- [158] P. O’Donovan, A. Agarwala, and A. Hertzmann, “DesignScape: Design with interactive layout suggestions,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’15, ACM, 2015, pp. 1221–1224. DOI: 10.1145/2702123.2702149.

- [159] A. Swearngin, C. Wang, A. Oleson, J. Fogarty, and A. J. Ko, “Scout: Rapid exploration of interface layout alternatives through high-level design constraints,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’20, ACM, 2020, pp. 1–13. DOI: 10.1145/3313831.3376593.
- [160] J. Mackinlay, “Automating the design of graphical presentations of relational information,” *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, 1986. DOI: 10.1145/22949.22950.
- [161] B. Saket, A. Endert, and Ç. Demiralp, “Task-based effectiveness of basic visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 7, pp. 2505–2512, 2019. DOI: 10.1109/TVCG.2018.2829750.
- [162] G. Brewka, T. Eiter, and M. Truszczyński, “Answer set programming at a glance,” *Communications of the ACM*, 2011. DOI: 10.1145/2043174.2043195.
- [163] V. Lifschitz, “What is answer set programming?” In *Proceedings of the AAAI Conference on Artificial Intelligence*, ser. AAAI ’23, AAAI, 2008, pp. 1594–1597.
- [164] H. Lin, D. Moritz, and J. Heer, “Dziban: Balancing agency & automation in visualization design via anchored recommendations,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’20, ACM, 2020, pp. 1–12. DOI: 10.1145/3313831.3376880.
- [165] R. Ma, H. Mei, H. Guan, W. Huang, F. Zhang, C. Xin, W. Dai, X. Wen, and W. Chen, “Ladv: Deep learning assisted authoring of dashboard visualizations from images and sketches,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 9, pp. 3717–3732, 2021. DOI: 10.1109/TVCG.2020.2980227.
- [166] C. Healey, S. Kocherlakota, V. Rao, R. Mehta, and R. St. Amant, “Visual perception and mixed-initiative interaction for assisted visualization design,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 396–411, 2008. DOI: 10.1109/TVCG.2007.70436.
- [167] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert, “Podium: Ranking data using mixed-initiative visual analytics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 288–297, 2018. DOI: 10.1109/TVCG.2017.2745078.

- [168] Z. Qu and J. Hullman, “Keeping multiple views consistent: Constraints, validations, and exceptions in visualization authoring,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 468–477, 2017. DOI: 10.1109/TVCG.2017.2744198.
- [169] W. Cui, J. Wang, H. Huang, Y. Wang, C.-Y. Lin, H. Zhang, and D. Zhang, “A mixed-initiative approach to reusing infographic charts,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 173–183, 2022. DOI: 10.1109/TVCG.2021.3114856.
- [170] N. Bremer, *Techniques for Data Visualization on both Mobile & Desktop*, <https://www.visualcinnamon.com/2019/04/mobile-vs-desktop-dataviz>, 2019. (visited on 12/12/2019).
- [171] I. Ros, *MobileVis*, Last accessed: Mar 1, 2020, <http://mobilev.is/>, 2014. (visited on 12/12/2019).
- [172] New York Times, *2016: The year in visual stories and graphics*, Last accessed Mar 1, 2019. <https://www.nytimes.com/interactive/2016/12/28/us/year-in-interactive-graphics.html>, 2016.
- [173] New York Times, *2017: The year in visual stories and graphics*, Last accessed Mar 1, 2019. <https://www.nytimes.com/interactive/2017/12/21/us/2017-year-in-graphics.html>, 2017.
- [174] Wall Street Journal, *The best wsj graphics and visual stories from 2016*, <https://www.wsj.com/graphics/graphics-year-in-review-2016/>, 2016. (visited on 03/01/2019).
- [175] Wall Street Journal, *The year in graphics: 2017*, <https://www.wsj.com/graphics/year-in-graphics-2017>, 2017. (visited on 03/01/2019).
- [176] J. M. Corbin and A. Strauss, “Grounded theory research: Procedures, canons, and evaluative criteria,” *Qualitative sociology*, vol. 13, no. 1, pp. 3–21, 1990. DOI: 10.1007/BF00988593.
- [177] J. Lofland and L. H. Lofland, *Analyzing social settings*. Wadsworth Pub, 1971.
- [178] J. Hullman, S. Drucker, N. H. Riche, B. Lee, D. Fisher, and E. Adar, “A deeper understanding of sequence in narrative visualization,” *IEEE Transactions on*

- visualization and computer graphics*, vol. 19, no. 12, pp. 2406–2415, 2013. DOI: 10.1109/TVCG.2013.119.
- [179] J. Hullman, N. Diakopoulos, and E. Adar, “Contextifier: Automatic generation of annotated stock visualizations,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’13, ACM, 2013, pp. 2707–2716. DOI: 10.1145/2470654.2481374.
- [180] H. W. J. Rittel and M. M. Webber, “Dilemmas in a general theory of planning,” *Policy Sciences*, vol. 4, no. 2, pp. 155–169, 1973, ISSN: 1573-0891. DOI: 10.1007/BF01405730.
- [181] R. Buchanan, “Wicked problems in design thinking,” *Design Issues*, vol. 8, no. 2, pp. 5–21, 1992, ISSN: 07479360, 15314790. DOI: 10.2307/1511637.
- [182] J. Heer, N. Kong, and M. Agrawala, “Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’09, ACM, 2009, pp. 1303–1312. DOI: 10.1145/1518701.1518897.
- [183] M. Correll, M. Li, G. Kindlmann, and C. Scheidegger, “Looks good to me: Visualizations as sanity checks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 830–839, 2019, ISSN: 2160-9306. DOI: 10.1109/TVCG.2018.2864907.
- [184] S. Burigat and L. Chittaro, “On the effectiveness of overview+detail visualization on mobile devices,” *Personal and Ubiquitous Computing*, vol. 17, no. 2, pp. 371–385, 2013. DOI: 10.1007/s00779-011-0500-3.
- [185] K. Hornbæk and M. Hertzum, “The notion of overview in information visualization,” *International Journal of Human-Computer Studies*, vol. 69, no. 7, pp. 509–525, 2011. DOI: 10.1016/j.ijhcs.2011.02.007.
- [186] J. Talbot, J. Gerth, and P. Hanrahan, “An empirical model of slope ratio comparisons,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2613–2620, 2012, ISSN: 2160-9306. DOI: 10.1109/TVCG.2012.196.
- [187] D. Haehn, J. Tompkin, and H. Pfister, “Evaluating ‘graphical perception’ with cnns,” *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 641–650, 2019. DOI: 10.1109/TVCG.2018.2865138.

- [188] J. Hullman, “Why authors don’t visualize uncertainty,” *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 130–139, 2020. DOI: 10.1109/TVCG.2019.2934287.
- [189] C. North, “Toward measuring visualization insight,” *IEEE Computer Graphics and Applications*, vol. 26, no. 3, pp. 6–9, 2006, ISSN: 1558-1756. DOI: 10.1109/MCG.2006.70.
- [190] R. Burns, S. Carberry, and S. E. Schwartz, “Modeling a graph viewer’s effort in recognizing messages conveyed by grouped bar charts,” in *Proceedings of Conference on User Modeling, Adaptation and Personalization*, ser. UMAP 2013, 2013, pp. 114–126. DOI: 10.1007/978-3-642-38844-6_10.
- [191] R. Amar, J. Eagan, and J. Stasko, “Low-level components of analytic activity in information visualization,” in *IEEE Symposium on Information Visualization*, ser. InfoVis ’05, IEEE Computer Society, 2005, pp. 111–117. DOI: 10.1109/INFVIS.2005.1532136.
- [192] M. Brehmer and T. Munzner, “A multi-level typology of abstract visualization tasks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2376–2385, 2013. DOI: 10.1109/TVCG.2013.124.
- [193] Q. Cui, M. Ward, E. Rundensteiner, and J. Yang, “Measuring data abstraction quality in multiresolution visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, 2006. DOI: 10.1109/TVCG.2006.161.
- [194] E. Zraggen, Z. Zhao, R. Zeleznik, and T. Kraska, “Investigating the effect of the multiple comparisons problem in visual analysis,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’18, ACM, 2018. DOI: 10.1145/3173574.3174053.
- [195] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.
- [196] D. A. Szafir, “Modeling color difference for visualization design,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 392–401, 2018. DOI: 10.1109/TVCG.2017.2744359.
- [197] Alžběta Brychtová and Arzu Çöltekin, “The effect of spatial distance on the discriminability of colors in maps,” *Cartography and Geographic Information Science*, 2017. DOI: 10.1080/15230406.2016.1140074.

- [198] S. Smart and D. A. Szafir, “Measuring the separability of shape, size, and color in scatterplots,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’19, ACM, 2019. DOI: 10.1145/3290605.3300899.
- [199] J. Heer and M. Bostock, “Crowdsourcing graphical perception: Using mechanical turk to assess visualization design,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’10, ACM, 2010. DOI: 10.1145/1753326.1753357.
- [200] M. D. Fairchild, *Color appearance models*. Wiley Blackwell, 2004.
- [201] Ç. Demiralp, M. S. Bernstein, and J. Heer, “Learning perceptual kernels for visualization design,” *IEEE Transactions on Visualization and Computer Graphics*, 2014. DOI: 10.1109/TVCG.2014.2346978.
- [202] C. Villani, “The wasserstein distances,” in *Optimal Transport: Old and New*. Springer Berlin Heidelberg, 2009. DOI: 10.1007/978-3-540-71050-9_6.
- [203] W. S. Cleveland, “Robust locally weighted regression and smoothing scatterplots,” *Journal of the American Statistical Association*, 1979, ISSN: 0162-1459. DOI: 10.1080/01621459.1979.10481038. [Online]. Available: <http://www.jstor.org/stable/2286407>.
- [204] S. van der Walt and N. Smith, *Mpl colormaps*, <https://bids.github.io/colormap/>. Last accessed: March 1, 2021, 2015.
- [205] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider, “Potassco: The potsdam answer set solving collection,” *AI Communications*, vol. 24, no. 2, pp. 107–124, 2011. DOI: 10.3233/AIC-2011-0491.
- [206] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, “Clingo ASP + control: Preliminary report,” 2014, <https://arxiv.org/abs/1405.3694>. arXiv: 1405.3694.
- [207] UW Interactive Data Lab, *Vega: View api*, 2017.
- [208] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero,

- C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, 2020. DOI: 10.1038/s41592-019-0686-2.
- [209] M. Cappellari, R. M. McDermid, K. Alatalo, L. Blitz, M. Bois, F. Bournaud, M. Bureau, A. F. Crocker, R. L. Davies, T. A. Davis, P. T. de Zeeuw, P.-A. Duc, E. Emsellem, S. Khochfar, D. Krajnović, H. Kuntschner, R. Morganti, T. Naab, T. Oosterloo, M. Sarzi, N. Scott, P. Serra, A.-M. Weijmans, and L. M. Young, “The atlas^{3D} project - xx. mass-size and mass- σ distributions of early-type galaxies: Bulge fraction drives kinematics, mass-to-light ratio, molecular gas fraction and stellar initial mass function,” *MNRAS*, 2013. DOI: 10.1093/mnras/stt644. eprint: 1208.3523.
- [210] M. Roser and E. Ortiz-Ospina, *Income inequality*, <https://ourworldindata.org/income-inequality> Last accessed: March 1, 2021, 2013.
- [211] J. Hasell, *Which countries have protected both health and the economy in the pandemic?* <https://ourworldindata.org/covid-health-economy> Last accessed: March 1, 2021, 2020.
- [212] M. Roser, *Human development index (hdi)*, <https://ourworldindata.org/human-development-index>. Last accessed: March 1, 2021, 2014.
- [213] Y. Luo, X. Qin, N. Tang, and G. Li, “Deepeye: Towards automatic data visualization,” in *IEEE 34th International Conference on Data Engineering*, ser. ICDE ’18, IEEE Computer Society, 2018. DOI: 10.1109/ICDE.2018.00019.
- [214] R. Herbrich, “Support vector learning for ordinal regression,” in *Proceedings of International Conference on Artificial Neural Networks*, ser. ICANN ’99, Institution of Engineering and Technology, 1999. DOI: 10.1049/cp:19991091.
- [215] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, “Learning to rank using gradient descent,” in *Proceedings of International Conference on Machine Learning*, ser. ICML ’05, ACM, 2005. DOI: 10.1145/1102351.1102363.
- [216] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo, “VizML: A Machine Learning Approach to Visualization Recommendation,” in *Proceedings of*

- Conference on Human Factors in Computing Systems (CHI)*, ser. CHI '19, ACM, 2019. DOI: 10.1145/3290605.3300358.
- [217] 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 É. Duchesnay, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, 2011, <http://jmlr.org/papers/v12/pedregosa11a.html>.
- [218] M. G. Kendall, *Rank correlation methods*. Griffin, 1948.
- [219] K. G. Jamieson and R. D. Nowak, “Active ranking using pairwise comparisons,” in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., <https://proceedings.neurips.cc/paper/2011/file/6c14da109e294d1e8155be8aa4b1ce8e-Paper.pdf>, Curran Associates, Inc., 2011.
- [220] T. Blascheck. and P. Isenberg., “A replication study on glanceable visualizations: Comparing different stimulus sizes on a laptop computer,” in *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP,, INSTICC, SciTePress*, 2021, ISBN: 978-989-758-488-6. DOI: 10.5220/0010328501330143.
- [221] Z. Qu and J. Hullman, “Evaluating visualization sets: Trade-offs between local effectiveness and global consistency,” in *Proceedings of Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, ser. BELIV '16, ACM, 2016, pp. 44–52. DOI: 10.1145/2993901.2993910.
- [222] W. Javed and N. Elmqvist, “Exploring the design space of composite visualization,” in *IEEE Pacific Visualization Symposium*, ser. PacificVis '12, IEEE Computer Society, 2012. DOI: 10.1109/PacificVis.2012.6183556.
- [223] B. Saket, A. Srinivasan, E. D. Ragan, and A. Endert, “Evaluating interactive graphical encodings for data visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 3, pp. 1316–1330, 2018. DOI: 10.1109/TVCG.2017.2680452.
- [224] Y. Ma, A. K. H. Tung, W. Wang, X. Gao, Z. Pan, and W. Chen, “Scatternet: A deep subjective similarity model for visual analysis of scatterplots,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1562–1576, 2020. DOI: 10.1109/TVCG.2018.2875702.

- [225] J. Bryant and M. Jones, “Responsive web design,” in *Pro HTML5 Performance*. Apress, 2012, pp. 37–49. DOI: 10.1007/978-1-4302-4525-4_4.
- [226] S. Mohorovičić, “Implementing responsive web design for enhanced web presence,” in *36th International Convention on Information and Communication Technology, Electronics and Microelectronics*, ser. MIPRO ’13, 2013, pp. 1206–1210.
- [227] MDN, *Using media queries*, Last accessed Sept 4, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries.
- [228] E. Bentley, “The web as medium for data visualization,” in *The Data Journalism Handbook: Towards a Critical Data Practice*, L. Bounegru and J. Gray, Eds., Amsterdam University Press, 2021, pp. 182–192. DOI: 10.5117/9789462989511.
- [229] C. Sam, “Ai2html and its impact on the news graphics industry,” in *MobileVis ’18 Workshop at CHI 2018*, https://mobilevis.github.io/assets/mobilevis2018_paper_20.pdf, 2018.
- [230] Google, *Using google charts*, Last accessed Sep 11, 2020. <https://developers-dot-devsite-v2-prod.appspot.com/chart/interactive/docs> Last accessed: Sept. 11, 2020., 2019.
- [231] R. Gal, *Responsive visualizations coming to power bi*, <https://powerbi.microsoft.com/en-us/blog/responsive-visualizations-coming-to-power-bi/> Last accessed: Jun. 2, 2021, 2017.
- [232] L. C. Rost and G. Aisch, *Our new tables: Responsive, with sparklines, bar charts and sticky rows*, Last accessed: Jun. 2, 2021. <https://blog.datawrapper.de/new-table-tool-barcharts-fixed-rows-responsive-2/>, 2020.
- [233] L. C. Rost, *Create better, more responsive text annotations (yes, also on maps)*, <https://blog.datawrapper.de/better-more-responsive-annotations-in-datawrapper-data-visualizations/> Last accessed: Jun. 2, 2021, 2020.
- [234] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert, “Altair: Interactive statistical visualizations for python,” *Journal of Open Source*

- Software*, vol. 3, no. 32, p. 1057, 2018. DOI: 10.21105/joss.01057. [Online]. Available: <https://doi.org/10.21105/joss.01057>.
- [235] R. A. Becker, W. S. Cleveland, and M.-J. Shyu, “The visual design and control of trellis display,” *Journal of Computational and Graphical Statistics*, vol. 5, no. 2, pp. 123–155, 1996. DOI: 10.1080/10618600.1996.10474701.
- [236] MDN, *Cascade and inheritance*, Last accessed Aug 13, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance.
- [237] MDN, *Specificity*, Last accessed Dec 15, 2021. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>.
- [238] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu, “Ai4vis: Survey on artificial intelligence approaches for data visualization,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2021. DOI: 10.1109/TVCG.2021.3099002.
- [239] G. G. Méndez, U. Hinrichs, and M. A. Nacenta, “Bottom-up vs. top-down: Trade-offs in efficiency, understanding, freedom and creativity with infovis tools,” in *Proceedings of CHI Conference on Human Factors in Computing Systems (CHI '17)*, CHI '17. Association for Computing Machinery, 2017, pp. 841–852. DOI: 10.1145/3025453.3025942.
- [240] IDL, *Documentation-vega*, Last accessed: Sept. 9, 2021, 2017. [Online]. Available: <https://vega.github.io/vega/docs/>.
- [241] G. Ellis and A. Dix, “Enabling automatic clutter reduction in parallel coordinate plots,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 717–724, 2006. DOI: 10.1109/TVCG.2006.138.
- [242] *Svelte*, <https://svelte.dev/>.
- [243] S. Ramírez, *FastAPI*, <https://fastapi.tiangolo.com/>.
- [244] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006. DOI: 10.1191/1478088706qp0630a.

- [245] L. Buchanan, J. Huang, and A. Pearce, *Nine rounds a second: How the las vegas gunman outfitted a rifle to fire faster*, Last accessed Sep. 3, 2023. <https://www.nytimes.com/interactive/2017/10/02/us/vegas-guns.html>, 2017.
- [246] G. Dubus and R. Bresin, “A systematic review of mapping strategies for the sonification of physical quantities,” *PLOS ONE*, vol. 8, no. 12, pp. 1–28, 2013. DOI: 10.1371/journal.pone.0082491.
- [247] D. Purves, G. Augustine, D. Fitzpatrick, L. Katz, A.-S. LaMantia, J. McNamara, and S. Williams, “The audible spectrum,” in *Neuroscience*, 2nd, <https://www.ncbi.nlm.nih.gov/books/NBK10924/>, Sinauer Associates, 2001.
- [248] L. Battle, D. Feng, and K. Webber, “Exploring d3 implementation challenges on stack overflow,” in *2022 IEEE Visualization and Visual Analytics*, ser. VIS ’22, 2022, pp. 1–5. DOI: 10.1109/VIS54862.2022.00009.
- [249] L. Wilkinson, *The grammar of graphics*. Springer, 2012.
- [250] *Vega-datasets*, Last accessed Sep. 1, 2023. <https://github.com/vega/vega-datasets>.
- [251] *Csound*, Last accessed Sep. 5, 2023. <https://csound.com/>.
- [252] MDN, *Web audio api*, Last accessed Aug 5, 2023. https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API.
- [253] MDN, *Web speech api*, Last accessed Aug 5, 2023. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API.
- [254] Apple, *Attack, decay, sustain, and release*, Last accessed Aug 5, 2023. <https://support.apple.com/guide/logicpro/attack-decay-sustain-and-release-lgsife419620/mac>.
- [255] MDN, *Audionode - web apis*, Last accessed Sep 12, 202. <https://developer.mozilla.org/en-US/docs/Web/API/AudioNode>. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries.
- [256] Wikipedia, *Pulse-code modulation*, Last accessed Sep. 6, 2023. https://en.wikipedia.org/wiki/Pulse-code_modulation.

- [257] F. Elavsky, L. Nadolskis, and D. Moritz, “Data navigator: An accessibility-centered data navigation toolkit,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 803–813, 2024. DOI: 10.1109/TVCG.2023.3327393.
- [258] M. Tory, L. Bartram, B. Fiore-Gartland, and A. Crisan, “Finding their data voice: Practices and challenges of dashboard users,” *IEEE Computer Graphics and Applications*, 2021. DOI: 10.1109/MCG.2021.3136545.
- [259] J. Seo, S. McCurry, and A. Team, “Latex is not easy: Creating accessible scientific documents with r markdown,” *Journal on Technology and Persons with Disabilities*, vol. 7, pp. 157–171, 2019.
- [260] J. Seo and G. Richard, “Coding through touch: Exploring and re-designing tactile making activities with learners with visual dis/abilities [conferencia],” in *International Conference of the Learning Sciences*, ser. ICLS ’20, 2020. DOI: 10.22318/icls2020.1373.
- [261] J. Seo, Y. Xia, B. Lee, S. McCurry, and Y. J. Yam, “Maidr: Making statistical visualizations accessible with multimodal data representation,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’24, ACM, 2024. DOI: 10.1145/3613904.3642730.
- [262] A. Sharif, O. H. Wang, A. T. Muongchan, K. Reinecke, and J. O. Wobbrock, “Voxlens: Making online data visualizations accessible with an interactive javascript plug-in,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’22, ACM, 2022. DOI: 10.1145/3491102.3517431.
- [263] J. Heer and D. Moritz, “Mosaic: An architecture for scalable & interoperable data views,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 436–446, 2024. DOI: 10.1109/TVCG.2023.3327189.
- [264] M. Borowski, L. Murray, R. Bagge, J. B. Kristensen, A. Satyanarayan, and C. N. Klokmoose, “Varv: Reprogrammable interactive software as a declarative data structure,” in *Proceedings of CHI Conference on Human Factors in Computing Systems*, ser. CHI ’22, ACM, 2022. DOI: 10.1145/3491102.3502064.
- [265] GitHub, *Copilot*, <https://github.com/features/copilot>.
- [266] *Rollup*, <https://rollupjs.org/>.

- [267] J. C. Koone, C. M. Dashnaw, E. A. Alonzo, M. A. Iglesias, K.-S. Patero, J. J. Lopez, A. Y. Zhang, B. Zechmann, N. E. Cook, M. S. Minkara, C. A. Supalo, H. B. Wedler, M. J. Guberman-Pfeffer, and B. F. Shaw, “Data for all: Tactile graphics that light up with picture-perfect resolution,” *Science Advances*, vol. 8, no. 33, eabq2640, 2022. DOI: 10.1126/sciadv.abq2640.
- [268] W. Oropallo and L. A. Piegl, “Ten challenges in 3d printing,” *Engineering with Computers*, vol. 32, pp. 135–148, 2016. DOI: 10.1007/s00366-015-0407-0.
- [269] C. Donalek, S. G. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, S. Davidoff, J. S. Norris, and G. Longo, “Immersive and collaborative data visualization using virtual reality platforms,” in *2014 IEEE International Conference on Big Data*, ser. Big Data '14, 2014, pp. 609–614. DOI: 10.1109/BigData.2014.7004282.
- [270] M. Bostock, *D3-format*, Last accessed Sept 4, 2021, 2015. [Online]. Available: <https://github.com/d3/d3-format>.
- [271] E. Kandogan, “Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations,” in *IEEE Conference on Visual Analytics Science and Technology*, ser. VAST '12, IEEE Computer Society, 2012. DOI: 10.1109/VAST.2012.6400487.

Appendix A

Responsive Visualization Cases for the Design Space and Tradeoff Analyses

Table A.1: List of responsive visualization cases used in Chapter 3 (part 1)

No.	Publisher	URL	Title	Chart type
E1	NYT	https://www.nytimes.com/elections/results/president	Presidential Election Results: Donald J. Trump Wins	Bar
E2	NYT	https://www.nytimes.com/interactive/2016/04/19/us/elections/new-york-city-republican-primary-results.html#11/40.7100/-73.9800	How Every New York City Neighborhood Voted in the Republican Primary	Choropleth
E3	NYT	https://www.nytimes.com/elections/2016/national-results-map	Detailed Maps of Where Trump, Cruz, Clinton and Sanders Have Won	Choropleth
E4	NYT	https://www.nytimes.com/interactive/2016/09/04/science/global-warming-increases-nuisance-flooding.html	A Sharp Increase In 'Sunny Day' Flooding	Map
E5	NYT	https://www.nytimes.com/interactive/2016/04/16/upshot/stephen-curry-golden-state-warriors-3-pointers.html	Stephen Curry's 3-Point Record in Context: Off the Charts	Line
E6	NYT	https://www.nytimes.com/interactive/2016/05/22/world/europe/europe-right-wing-austria-hungary.html	How Far Is Europe Swinging to the Right?	Bar, Stacked
E7	NYT	https://www.nytimes.com/interactive/2016/04/29/upshot/money-race-and-success-how-your-school-district-compares.html	Money, Race and Success: How Your School District Compares	Bubble, Scatterplot
E8	NYT	https://www.nytimes.com/interactive/2016/02/19/us/2015-year-in-weather-temperature-precipitation.html	How Much Warmer Was Your City in 2015?	Area, Bar, Line
E9	NYT	https://www.nytimes.com/interactive/2016/04/05/us/elections/state-voter-histograms.html	What's Driving Trump and Clinton Voters to the Polls	Bar
E10	NYT	https://www.nytimes.com/interactive/2016/03/11/us/elections/what-parties-debate-or-ignore.html	Which Issues Each Party Debates, or Ignores	Bubble, Dot

List of responsive visualization cases used in Chapter 3 (part 2)

No.	Publisher	URL	Title	Chart type
E11	NYT	https://www.nytimes.com/interactive/2016/12/06/upshot/how-to-know-what-donald-trump-really-cares-about-look-at-who-hes-insulting.html	How to Know What Donald Trump Really Cares About: Look at What He's Insulting	Area, Stacked
E12	NYT	https://www.nytimes.com/interactive/2016/12/26/upshot/duck-dynasty-vs-modern-family-television-maps.html	'Duck Dynasty' vs. 'Modern Family': 50 Maps of the U.S. Cultural Divide	Choropleth
E13	NYT	https://www.nytimes.com/interactive/2016/02/27/upshot/republican-delegate-calculator-how-trump-can-win.html	The Cold Hard Math of How Trump Can Win, and How Rubio Can Stop Him	Line
E14	NYT	https://www.nytimes.com/interactive/2016/05/18/upshot/which-buildings-in-manhattan-couldnt-be-built-again-today.html	Which Buildings in Manhattan Couldn't Be Built Today?	Choropleth
E15	NYT	https://www.nytimes.com/interactive/2016/10/30/upshot/florida-poll.html	Latest Upshot Poll Shows Trump With a Lead in Florida	Dot-Map
E16	NYT	https://www.nytimes.com/interactive/2016/01/07/us/drug-overdose-deaths-in-the-us.html	How the Epidemic of Drug Overdose Deaths Rippled Across America	Choropleth
E17	NYT	https://www.nytimes.com/interactive/2016/05/18/us/chicago-murder-problem.html	Chicago's Murder Problem	Line
E18	NYT	https://www.nytimes.com/interactive/2016/09/08/us/us-murder-rates.html	Murder Rates Rose in a Quarter of the Nation's 100 Largest Cities	Shape, Map
E19	NYT	https://www.nytimes.com/interactive/2016/07/25/us/politics/political-firsts.html	Hillary Clinton Broke One Glass Ceiling. When Were Others Broken?	Heatmap
E20	NYT	https://www.nytimes.com/interactive/2016/07/22/us/politics/trump-immigration-ban-how-could-it-work.html	Millions Could Be Blocked From Entering the U.S. Depending on How Trump Would Enforce a Ban on Muslim Immigration	Cartogram
E21	NYT	https://www.nytimes.com/2016/09/02/upshot/new-geography-of-prisons.html	This small Indiana county sends more people to prison than San Francisco and Durham, N.C., combined. Why?	Line
E22	NYT	https://www.nytimes.com/interactive/2017/08/24/us/hurricane-harvey-texas.html	Maps: Tracking Harvey's Destructive Path Through Texas and Louisiana	Choropleth, Line
E23	NYT	https://www.nytimes.com/interactive/2017/10/02/us/vegas-guns.html	Nine Rounds a Second: How the Las Vegas Gunman Outfitted a Rifle to Fire Faster	Scatterplot
E24	NYT	https://www.nytimes.com/interactive/2017/10/06/us/las-vegas-gun-deaths.html	Comparing the Las Vegas Attack With Daily Gun Deaths in U.S. Cities	Isotype
E25	NYT	https://www.nytimes.com/interactive/2017/06/22/climate/95-degree-day-maps.html	95-Degree Days: How Extreme Heat Could Spread Across the World	Choropleth
E26	NYT	https://www.nytimes.com/interactive/2017/08/07/upshot/music-fandom-maps.html	What Music Do Americans Love the Most? 50 Detailed Fan Maps	Choropleth
E27	NYT	https://www.nytimes.com/interactive/2017/10/22/world/middleeast/isis-the-islamic-state-from-insurgency-to-rogue-state-and-back.html	The Islamic State: From Insurgency to Rogue State and Back	Bubble, Map
E28	NYT	https://www.nytimes.com/interactive/2017/01/15/us/politics/you-draw-obama-legacy.html	You Draw It: What Got Better or Worse During Obama's Presidency	Area, Line
E29	NYT	https://www.nytimes.com/interactive/2017/01/18/upshot/some-colleges-have-more-students-from-the-top-1-percent-than-the-bottom-60.html	Some Colleges Have More Students From the Top 1 Percent Than the Bottom 60. Find Yours.	Treemap
E30	NYT	https://www.nytimes.com/interactive/2017/02/27/upshot/whats-normal-whats-important-a-ranking-of-20-events-in-the-trump-administration.html	Just How Abnormal Is the Trump Presidency? Rating 20 Events	Scatterplot

List of responsive visualization cases used in Chapter 3 (part 3)

No.	Publisher	URL	Title	Chart type
E31	NYT	https://www.nytimes.com/interactive/2017/03/08/upshot/who-wins-and-who-loses-under-republicans-health-care-plan.html	Who Wins and Who Loses Under Republicans' Health Care Plan	Choropleth
E32	NYT	https://www.nytimes.com/interactive/2017/03/09/us/politics/who-is-really-affected-by-rising-obamacare-premiums.html	How Many People Are Affected by Obamacare Premium Increases? (Hint, It's Fewer Than You Think)	Bar, Stacked
E33	NYT	https://www.nytimes.com/interactive/2017/04/01/us/politics/how-much-people-in-the-trump-administration-are-worth-financial-disclosure.html	How Much People in the Trump Administration Are Worth	Bubble
E34	NYT	https://www.nytimes.com/interactive/2017/04/23/world/europe/french-election-results-maps.html	How the Election Split France	None
E35	NYT	https://www.nytimes.com/interactive/2017/05/12/world/europe/wannacry-ransomware-map.html	Animated Map of How Tens of Thousands of Computers Were Infected With Ransomware	Dot-Map
E36	NYT	https://www.nytimes.com/interactive/2017/05/25/sports/basketball/lebron-career-playoff-points-record.html	LeBron James Scores 5,995th Playoff Point, Taking the Record From Michael Jordan	Line
E37	NYT	https://www.nytimes.com/interactive/2017/06/08/world/europe/british-general-election-results-analysis.html	How Britain Voted	Bar, Map
E37b	NYT	https://www.nytimes.com/interactive/2017/08/09/upshot/game-of-thrones-chart.html	Good, Evil, Ugly, Beautiful: Help Us Make a 'Game of Thrones' Chart	Scatterplot
E38	NYT	https://www.nytimes.com/interactive/2017/09/09/us/hurricane-irma-records.html	Hurricane Irma Is One of the Strongest Storms In History	Flowmap, Line
E39	NYT	https://www.nytimes.com/interactive/2017/11/15/us/politics/every-tax-cut-in-the-house-tax-bill.html	Every Tax Cut and Tax Increase in the House G.O.P. Bill and What It Would Cost	Bar, Bubble
E40	NYT	https://www.nytimes.com/interactive/2017/12/05/us/politics/tax-bill-salt.html	Among the Tax Bill's Biggest Losers: High-Income, Blue State Taxpayers	Bubble, Line, Scatterplot
E41	NYT	https://www.nytimes.com/elections/results/alabama-senate-special-election-roy-moore-doug-jones	Alabama Election Results: Doug Jones Defeats Roy Moore in U.S. Senate Race	Bar, Choropleth
E42	NYT	https://www.nytimes.com/interactive/2017/12/17/upshot/tax-calculator.html	Tax Bill Calculator: Will Your Taxes Go Up or Down?	Scatterplot
E43	NYT	https://www.nytimes.com/interactive/2017/01/13/us/politics/trump-cabinet-women-minorities.html	Trump's Cabinet So Far Is More White and Male Than Any First Cabinet Since Reagan's	Bar
E44	NYT	https://www.nytimes.com/interactive/2017/01/18/science/earth/2016-hottest-year-on-record.html	How 2016 Became Earth's Hottest Year on Record	Scatterplot
E45	NYT	https://www.nytimes.com/interactive/2017/04/11/business/oreilly-advertisers.html	Bill O'Reilly's Show Lost More Than Half Its Advertisers in a Week	Bar
E46	NYT	https://www.nytimes.com/interactive/2017/06/05/upshot/opioid-epidemic-drug-overdose-deaths-are-rising-faster-than-ever.html	Drug Deaths in America Are Rising Faster Than Ever	Line
E47	NYT	https://www.nytimes.com/interactive/2017/06/14/world/europe/migrant-rescue-efforts-deadly.html	Efforts to Rescue Migrants Caused Deadly, Unexpected Consequences	Dot-Map
E48	NYT	https://www.nytimes.com/interactive/2017/06/28/nyregion/subway-delays-overcrowding.html	Every New York City Subway Line Is Getting Worse. Here's Why.	Line
E49	NYT	https://www.nytimes.com/interactive/2017/05/14/upshot/if-americans-can-find-north-korea-on-a-map-theyre-more-likely-to-prefer-diplomacy.html	If Americans Can Find North Korea on a Map, They're More Likely to Prefer Diplomacy - A	Dot-Map

List of responsive visualization cases used in Chapter 3 (part 4)

No.	Publisher	URL	Title	Chart type
E50	NYT	https://www.nytimes.com/interactive/2017/07/28/climate/more-frequent-extreme-summer-heat.html	It's Not Your Imagination. Summers Are Getting Hotter.	Area
E51	NYT	https://www.nytimes.com/interactive/2017/08/07/nyregion/new-yorks-subways-are-not-just-delayed-some-trains-dont-run-at-all.html	New York's Subways Are Not Just Delayed. Some Trains Don't Run at All.	Bar, Line
E52	NYT	https://www.nytimes.com/interactive/2017/08/24/us/affirmative-action.html	Even With Affirmative Action, Blacks and Hispanics Are More Underrepresented at Top Colleges Than 35 Years Ago	Line
E53	NYT	https://www.nytimes.com/interactive/2017/09/27/us/politics/six-charts-to-explain-the-republican-tax-plan.html	Six Charts That Help Explain the Republican Tax Plan	Dot, Line
E54	NYT	https://www.nytimes.com/2017/11/07/world/americas/mass-shootings-us-international-comparisons-suggest-an-answer.html	What Explains U.S. Mass Shootings? International Comparisons Suggest an Answer	Scatterplot
E112	WSJ	https://www.wsj.com/graphics/a-year-of-suicide-bombings/	A Year of Suicide Bombings	Bubble
E113	WSJ	http://www.wsj.com/graphics/coming-to-america/	Coming to America	Area, Line, Stacked
E114	WSJ	https://www.wsj.com/graphics/uk-eu-trade/	Mind the Gap: The U.K.-EU Trade Challenge	Bar
E115	WSJ	https://www.wsj.com/graphics/french-assembly-2017/	France's New Political Class	Bar, Stacked
E116	WSJ	https://www.wsj.com/articles/one-nation-under-xi-jinping-1508224864?tesla=y	One Nation Under Xi Jinping	Area, Bar, Line
E117	WSJ	https://www.wsj.com/graphics/how-this-tech-rally-is-different-from-1999/	How This Tech Rally Is Different From 1999	Scatterplot
E118	WSJ	http://www.wsj.com/graphics/trump-market-tweets/	Think a Negative Tweet From Trump Crushes a Stock? Think Again	Line
E119	WSJ	https://www.wsj.com/articles/octobers-not-as-bleak-as-its-reputation-for-stock-markets-1507384342	October's Not as Bleak as Its Reputation for Stock Markets	Bar
E120	WSJ	https://www.wsj.com/graphics/gop-corporate-tax-plan-winners-losers/	Winner or Loser? How Corporations Could Fare Under the GOP Tax Plan	Bar
E121	WSJ	https://www.wsj.com/graphics/ceopay-2017/	Top to Bottom: Pay for 500 CEOs	Bar
E122	WSJ	https://www.wsj.com/graphics/republican-tax-plan-calculator/	GOP Tax Plan Calculator	Line
E123	WSJ	https://www.wsj.com/graphics/amazon-headquarters/	Courting a Giant	Polar
E124	WSJ	https://www.wsj.com/graphics/big-companies-get-bigger/	Why You Probably Work for a Giant Company, in 20 Charts	Line
E125	WSJ	https://www.wsj.com/graphics/donald-trump-potential-conflicts-of-interest/	Trump, His Children, and 500+ Potential Conflicts of Interest	Network
E126	WSJ	https://www.wsj.com/graphics/house-health-care-holdouts-round-two/	Health-Care Holdouts in the House	Scatterplot
E127	WSJ	https://www.wsj.com/graphics/american-workplace-then-and-now/	Then and Now: The Big Shift at Work	Icon array
E128	WSJ	http://www.wsj.com/graphics/elections/2016/where-they-won/	Where Trump and Clinton Won	Choropleth
E129	WSJ	http://www.wsj.com/graphics/elections/2016/bellwether-counties/	The Most Important Counties to Watch on Election Night	Choropleth
E130	WSJ	http://graphics.wsj.com/how-the-worlds-best-fighter-jets-measure-up/	Comparing the World's Fighter Jets	Bar

List of responsive visualization cases used in Chapter 3 (part 5)

No.	Publisher	URL	Title	Chart type
E131	WSJ	http://graphics.wsj.com/time-use/	Changing Times	Bar
E132	WSJ	http://graphics.wsj.com/citi-revenue/	The Rise and Retreat of the Megabank	Area, Line, Stacked
E133	WSJ	http://www.wsj.com/graphics/markets-splinter-postelection-reaction/	Markets Splinter	Heatmap
E134	WSJ	http://www.wsj.com/graphics/where-zika-can-thrive/	Where the Zika Virus Can Thrive and Take Its Toll in the United States	Choropleth
E135	WSJ	http://graphics.wsj.com/eba-stress-tests-2016/	European Bank Stress Tests 2016	Bar, Dot
E136	WSJ	http://graphics.wsj.com/elections/2016/lining-up-support/	When Losing Presidential Candidates Endorse	Bar, Dot
E137	WSJ	http://graphics.wsj.com/elections/2016/field-guide-red-blue-america/	A Field Guide to Red and Blue America	Bar, Cartogram
E138	WSJ	http://graphics.wsj.com/gender-pay-gap/	What's Your Pay Gap?	Bar, Dot
E139	WSJ	http://graphics.wsj.com/elections/2016/how-trump-happened/	HOW TRUMP HAPPENED	Bubble
E140	WSJ	http://graphics.wsj.com/elections/2016/2016-electoral-college-map-predictions/	Draw the 2016 Electoral College Map	Cartogram, Choropleth
E141	WSJ	http://www.wsj.com/graphics/elections/2016/swing-the-swing-states-to-see-if-hillary-clinton-or-donald-trump-will-win/	Who Will Swing the Swing States?	Bar, Stacked
E142	WSJ	http://graphics.wsj.com/how-men-and-women-see-the-workplace-differently/	How Men & Women See the Workplace Differently	Bar
E143	WSJ	http://graphics.wsj.com/what-percent/	What Percent Are You?	Area, Bar
E144	WSJ	http://graphics.wsj.com/global-growth/	The Global Economy's Shifting Center of Gravity in 11 Charts	Area
E145	WSJ	http://graphics.wsj.com/elections/2016/donald-trump-and-hillary-clintons-popularity-problem/	Trump and Clinton's Popularity Problem	Bar, Dot
E146	WSJ	http://www.wsj.com/graphics/elections/2016/divided-america/	A Divided America	Scatterplot
E147	WSJ	http://graphics.wsj.com/elections/2016/polls/	Polling for the 2016 Election	Line, Scatterplot
E148	WSJ	http://graphics.wsj.com/boards-of-directors-at-SP-500-companies/	Inside America's Boardrooms	Icon array
E149	WSJ	http://graphics.wsj.com/ceo-salary-vs-company-performance/	How Much Do Top CEOs Make?	Bar, Bubble
E150	WSJ	https://www.wsj.com/graphics/tech-startup-stocks-to-watch/	The Startup Stock Tracker	Line
E151	WSJ	http://graphics.wsj.com/stock-game/	CAN YOU PICK A WINNING STOCK?	Line
E152	WSJ	http://www.wsj.com/graphics/how-bond-yields-got-this-low/	How Bond Yields Got This Low	Line
E153	WSJ	http://www.wsj.com/graphics/housing-market-recovery/	Home Values Rebound, But Not For Everyone	Line
E154	WSJ	http://www.wsj.com/graphics/elections/2016/in-polling-who-got-it-right/	In Polling, Who's Getting It Right?	Bar, Dot
E155	WSJ	http://www.wsj.com/graphics/elections/2016/what-to-watch-on-election-night/	What to Watch On Election Night As Results Roll In	Bar
E156	WSJ	http://graphics.wsj.com/elections/2016/hillary-vs-hillary/	Hillary vs. Hillary	Area
E157	WSJ	http://graphics.wsj.com/elections/2016/delegate-simulator/	The Delegate Simulator	Area, Line
E158	WSJ	http://graphics.wsj.com/elections/2016/democratic-delegate-math/	The Delegate Math Facing Clinton and Sanders	Area, Line
E159	WSJ	http://graphics.wsj.com/elections/2016/rnc-convention-delegates/	Republican Convention's Delegate Math Explained	Cartogram
E160	WSJ	http://www.wsj.com/graphics/elections/2016/how-is-your-state-swinging/	How Far is Your State Swinging?	Dot, Line

List of responsive visualization cases used in Chapter 3 (part 6)

No.	Publisher	URL	Title	Chart type
E201	Research	https://projects.isds.tugraz.at/respvis/	Responsive Visualization	Line
E202	Blog post	http://www.datasketch.es/january/code/nadielh/	All fights from DRAGON BALL Z	Network
E203	Blog post	http://nbremer.github.io/sotshouse/	Woningmarkt	Choropleth, Treemap
E204	Blog post	https://whydocatsanddogs.com/cats#chart-vs	Why do cats...?	Bubble
E205	Organizations	http://uis.unesco.org/apps/visualisations/laci/map.html#	UNESCO Learning Assessment Capacity Index	Choropleth
E206	Organizations	https://ich.unesco.org/dive/constellation/?language=en	UNESCO Dive	Network
E207	Organizations	http://www.oecdbetterlifeindex.org	OECD Better Life Index	Bar, Bubble, Bubble-Map, Dot
E208	Organizations	https://data.oecd.org	OECD Data	Bar, Dot, Line
E209	News-Misc	http://newtown.sisajournal-e.com/skin/page/Price01.html	New Town, 30 Years (using Tableau)	Bar, Map, Treemap
E210	Misc	https://www.gapminder.org/tools/	GapMinder Tools	Bubble, Scatterplot
E211	Organizations	https://data.wgea.gov.au/overview	WGEA Data Explorer	Scatterplot
E212	News-Misc	http://0media.tw/t/geoevent/	Islamic Terrorist Attacks All over the World, 1983 - 2015	Bubble-Map
E213	News-Misc	https://hbr.org/2016/06/the-decline-of-yahoo-in-its-own-words	The Decline of Yahoo in Its Own Words	Line
E214	News-Misc	https://hbr.org/2016/04/how-corporate-boards-connect-in-charts	How Corporate Boards Connect, in Charts	Network
E215	NYT	https://www.nytimes.com/interactive/2014/07/01/sports/worldcup/usa-belgium.html?smid=tw-share&r=1	Record-Setting Effort by Howard, but Belgium Ousts U.S.	Dot, Etc
E216	Quartz	https://qz.com/209809/indias-election-trends-live/	India election results—a landslide for the BJP and crushing defeat for Congress	Bar, Icon array
E217	Quartz	https://qz.com/229731/today-s-jobs-report-us-june-july-2014/	Live: US jobs report	Bar, Line
E218	Quartz	https://qz.com/165238/the-biggest-land-rush-in-the-history-of-the-internet-begins-on-february-4/	The biggest land rush in the history of the internet starts on February 4	Icon array
E219	Quartz	https://qz.com/200594/where-the-best-and-worst-tippers-in-america-live/	Where the best and worst tipplers in America live	Scatterplot
E220	Quartz	https://qz.com/109317/latin-america-soccer-player-exports-are-worth-more-than-its-animal-exports/	Latin America earns more from exporting soccer players than live animals	Bubble-Map
E221	Quartz	https://qz.com/98373/the-state-of-us-trade-with-africa-as-obama-visits/	The state of US trade with Africa as Obama visits	Bubble-Map
E222	WP	http://www.washingtonpost.com/wp-srv/special/sports/leagues-of-the-world-cup/	Leagues of the World Cup	Cartogram, Icon array
E223	NYT	http://www.nytimes.com/interactive/2014/06/25/upshot/984-ways-the-united-states-can-advance-to-the-next-round-of-the-world-cup.html?_r=0	984 Ways the United States Can Advance to the Next Round of the World Cup	Heatmap
E224	Organizations	https://oecdregionalwellbeing.org/US17.html	OECD Regional Well Being	Polar
E225	NYT	http://www.nytimes.com/interactive/2014/03/31/science/motorcycle-helmet-laws.html	Fewer Helmets, More Deaths	Area, Bar, Line
E226	Misc	http://www.therefugeeproject.org/#/2018	The Refugee Project	Bar, Bubble-Map, Flowmap
E227	NYT	http://www.nytimes.com/interactive/2014/05/12/upshot/12-upshot-nba-basketball.html?ref=multimedia	Which Team Do You Cheer For? An N.B.A. Fan Map	Choropleth
E228	NYT	http://www.nytimes.com/newsgraphics/2013/09/28/eli-manning-milestone/	For Eli Manning, 150 Games and Counting	Bar
E229	NYT	http://www.nytimes.com/interactive/2014/01/11/us/politics/who-controls-the-states-and-where-they-stand.html	Taking the Battle to the States	Bar, Choropleth
E230	NYT	http://www.nytimes.com/newsgraphics/2014/senate-model/index.html	Who Will Win The Senate?	Line
E231	NYT	https://www.nytimes.com/interactive/2014/upshot/buy-rent-calculator.html	Is It Better to Rent or Buy?	Bar

List of responsive visualization cases used in Chapter 3 (part 7)

No.	Publisher	URL	Title	Chart type
E232	Propublica	https://projects.propublica.org/graphics/temps-around-the-world	Temp Worker Regulations Around the World	Bar, Bubble-Map
E233	Propublica	https://projects.propublica.org/segregation-now/	School Segregation after Brown	Bar, Choropleth
E235	Propublica	https://projects.propublica.org/emails	Message Machine: Reverse-Engineering the 2012 Campaign	Bar
E236	Propublica	https://projects.propublica.org/schools/	The Opportunity Gap	Bar, Heatmap
E237	Propublica	https://projects.propublica.org/nursing-homes/	Nursing Home Inspect	Bar, Choropleth
E239	Propublica	https://projects.propublica.org/emergency/	ER Wait Watcher	Bar
E240	Propublica	https://projects.propublica.org/graphics/koch	How Dark Money Flows Through the Koch Network	Flow
E241	Boston Globe	http://www.bostonglobe.com/news/world/2013/02/27/pope/UvB5Shk5EkawzTG0kRNdCN/story.html	Explore the College of Cardinals	Icon array
E243	Boston Globe	http://www.bostonglobe.com/2012/09/24/schoolplans/t60CkzkzBf0e03IQkLzn8J/story.html	Overhauling the school assignment system	Map
E244	Boston Globe	http://www.bostonglobe.com/2013/01/21/main/4kXqydaTvcSf8WdIt6F1hM/story.html#2013entrusting	Inaugural language	Radar
E245	Boston Globe	http://www.bostonglobe.com/2013/01/30/mai/yWQCjhK71yBaqqgFrakr1M/story.html	Women' a central theme in Menino's speech	Bubble
E246	Boston Globe	http://www.bostonglobe.com/Page/Boston/2011-2020/WebGraphics/Metro/BostonGlobe.com/2013/02/spEvolution/main.xml	Evolution of a tycoon	Icon array
E247	Boston Globe	http://www.bostonglobe.com/2013/06/02/analyzing-ticks-town-town/u30a7MHbJ1yHT7DgDpjTdN/story.html	Analyzing ticks, town by town	Choropleth
E249	Boston Globe	http://www.bostonglobe.com/2013/09/24/results/oFoYkTunQrfNzcp1UtboCP/story.html	Results of the preliminary election for mayor	Bar
E251	Boston Globe	http://www.bostonglobe.com/2014/01/09/snow-totals-mass-northeast/badi2tA5EWgipQN6xjPX3L/story.html	Snow totals in Mass., Northeast	Choropleth
E252	Propublica	https://projects.propublica.org/treatment/	Treatment Tracker	Pie, Polar
E261	Scientific American	https://www.scientificamerican.com/article/viruses-thrive-in-the-arctic-ocean/	Viruses Thrive in the Arctic Ocean	Heatmap
E262	Scientific American	https://www.scientificamerican.com/article/solar-eclipse-charted-for-the-next-1000-years/	Solar Eclipse Charted for the Next 1,000 Years	Map
E263	Scientific American	https://www.scientificamerican.com/article/only-150-of-your-facebook-contacts-are-real-friends/	Only 150 of Your Facebook Contacts Are Real Friends	Line
E264	Scientific American	https://www.scientificamerican.com/article/coming-soon-a-solar-eclipse-near-you/	Coming Soon? A Solar Eclipse Near You	Map, Bar
E265	Misc	http://inauguratespeeches.com/	INAUGURATE	Bar, Heatmap, Line
E266	Scientific American	https://blogs.scientificamerican.com/a-visual/a-visual-guide-to-the-search-for-exoplanets/	A Visual Guide to the Search for Exoplanets	Bubble, Bar
E267	Scientific American	https://blogs.scientificamerican.com/a-visual/china-takes-the-lead-on-clean-energy/	China Takes the Lead on Clean Energy	Line
E268	Scientific American	https://www.scientificamerican.com/article/cetaceans-quo-big-brains-are-linked-to-their-rich-social-life/	Cetaceans' Big Brains Are Linked to Their Rich Social Life	Bubble
E269	Scientific American	https://blogs.scientificamerican.com/a-visual/unexplained-light-curves/	Unexplained Light Curves	Line
E270	Scientific American	https://blogs.scientificamerican.com/a-visual/sleeping-beauties-of-science/	Sleeping Beauties of Science	Bubble

Appendix B

Survey of Responsive Visualization Practitioners

B.1 Methods

We recruited a convenience sample of 19 responsive visualization authors with an average of 5 years of experience with visualization authoring. From the contact information the participants provided, there were four practitioners, three journalists, and six researchers (not every participant provided their contact). We surveyed the participants via Google Forms and attached the screenshots of the actual questionnaire page in Section B.3.

We asked

- whether they start with designing desktop views or mobile views,
- frequency of design process that explicitly considers mobile views,
- rank of seven possible design guidelines for responsive visualization (in 7-point scale), and
- two qualitative questions regarding “rule of thumbs” and difficulties in respon-

sive visualization.

B.2 Results

B.2.1 Design Process

As illustrated in Table B.1, more than half of the authors (11, 58%) responded that they start designing the desktop view and then move on to the mobile version.

Table B.1: Directions for responsive visualization design process. Responses in “ ” are suggested by participants.

Process	Responses
Desktop first	11 (58 %)
- After designing the desktop version, I (or my team) start on the mobile version.	11
Both	5 (26 %)
- I (or my team) design the desktop and mobile versions at the same time.	4
- ‘Drawing the basic visualization on paper and then trying it on both the devices’	1
Mobile first	3 (16 %)
- After designing the mobile version, I (or my team) start on the desktop version.	2
- ‘Optimized for mobile versions (Because of tight deadlines)’	1

13 authors responded that they consider mobile views at least half of the time, and eight authors among them consider more than half of the time (Table B.2).

Table B.2: Frequency of considering mobile views.

Frequency	Responses
10% or less of the time	2 (11%)
More than 10%, but less than half of the time	4 (21%)
About half of the time	5 (26%)
More than half of the time, but less than 90% of the time.	3 (16%)
90% or more of the time.	5 (26%)

B.2.2 Rank of guidelines

In general, maintaining takeaways and information and adjusting interactivity were ranked high.

Table B.3: Average rank indicating perceived importance (out of 7) for seven possible guidelines.

Guideline	Avg. Rank
Maintaining the main “takeaways” or message.	2.12
Maintaining the same information across versions.	3.42
Changing design to acknowledge greater difficulty users face interacting on a SS (e.g., reducing interactivity)	3.42
Maintaining the “information density” (i.e., the amount of information conveyed relative to the screen size)	4.32
Changing design to acknowledge attention limits in mobile context (e.g., reducing information)	4.53
Changing design to acknowledge other technical constraints (e.g., computing power)	4.53
Maintaining aspect ratio across versions.	5.68

B.2.3 Qualitative responses

We open coded qualitative responses in terms of whether they mentioned information density problems. 10 participants appeared to negotiate maintaining message and adjusting information density. We attached relevant responses.

- The simpler, the better (P4)
- Maximize screen space. Not accidentally creating graphics/tables/etc. where the user has to scroll forever to get to the next section. (P6)
- Try to keep everything on a single screen if possible so that people don’t miss something, or have to scroll up and down. (P7)
- (Use) single column layouts. (P11)
- Maximize main view, hide secondary views with toggles. (P14)
- Step by step information reveal rather than showing everything at once. (P15)
- Help users stay ‘oriented’ within the narrative of the visualizations, because only a fraction of the storyline is visible on mobile compared with large-screen

format. (P16)

- Readability. (P17)
- Creating a similar experience without overwhelming the user. (P18)
- Many things work better when rotated 90 degrees, e.g., vertical timelines on mobile instead of horizontal. Little screen space and low touch precision. (P19)

Below, we listed the entire response of two qualitative items.

What other design guidelines or “rules of thumb” do you use in deciding how to design the small screen version of a visualization?

- (P1) Limit interactivity because accidental touches that change experience can ruin the viewing experience.
- (P2) Viz should convey the main message without any interaction. Interaction conveys extra/deeper info.
- (P3) zoomable
- (P4) the simpler the better
- (P5) I’ll sometimes consider a points per pixel and make changes to the layout based on the amount of size I have. I’ll also sometimes add or remove affordances depending on the size (e.g. if on a smaller screen remove some tap interactions because you are more likely to hit them).
- (P6) Maximize screen space (Don’t be afraid to turn maps on their “side.” North is a social construct), reduce tooltips (or make them stick to bottom of screen), make sure anything that’s “touchable” isn’t tiny
- (P7) Try to keep everything on a single screen if possible so that people don’t miss something, or have to scroll up and down.
- (P8) variable font size, horizontal scroll bar on mobile devices

- (P9) Maintain a consistent appearance (sometimes) across different resolutions and devices
- (P10) -
- (P11) Single Column layouts, Reduce to the max
- (P12) Top load critical info needed for understanding, so the reader doesn't need to scroll down and then back up to read, for example, axis labels.
- (P13) Less pointer accuracy
- (P14) Maximize main view, hide secondary views with toggles.
- (P15) Step by step information reveal rather than showing everything at once. Layering the information so that short attention users can see the gist and explorers can dig into the details.
- (P16) Note that my app development work is primarily for research projects, and I have spent only a few months converting research-oriented dataviz into 'consumer' apps. Given that caveat, I included my email below in case of follow-up (also because I am interested in following your research), not for lottery. For small screen development, I have had to pay much more attention to how to help users stay 'oriented' within the narrative of the visualizations, because only a fraction of the storyline is visible on mobile compared with large-screen format. So, I have experimented with ways to repeat, summarize, and connect 'messages' across dataviz screens.
- (P17) readability
- (P18) Visual consistency
- (P19) Many things work better when rotated 90 degrees, e.g. vertical timelines on mobile instead of horizontal

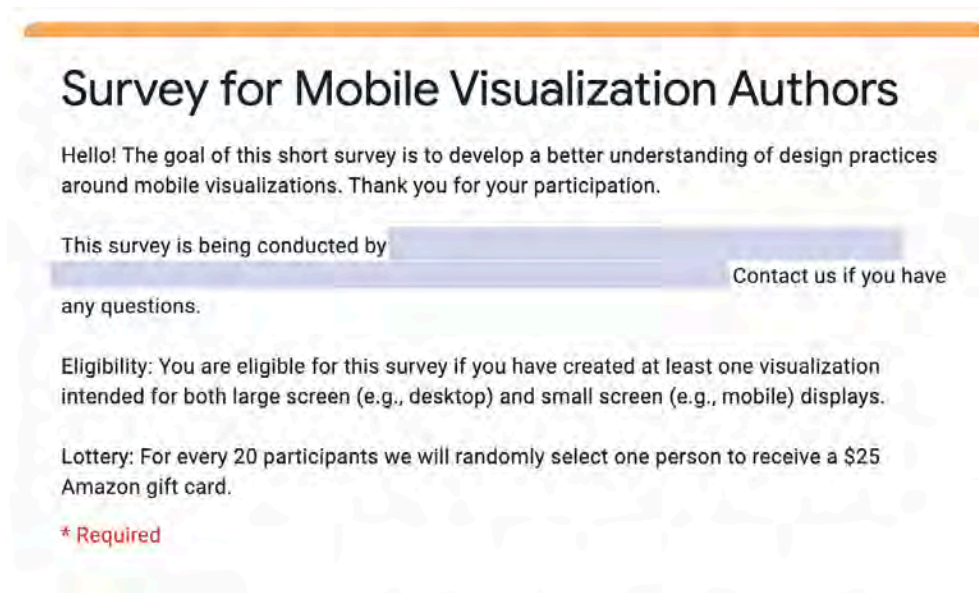
What are the most difficult aspects of creating mobile versions of visualizations?

- (P1) Tooltips. Engineering them to work on both mobile and desktop screen sizes is difficult for me, an intermediate coder at best.
- (P2) Size constraints. Since data viz involves comparison, less space limits the information that can be onscreen at one time for comparison.
- (P3) Aspect, sides, make it zoomable
- (P4) legibility and range of devices
- (P5) Emulating the actual experience of developing for a small device from a computer. While browser dev tools do a lot of work to help smooth the iteration cycle, it only gives glimpses of the experience of looking at it on mobile.
- (P6) Providing the same information while limiting interaction. Not accidentally creating graphics/tables/etc where the user has to scroll forever to get to the next section.
- (P7) Testing. Lot's of different devices and browsers and screen sizes. Difficult to test for everyone.
- (P8) double work if creating mobile specific vis
- (P9) It's often a lot of effort
- (P10) Variety of devices and standards, lack of hover interaction, fat thumb
- (P11) Space, Zooming/ Panning, Tooltips
- (P12) Being unable to rely on reader making immediate cross comparisons between all visual elements
- (P13) Accessibility and text editing
- (P14) Different devices.
- (P15) Screen and UI optimization and dealing with fat finger problems when allowing users to explore the visualizations.
- (P16) The 'guideline' I mentioned above leads to problems of balancing repetitiveness, density of text, inconsistency of visual representation, and oversim-

plification of the ‘messages’ that might cause users to feel disoriented and confused. I also am challenged by users’ expectations of increased, not decreased, interactivity on mobile, because many have expected a more tactile relationship with consumer mobile apps.

- (P17) It is not easy to omit information to reduce complexity (It may makes difficult to communicate the context.)
- (P18) Creating a similar experience without overwhelming the user: Keeping a similar level of functionality and visual representation.
- (P19) Little screen space and low touch precision

B.3 Actual Questionnaire



Survey for Mobile Visualization Authors

Hello! The goal of this short survey is to develop a better understanding of design practices around mobile visualizations. Thank you for your participation.

This survey is being conducted by [redacted] [Contact us if you have any questions.](#)

Eligibility: You are eligible for this survey if you have created at least one visualization intended for both large screen (e.g., desktop) and small screen (e.g., mobile) displays.

Lottery: For every 20 participants we will randomly select one person to receive a \$25 Amazon gift card.

* Required

Choose the option below that best describes your typical design process for creating a visualization for multiple devices. *

- After designing the desktop version, I (or my team) start on the mobile version.
- I (or my team) design the desktop and mobile versions at the same time.
- After designing the mobile version, I (or my team) start on the desktop version.
- Other: _____

When creating visualizations, how often would you say your design process involves explicitly considering mobile views? *

- 10% or less of the time
- More than 10%, but less than half of the time
- About half of the time
- More than half of the time, but less than 90% of the time.
- 90% or more of the time

Changing design to acknowledge attention limits in mobile context (e.g., reducing information)

Changing design to acknowledge other technical constraints (e.g., computing power)

What other design guidelines or "rules of thumb" do you use in deciding how to design the small screen version of a visualization? *

Your answer

What are the most difficult aspects of creating mobile versions of visualizations? *

Your answer

How many years have you been creating both mobile and desktop versions of visualizations? *

Your answer

Appendix C

Detailed Results for Task-oriented Insight Loss Measure Evaluation

C.1 Rank Correlation Results

Figure C.1 shows the rank correlations between disaggregated loss measures for different source visualizations. Based on this graph, it is clear that we see very few correlations between different categories of disaggregated loss measures. In other words, we do not observe strong correlations between any of the three trend loss measures (*trend.y.x*, *trend.size.x.y*, *trend.color.x.y*) and the individual disaggregated loss measures of comparison and identification. This suggests that the loss measures, which we use as features in the model are orthogonal and capture different information about the transformation.



Figure C.1: Rank correlations between disaggregated loss measures.

C.2 Training Results

Table C.1 show the training results from all the model configurations. Baseline 1 refers to changes to chart size, and Baseline 2 refers to transposing axes. *nan** errors are due to divided by zero.

Table C.1: Training results for all model configurations (part 1)

Model Names	Feature Sets	Mapping	Accuracy	F1-Score	Recall	Precision	AUC
1NN	Disaggregated	Concatenate	0.78725398	0.79116835	0.80675422	0.77617329	0.78727224
1NN	Disaggregated	Difference	0.77881912	0.78066914	0.7879925	0.77348066	0.77882771
1NN	Aggregated	Concatenate	0.71602624	0.71970398	0.72983114	0.70985401	0.71603917
1NN	Aggregated	Difference	0.67478913	0.67780873	0.684803	0.67095588	0.6747985
1NN	Disagg+Agg	Concatenate	0.77788191	0.7827681	0.8011257	0.76523297	0.77790368
1NN	Disagg+Agg	Difference	0.77132146	0.77153558	0.77298311	0.77009346	0.77132302
1NN	Baseline 1	Concatenate	0.51077788	0.51576994	0.52157598	0.51009174	0.51078799
1NN	Baseline 1	Difference	0.49015933	0.5125448	0.53658537	0.49056604	0.4902028
1NN	Baseline 2	Concatenate	0.62324274	0.57324841	0.5065666	0.6601467	0.62313349
1NN	Baseline 2	Difference	0.59700094	0.67814371	0.84990619	0.5641345	0.59723774
kNN (k=5)	Disaggregated	Concatenate	0.7628866	0.76852699	0.7879925	0.75	0.76291011
kNN (k=5)	Disaggregated	Difference	0.7769447	0.77962963	0.78986867	0.76965265	0.77695681
kNN (k=5)	Aggregated	Concatenate	0.73477038	0.73772011	0.7467167	0.72893773	0.73478157
kNN (k=5)	Aggregated	Difference	0.72727273	0.72829132	0.73170732	0.72490706	0.72727688
kNN (k=5)	Disagg+Agg	Concatenate	0.76382381	0.76838235	0.78424015	0.75315315	0.76384292
kNN (k=5)	Disagg+Agg	Difference	0.7769447	0.77881041	0.78611632	0.77163904	0.77695329
kNN (k=5)	Baseline 1	Concatenate	0.53327085	0.52930057	0.52532833	0.53333333	0.53326342
kNN (k=5)	Baseline 1	Difference	0.47891284	0.41962422	0.37711069	0.47294118	0.47881752
kNN (k=5)	Baseline 2	Concatenate	0.64761012	0.68243243	0.75797373	0.62058372	0.64771346
kNN (k=5)	Baseline 2	Difference	0.62324274	0.50492611	0.38461538	0.73476703	0.6230193
kNN (k=10)	Disaggregated	Concatenate	0.76476101	0.75701839	0.73358349	0.782	0.76473182
kNN (k=10)	Disaggregated	Difference	0.77788191	0.77277085	0.75609756	0.79019608	0.77786151
kNN (k=10)	Aggregated	Concatenate	0.72727273	0.70870871	0.6641651	0.75965665	0.72721364
kNN (k=10)	Aggregated	Difference	0.72258669	0.70923379	0.67729831	0.7443299	0.72254429
kNN (k=10)	Disagg+Agg	Concatenate	0.76007498	0.75097276	0.72420263	0.77979798	0.76004139
kNN (k=10)	Disagg+Agg	Difference	0.77319588	0.77039848	0.76172608	0.77927063	0.77318514
kNN (k=10)	Baseline 1	Concatenate	0.55107779	0.49631966	0.44277674	0.5645933	0.55097638
kNN (k=10)	Baseline 1	Difference	0.50421743	0.44491081	0.39774859	0.5047619	0.50411774
kNN (k=10)	Baseline 2	Concatenate	0.62605436	0.59737639	0.55534709	0.64628821	0.62598815
kNN (k=10)	Baseline 2	Difference	0.62324274	0.50492611	0.38461538	0.73476703	0.6230193

Training results for all model configurations (part 2)

Model Names	Feature Sets	Mapping	Accuracy	F1-Score	Recall	Precision	AUC
Logistic Regression	Disaggregated	Concatenate	0.78631678	0.78651685	0.7879925	0.78504673	0.78631835
Logistic Regression	Disaggregated	Difference	0.78069353	0.77840909	0.77110694	0.78585086	0.78068456
Logistic Regression	Aggregated	Concatenate	0.75913777	0.7563981	0.74859287	0.76436782	0.7591279
Logistic Regression	Aggregated	Difference	0.76382381	0.76091082	0.75234522	0.7696737	0.76381306
Logistic Regression	Disagg+Agg	Concatenate	0.78631678	0.78651685	0.7879925	0.78504673	0.78631835
Logistic Regression	Disagg+Agg	Difference	0.77975633	0.77767266	0.77110694	0.78435115	0.77974823
Logistic Regression	Baseline 1	Concatenate	0.53701968	0.53571429	0.53470919	0.53672316	0.53701752
Logistic Regression	Baseline 1	Difference	0.54076851	0.56171735	0.5891182	0.53675214	0.54081378
Logistic Regression	Baseline 2	Concatenate	0.62417994	0.50186335	0.37898687	0.74264706	0.62395036
Logistic Regression	Baseline 2	Difference	0.62417994	0.50186335	0.37898687	0.74264706	0.62395036
Linear SVM	Disaggregated	Concatenate	0.78350515	0.78104265	0.77298311	0.78927203	0.7834953
Linear SVM	Disaggregated	Difference	0.78819119	0.7835249	0.7673546	0.80039139	0.78817168
Linear SVM	Aggregated	Concatenate	0.75351453	0.75023742	0.74108818	0.75961538	0.75350289
Linear SVM	Aggregated	Difference	0.76007498	0.75757576	0.75046904	0.76481836	0.76006598
Linear SVM	Disagg+Agg	Concatenate	0.78444236	0.78178368	0.77298311	0.79078695	0.78443163
Linear SVM	Disagg+Agg	Difference	0.78350515	0.77894737	0.76360225	0.79492188	0.78348652
Linear SVM	Baseline 1	Concatenate	0.55014058	0.55637708	0.56472795	0.54826958	0.55015424
Linear SVM	Baseline 1	Difference	0.59044049	0.58967136	0.5891182	0.59022556	0.59043925
Linear SVM	Baseline 2	Concatenate	0.62886598	0.65920826	0.71857411	0.60890302	0.62894998
Linear SVM	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
RBF SVM	Disaggregated	Concatenate	0.6579194	0.65919701	0.66228893	0.65613383	0.65792349
RBF SVM	Disaggregated	Difference	0.76757263	0.75733855	0.7260788	0.79141104	0.76753378
RBF SVM	Aggregated	Concatenate	0.75070291	0.75818182	0.78236398	0.73544974	0.75073255
RBF SVM	Aggregated	Difference	0.74789128	0.75523203	0.77861163	0.73321555	0.74792005
RBF SVM	Disagg+Agg	Concatenate	0.69540769	0.63848721	0.53846154	0.78415301	0.69526073
RBF SVM	Disagg+Agg	Difference	0.74414246	0.7239636	0.67166979	0.78508772	0.7440746
RBF SVM	Baseline 1	Concatenate	0.56888472	0.56273764	0.55534709	0.57032755	0.56887205
RBF SVM	Baseline 1	Difference	0.55014058	0.53488372	0.51782364	0.55310621	0.55011032
RBF SVM	Baseline 2	Concatenate	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
RBF SVM	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
Decision Tree	Disaggregated	Concatenate	0.76194939	0.7612782	0.75984991	0.76271186	0.76194743
Decision Tree	Disaggregated	Difference	0.75913777	0.76137419	0.76923077	0.75367647	0.75914722
Decision Tree	Aggregated	Concatenate	0.70196813	0.69829222	0.69043152	0.70633397	0.70195733
Decision Tree	Aggregated	Difference	0.67947516	0.67977528	0.68105066	0.67850467	0.67947664
Decision Tree	Disagg+Agg	Concatenate	0.76194939	0.76261682	0.76547842	0.75977654	0.7619527
Decision Tree	Disagg+Agg	Difference	0.74695408	0.74813433	0.75234522	0.74397032	0.74695912
Decision Tree	Baseline 1	Concatenate	0.5004686	0.47384008	0.45028143	0.5	0.50042161
Decision Tree	Baseline 1	Difference	0.51077788	0.496139	0.48217636	0.51093439	0.5107511
Decision Tree	Baseline 2	Concatenate	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
Decision Tree	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
Random Forest (k=10)	Disaggregated	Concatenate	0.81068416	0.80725191	0.79362101	0.82135922	0.81066818
Random Forest (k=10)	Disaggregated	Difference	0.8022493	0.79923882	0.7879925	0.81081081	0.80223595
Random Forest (k=10)	Aggregated	Concatenate	0.74882849	0.73980583	0.71482176	0.7665996	0.74879665
Random Forest (k=10)	Aggregated	Difference	0.71602624	0.70724638	0.68667917	0.72908367	0.71599876
Random Forest (k=10)	Disagg+Agg	Concatenate	0.79756326	0.79150579	0.76923077	0.81510934	0.79753673
Random Forest (k=10)	Disagg+Agg	Difference	0.80131209	0.79962193	0.79362101	0.80571429	0.80130489
Random Forest (k=10)	Baseline 1	Concatenate	0.50984067	0.49856184	0.48780488	0.50980392	0.50982004
Random Forest (k=10)	Baseline 1	Difference	0.50515464	0.51470588	0.52532833	0.5045045	0.50517353
Random Forest (k=10)	Baseline 2	Concatenate	0.61480787	0.6661251	0.76923077	0.58739255	0.61495246
Random Forest (k=10)	Baseline 2	Difference	0.62136832	0.68188976	0.81238274	0.58751696	0.62154717

Training results for all model configurations (part 3)

Model Names	Feature Sets	Mapping	Accuracy	F1-Score	Recall	Precision	AUC
Random Forest (k=50)	Disaggregated	Concatenate	0.83036551	0.83036551	0.83114447	0.82958801	0.83036624
Random Forest (k=50)	Disaggregated	Difference	0.82380506	0.82720588	0.84427767	0.81081081	0.82382423
Random Forest (k=50)	Aggregated	Concatenate	0.76663543	0.76575729	0.76360225	0.76792453	0.76663259
Random Forest (k=50)	Aggregated	Difference	0.74695408	0.74576271	0.74296435	0.74858223	0.74695034
Random Forest (k=50)	Disagg+Agg	Concatenate	0.82193065	0.82142857	0.81988743	0.82297552	0.82192873
Random Forest (k=50)	Disagg+Agg	Difference	0.82567948	0.82841328	0.8424015	0.81488203	0.82569513
Random Forest (k=50)	Baseline 1	Concatenate	0.50890347	0.5	0.49155722	0.50873786	0.50888723
Random Forest (k=50)	Baseline 1	Difference	0.50609185	0.51338873	0.52157598	0.50545455	0.50610634
Random Forest (k=50)	Baseline 2	Concatenate	0.61199625	0.68445122	0.8424015	0.57637997	0.61221199
Random Forest (k=50)	Baseline 2	Difference	0.62886598	0.70491803	0.88742964	0.58467244	0.62910808
Random Forest (k=100)	Disaggregated	Concatenate	0.84067479	0.84259259	0.85365854	0.83180987	0.84068695
Random Forest (k=100)	Disaggregated	Difference	0.82286785	0.82802548	0.85365854	0.80388693	0.82289668
Random Forest (k=100)	Aggregated	Concatenate	0.77413308	0.77413308	0.77485929	0.77340824	0.77413376
Random Forest (k=100)	Aggregated	Difference	0.74320525	0.7424812	0.74108818	0.74387947	0.74320327
Random Forest (k=100)	Disagg+Agg	Concatenate	0.82755389	0.82899628	0.83677298	0.8213628	0.82756252
Random Forest (k=100)	Disagg+Agg	Difference	0.82380506	0.82720588	0.84427767	0.81081081	0.82382423
Random Forest (k=100)	Baseline 1	Concatenate	0.51546392	0.50808754	0.50093809	0.51544402	0.51545032
Random Forest (k=100)	Baseline 1	Difference	0.50984067	0.51258155	0.51594747	0.50925926	0.50984639
Random Forest (k=100)	Baseline 2	Concatenate	0.62043112	0.69662921	0.87242026	0.5798005	0.62066706
Random Forest (k=100)	Baseline 2	Difference	0.63074039	0.70728083	0.89305816	0.58548585	0.63098601
Naive Bayes	Disaggregated	Concatenate	0.71696345	0.71563089	0.71294559	0.71833648	0.71695969
Naive Bayes	Disaggregated	Difference	0.76007498	0.77504394	0.82739212	0.72892562	0.76013801
Naive Bayes	Aggregated	Concatenate	0.71415183	0.71415183	0.71482176	0.71348315	0.71415245
Naive Bayes	Aggregated	Difference	0.74414246	0.75559534	0.79174484	0.72260274	0.74418703
Naive Bayes	Disagg+Agg	Concatenate	0.72071228	0.71673004	0.70731707	0.72639692	0.72069974
Naive Bayes	Disagg+Agg	Difference	0.7628866	0.7755102	0.81988743	0.73569024	0.76293997
Naive Bayes	Baseline 1	Concatenate	0.54170572	0.56763926	0.60225141	0.5367893	0.54176241
Naive Bayes	Baseline 1	Difference	0.54076851	0.56171735	0.5891182	0.53675214	0.54081378
Naive Bayes	Baseline 2	Concatenate	0.45923149	0.41182467	0.37898687	0.45089286	0.45915635
Naive Bayes	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
Adaptive Boosting	Disaggregated	Concatenate	0.81162137	0.81197381	0.81425891	0.80970149	0.81162384
Adaptive Boosting	Disaggregated	Difference	0.7900656	0.79220779	0.8011257	0.78348624	0.79007596
Adaptive Boosting	Aggregated	Concatenate	0.72727273	0.72727273	0.72795497	0.72659176	0.72727337
Adaptive Boosting	Aggregated	Difference	0.73008435	0.74147217	0.77485929	0.71084337	0.73012627
Adaptive Boosting	Disagg+Agg	Concatenate	0.79943768	0.79962547	0.8011257	0.79813084	0.79943926
Adaptive Boosting	Disagg+Agg	Difference	0.78725398	0.79307201	0.81613508	0.7712766	0.78728103
Adaptive Boosting	Baseline 1	Concatenate	0.56419869	0.54277286	0.51782364	0.57024793	0.56415527
Adaptive Boosting	Baseline 1	Difference	0.53045923	0.52601703	0.52157598	0.53053435	0.53045091
Adaptive Boosting	Baseline 2	Concatenate	0.62417994	0.50186335	0.37898687	0.74264706	0.62395036
Adaptive Boosting	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241

Training results for all model configurations (part 4)

Model Names	Feature Sets	Mapping	Accuracy	F1-Score	Recall	Precision	AUC
QDA	Disaggregated	Concatenate	0.74601687	0.74649205	0.74859287	0.74440299	0.74601928
QDA	Disaggregated	Difference	0.74507966	0.74953959	0.76360225	0.73598553	0.74509701
QDA	Aggregated	Concatenate	0.74507966	0.74531835	0.7467167	0.74392523	0.7450812
QDA	Aggregated	Difference	0.76476101	0.76823638	0.7804878	0.75636364	0.76477574
QDA	Disagg+Agg	Concatenate	0.70759138	0.71891892	0.74859287	0.6915078	0.70762977
QDA	Disagg+Agg	Difference	0.67197751	0.64212679	0.5891182	0.70561798	0.67189992
QDA	Baseline 1	Concatenate	0.5004686	nan*	nan*	nan*	0.5
QDA	Baseline 1	Difference	0.5004686	0	0	0	0.5
QDA	Baseline 2	Concatenate	0.54358013	0.6113328	0.71857411	0.53194444	0.54374398
QDA	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
Gradient Boosting	Disaggregated	Concatenate	0.81818182	0.81835206	0.81988743	0.81682243	0.81818342
Gradient Boosting	Disaggregated	Difference	0.80880975	0.8121547	0.82739212	0.79746835	0.80882715
Gradient Boosting	Aggregated	Concatenate	0.77319588	0.77169811	0.7673546	0.77609108	0.77319041
Gradient Boosting	Aggregated	Difference	0.75257732	0.76130199	0.78986867	0.73472949	0.75261224
Gradient Boosting	Disagg+Agg	Concatenate	0.8294283	0.82990654	0.83302064	0.82681564	0.82943167
Gradient Boosting	Disagg+Agg	Difference	0.80037488	0.80440771	0.8217636	0.78776978	0.80039491
Gradient Boosting	Baseline 1	Concatenate	0.53514527	0.51750973	0.49906191	0.53737374	0.53511148
Gradient Boosting	Baseline 1	Difference	0.53045923	0.52601703	0.52157598	0.53053435	0.53045091
Gradient Boosting	Baseline 2	Concatenate	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
Gradient Boosting	Baseline 2	Difference	0.6316776	0.70824053	0.89493433	0.58599509	0.6319241
MLP (4-layer, 128-128-128-128)	Disaggregated	Concatenate	0.8172446111	0.8212648946	0.8405253283	0.8028673835	0.8172664095
MLP (4-layer, 128-128-128-128)	Disaggregated	Difference	0.8209934396	0.8203198495	0.818011257	0.8226415094	0.8209906472
NN (4-layer, 128-128-128-128)	Aggregated	Concatenate	0.7722586692	0.7718309859	0.7711069418	0.772556391	0.7722575908
NN (4-layer, 128-128-128-128)	Aggregated	Difference	0.7628865979	0.7676767677	0.7842401501	0.7517985612	0.7629065919
NN (4-layer, 128-128-128-128)	Disagg+Agg	Concatenate	0.8097469541	0.8122109158	0.8236397749	0.8010948905	0.8097599623
NN (4-layer, 128-128-128-128)	Disagg+Agg	Difference	0.8106841612	0.8119180633	0.818011257	0.8059149723	0.8106910218
NN (4-layer, 128-128-128-128)	Baseline 1	Concatenate	0.5492033739	0.5762114537	0.6135084428	0.5431893688	0.5492635847
NN (4-layer, 128-128-128-128)	Baseline 1	Difference	0.5360824742	0.582278481	0.6472795497	0.5291411043	0.5361865913
NN (4-layer, 128-128-128-128)	Baseline 2	Concatenate	0.6194939082	0.6381461676	0.6716697936	0.6078098472	0.619542762
NN (4-layer, 128-128-128-128)	Baseline 2	Difference	0.6204311153	0.6361185984	0.6641651032	0.6103448276	0.6204720647

Appendix D

Formal Definitions of Cicero and Extended Vega-Lite

D.1 Cicero Formal Definition

Figure D.1 and Figure D.2 provide a formal definition of the Cicero specification syntax.

Notations	Examples	Description
" <code>:=</code> ": is defined as a tuple of	<code>a := b, c, d</code>	<code>a</code> is defined as a tuple of <code>b</code> , <code>c</code> , and <code>d</code> .
" <code>~</code> ": possible names for ... are ...	<code>a ~ b, c</code>	The possible names of <code>a</code> are <code>b</code> and <code>c</code> .
" <code> </code> ": alternate argument	<code>a b c</code>	Either one of <code>a</code> , <code>b</code> , or <code>c</code> .
" <code>...</code> ": extensible arguments	<code>a := b c ...</code>	<code>a</code> can be either <code>b</code> , <code>c</code> , or something else.
" <code><></code> ": datatype	<code>A<String></code>	<code>A</code> is a string type argument.
" <code>[]</code> ": a list of	<code><String>[]</code>	A list of string elements.
" <code>?</code> ": optional argument	<code><a, b>[]</code> <code>a?</code> <code>...?</code>	A list of tuples composed of <code>a</code> and <code>b</code> . <code>a</code> is optional. optional additional arguments
" <code>{}</code> ": key-value map (e.g., JavaScript Object)	<code>{ <A> : }</code>	An object with a type <code>A</code> key and type <code>B</code> value.
" <code>()</code> ": An omissible part of a string	<code>(a.)b</code>	" <code>a.</code> " can be omitted.

Notes
"`<Number>`" either a number or a string of a number with its unit (e.g., 350, "350px").

Formal Specification

`CiceroSpec` := Name?, Metadata?, Transformations

`Name` := <String>

`Metadata` := Condition?, MediaType?, AspectRatio?, ...?

`Condition` := xsmall | small | medium | large | xlarge | ...

`MediaType` := screen | paper | ...

`AspectRatio` := portrait | landscape | <Number> | ...

`Transformations` := <Rule>[]

`Rule` := Specifier, Action, Option?

`Specifier` := Role, Mark?, Index?, Id?, Data?, Field?, Values?, Datatype?,

Structure query

Data query

Channel?, Operation?, Interaction?, \$OtherAttributes?

Attribute query

\$OtherAttributes include encoding channels, role values, and other appearance-related properties (e.g., font styles, stroke styles, etc.).

`Role` := data | (data.)transform | view | (view.)row | (view.)column | (view.)facet | (view.)axis

data data transformation(s) chart view row (y) / column (x) encodings chart facets axes

| (view.)hAxis | (view.)vAxis | (view.)axis.grid | (view.)axis.domain | (view.)axis.tick

horizontal axes vertical axes axis grids axis domain(s) axis tick(s)

| (view.)axis.label | (view.)axis.title | (view.)hAxis.grid | (view.)hAxis.domain

axis labels axis title horizontal axis grids horizontal axis domain(s)

| (view.)hAxis.tick | (view.)hAxis.label | (view.)axis.title | (view.)vAxis.grid

horizontal axis ticks horizontal axis label(s) horizontal axis title vertical axis grids

| (view.)vAxis.domain | (view.)vAxis.tick | (view.)vAxis.label | (view.)vAxis.title

vertical axis domain(s) vertical axis ticks vertical axis label(s) vertical axis title

| (view.)layer | (view.)layer.transform | (view.layer.)mark | (view.layer.)mark.label

layer(s) layer-wise data transformation(s) mark(s) mark label(s)

| (view.layer.mark.)tooltip | (view.layer.)legend | (view.layer.)legend.title

tooltip(s) legend(s) legend title

| (view.layer.)legend.label | (view.layer.)legend.mark | (view.)title | (view.)annotation

legend label(s) legend mark(s) chart title(s) (non-data) annotations

| (view.)emphasis

(non-data) emphases

Figure D.1: The detailed formal specification of Cicero (part 1)

```

Mark := point | circle | rect | bar | line | ...
For information, read https://vega.github.io/vega-lite/docs/mark.html
Index := <Number> | first | last | even | odd
Id := <String>

Data := Datum | <Datum>[]
Datum := { <Field>: (<Any> | <Any>[] | <Op>[] ) }
Op := { <Operator>: <Any> }
Operator := not | and | or | == | > | >= | startsWith | ...
Field := <String>
Values := <Any>[]
Datatype := nominal | ordinal | quantitative | temporal ...

Channel := x | y | color | fill | stroke | opacity | fillOpacity | strokeOpacity | strokeDash | size
         | shape | arc | radius | theta | ...
Operation := OperationType | <OperationType>[]
OperationType := filter | aggregate | bin | ...
Interaction := InteractionType | <InteractionType>[]
InteractionType := zoom | context | ...

$OtherAttributes ~ position, x, y, dx, dy, color, fill, stroke, opacity, shape, size, width, height,
                 strokeWidth, strokeDash, label, title, fontColor, fontSize, bin, aggregate, scale, translate, ...
$OtherAttributes := <Any> | By | Prod
By := <Number> Addition to an existing value
Prod := <Number> Product with an existing value

Action := modify | reposition | transpose | add | duplicate | remove | replace | swap

Option := Specifier | To?, From?
To := Specifier
From := Specifier

```

Role semantics in Option

When an Option has a <Specifier> form, then Role is optional. If Role is not provided, then the Role is automatically that of the specifier. If provided, the Role of the Option should be subordinate to that of the specifier, and the other properties in the Option is considered as the elements specified by the Option's role keyword. If a Role keyword B can be directly concatenated after A with the dot operator, then B is subordinate to A (e.g., A.B). However, Role is required for To and/or From, and their Roles are considered independent of the Specifier (i.e., not subordinate to the Specifier).

Examples

```
{specifier: {
  role: layer,
  ... },
action: ...,
option: {
  role: mark,
  ... }}
```

The Option's role: layers' mark(s).

```
{specifier: {
  role: view,
  ... },
action: ...,
option: {
  role: mark,
  ... }}
```

The Option's role is **not valid** because view.mark is not possible.

```
{specifier: {
  role: legend,
  ... },
action: ...,
option: {
  to: { axis.label }
  ... }}
```

The Option's To's role: axis label(s), irrelevant to the legend(s) specified by the specifier.

Figure D.2: The detailed formal specification of Cicero (part 2)

D.2 An Extended Version of Vega-Lite

Currently, Vega-Lite [96] often makes it complicated or difficult to express common techniques used in many public-facing visualizations where responsive design is critical, limiting us to observe realistic use cases. We extended Vega-Lite [96] primarily to enhance the expressiveness for public-facing visualizations that include additional text elements and informational marks. For example, the latest version of Vega-Lite (5.3) is limited in expressing common strategies for communicative, narrative visualizations in the responsive context, including text wrapping, externalization of elements, complex labeling (e.g., mark labels with multiple data values and varying styles). Furthermore, it is complicated to declare mobile-specific strategies like label-mark serialization and callout lines for annotations to data points using Vega-Lite. Thus, our extended version of Vega-Lite (ExVL) makes it easier to express and render various techniques for communicative visualizations by providing simpler expressions. A ExVL specification is converted to Vega-Lite that handles the primary visualization rendering. ExVL additionally includes separate rendering modules for annotations, informational marks, and title elements that are not fully supported by Vega-Lite. The example cases in our gallery in supplementary material provide various ExVL specifications with descriptions.

An ExVL specification consists of `data`, `transform`, `layout`, `layer`, `nodata`, `interaction`, and `title` objects. A `data` object is an array of JSON-formatted data points, and a `transform` object conveys a set of global data transformations (similar to that in Vega-Lite). A `layout` object includes information about size (`width` and `height`), `composition` (single view, small multiples, and map), map projection details (e.g., `translates`, `scale`), axis designs, and `row` and `column` elements. `row` and `column`

elements are inspired by Tableau's shelves design [114]. A user can declare at most two `row` and `column` elements each. The first and second `row` elements are converted to `row` and `y` encodings in Vega-Lite, respectively. For small multiples, a user can define a list of filter statements or data values for small multiples using `split` keyword.

A `layer` object is composed of `mark`, `text`, `tooltip`, and `transform` objects. A `mark` object states the mark type of the layer and its visual properties (e.g., color, size, stroke, etc) which are then converted to encoding channels or static mark properties in Vega-Lite. A `text` object contains a list of text elements associated to the current layer. A `text` element can be on one of the mark, axis, and legend. Visibility options for a text element includes externalized (with or without numbering), serialized (to the mark), and callout lines (or ticks). By setting `width`, a `text` element can be wrapped. Furthermore, we added various expressions for complex labeling (e.g., axis title with summary statistics). The `tooltip` object contains a list of data fields to include in a tooltip and its positioning options (at its triggered position or fixed at the bottom of screen). A layer may have its local data transformations (`transform`).

A `nondata` object contains a list of annotation and emphasis elements that are not associated to data objects (marks, axes, and mark labels). A `nondata` element can be either `text` (annotation) or `mark` (emphasis). They are manually positioned, and an annotation element can be externalized (with or without numbering). An `interaction` object includes definitions of user interactions. The types of supported user interactions include zoom and pan for map, context view, and interactive filtering, while tooltip is separately defined in each layer item).

Notations	Examples	Description
“:=”: is defined as a tuple of	<code>a := b, c, d</code>	<code>a</code> is defined as a tuple of <code>b</code> , <code>c</code> , and <code>d</code> .
“=”: set as value of	<code>A=b</code>	An argument <code>A</code> set as <code>b</code> .
“~”: possible names for ... are ...	<code>a ~ b, c</code>	The possible names of <code>a</code> are <code>b</code> and <code>c</code> .
“ ”: alternate argument	<code>a b c</code>	Either one of <code>a</code> , <code>b</code> , or <code>c</code> .
“...”: extensible arguments	<code>a := b c ...</code>	<code>a</code> can be either <code>b</code> , <code>c</code> , or something else.
“<”: datatype	<code>A<String></code>	<code>A</code> is a string type argument.
“[]”: a list of	<code><String>[]</code>	A list of string elements.
“[0..n]”: a list of length 0 to n	<code><a, b>[]</code>	A list of tuples composed of <code>a</code> and <code>b</code> .
“?”: optional argument	<code>[0..5]</code>	A length-5 array.
	<code>a?</code>	<code>a</code> is optional.
	<code>...?</code>	optional additional arguments

Notes
“<Number>” either a number or a string of a number with its unit (e.g., 350, “350px”).

Formal Specification

ExVLSpec := Name?, Data, Layout, Layer, Transform?, Interaction?, Title?, NonData?

Name := <String> the name of a visualization

Data := <JSON> the dataset of a visualization

Layout := Width?, Height?, Composition, Row, Column, HAxis?, VAxis?, NColumns?, Projection?

Width := <Number> **Height** := <Number>

Composition := single | repeated | projection | ...

the type of a visualization layout; repeated refers to small multiples

Row := <RCItem>[0..2] **Column** := <RCItem>[0..2] row and column items (as used in a Trellis plot)

RCItem := Field<String> | FieldObject

HAxis := AxisItem **VAxis** := AxisItem the design of an X and Y axis, respectively

AxisItem* := Domain<Boolean>?, DomainColor<Color>?, DomainDash<Dash>?, DomainWidth<Number>?, DomainOpacity<Opacity>?, Grid<Boolean>?, GridColor<Color>?, GridDash<Dash>?, GridWidth<Number>?, GridOpacity<Opacity>?, Offset<Number>? <https://vega.github.io/vega-lite/docs/axis.html>

NColumns := <Integer> the number of columns in a “repeated” composition

Projection* := ProjectionType, ProjectionScale?, ProjectionTranslate?, ...? the details of a map projection

<https://vega.github.io/vega-lite/docs/projection.html>

FieldObject* := Field<String>, DataType?, Scale?, Sort?, Aggregate?, Bin? | Aggregate=count, Scale? details about a data field encoded to a row/column/channel <https://vega.github.io/vega-lite/docs/encoding.html#field-def>

DataType := nominal | ordinal | quantitative | temporal

Scale* := Domain?, Range?, Scheme?, Reverse<Boolean>?, ...? <https://vega.github.io/vega-lite/docs/scale.html>

Domain := <Any>[] **Range** := <Any>[] **Scheme** := <String> (e.g., “magma”)

Sort* := ascending | descending | SortBy <https://vega.github.io/vega-lite/docs/sort.html>

SortBy := SortOrder?, SortField<String>? sort by a certain field

SortOrder := ascending | descending | <Any>[] an ascending, descending or custom order

Aggregate* := count | mean | max | median | ... <https://vega.github.io/vega-lite/docs/aggregate.html>

Bin* := Maxbins?, BinSteps?, Nice?, ...? <https://vega.github.io/vega-lite/docs/bin.html>

Layer := <LayerItem>[]

LayerItem := Mark, Text?, Tooltip?, Transform?

Mark := MarkType, \$MarkProperty?

MarkType* := circle | point | bar | rect | ... <https://vega.github.io/vega-lite/docs/mark.html#types>

\$MarkProperty ~ color, shape, size, stroke, ... mark properties or such channels

\$MarkProperty := Value | FieldObject if “Value” is used, then it is a static property. Otherwise, it is an encoding channel.

Figure D.3: The formal specification of Extended Vega-Lite (part 1)

Text := TextType, TextField, Values?, Anchor?, Orient?, TextItems?, Tick?, TextVisibility? ...?
TextType := on-mark | on-axis | legend mark labels, axis labels, and legends, respectively
TextField := FieldName<String> the reference field of the text item (i.e., text element for each value of the “TextField”)
Values := <Any>[] a subset of elements of the TextField to show the labels
Anchor, Orient := start | end | middle | right-start | right-end | ...
Anchor: the reference position to the corresponding mark/axis element; Orient: the reference position to the text element itself
TextItems := <TextItem>[] each line of the text element
TextItems := Format?, FontColor<Color>?, Width?, ...?
Tick := <Boolean> | TickColor<Color>?, TickWidth?, ...?
the design of the line segment between the text element and referred visual element.
TextVisibility := External<Boolean>?, Numbering<Boolean>?, Position<top|bottom|left|right>?
internalization/externalization of mark-labels; If Numbering = true, then reference numbers are shown on corresponding marks.

Tooltip := TooltipVisibility, TooltipFields
TooltipVisibility := on-mark | fixed | hidden the position of a tooltip
TooltipFields := <FieldName, Format>[] the information shown in a tooltip

Transform* := <TransformItem>[] global/layer-specific data transformation
TransformItem* := Filter* | Aggregate*, As* | Bin*, As* | Compute*, As*, Op* | ...
<https://vega.github.io/vega-lite/docs/transform.html>

Interaction := <InteractionItem>[] global interactions
InteractionItem := ZoomPan | Context | Filter zoom+pan, brush-based context view, and interactive filter, respectively

Title := Width<Number>?, Align?, TitleItems
TitleItems := <Name<String>?, Text<String>, Align?, FontSize<Number>?, FontWeight?, ...?>[]
the design and content of each title element

NonData := NonDataVisibility?, NonDataGlobalStyle?, NonDataItems
annotations and emphases unbound to data
NonDataVisibility := External<Boolean>?, Numbering<Boolean>?, Position<top|bottom|left|right>?
NonDataGlobalStyle := FontFamily<String>?, FontWeight?, BoxStroke<Color>?,
BoxStrokeWidth<Number>?, LineHeight<Number>, ...? global style of non-data items

NonDataItems := <NonDataType, Name<String>?, X<Number>, Y<Number>, DX<Number>, DY<Number>, Width<Number>, Height<Number>, Rotate<Number>, NonDataText, NonDataBox, NonDataMark>[]
the appearance and contents of nondata items / X, Y: absolute position, DX, DY: relative position
NonDataType := text annotations | mark emphases
NonDataText := <Text<String>, Align?, Overflow?, FontSize<Number>?, FontWeight?, LineHeight<Number>?, FontColor<Color>?, Opacity<Number>?>[]
NonDataBox := Padding<Number>?, Fill<Color>?, Stroke<Color>?, StrokeWidth<Number>?, Radius<Number>? StrokeStyle?
StrokeStyle := solid | dashed | dotted | ...
NonDataMark := NonDataMarkDef, Fill<Color>?, Opacity<Number>?, Stroke<Color>?, StrokeOpacity<Number>?, StrokeWidth<Number>?, Radius<Number>?
NonDataMarkDef := Icon<String> Bootstrap icon names | Image<URI> | Shape
Shape := circle | rect | rule | ...

Detailed documents for the following common value types (available as of Feb. 2022).

Number	https://developer.mozilla.org/en-US/docs/Web/CSS/length
Align	https://developer.mozilla.org/en-US/docs/Web/CSS/text-align
Color	https://developer.mozilla.org/en-US/docs/Web/CSS/color
FontWeight	https://developer.mozilla.org/en-US/docs/Web/CSS/font-weight
Dash	https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray
Opacity	https://developer.mozilla.org/en-US/docs/Web/CSS/opacity
LineHeight	https://developer.mozilla.org/en-US/docs/Web/CSS/line-height
Overflow	https://developer.mozilla.org/en-US/docs/Web/CSS/overflow
Icon	https://getbootstrap.com/docs/5.1/content/typography/

Figure D.4: The formal specification of Extended Vega-Lite (part 2)

Appendix E

Technical Details for Cicero

This appendix outlines further technical details about our Cicero compiler to this extended version of Vega-Lite. Then, this document describes our prototype recommender for responsive visualization transformations.

E.1 Cicero Compiler API for Extended Vega-Lite

Our Cicero compiler API provides an architecture for handling, compiling, and rendering the source view and Cicero specifications with a `Cicero` class and `loadCicero` function. Developers for a rendering grammar can attach a Cicero compiler and renderer (i.e., the compiler of the rendering grammar that actually draws the visualization) specific to that grammar. A `Cicero` class instance contains a source specification in any declarative rendering grammar, a Cicero specification, a transformation compiler that compiles the source and Cicero specifications to a target specification, and the compiler function (or equivalent) for the rendering grammar. Our API also provide the `loadCicero` function as a wrapper for creating a `Cicero` class instance, compiling the specifications, and rendering the transformed view. While we developed a Cicero compiler for an extended version of Vega-Lite, other render-

ing grammars like the default Vega-Lite [96] or ggplot2 [26] can be similarly developed. To fit our Compiler API, a Cicero compiler for a rendering grammar should take source view and Cicero specifications and return the transformed specification (not the rendered view), and a renderer should return a rendered view given a visualization specification and a DOM element (or selector) to draw the visualization in.

The `Cicero` class is instanced with `name` (the name of a transformed view or Cicero specification), `source` (the source view specification), `metadata` (a Cicero `metadata` object), and `description` (detailed description for the Cicero specification) as shown in line 3–4 of Figure E.1. After instancing, users can add a list of Cicero `rule` objects using `addTransformations` method (line 7–8). Users can set a Cicero compiler and renderer for the rendering grammar using a `setCompiler` method (`CiceroToExVL` and `renderExVL` in line 10, respectively). To get the transformed specification in the rendering grammar, users can call a `getTransformed` method (line 12). To render the transformed view, users can use a `getRendered` method with a CSS selector for the HTML element (or DOM) to insert the rendered visualization in (line 14).

The `loadCicero` function makes the above job more pipelined. The `loadCicero` function takes a `ciceroSpec`, `source`, a Cicero compiler and renderer for the rendering grammar (line 18–19). Then, it returns a Promise object ¹, and the `then` method of the returning Promise takes a callback function with a `Cicero` class instance as an argument (line 20–22).

¹A type of JavaScript object that better enables asynchronous operation of functions. See https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise for details.

```

1 // create a config object
2 let metadata = { condition: "small" };
3 // create a Cicero instance
4 let cicero = new Cicero("target-view", source,
5   metadata, "This is a Cicero Spec for X");
6 // add Cicero transformation rules
7 let rules = [ ... Cicero transformations ];
8 cicero.addTransformations(rules);
9 // set compilers for Cicero and rendering (ExVL)
10 cicero.setCompiler(CiceroToExVL, renderExVL);
11 // get the transformed specification
12 let transformed = cicero.getTransformed();
13 // render the transformed view
14 cicero.getRendered("#dom-element");
15
16 // Using loadCicero function
17 let ciceroSpec = { ... a Cicero specification };
18 loadCicero(ciceroSpec, source, CiceroToExVL,
19   renderExVL) // returns a Promise
20   .then(cicero => {
21     cicero.getRendered("#dom-element");
22   });

```

Figure E.1: A Cicero Compiler API use cases. Line 1–14: instancing a `Cicero` class object. Line 16–22: pipelining the job using the `loadCicero` function.

E.2 Recommender Prototype for Responsive

Visualization

Below, we describe the pipeline of our prototype recommender and the strategies that we encoded.

E.2.1 pipeline

Input

Our prototype recommender (Figure E.2A) takes as inputs the ExVL specification of a source view and user preferences for responsive designs. User preferences for our recommender include (1) the intended size of a responsive view, (2) hard constraints for transformation strategies, and (3) a subset of data that can be omitted or added. Users can specify hard constraints including whether to allow for transposing axes, changing encoding channels, modifying mark types, and altering the

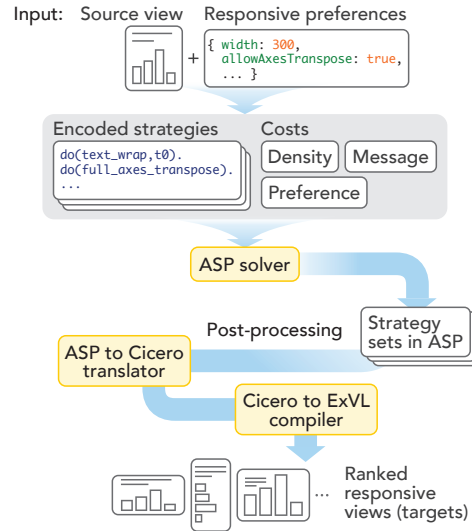


Figure E.2: The pipeline of our prototype recommender.

aspect ratio. For example, if ‘allow for modifying mark type’ is set as false, then recommendations with changes to mark type are ignored. One can specify a subset of data to omit or add using a filter statement.

Search space generation

Given the source view, users’ preferences, and transformations strategies, our recommender generates sets of responsive strategies (strategy sets). The source view design and preferences are converted to ASP expressions. We encoded a set of transformation strategies in ASP (see the next section). A transformation strategy can be applied or not applied when its condition is satisfied. For example, when the intended chart size is smaller than that of the source view, then our recommender may or may not apply a text-wrapping strategy to text element. We use Clingo [205, 206], a Python library, as our solver over these rules and constraints, similar to past uses of ASP for visualization recommenders, namely Draco [148].

Cost evaluation

For each strategy set, our recommender assigns three types of costs (*density*, *message*, and *preference*) to the application or omission of each transformation strategy. Then, these costs are normalized to be in the same scale, and then aggregated as a final cost. First, for changes from bigger screen to smaller screen, if a strategy reduces the number of elements or spreads them on screen, it has a density cost of 1, otherwise 0. For instance, a strategy of removing every other axis labels for mobile views has a density cost of 1 while not applying that strategy has a density cost of 0. Similarly, for changes in the opposite direction, if a strategy increases the number of elements or gathers them on screen, it has a density cost of 1, otherwise 0. In this case, adding an axis label between each pair of existing labels has a density cost of 1.

Second, Kim et al. [37] propose five forms of changes to implied messages in visualization under responsive transformations: omitting or adding information and interaction, changes to the amount of concurrent information within a single scroll height, the discoverability of information (i.e., whether it is toggled), and changes to graphical perception (e.g., aspect ratio changes). We assign message cost of 1 for each transformation that cause such changes to visualization messages. For example, transposing an overly wide view to a longer visualization has a message cost of 1 because it reduces the amount of information that are concurrently visible within a single scroll height.

Third, to reflect preferences in applying responsive transformation, our prototype includes a preference cost according to their popularity or frequency in use cases [37, 99]. Given a strategy, if it is commonly applied (more than 50% of the

cases), applying it has preference cost of 2, otherwise 0. If it is less commonly applied (more than 10% of the cases), then it has preference cost of 1, otherwise 1, which makes it more random. If it is rarely used (less than 10% of the cases), then it has preference cost of 0, otherwise 2. For instance, disproportionate rescaling is highly common, so it has preference cost of 2, while it has message cost of 1 (as it causes changes to graphical perception). Users may change this preferences cost based on their own preferences like style guidelines of an organization. To prevent too many transformations, we assign a preference cost of $20 - \text{count}(\text{strategies})$.

Post-processing

The ASP to Cicero translator converts each of the stable strategy sets generated by ASP solver into a Cicero specification. Then, our Cicero compiler for ExV produces ExVL specifications for target views.

E.2.2 Encoded strategies

We encode a diverse set of automatable strategies that we observed in our case studies with 13 realistic use cases and prior design pattern analyses [37, 99]. Strategies denoted by M are for desktop-to-mobile transformations, and those denoted by D are for mobile-to-desktop transformations. The non-prefixed strategies can be applied to both directions of transformation.

- **Changes to layout**
 - Transposing axes
 - Partial axes transpose (see the Aid Budget case in the paper)
 - Resizing the chart (proportionately or disproportionately)
- **Changes to data**

- M-Omitting a specified subset of data
- D-Filtering in a specified subset of data
- **Changes to mark properties and encoding channels**
 - Rescaling the size channel.
 - M-Removing detail encodings like image, color, and size
 - M-Changing the mark type (from bar, line, scatterplot to heatmap)
 - M-Changing small multiples to a heatmap
- **Changes to text elements**
 - M-Externalizing non-data/data annotations
 - M-Numbering externalized data annotations
 - D-Internalizing non-data/data annotations
 - M-Wrapping text elements
- **Changes to references**
 - Repositioning legends
 - M-Serializing axis labels
 - D-Parallelizing axis labels
 - M-Converting axis labels to legends
 - M-Removing every other axis labels
 - D-Adding every other axis labels
 - Adding ticks for mark labels
- **Changes to interaction**
 - M-Fixing the tooltip position
 - D-Unfixing the tooltip position
 - M-Removing tooltip
 - D-Adding tooltip
 - Removing a context view

- Adding a context view (for time-serial visualizations)
- Removing zoom
- Adding zoom (for map visualizations)

Appendix F

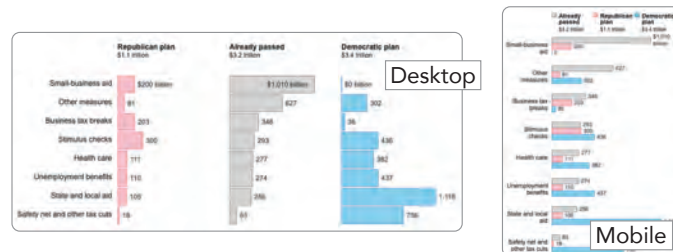
Additional Walkthrough Cases for Cicero

F.1 Aid Budget

The Aid Budget example¹ summarizes the COVID-19 aid budget plans for each business sector suggested by the Democratic (blue) and Republican (pink) U.S. parties compared to the budgets already passed by the U.S. Congress (lightgray). The desktop version in Figure F.1 has three columns for the Republican, already passed, and Democratic **plans** in this order, and rows for eight business **sectors**, composing a tabulated format. Each bar encodes the budget **amount** of the corresponding **plan** and **sector** categories. To address the significantly reduced screen width in mobile screens, transformations for mobile include (1) changing the chart layout to a single-column grouped bar chart with two row fields (**sector** → **plan**), (2) converting the axis for **plan** to a legend for the color channel, (3) reordering the **plan** row, and (4) miscellaneous changes to text elements. The Cicero spec is shown in Figure F.1.

To fit the mobile screen, the resizing rule in line 2–4 modifies the size to 375 ×

¹<https://www.nytimes.com/interactive/2020/07/30/upshot/coronavirus-stimulus-bill.html>



Cicero transformations

```

1 ...
2 { specifier: { role: "view" },
3   action: "modify",
4   option: { size: [375, 350] } },
5 { specifier: { role: "view" },
6   action: "replace",
7   option: {
8     from: { role: "column", index: 0 },
9     to: { role: "row", index: 1 } },
10 { specifier: { role: "row", field: "plan" },
11   action: "modify",
12   option: {
13     sortBy: [
14       "Already passed",
15       "Republican plan",
16       "Democratic plan" ] } },
17 { specifier: { role: "axis", field: "plan" },
18   action: "replace",
19   option: {
20     to: {
21       role: "legend",
22       channel: "color",
23       position: "top",
24       direction: "horizontal",
25       offset: 40, width: 80 } },
26 { specifier: { role: "axis.label", field: "sector" },
27   action: "modify",
28   option: { width: 100 } },
29 { specifier: { role: "mark.label",
30   data: { amount: [200, 0] } },
31   action: "modify",
32   option: { format: "d" } },
33 { specifier: { role: "mark.label",
34   data: { amount: [1010] } },
35   action: "modify",
36   option: { anchor: "start", dx: 5, width: 60 }

```

Transformed

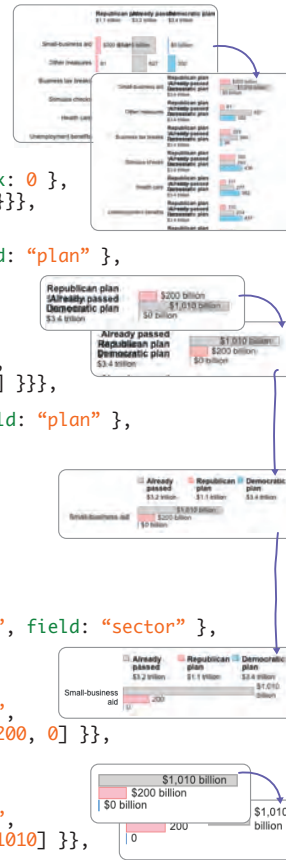


Figure F.1: A walk-through example case of Aid Budget (Section F.1).

350. This transformation results in an overly narrow resolution for the horizontal bars. To address the narrowed column width, one can partially transpose the column field (`plan`) to the row, as indicated in line 5–9. In the `option`, the `from` keyword (line 8) specifies the `column` at index of `0` (the second column is the budget value field). The `to` keyword (line 9) indicates the `row` element at index of `1`, which does not exist before the transformation (i.e., after the first row element). Using this `index` keyword instead of the `field` keyword enhances the reusability of this partial transpose rule by making it agnostic to the underlying data set. This rule is equivalent to using a specifier of `{role:"column", "index": 0}` instead of having the `from` keyword.

By changing the layout as above, one might want to change the order of bars to emphasize the “already passed” data as a reference point for both plans (previously shown at the center of the desktop view). In the mobile view, one can emphasize those reference points by placing them above the other bars. To enable such a layout transformation, the reordering rule in line 10–14 first specifies the the `row` of the `plan` field (line 10). This specifier reflects the previous change from the `row` to the `column` (P7). Then the `sort` keyword and its subproperty `sortBy` in the `option` indicate the new order of the field (line 13–14).

The previous changes cause the vertical axis labels for the `plan` field to overlap with each other and repeat for each `sector` (P2). At the same time, another encoding channel (`color`) also encodes the same field. Using this redundant encoding, one can change the axis labels to a color legend: the replacement rule in line 15–23 converts the `axis` for the `plan` field (the `specifier` in line 15) to the `legend` for the `color` channel (the `option` in line 19–20). The additional keywords (`position`, `direction`, `offset`, and `width`) in line 21–23 set additional layout properties.

One can further increase the space for the bars by reducing the space for the remaining vertical axis labels for the `sector` field. One can reduce the space for axis labels by truncating or wrapping. To maintain the same amount of information, the text wrapping rule in line 24–26 changes the width of axis labels as `100px`. In the desktop version, the mark labels for the first rows have units (`$00 billion`). However, the partial transpose earlier (line 5–9) makes the unit repeat unnecessarily. To address the repeating labels, the modification rule in line 27–30 selects `mark.labels` for the `amount` value of `[200,0]` using the `data` keyword and changes the `format` as `",d"` (the format expression of `d3.js` [270]). Lastly, the mark label for the `amount` value of `1,010` (“\$1,010 billion”) is positioned inside the mark, the size of which is reduced after the earlier resizing and re-layout transformations. To reposition the mark label with text wrapping, the modification rule in line 31–34 queries the label similarly to the previous rule, and the `option` defines the layout properties (`anchor`, `dx`, and `width`).

F.2 Disaster Cost

The Disaster Cost chart² depicts the losses associated with major natural disasters in the U.S. In Figure F.2, the x position encodes the year, while bar height represents the loss in USD. Bars for five disasters are red and have labels (their names) in the mobile version. There is a title element (two lines) above the visualization. In addition to resizing, changes for the desktop view involve (1) internalizing the title, (2) using a longer form of disaster names (‘Hurricane’ is added) with the loss amount in USD, (3) adding y axis labels for 50, 150, and 250 billion, (4) lengthening the x axis labels

²<https://nyti.ms/38VBIzz>

(e.g., '80 to 1980). The Cicero spec is shown in Figure F.2.

The resizing transformation in line 2–4 expresses `modify`ing the `size` of the `view` (reusing the previous rule but changing the size value). After resizing, there is a large empty space on the left side of the chart. One way to utilize that space is to internalize the existing title element into the chart, which can be done using the rule in line 5–10. In this case, one can use the `replace` action instead of `reposition` to express converting the title element to an internal non-data annotation. To indicate the new role, the `option` has the `to` keyword (line 8–9). The `internal` keyword in line 9 expresses that the converted annotation should appear inside of the visualization. However, this rule provides no designated position for the internalized annotation, so our Cicero compiler relies on the default behavior of the rendering grammar (our extended Vega-Lite) for choosing the center of the largest empty area (P6). The `separate` keyword set to `false` in line 10 means that the two title lines move together as a single annotation to prevent the automated positioning from separating them. This transformation can be done using the `reposition` action and `internal` keyword in the `option` as well. One may specify the absolute position of the internalized annotation using the `x` and `y` keywords.

The increased space can also add more axis labels and make labels longer to enhance the clarity of reference elements. The addition rule in line 11–14 adds the values of 50, 150, and 250 (in the `option` at line 13–14) to the vertical axis (`vAxis`), resulting in new axis labels and grid lines for the corresponding values (P3). The modification rule in line 6–10 lengthens the text format (`expression`) of the axis labels for the `year` field (the `specifier` in line 15–17). The `null` value for the `expression` keyword in line 19 means ‘no particular format,’ showing the original four digits for

the years.

In addition to the larger screen space, desktop screens allow for utilizing the horizontal offsets of visualizations (e.g., the visualization margins). Therefore, the modification rule in line 20–28 changes the position and text format of mark labels. In the `option`, line 23–25 modify the mark labels to be 10 pixels away from the corresponding marks horizontally (`dx`) using the `top-right` position of each mark and the `start` of each label as reference points (`orient` and `anchor`, respectively). Line 26 sets the text `alignment` as `left` as the labels are moved to the right of the marks. As it becomes easier to distinguish the relationship of each mark-label pair due to proximity and the added spacing between labels, line 27 indicates that no label ticks are needed for the desktop view, expressed as a shortcut for `{specifier: "mark.label.tick", action: "remove"}`. To lengthen the mark labels, line 28 changes their `expression` using a JavaScript-based expression meaning “if the label value is not ‘California wildfires’ then add ‘Hurricane’ before it with a space.”

In addition to lengthening the mark labels, one can add more information like the loss cost of each of the specified disasters to the existing mark labels, as expressed in line 29–37. In line 32, the `items` keyword in `option` indicates subelements of the element selected by the `specifier` (i.e., text lines of each mark label). The `add` action in line 30 adds the specified `items` as part of the mark labels. Line 33–34 sets text style properties; other properties like text alignment does not need to be specified as the compiler looks for those properties from other similar elements (i.e., the existing labels; **P3**). Lastly, line 35 formats the new text element as “\$00 billion” using the `expression` keyword.



Figure F.2: A walk-through example case of Disaster Cost (Section F.2).

Appendix G

Applying Cicero to MobileVisFixer

Refer to Figure G.1

Cicero expressions for the MobileVisFixer rules

```

1 // For A0--7, we assume specific
  chart size intended for a view
2 // A0&2: decrease X axis scale range
  = chart width
3 { specifier: { role : "view" },
4   action: "modify",
5   option: { width: 375 }}
6 // A1&3: increase X axis scale range
7 { specifier: { role : "view" },
8   action: "modify",
9   option: { width: 1024 }}
10 // A4&6: decrease Y axis scale range
  = chart height
11 { specifier: { role : "view" },
12   action: "modify",
13   option: { height: 200 }}
14 // A5&7: increase Y axis scale range
15 { specifier: { role : "view" },
16   action: "modify",
17   option: { height: 600 }}
18 // A9: decrease font size
19 { specifier: { role : "text" },
20   action: "modify",
21   option: { fontSize: { prod: 0.8 }}}
22 // A10: increase # of axis lables/ticks
23 { specifier: { role : "axis" },
24   action: "add",
25   option: { values : "odd" }}
26 // A11: decrease # of axis lables/ticks
27 { specifier: { role : "axis" },
28   action: "remove",
29   option: { values : "even" }}
30 // A12: skip, unrecognizable
31 // A13: break text line = wrapping text
32 { specifier: { role : "text" },
33   action: "modify",
34   option: {
35     width: { prod: 0.8 },
36     wrap: true }}
37 // A14: decrease the X offset min
  = left margin
38 { specifier: { ... },
39   action: "modify",
40   option: { marginLeft: { prod: 0.9 }}}
41 // A15: increase the X offset min
42 { specifier: { ... },
43   action: "modify",
44   option: { marginLeft: { prod: 1.1 }}}
45 // A16: decrease the X offset max
  = right margin
46 { specifier: { ... },
47   action: "modify",
48   option: { marginRight: { prod: 0.9 }}}
49 // A17: increase the X offset max
50 { specifier: { ... },
51   action: "modify",
52   option: { marginRight: { prod: 1.1 }}}
53 // A18: decrease the Y offset min
  = top margin
54 { specifier: { ... },
55   action: "modify",
56   option: { marginTop: { prod: 0.9 }}}
57 // A19: increase the Y offset min
58 { specifier: { ... },
59   action: "modify",
60   option: { marginTop: { prod: 1.1 }}}
61 // A20: decrease the Y offset max
  = bottom margin
62 { specifier: { ... },
63   action: "modify",
64   option: { marginBottom: { prod: 0.9 }}}
65 // A21: increase the Y offset max
66 { specifier: { ... },
67   action: "modify",
68   option: { marginBottom: { prod: 1.1 }}}

```

The list of rules supported by MobileVisFixer

```

A0: decXAxisRangemin
A1: incXAxisRangemin
A2: decXAxisRangemax
A3: incXAxisRangemax
A4: decYAxisRangemin
A5: incYAxisRangemin
A6: decYAxisRangemax
A7: incYAxisRangemax
A8: incFontSize
A9: decFontSize
A10: incTickNumber
A11: decTickNumber
• A12: callVegaLabel
A13: breakTextLine
A14: decXOffsetmin
A15: incXOffsetmin
A16: decXOffsetmax
A17: incXOffsetmax
A18: decYOffsetmin
A19: incYOffsetmin
A20: decYOffsetmax
A21: incYOffsetmax
• A22: changeOffset

```

- When the meaning of the rule cannot be determined because they are not defined in the paper.
- For A14–21, we assume that the specifier of each rule (i.e., the element to transform the offset) can be parameterized within the recommender's pipeline.

Figure G.1: Re-expressing actions in MobileVisFixer [93] using Cicero

Appendix H

Technical Details for Dupo

H.1 Rationales for Exploration Suggestions

H.1.1 Conversion 1: Desktop to Phone

H.1.1.1 Case 1: Default

Rationale 1: Minimize changes

- Rescale (chart size, mark size, and text size) to the intended size.

Rationale 2: Avoid overplotting

- Aggregate if there is a hierarchy that is stated or natural.

H.1.1.2 Case 2: Non-geospatial, position-based charts

Mark types: bar, line, point, etc.

Rationale 3: Fit to a different aspect ratio

- Transpose a chart.
- In case of a 1D dot plot, convert to a bar chart.

Rationale 4: Avoid overplotting

- Convert to a heatmap if there is a nominal encoding.
- Separate into small multiples if there is a nominal encoding, with a default column number of 3.

H.1.1.3 Case 3: Geo-spatial charts

Mark types: map

Rationale 5: Avoid overplotting

- Convert to a bar chart.
- Convert to small multiples while keeping the mark type if it is not a choropleth.

H.1.1.4 Case 4: Non-position-based charts

Mark types: pie

Rationale 6: Minimize changes

No strategies other than minimal changes.

H.1.1.5 Case 5: Small multiples**Rationale 7: Adjust graphical density**

- Reduce the number of columns: 1, 2, and 3.

H.1.1.6 Case 6: Multi-layered charts (with different mark types)**Rationale 8: Adjust graphical density**

- Remove a less important (e.g., decorative) redundant layer (e.g., line vs. area to line).

H.1.1.7 Case 7: Heavily annotated charts (4+ annotations)

Rationale 9: Adjust graphical density

- Externalize annotations with numbering.

H.1.2 Conversion 2: Desktop to Tablet/Print

H.1.2.1 Case 1: Default

Rationale 1: Minimize changes

- Rescale (chart size, mark size, and text size) to the intended size.

H.1.2.2 Case 2: For print

Rationale 2: Fit to the medium

- Remove interactions.

H.1.3 Conversion 3: Desktop to Thumbnail/Watch

H.1.3.1 Case 1: Default

Rationale 1: Adjust graphical density and this is only for skimming

- Remove references.
- Remove secondary titles.
- Remove annotations.

Rationale 2: Minimize changes

- Rescale (chart size, mark size, and text size) to the intended size.

Rationale 3: Avoid overplotting

- Aggregate if there is a hierarchy that is stated or natural.

H.1.3.2 Case 2: Non-geospatial, position-based charts

Mark types: bar, line, point, etc.

Rationale 4: Avoid overplotting

- Convert to a heatmap if there is a nominal encoding.
- Leave only arg-min and arg-max given a temporal x axis.

H.1.3.3 Case 3: Geo-spatial charts

Mark types: map

Rationale 5: Avoid overplotting

- Convert to a bar chart.

H.1.3.4 Case 4: Non-position-based charts

Mark types: pie

Rationale 6: Minimize changes

No strategies other than minimal changes.

H.1.3.5 Case 5: Small multiples**Rationale 7: Adjust graphical density**

- Reduce the number of columns to 1.

H.1.3.6 Case 6: Multi-layered charts (with different mark types)

Rationale 7: Adjust graphical density

- Remove a less important (e.g., decorative) redundant layer (e.g., line vs. area to line).

H.1.4 Conversion 4: Phone to Desktop

Basically, the reverse operations of the conversion methods from Desktop to Phone.

H.1.4.1 Case 1: Default

Rationale 1: Minimize changes

- Rescale (chart size, mark size, and text size) to the intended size.

Rationale 2: Adjust graphical density

- Disaggregate if there is a hierarchy that is stated or natural.

H.1.4.2 Case 2: Non-geospatial, position-based charts

Mark types: bar, line, point, etc.

Rationale 3: Fit to a different aspect ratio

- Transpose a chart.

Rationale 4: Choose a more effective encoding channel given a more space available

- Convert a heatmap to a bar chart.
- Convert a heatmap to a line chart.

- Convert to a map if there is longitude and latitude fields.

H.1.4.3 Case 3: Geo-spatial charts

Mark types: map

Rationale 5: Minimize changes

No strategies other than minimal changes.

H.1.4.4 Case 4: Non-position-based charts

Mark types: pie

Rationale 6: Minimize changes

No strategies other than minimal changes.

H.1.4.5 Case 5: Small multiples

Rationale 6: Adjust graphical density

- Increase the number of columns: 4, 5, and 6.
- Convert to a map if there are longitude and latitude fields.

H.1.4.6 Case 6: Single-layered charts

Rationale 7: Adjust graphical density

- Add a decorative layer if possible (e.g., points on line, area below line).

H.2 Rationales for Quick Edits

H.2.1 Condition 1: When adding a new tooltip to a Phone artboard.

- Quick edit 1: Fix tooltip at the bottom of the device screen.

H.2.2 Condition 2: When externalizing an annotation to the outside of the chart.

H.2.2.1 Condition 2a: if not numbered.

- Quick Edit 2a: Number the annotation.

H.2.2.2 Condition 2b: if not left-aligned.

- Quick Edit 2b: Left-align the annotation.

H.2.3 Condition 3: When repositioning an internal annotation that is associated with a data mark.

H.2.3.1 Condition 3a: If the repositioned distance between the annotation and the corresponding data mark is increased and if the annotation and data mark are not connected to the data mark by a tick line.

- Quick Edit 3a: Add a tick line between them.

H.2.3.2 Condition 3b: If the repositioned distance between the annotation and the corresponding data mark is decreased and if the annotation and data mark are connected to the data mark by a tick line.

- Quick Edit 3b: Remove the tick line.

H.2.4 Condition 4: When removing a title from a Desktop/Tablet artboard (in case that the user is trying to change the title to an internal annotation).

- Quick Edit 4: Convert the removed title to an internal annotation.

H.2.5 Condition 5: When changing the chart size.

- Quick Edit 5: Rescale the mark channel (static or encoding) proportionately.

H.2.5.1 Condition 5a: If it is a static map visualization.

- Quick Edit 5a: Add a zoom + pan interaction.

H.3 Loss Measures for the Recommender

Responsive visualization can be characterized as a trade-off between preserving takeaways and adjusting density (Section 3.2): Applying a set of responsive transformations to adjust graphical density (e.g., resizing, rescaling, aggregating, etc.) may cause changes to messages implied in the visualization. Thus, Dupo evaluates Exploration and Alteration suggestions in comparison with their source design in terms of “message” and density. To outline, we use the task-oriented loss mea-

asures for identification, comparison, and trend proposed in Chapter 4 to approximate changes to “message.” In addition to them, we develop a text loss to approximate changes to explicit takeaways conveyed in text elements (title and annotation). Lastly, we use overplotting and occupied ratios to approximate changes to graphical density, inspired by Ellis and Dix [241]. An occupied ratio is how much space visual elements take in a visualization (i.e., the ratio of non-white space). We introduce the technical details of these measures for background. Our goal is to integrate those measures with the authoring interface, and the evaluation of each measure (except the text loss) can be found in the relevant prior work.

H.3.1 Text Loss

In addition to the task-oriented insight loss measures in Chapter 4, the **text loss** estimates the changes to text contents in the visualization, such as annotations and titles, given their importance in narrative visualizations. Given a source design S and a target design T , the text loss from S to T , $L(\text{text}; S \rightarrow T)$, is formalized as:

$$L(\text{text}; S \rightarrow T) = \sum_{t \in S, t' \in T} I(t, t'), \quad (\text{H.1})$$

where t and t' are corresponding text elements in the source and target designs, respectively. $I(t, t') = 0$ if they are same, $I(t, t') = 0.3$ if they are different, and $I(t, t') = 1$ if either one of them does not exist. We heuristically chose the value of 0.3 for inequality given that the pair of text elements are supposed to contain similar information.

H.3.2 Density Loss Measures

To approximate changes to graphical density, we use overplotted and occupied ratios as major sources of changes to graphical density under responsive transformation. The overplotted ratio of a visualization refers to the overplotted area (i.e., occupied by two or more elements) divided by the occupied area (i.e., containing one or more elements) [241]. An occupied ratio is defined as the occupied area (i.e., non-white space) divided by the entire visualization area. Dupo only considers the chart area for changes to overplotted and occupied ratios because the ratios for out-chart text (e.g., title) are constant.

However, precisely measuring overplotted and occupied ratios lacks scalability, taking $O(n^2p^2)$ time (n : the number of elements, p : the size of the biggest element). Instead, we approximate a density grid, inspired by related work [241, 271]. In particular, Dupo divides a chart space by four-by-four (pixel) grid cells, as recommended by prior work [241]. Given a visualization V , we check the membership of the graphical and text elements in each grid cell, where $G_k(V)$ denotes the number of grid cells with k or more elements in V (G_0 : entire cells, G_1 : occupied cells, G_2 : overplotted cells). Figure H.1 shows an example density grid.

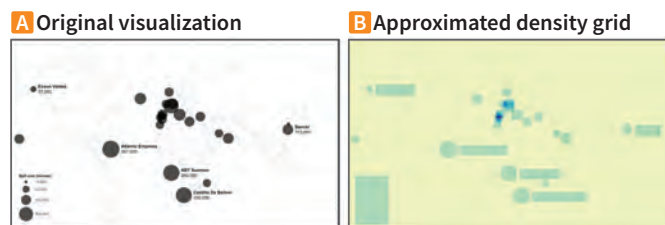


Figure H.1: An example density grid (B) approximated for a map with circle marks (A). The color in the density grid indicates the number of occupying visualization elements in each grid cell. Geographic shapes are excluded when it has no encoding.

Using this approximation, the **occupied ratio** (C) and **overplotted ratio** (P), given a visualization V , are formalized as:

$$\begin{aligned} C(V) &= G_1(V)/G_0(V) \\ P(V) &= G_2(V)/G_1(V) \end{aligned} \tag{H.2}$$

Then, we formalize the changes to graphical density $L(\text{Density}, S \rightarrow T)$, from S to T , with the weight terms (w_C, w_P) as:

$$L(\text{Density}, S \rightarrow T) = w_C|C(S) - C(T)| + w_P|P(S) - P(T)|, \tag{H.3}$$

H.3.3 Combining Loss Measures

To combine these message and density loss measures, Dupo first normalizes them using their maximum value (\hat{L}) as each loss measures are in different scales but they evaluate each target design in comparison with the same source design. After normalization, the final loss value is the weighted sum of individual loss values. To formalize,

$$L(S \rightarrow T) = \sum_i w_i \hat{L}(i, S \rightarrow T) \tag{H.4}$$

where i is a loss type (identification, comparison, trend, text, density) and w_i is a weight term for each loss type. We initially decided the weight terms by heuristically testing them out, as shown in Table H.1. Users can fine-tune weight terms to reflect their priorities in the preferences menu.

Loss type	Channel/model	Wieght
Identification/ Comparison	Total	1.5
	Position	3
	Length	2.5
	Color	2.2
	Size	2.2
	Shape	1.5
	Text	1.5
	Others	1
Trend	Total	5
	Y~X	3
	Color~x+y	2
	Size~X+Y	2.3
	Color~X	1
	Others	1
Text	Total	0.5
Density	Total	6
	Occupied ratio	3
	Overplotted ratio	5

Table H.1: Initial weight terms

Appendix I

Dupo Pilot Study

After initially implementing Dupo, we conducted a pilot study with four visualization authors. In this appendix, we outline the pilot study methods and impacts to our system design (Section I.2) and the main study(Section I.3) .

I.1 Method

I.1.1 Participants

We recruited four visualization authors through social media and snowballing as shown in Table I.1

ID	Characteristics	Occupation	Data visualization experience
P1	Novice to responsive visualization	PhD Student	7 years as a researcher
P2	Novice to responsive visualization	Data analyst	6 years as an analyst
P3	Novice to responsive visualization	Designer	2 years as a UI designer
P4	Non-novice	PhD Student, Former Designer	10 years as a creative visualization and UI designer

Table I.1: Dupo pilot study participants

I.1.2 Procedure

1. Introduction and consent (10 minutes)
2. Trial 1 (manual version—Dupo without the recommender)

- Training (7 minutes)
 - Task instruction (5 minutes)
 - Task (25 minutes)
3. Trial 2 (recommender version)
 - Training (7 minutes)
 - Task instruction (5 minutes)
 - Task (25 minutes)
 4. Post-study interview (15 minutes)
 5. Wrap-up (5 minutes)

For the training, task, and interview parts, refer to Section J.1.

I.2 Changes to the System Design

I.2.1 Integrate Multiple Options for the Same Task

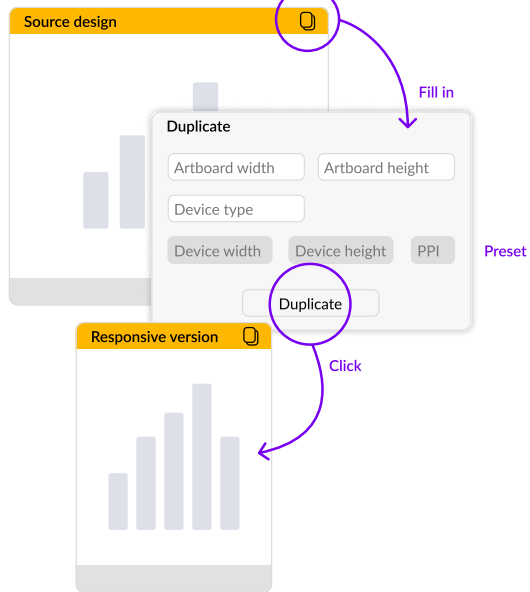
- Duplicate \Rightarrow Create a Responsive Version with preset sizes (Figure I.1).
- Change the default exploration view (a column view \Rightarrow a full view; Figure I.1).

I.2.2 Recommender interface

- Remove “see similar” for artboards (Figure I.2).
- Make the artboard area more compact (Figure I.2).
- Add an option to applying existing edits (Figure I.3).

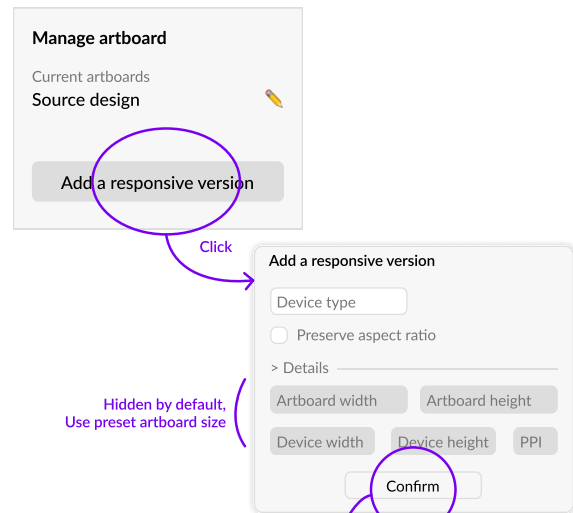
Duplicate/Default Explanation View

Current Duplicate

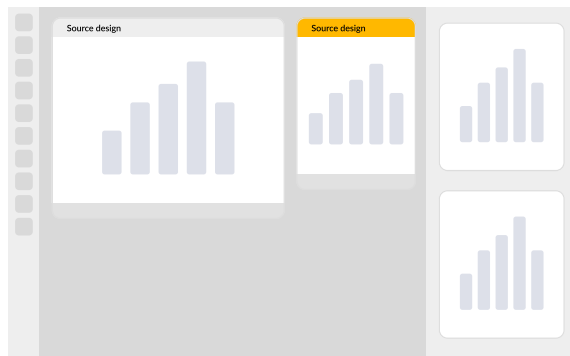


Changed Add a Responsive Version

Editing panel for the Artboard menu



Exploration view as a column



Exploration view as a pop-up window

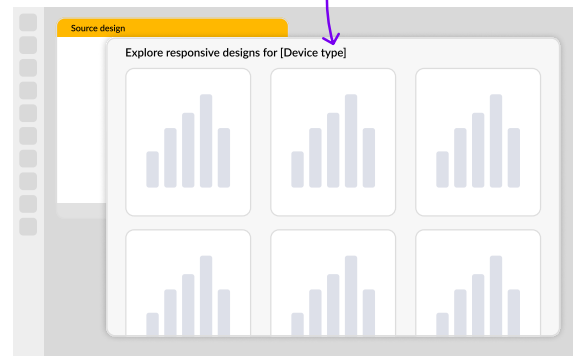
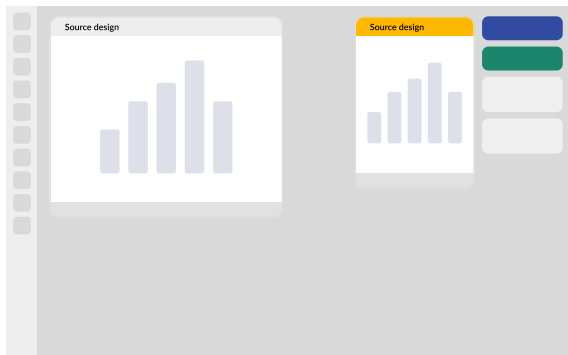


Figure I.1: Updates to duplicate and artboard creation

See Similar/Artboard Panel

Current

Two buttons/Wide spacing



Changed

Single button & Reposition/Narrow spacing



Explore alternatives

See similar

Quick edits

Figure I.2: Updates to the recommender buttons and artboard area

From Scratch/Apply My Edits

If checked, manual edits applied to the existing version is also applied to the recommendations in order to save time for doing same edits again. Cicero by default checks applicability of edits.
If not checked, then recommendations do not consider the existing edits in case a user wants to start a new edit path.

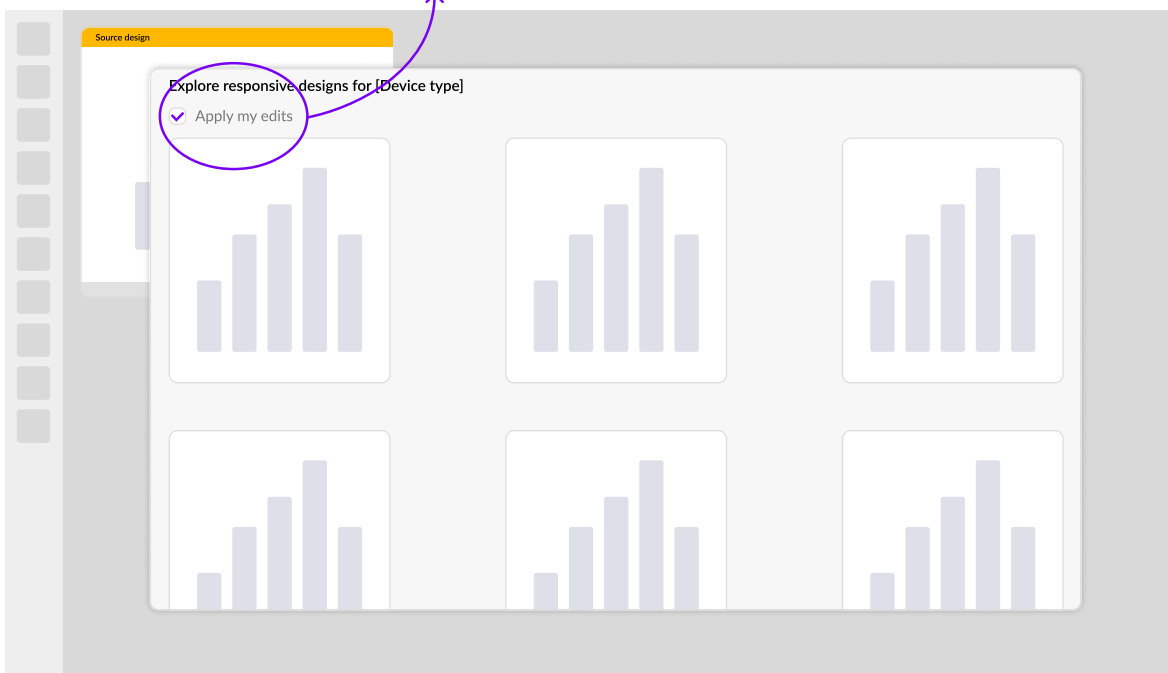


Figure I.3: Updates to the Control Panel

I.3 Changes Made to the Study Design

I.3.1 Study Material

Ask participants to bring several visualization designs they want to make alternative responsive versions to help participants to have more ownership to work and hence more engagement with the study. Participants had to spend more time on understanding the data and making non-essential comments (e.g., the color choice of a given source version).

I.3.2 Shorten the Trial Time

Focus only on the mixed-initiative version but increase the trial time (25 to 30 minutes) because longer study time limits the participation of expert users.

Appendix J

Dupo User Study Details

J.1 Study Protocols

J.1.1 Overview

1. Procedure overview (1 hour)
2. Introduction and consent (3 minutes)
3. Training (7 minutes)
4. Task instruction (3 minutes)
5. Trial (up to 30 minutes)
6. Self-evaluation (2 minutes)
7. Interview (15 minutes)

J.1.2 Protocol

J.1.2.1 Introduction and consent

Hi, I am [____] from [____]. Thank you for participating in this study. Have you had a chance to read the consent form that I sent you previously? I

will first briefly recap the study, and you will have two-three minutes to review the document again. Does this sound okay?

Alright, we are running this study to understand your experience of using a newly designed authoring system for responsive visualizations. You will have a training session for five minutes. Then, you will do one trial. In the trial, you will be creating responsive versions for the visualization we chose for 30 minutes. I will provide further instructions right before the trial. After creating them, you will respond to a short survey for about five minutes. After the trial, you will have an interview session for 15 minutes, and I will ask some questions about your experience. Do you have any questions so far?

Great. For each trial, when you are creating visualizations, you will have remote control access to my device while thinking aloud. Your voice for think-aloud and my screen will be recorded. The log recorded in our system will also be collected. You may mute your video while you are creating visualizations. Do you have any questions so far?

Okay. In the interview, your voice will also be recorded for analysis. However, the voice for the interview and think-aloud will all be deleted after transcribed. Any questions?

Your visualizations will be later evaluated by a panel NOT to evaluate your work BUT to evaluate our system's effectiveness. Your creation will always be anonymized for evaluation and in our paper. Do you have any questions?

Thank you. You can withdraw this study at any time, just let me know. After fully completing this study, you will be compensated with an Amazon gift card for USD 60. This might take a few weeks. Are we good to proceed?

Alright. Do you provide your consent for participating in this research study?

[If a participant provides a consent] Thank you for the consent. [If not] I am sorry that you don't agree. Thank you for the time!

J.1.2.2 Training

Now, you have a five-minute training session. While you are using the system, I will give you instructions and check your connectivity. This is not part of the study. I will ask you to join a remote control over Zoom, if you accept, then you will be able to use the system. Please mute your video for this training session. Also, feel free to ask questions.

See the Training section for more details.

J.1.2.3 Task Instruction

Now we will start the trial. In the trial, your role is a data journalist. So far, you have created a news visualization—the one that we chose for this study, and you need alternative versions for responsive contexts for it. That is, you have already confirmed the desktop version, and now it's time to adapt this desktop version for readers with different screen types.

You are going to create several responsive versions for other devices. That is, you will be creating those for smartphones, tablets, thumbnail, and paper printing given a desktop version. You must create a smartphone version for this task. You have up to 30 minutes total. If the time allows, you should also try to create other versions to cover other screen sizes like tablets, thumbnails, and printing. Do you have any questions about this trial?

As you start this trial, you will see the first version already loaded in the system.

It's up to you to decide when you are done with any given responsive version you create. We encourage you to make responsive versions carefully, as you would if you were going to publish them online. But don't be too bothered by tiny details. I will let you know every 10 minutes. While using this system, please let me know if you have any questions about how to use it.

While doing the trial, please keep thinking aloud—saying what you think while using the system. Again, I'm interested in hearing how you interact with the system, including what you like, what you are confused with, etc. Again, the whole study's point is to evaluate the system, not you, so don't be shy about speaking aloud your thoughts. For example, if you find something confusing, it would be nice if you say "I'm confused" and describe the confusing part. I'll keep reminding you to think aloud. Now, I will ask you to join a remote access session over Zoom/Microsoft Teams, if you accept, then you will be able to use the system. Please mute your

video for bandwidth during the trial.

J.1.2.4 Trial

We allowed users to ask questions regarding how to use the system.

Questions to answer: “How to do *specific edit* (e.g., *change the color of text*)?”

Questions to not answer: “How should I change the design” types of questions.

J.1.2.5 Self-evaluation question

Now, you are going to respond to a self-evaluation question for each version you created.

The below evaluation prompt was shown below each artboard.

Prompt: Given the limited design time you had, how satisfied are you with this draft?

Response options: Very satisfied, Satisfied, Just okay, Unsatisfied, Very unsatisfied

After each response, we followed up with the below question:

What changes or refinements would you make to your final design given more time?

J.1.2.6 Interview questions

What was your overall reaction to seeing design suggestions at the beginning?

How was being able to make edits to the design suggestion that you have chosen?

What was different between using this system and your day-to-day visualization tasks?

If you can take out individual features from the system and embed/apply them to your whatever existing tools, which feature would you choose, and why?

[With showing the suggestions]

How useful were the suggested design examples for responsive versions?

How would you rank them?

Why did you rank them in that way?

Any suggestions for improvements?

J.1.3 Training Details

The training goal was to create the below desktop version and then explore design suggestions for the mobile version.

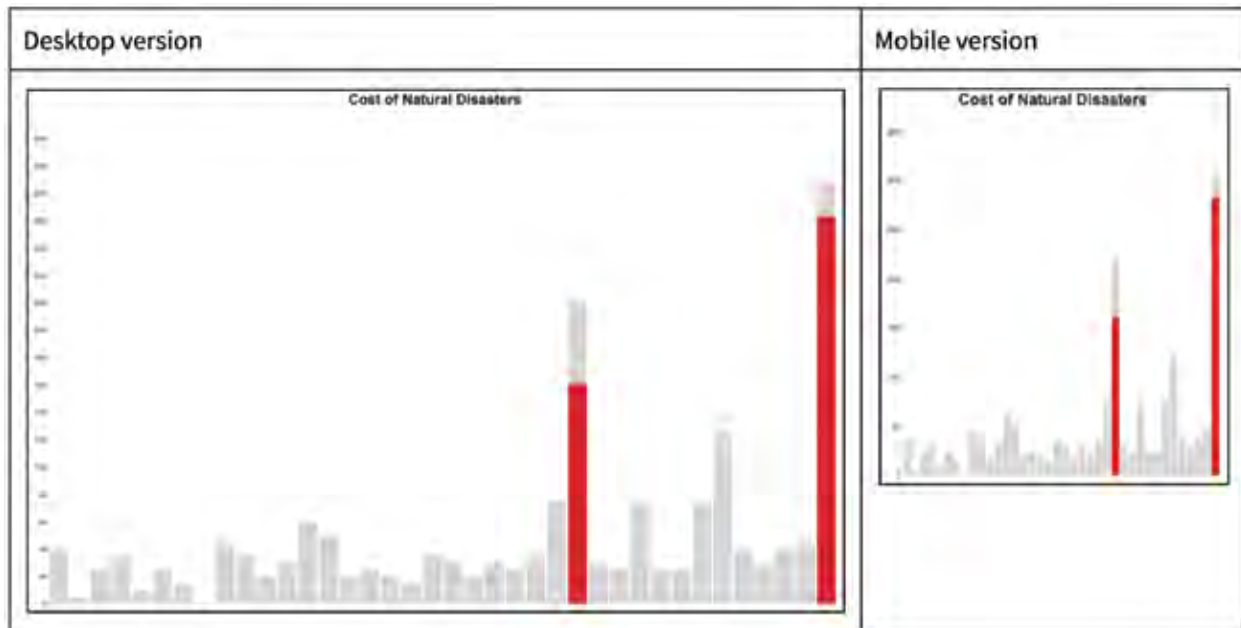


Figure J.1: Training visualizations for Dupo's user study.

J.1.3.1 Instructions

Check dataset:

1. For this study, the dataset for visualization is already loaded. To check, click the data icon, it looks like a table.

Creating an artboard:

1. To create a visualization, you need to create an artboard for a screen type.
2. Click the artboard icon with an easel on the left sidebar to create one.
3. You can also see the tooltip messages to see the menu description.
4. Then, click "New responsive artboard."
5. For the name, use "cost".
6. Choose a device type as "Desktop-Landscape," then a preset device screen dimension will be loaded.

7. If you click “Advanced,” then you’ll be able to change the size.
8. For size, enter 1,000 for width and 600 for height.
9. For this study, you don’t need to customize the device screen dimensions.
10. Lastly, click “create” to finalize.

Setting layout (use shelves):

1. Assuming that the data is already loaded, you need to set the layout of a visualization first, like which fields you want to use for x and y axes.
2. To do so, click the layout menu icon with four filled squares in the left sidebar.
3. First, set “layout type” as “single”.
4. Next, we will assign data fields to the x and y axes.
5. You will see a small widget for data fields.
6. If you hover on a badge, then you will see a preview of that field.
7. Rows are vertical axes and columns are horizontal axes.
8. Drag the “year” field badge and drop it on the “columns” shelf.
9. Drag the “cost” field badge and drop it on the “rows” shelf.

Change mark type:

1. The previous step automatically generates point marks.
2. Now, let’s change the mark type.
3. Click the mark menu icon with a filled circle overlaid on a square. It’s in the third icon group.
4. You’ll see a list of mark layers. Click the pencil icon next to the one for the point mark.
5. Then, you’ll see different mark types. Click the “bar” mark for this training session.

Set a mark property encoding:

1. Now, you will change the fill color of the mark.
2. Click the fill color attribute below.
3. If you want a static value, you can click “static.” Or, if you want to pipe a data field to the marks, click “encoding.” Let’s choose “encoding” for now.
4. You’ll see a data field widget. Drag the “labeled” field badge to the shelf.
5. Then, some preset values are shown.
6. To change the scale by providing specific domain and range values, you can change it below. Domain means the data values and range values are visual values.
7. For this session, enter “true” in the domain and click the color picker. Choose some red color, and click “confirm” in the picker.
8. Then, click “add a scale row” for another value.
9. Enter “false” in the domain and click the color picker.
10. Choose some gray color and click “Confirm.”
11. Click “Apply” to finalize this edit.

Selection menu:

1. You can also double click an element, and a context menu will show up on the left.

Duplication:

In the pilot study, this and later instructions were for the second trial training

1. Now, we are going to create a mobile version.

2. To do so, click the artboard menu icon.
3. Click “new responsive artboard.”
4. Choose a device type “Phone-portrait,” then the size preset will be loaded. You can check it by clicking “Advanced.”

Exploration:

1. You will see our exploration panel that provides some design ideas for bigger responsive transformations.
2. For each design,
 - (a) If you click “apply” then the design applies to your mobile version.
 - (b) If you click “branch” then it will create another artboard for your mobile version.
 - (c) If you click the “hide this” button, then the design is removed and similar designs will not be suggested in the future.
3. To access again, click the magic wand button.

Alteration:

1. To explore some subtle changes, click the “see similar” button when you hover an example.
2. Close the exploration panel.

Active, propagation, lock, and solo:

1. Double click the title, and change the text in any way.
2. Now, you’ll see the title is also applied to the duplicated version.
3. To make changes to specific artboards, you can “lock” an artboard by clicking the lock icon on the top right corner of the artboard. Now you’ll see that the

locked artboard is less visible.

4. Let's lock the desktop version, and then change the text alignment to "left." Then this change applies only to the mobile version.
5. If you want to apply edits only to the current artboard, click the "S in a square" icon to solo that artboard.
6. To easily lock and solo artboards, you can use the artboard list on the bottom right of this application.

J.2 Analysis Details

Overall, we used our research questions and structured interview questions as initial themes for the analysis of the think-aloud and interview data. In addition, we looked at the log data and screen recordings to understand how they make manual edits and use the recommender. Below, we outline how we filtered 'experimenting edits' from the log data and tagged reasoning behaviors in the screen recordings.

J.2.1 Filtering Experimenting Edits

Each manual edit that a participant made was expressed as a Cicero rule. A Cicero expression fundamentally includes a specifier (an element to change), action (the direction of change), and option (property(s) to change). After making a first edit, a user makes the same edit with different values for the same element and property to experiment with that property. In the below example from E4, he made an initial edit to annotation_6 for the tick's dy (relative vertical position) value (-7). As shown in the callTime, right after making that initial edit, E4 made an experimenting edit to the same element and property with a different value (-6). We captured this kind

of edits as experimenting edits, defining them as “a series of consecutive rules for the same specifier, action, and option property(s) with different option values.”

An initial edit

```

1 {
2   "description": "Edit annotation 6 tick position from mark (dy)",
3   "specifier": {
4     "role": "annotation",
5     "id": "annotation_6"
6   },
7   "action": "modify",
8   "option": {
9     "tick": {
10      "from": { "dy": -7 }
11    }
12  },
13  "callTime": "11:28:03 AM"
14 }

```

An additional experimenting edit

```

1 {
2   "description": "Edit annotation 6 tick position from mark (dy)",
3   "specifier": {
4     "role": "annotation",
5     "id": "annotation_6"
6   },
7   "action": "modify",
8   "option": {
9     "tick": {
10      "from": { "dy": -6 }
11    }
12  },
13  "callTime": "11:28:03 AM"
14 }

```

J.2.2 Tagging Screen Recordings for the Use of Recommender

We tagged screen recordings into six categories as below. The main goal of this tagging is to understand what kind of reasoning behaviors the participants performed rather than how long they spent for each behavior, or what suggestion they paid

more attention to. We note that this tagging is not intended for precise time measurement.

J.2.2.1 Tag categories

- **Load and overview**
 - Obtain Exploration designs: A participant requested Exploration suggestions and saw them before they jumped to a specific suggestion(s).
 - Obtain Alteration designs: A participant clicked the “see similar” button for an Exploration suggestion and saw them before they jumps to a specific suggestion.
- **Reason and inspect**
 - For a single design: A participant read the descriptions in the action widget and made any comment relating to what they likes or not for one particular suggestion.
 - For multiple designs: A participant compared two or more suggestions.
- **Next steps**
 - Brainstorm possible edits: A participant thought of potential manual edits they could make for a design suggestion.
 - Summarize thoughts: A participant rationalized their motivations or final decisions.

J.2.2.2 Log data

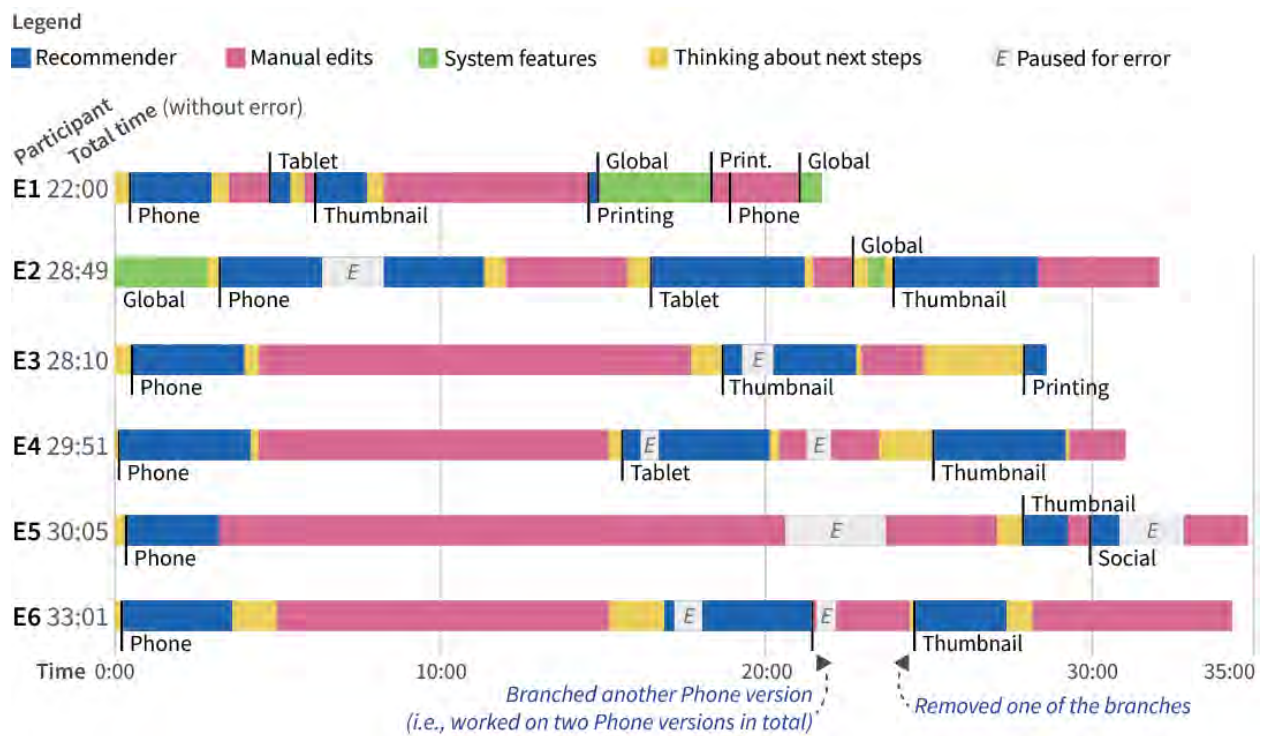


Figure J.2: Dupo user study log data for overall usages.

Legend

Load and overview

- Obtain *Exploration* designs
- Obtain *Alteration* designs
A participant requested design suggestions and took an overview of them.

Reason and inspect

- For a single design
- For multiple designs
A participant reason about and inspect designs individually or by comparing.

Next steps

- Brainstorm possible edits
A participant thought of potential manual edits they could make for a design.
- Summarize thoughts
A participant rationalized their motivations or final decisions.

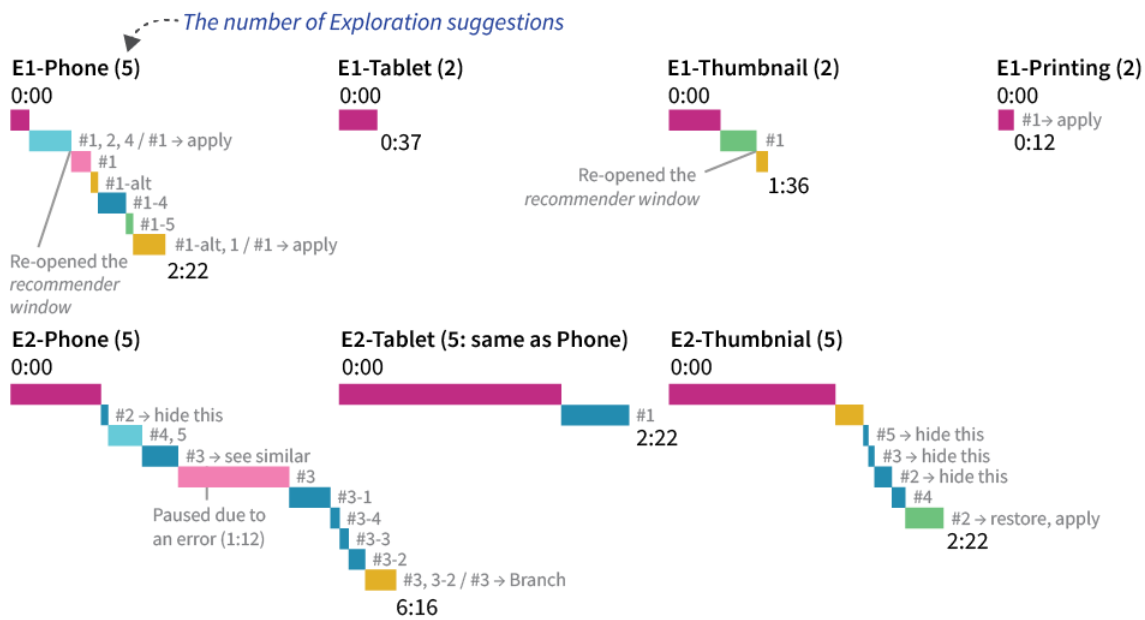


Figure J.3: Dupo user study log data for recommender usages (part 1).

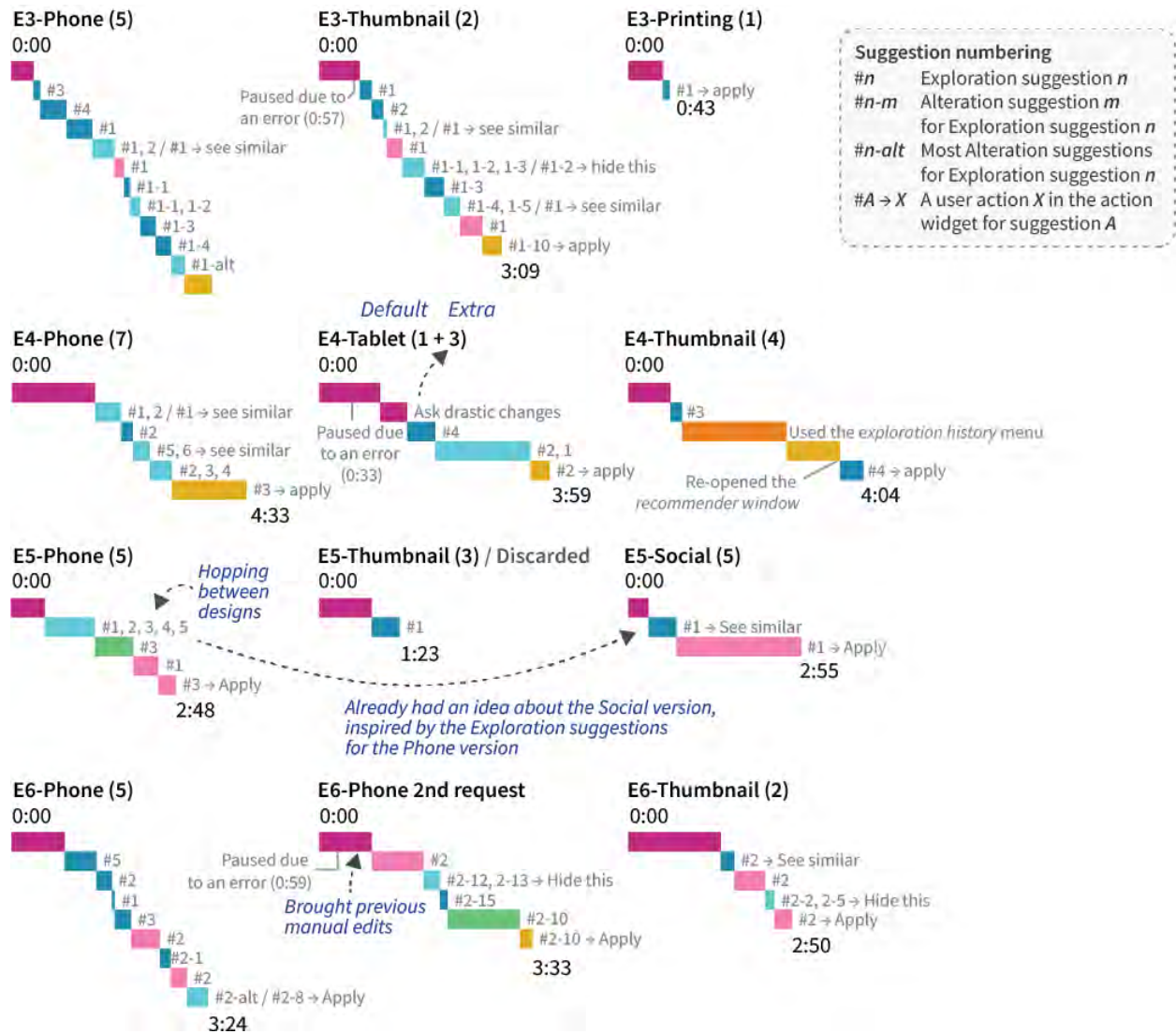


Figure J.4: Dupo user study log data for recommender usages (part 2).

Appendix K

Sonification Tutorials and Design Cases for the Formative Study of ERIE

K.1 Collected Tutorials

Learning Data Sonification (2021)

- URL: https://reginaldbain.com/vc/musc540/pub/learning/data_sonification.html
- A course material
- Introduces toolkits (most of which we already cited)

Sonification 101: How to convert data into music with Python (2022)

- URL: <https://medium.com/@astromattrusso/sonification-101-how-to-convert-data-into-music-with-python-71a6dd67751c>
- Video tutorial: <https://www.youtube.com/watch?v=DUdLRy8i9qI>
- Notebook: <https://github.com/SYSTEMSounds/sonification-tutorials/blob/main/data2midi-part1.ipynb>
- Introduces how to convert variables to time, pitch, and velocity (= gain) channels.

- Sound realization through PyGame library (a python library for video games); output is MIDI.

I Created Music From Data Using Python (2022)

- URL: <https://towardsdatascience.com/i-created-music-from-data-using-python-adfc349f55f1>
- Introduces how to convert variables to time, pitch, and amplitude channels.
- Sound realization through music21 library (a python library for MIDI creation); output is MIDI.

Literate Computing for Art and Science (post-2020 Based on the library usages)

- URL: <https://jupyter.brynmawr.edu/services/public/dblank/jupyter.cs/Sonification.ipynb>
- Introduces how to create tone and scale functions in Python.

Making an animated data visualisation / sonification (2021)

- URL: <https://propolis.io/articles/making-animated-dataviz-sonification.html>
- Introduces how to encode time/duration and pitch (note to frequency).
- Use R functions to do so, but the actual sound processing was done in After Effects.

Turning Data Into Sound (2020)

- URL: https://jbrussell.github.io/eilive2020/part2a_sonification/
- Code: https://nbviewer.org/github/jbrussell/EI_Live_2020/blob/master/earthquake_sonification/1_bokeh_notebooks/1_simple_sounds.ipynb

- Introduces how to sonify using a custom UI element in Jupyter Notebook using Bokeh (a UI toolkit).
- But the custom UI element does not have “scale” or “tone” parts, all those audio value calculations are done before using it.
- Sound is realized using Howler.js (a popular audio creation library in JS).

Turn your data into sound using our new MIDITime library (2015)

- URL: <https://revealnews.org/blog/turn-your-data-into-sound-using-our-new-miditime-library/>
- Introduces how to encode time and pitch.
- Sound realization through MIDITime (a python MIDI library).

From Data to Music – Max as a Sonification Algorithmic Composition Tool (2016)

- URL: <http://www.algorithmiccomposer.com/2016/01/from-data-to-music-max-as-a-sonification.html>
- Uses MAX (a visual programming language)
- Developers need to creating a scale function by themselves.

SoniPy (2009)

- URL: <https://www.sonification.com.au/sonipy/index.html>
- A collection of python libraries could be used for sonification.
- Mostly a collection of low-level music programming libraries.

Table K.1: The distribution of sonification use cases. *Audio tools refer to professional audio editing tools (e.g., Ableton Live) and electronic music equipment (e.g., sequencers). **Circuit-based sonifications mean devices that generate sonifications by physical inputs through some kind of electronic circuits. ***While I could sense that the sounds are electronically generated (e.g., sine-wave oscillator or synthesizers) but they did not provide clear creation methods.

Types		Total	Inspectable	Non-inspectable
Programming	Codes	28	13	5
	Visual Programming	2	2	0
	Codes + Audio Tools*	10	6	4
	Circuits**	5	1	4
	Domain-specific applications	3	2	1
	Digitally created with unknown creation methods***	26	0	26
Instruments		11	0	11
Unavailable		3	0	3
Total		88	24	54

K.2 Collected Sonification Designs

K.2.1 Overview

K.2.2 Collected Designs

Spiders Song: The Music of Evolution (Phylogenetic Sonification) (2023)

- Authors: Future Ecologies
- URL: <https://www.youtube.com/watch?v=Z8c7rvGrNko>
- Methodology: Visual programming
- Notes: Created using Max (a visual programming language for multimedia)

Hydrological Soundscapes (2023)

- Authors: Ivan Horner and Benjamin Renard
- URL: <https://hydrologicalsoundscapes.github.io/>
- Methodology: Programming
- Notes: Used Tone.js; custom scale functions, specifically designed for the data

set of this project. Data processing and audio conversion were done separately.

Solar System Orbital data (2023)

- Authors: Philip Bergstrom
- URL: <https://github.com/phber/harmonices-mundi>
- Methodology: Programming
- Notes: Used Tone.js and Three.js; sounds are played as objects are controlled by Three.js. A planet moves, controlled by Three.js; then when it reaches at a certain position, Tone.js was used to make a related sound.

Degen Blues (2022)

- Authors: Andy Szybalski and Stephen Chau
- URL: <http://degenblues.xyz/>
- Methodology: Programming
- Notes: Offers an API for making time + pitch sonification. Uses Tone.js

Mass Extinction Magnitude Data Sonification (2022)

- Authors: Barnas Monteith
- URL: <https://www.youtube.com/watch?v=0nGue7Nso5I>
- Methodology: Programming
- Notes: Uses MIDITime (raw data to MIDI values); then post-processed using music software.

A postcard from Antarctica (2022)

- Authors: Benjamin Renard
- URL: <https://vimeo.com/653705727>
- Methodology: Programming

- Notes: Use sampled mp3 files and uses sequenceR package to mix those sampled sounds. sequenceR is a sequencer. It does not calculate actual time and pitch values.

The rhythm of flood occurrences (2022)

- Authors: Benjamin Renard
- URL: <https://vimeo.com/653714131>
- Methodology: Programming
- Notes: (Same approach) Use sampled mp3 files and uses sequenceR package to mix those sampled sounds. sequenceR is a sequencer. It does not calculate actual time and pitch values.

NASA'S Space Jam (2022)

- Authors: NASA/CXC/SAO/K. Arcand/A. Jubett/K. DiVona, SYSTEM Sounds (M.Russo, A. Santaguida)
- URL: <https://chandra.si.edu/sound/code/>
- Methodology: Visual programming
- Notes: Used a visual programming interface (Max); pitch and drum sounds; but does not provide customizable data mappings

The Sound of Sight (2022)

- Authors: Omar Shehata
- URL: <https://omarshehata.itch.io/sound-of-sight>
- Methodology: Programming
- Notes: Uses Tone.js, can customize the sound it generates, but there's no part that shows mapping from data to sound (no dynamic)

Talkative Men—The Gender Difference in the Zurich Cantonal Council (2022)

- Authors: Simon Huwiler
- URL: <https://www.youtube.com/watch?v=RhDcTuwuAq8>
- Methodology: Codes + audio tools
- Notes: Used Sonic Pi. Uniformly timed; custom scale functions (nominal); backgrounded by a composed music

Transforming Realtime Air Quality and Asteroid Data into MIDI (2022)

- Authors: uisato
- URL: <https://www.youtube.com/watch?v=NLmNGdSAPdA>
- Methodology: Codes + audio tools
- Notes: Mixed use of TouchDesigner (a visual programming tool for multimedia) + Ableton Live (a music workstation) + manual control of electrical instruments

Extreme Weather in Three Movements (2021)

- Authors: Andras Blazsek
- URL: https://soundcloud.com/user-763903752/sets/andras-blazsek-extreme-weather-in-three-movements?utm_source=clipboard&utm_campaign=wtshare&utm_medium=widget&utm_content=https%253A%252F%252Fsoundcloud.com%252Fuser-763903752%252Fsets%252Fandras-blazsek-extreme-weather-in-three-movements
- Methodology: Codes + audio tools
- Notes: Seems to have used various music production tools with custom data cleaning and scaling to audio (see more at here: <https://dataclimate.org/2022/09/09/daca-workshop-day-1/>)

Hydrology basics: the Ardèche river at Sauze (2021)

- Authors: Benjamin Renard and Chloé Le Bescond
- URL: <https://globxblog.inrae.fr/hegs/>
- Methodology: Programming
- Notes: (Same approach) Use sampled mp3 files and uses sequenceR package to mix those sampled sounds. sequenceR is a sequencer. It does not calculate actual time and pitch values.

Hydrological Principal Component Analysis (2021)

- Authors: Benjamin Renard and Chloé Le Bescond
- URL: <https://globxblog.inrae.fr/pca/>
- Methodology: Programming
- Notes: (Same approach) Use sampled mp3 files and uses sequenceR package to mix those sampled sounds. sequenceR is a sequencer. It does not calculate actual time and pitch values.

Hidden Climate Indices (2021)

- Authors: Benjamin Renard and Chloé Le Bescond
- URL: <https://vimeo.com/600898709>
- Methodology: Programming
- Notes: (Same approach) Use sampled mp3 files and uses sequenceR package to mix those sampled sounds. sequenceR is a sequencer. It does not calculate actual time and pitch values.

Entity (2021)

- Authors: Charlotte Roe

- URL: <https://charlotteroe.space/entity-2021/>
- Methodology: Programming
- Notes: Seems to be using Sonic Pi and Processing

A Sonification of the zCOSMOS Galaxy Dataset: a history of galaxies (2021)

- Authors: Sandro Bardelli, Claudia Ferretti, Giorgio Presti, Maurizio Rinaldi
- URL: <https://www.lim.di.unimi.it/demo/zcosmos.php>
- Methodology: Codes + audio tools
- Notes: MATLAB for data inspection and preprocessing, Supercollider to parse the CSV exported by MATLAB and perform real-time sound synthesis, Ableton Live to record the distinct audio tracks generated by Supercollider, and, finally, Steinberg Cubase for postproduction (see Page 7 in <https://arxiv.org/pdf/2202.05539.pdf>).

Soundscape in Times of Change (2021)

- Authors: Sara Lenzi, Permagnus Lindborg, Juan Sádaba
- URL: <https://www.frontiersin.org/articles/10.3389/fpsyg.2021.570741/full>
- Methodology: Codes + audio tools
- Notes: It is more about sound recording analysis

The Sound of Crowns and Tears (2021)

- Authors: Simon Huwiler
- URL: <https://www.youtube.com/watch?v=Dqfr0Ps2pKM>
- Methodology: Codes + audio tools
- Notes: Converted data points to musical notes and time, then it was printed as a music box punch card

Broadcast (2021)

- Authors: Studio Tilt
- URL: <https://www.studiotilt.design/broadcast>
- Methodology: Circuit
- Notes: Audio-based installed interface

Tiltification (2021)

- Authors: Tim Ziemer and the Master programs' students
- URL: <https://tiltification.uni-bremen.de/index.html>
- Methodology: Domain-specific application
- Notes: Audio feedback application with specific use case (tilt leveling; <https://github.com/Tiltification/sonic-tilt>)

Sonification-Enhanced Lattice Model Animations of the Protein Folding Reaction (2022)

- Authors: Carla Scaletti, Meredith M. Rickard, Kurt J. Hebel, Taras V. Pogorelov, Stephen A. Taylor, and Martin Gruebele
- URL: <https://pubs.acs.org/doi/full/10.1021/acs.jchemed.1c00857>
- Methodology: Domain-specific application
- Notes: Build a domain-specific tool for sonifying organic proteins

The UK's July 2022 Heatwave Turned Into Sound (2021)

- Authors: Duncan Geere and Miriam Quick (Loud Numbers)
- URL: <https://www.youtube.com/watch?v=bWmkTiemeEk>
- Methodology: Digitally created with unknown creation methods
- Notes: Programmed custom data cleaning and data-to-audio conversion func-

tions (only supports a linear mapping; see here: <https://observablehq.com/@duncangeere/data-mapper>). Then, the converted audio values were fed to digital instruments and then to a sequencer.

Sonoplanet (2021)

- Authors: Phia Damsma and John Norgaard (Sonokids)
- URL: <https://www.youtube.com/watch?v=baVtw9MBrA8>
- Methodology: Digitally created with unknown creation methods
- Notes: An educational app for teaching kids sonification

Appendix L

Technical Details of Erie

This appendix details the ERIE grammar.

L.1 Customizing a Tone

The *tone* of a single sonification design is defined in terms of instrument *type*, whether the sound is *continued*, and a set of audio *filters*. To use custom instruments to express diverse sonification designs, a developer can define new instruments using *synth* (for FM and AM synthesizer), *wave* (directly defining a wave function), and *sampling* (importing external audio files) objects in a top-level spec. The developer can apply custom instruments to the *tone type* and a *timbre* encoding channel by referencing their names.

A dataset typically exists as a set of data points; even if it represents a continuous distribution, its digital format is a set of approximated data points. Thus, a data representation should be able to capture the continuity or discreteness between data points (e.g., line chart vs. scatterplot). In the walkthrough, for example, we used a *discrete* (*continued = false*) tone to indicate that each sound represents a dis-

crete bin. On the other hand, a developer could use a *continuous* (*continued = true*) tone for a sonification of a continuous distribution. A sound is *discrete* if it is momentarily paused and resumed as auditory values change (e.g., a sound “beep Beep BEEP” with varying loudness). A sound is *continuous* if it is not paused as its auditory values change (e.g., a sweeping sound “bee^{C3}-ee^{C#3}-eep^{D3}” with increasing pitch).

When more artistic sound effects (e.g., dynamic compression, distortion) are needed, a developer can apply them using the *filter* property of a *tone*. A *filter* object is an ordered list of filter names, and each filter is applied after the previous filter, reflecting how audio filters are commonly applied to electric sounds. ERIE considers the properties of an audio filter (e.g., level of compression) as encoding channels so that a developer can configure audio filters both statically and dynamically (mapped to data variables). Our implementation offers several preset filters (e.g., dynamic compressor) and APIs for audio experts to define and use custom filters.

L.2 Encoding

Below, we detail how to indicate specific properties for different encoding channels and auxiliary or shortcut properties for diverse sonification designs.

L.2.1 Expressing Time as an Encoding

Time is to sonification as position is to visualization. An audio graph arranges its auditory values along a time axis. ERIE expresses time as encoding to enable data-driven time control. For example, the start time of each sound can be mapped to a certain data variable.

The time axis of a sonification encodes data either in terms of the start and end times of a sound (*time* and *time2*) or the start time and duration of a sound (*time* and *duration*). On the one hand, two data variables sharing the same unit (e.g., monthly lowest and highest temperature) can be mapped to start and end times. On the other hand, two data variables with different units (e.g., country names and CO2 emissions) can be mapped to start time and duration. ERIE supports expressing when a sound starts and ends (*time + time2*) or when and how long it is played (*time + duration*).

The length of a sonification is also the *range* of its time channel. Thus, ERIE provides another shortcut, *length*, for the *range* of time scale (i.e., $[0, length]$). When there is no need to encode end time or duration, *time* channel can have *band* to set the duration of each sound uniformly (for discrete tones).

ERIE makes a distinction between *when a sound starts* (the value of the *time* channel) and *how a sound is timed* in relation to other sounds (*timing*). For example, a developer wants a sound to be played after the previous sound (*relative*), to start on an exact time (*absolute*), or to start with the other sounds (*simultaneous*). To control how a *time* channel assigns times, the developer can use the *timing* property of the *time* channel's scale. The above extensions to the *time* channel's scale is formalized as:

$$scale_{time} := \{ \dots, timing, length, band \}$$

These time-related channels and the *timing* option produce the following high-level combinations:

Case 1: *time*(*field* = *x*, *scale.band* = *n*)

A *time* channel with a fixed *scale.band* value defines sounds with a fixed duration (*n*) and start times varied by the encoded data variable (*x*). If *scale.timing* is *simultaneous*, then all of the sounds are played at the same time.

Case 2: *time*(*field* = *x*) + *duration*(*field* = *y*)

Using both *time* and *duration* channels defines sounds with varying durations and start times.

Case 3: *time*(*field* = *x*, *scale.timing* = *absolute*) + *time2*(*field* = *y*)

A *time* channel with *absolute timing* and a *time2* channel specify sounds with varying start and end times. Note that the two fields mapped to the *time* and *time2* channel must be defined on the same data unit, such as bin start and endpoints.

L.2.2 Channel-specific properties

Specific auditory encoding channels may have different physical constraints that need channel-specific properties in addition to the above definition. ERIE considers such physical constraints in defining encodings for canonical auditory channels. For example, *pitch* can have raw pitch frequency values or have them rounded to musical notes. To enable this rounding, a developer can set *round-to-note* to *true* for the *pitch* channel.

L.2.3 Providing auditory reference elements

Tick for time channel

A longer sonification may need to provide a sense of the progression of time as Cartesian plots have axis ticks and grids. To do so, a developer could use a *tick* sound that repeats every certain time interval. The developer can define a *tick* directly in the *time* channel or refer to a tick definition in the top-level *tick*.

Scale description markup

As we use legends for data visualizations, it is important to provide the overview of the scales used in a sonification [29]. The *description* of a scale can be skipped, defined as a default audio legend set by a compiler, or customized. To customize a scale description, ERIE employs a markup expression that can express literal texts, audio legends (*<sound>*), a list of items (*<list>*), and reserved keywords, such as *<domain.min>* (for the minimum domain value) and *<field>* (for the data field's name). A developer can also pass a number or date-time *format* in the channel definition.

L.2.4 Inline Transform

Inspired by Vega-Lite [96], it is possible to provide an inline data transform: *aggregate* or *bin*. This is a shortcut for defining a corresponding *transform* item and use the resulting variables in the channel's *field*. For example, the separately defined transforms in the walkthrough can be rewritten as:

$$time = \{field = miles-per-gallon, bin = true, \dots\}$$

$$pitch = \{aggregate = count, \dots\}$$