Blitz: Automating Interaction in Visualization

Catherine Mullings
Stanford University
cmulling@stanford.edu

Ben-han Sung
Stanford University
bsung93@stanford.edu

Andrei Terentiev
Stanford University
andreit1@stanford.edu

ABSTRACT

Many data visualization tools only produce static visualizations, yet a key aspect of a good data visualization is interaction. Interactivity allows one to fluidly explore and analyze data. However, designing a good interactive visualization usually requires significant time and effort to build a solution customized for a specific dataset. In this paper, we present Blitz, a system that can take any dataset and uses simple mapping rules from data types to interaction paradigms to automatically generate interactive visualizations. Blitz leverages prior work on interaction mapping rules with the goal of making the visualization design process simpler while maintaining a robust set of interactions. Overall, users found that Blitz is effective in generating interaction paradigms that fit a given data set and is useful for exploratory data analysis. Additionally, participants commented that it was much easier and faster to generate an interactive visualization with Blitz than with Tableau.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Interaction styles

INTRODUCTION

Traditional visualization software like Tableau support limited interactivity compared to rich web frameworks like D3. On the other hand, while D3 can produce visualizations many interaction possibilities, building such visualizations typically requires many hours of development time.

Despite the importance of interaction in visualization, the majority of attention in information visualization research centers on static representation of data on a display [8]. Research on interaction in visualization often plays a secondary role [8]. Nevertheless, for meaningful analysis and iterative exploration of data, interaction in visualization is essential. Despite little research on automating interaction, the problem itself stands as a ripe ground for innovation. Automating interaction would allow designers and analysts to fluently and flexibly explore their data.

CHI 2009, April 4 - 9, 2009, Boston, Massachusetts, USA.

That being said, there are many challenges to automated interaction design. Consider the simple task of adding a GUI to a visualization that allows the user to filter data elements. It is not enough to simply select the correct GUI element for the data (ex. range slider for quantitative data) - one needs to have a deeper understanding of the underlying data in order to avoid a subpar result. For example, setting an equal-step size for a range slider works well for evenly distributed quantitative data, but leads to problems if the quantitative data is distributed unevenly. In such a case, many subranges in the slider may have no data bound to it, which may confuse a user in addition to being a waste of space.

Another challenge in automating interaction is choosing a data representation (in the program) that is amenable to interaction, i.e. one that supports efficient search and filtering so that dynamic queries can be implemented on top. For example, date strings in a dataset need to be converted to a quantitative scale like seconds in order to be compared and plotted in a meaningful way.

Finally, all visualizations are not the same; although there are general interaction paradigms that work well for most datasets, the reason that interactive visualizations are usually created in a custom manner is that designers want to guide the user in a specific way. Algorithmically generating all of the interactive aspect of a visualization needs to be able to make a good trade off between simplicity and the expressiveness that custom code can offer.

In summary, automated interaction design is a difficult but desirable goal in information visualization, because it allows any user to do basic exploration of high dimensional sets, without the overhead of having to create their own interactive tools.

RELATED WORK

Below we explore related prior work on interaction in digital visualization, with various degrees of automation vs. manual coding required for the interaction design.

One example of semi-automated interaction in visualization design can be found in Heer et al's Generalized selection via interactive query relaxation [6]. Their query relaxation engine takes a scatter plot and bar chart and enables users to interactively relax lasso (click and/or drag selections) selections. Blitz takes inspiration from this work and aims to explore interaction paradigms more generally.

Another example of semi-automated interaction is VisDock [5]. It is best described as a toolbar that can be attached to any visualization, which it will augment by providing navigation (rotation, panning, zooming) and annotation functionality. However, VisDock is data-agnostic; Blitz seeks to make a more deliberate approach to interaction, applying only the appropriate interactions to a data type. For example, panning is more relevant to a spatial visualization like a scatter plot than an axis-free visualization like a pie chart. Additionally, in contrast to VisDock, Blitz focuses more on dynamic query selection and filtering.

Another approach to interactive visualization design is taken by Vega [11] and Vega-Lite [3], a pair of visualization frameworks built on top of D3. Vega proposes a declarative style for creating visualizations - the user creates a JSON specification of what they want to visualize instead of writing D3 code. However, while a declarative style works well for static visualizations, interaction is more difficult to encode. Vega uses a system of signals that actually manages to express the scope of interaction possibilities quite well, but learning how to use this system requires a steep learning curve. Blitz seeks to builds on top of Vega by abstracting away this difficult process of making an interaction specification.

In the same vein as APT [9], Blitz uses Bertin's visual encodings [4] and the taxonomies of interactions researched by Yi et al. [8] and Heer et al. [7]. Particularly useful to Blitz is Heer et al.'s theoretical mapping from data type to interaction paradigm (e.g. range slider, checkbox). Blitz implements a subset of this mapping (as discussed in the Methods section below). Additionally, Blitz's implementation leverages Heer et al's theoretical mapping of user purpose (need to perform visual analysis) to interaction paradigms [7]. For example, if a user wants to highlight aspects of the data, Blitz allows users to hover or mouse click.

BLITZ

Here we present Blitz, a new system for generating interactive data visualizations. Blitz is designed with the goal of streamlining the process of creating interactive visualizations as much as possible. It leverages prior work [3, 7, 8, 11] to create mappings from low-level data types to high-level interaction paradigms such as filtering, hovering, and selecting, and uses this mapping to add interaction to static visualizations based on the contained data types.

Specifically, Blitz supports the following interactions on visualizations (e.g. bar charts, line plots, etc.): hovering and selecting. Additionally, Blitz showcases linked visualization, wherein filtering in one visualization (e.g highlighting a datum) affects another visualization. Finally, Blitz includes a filter panel that maps data types to filter interaction widgets that control the visualizations.

METHODS

As shown in Figure 1, Blitz uses a three-page work flow where it (1) receives a text dataset from the user and (2) asks the user label the data types present in the data set, and then (3) presents an interactive visualization of that data set.

Figure 1. Our work flow

Input Dataset

In the first step, we ask the user to give us a dataset via copy-and-paste into a text box. We would like to point out importance of this step, which is that we are striving to make Blitz a general interactive visualization framework that is compatible with any input that the user gives us. All we require is a standard format (CSV or JSON) so that we can parse the data.

```
}name,age,gender,raceethnicity,month,day,year,streetaddress,city,state,latitude,longit
ude,state_fp,county_fp,tract_ce,geo_id,county_id,namelsad,lawenforcementagency,cause,
armed,pop,share_white,share_black,share_hispanic,p_income,h_income,county_income,comp
_income,county_bucket,nat_bucket,pov,urate,college
A'donte Washington,16,Male,Black,February,23,2015,Clearview Ln,Millbrook,AL,
32.529577,-86.362829,1,51,30902,1051030902,1051,Census Tract 309.02,Millbrook Police
Department,Gunshot,No,
3779,60.5,30.5,5.6,28375,51367,54766,0.937935946,3,3,14.1,0.007686375,0.168509509
Aaron Rutledge,27,Male,White,April,2,2015,300 block Iris Park Dr,Pineville,LA,
31.3217392,-92.4348602,22,79,11700,22079011700,22079,Census Tract 117,Rapides Parish
Sheriff's Office,Gunshot,No,
2769,53.8,36.2,0.5,14678,27972,40930,0.683410701,2,1,28.8,0.865723794,0.111402359
Aaron Siler,26,Male,White,March,14,2015,22nd Ave and 56th St,Kenosha,WI,
42.5835597,-87.8357101,55,59,1200,55059001200,55059,Census Tract 12,Kenosha Police
Department,Gunshot,No,
4079,73.8,7.7,16.8,25286,45365,54930,0.825869288,2,3,14.6,0.166293142,0.147312269
Aaron Valdez,25,Male,Hispanic/Latino,March,11,2015,3000 Seminole Ave,South Gate,CA,
33.9392076,-118.2194634,6,37,535607,6037535607,6037,Census Tract 5356.07,South Gate
Police Department,Gunshot,Firearm,
4343,1.2,0.6,98.8,17194,48295,55909,0.863814413,3,3,11.7,0.124827269,0.050132928
Adam Jovicic,29,Male,White,March,19,2015,364 Hiwood Ave,Munroe Falls,OH,
41.1485748,-81.4290782,39,153,530800,39153530800,39153,Census Tract 5308,Kent Police
Department,Gunshot,No,
6809,92.5,1.4,1.7,33954,68785,49669,1.384867825,5,4,1.9,0.063549832,0.403954214
Adam Reinhart,29,Male,White,March,7,2015,18th St and Palm Ln,Phoenix,AZ,
33.4693799,-112.0433197,4,13,111602,4013111602,4013,Census Tract 1116.02,Phoenix
Police Department,Gunshot,No,
4602,7,7.7,79,15523,20033,53596,0.388704381,1,1,58,0.073651452,0.102955195
Adrian Hernandez,22,Male,Hispanic/Latino,March,27,2015,4000 Union Ave,Bakersfield,CA,
35.3956975,-119.0027449,6,29,700,6029000700,6029,Census Tract 7,Bakersfield Police
```

Figure 2. Sample CSV dataset

Data Tagging

In the second step, we detect and extract all the data fields from the given dataset, and ask the user to label the type of each column as either ordinal, nominal, temporal, or quantitative. Blitz attempts to reduce the manual tagging by automatically detecting if a data field does not fit a particular data type. Tagging the data is necessary for Blitz to auto-generate visualizations and interactions.

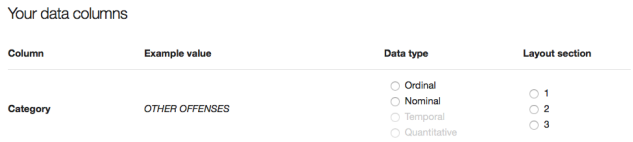

Figure 3. Tagging data types

Automating Interaction in visualizations and UI widgets

Finally, we require the user to tell us which data fields they would like to plot against each other, since there are many possibilities and the "correct" choice is simply what the user wants to do. In the future, we believe that this task could also be automated.

We break up our main visualization into three sections - a primary visualization, a secondary visualization, and a filters section, and the user can assign each data field to one of these three sections. The primary visualization should display the "main" data fields that the user would like to compare. The secondary visualization should display less important data fields that relate to the primary visualization. The filters section displays data fields that can be used to filter the content of the primary and secondary visualizations.

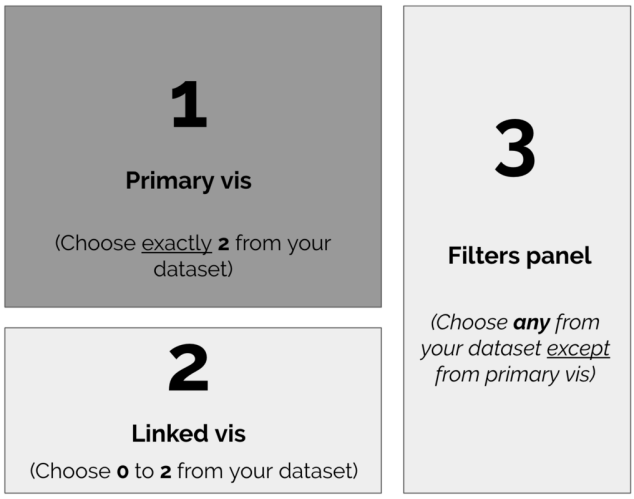

Figure 4. Our visualization layout

We have broken up our visualization in this way because it allow us to make use of a wider range of established interaction techniques. For example, cross filtering is not possible with a single visualization, so it motivated us to provide two output visualizations (primary and secondary) instead of a one. In the future, it should be possible to extend Blitz so that it can output more than just two visualizations linked together. This would lead to even more possibilities for inter-visualization interaction, but for now, two is is sufficient to demonstrate the cross-filtering interaction idea.

Primary and Secondary Visualizations

If the user assigns the data fields to the primary or secondary layout component, Blitz maps the user's tagging of data fields (i.e. ordinal, temporal, etc.) to a chart type, as defined by Mackinlay et al. [10]. For example, if the user wants to plot two quantitative axes, the most appropriate visualization would be a scatter plot [10]. Knowing the appropriate chart type to represent data fields, Blitz constructs an appropriate specification for Vega-Lite, which then constructs a visualization according Bertin's visual encodings [4] to chart type mapping. Blitz currently supports bar charts, line charts, scatter plots, and matrix plots.

Using a combination of D3 and jQuery, interactivity is grafted on top of the base visualization created with Vega-Lite. Blitz adds hovering and cross filtering (linking) to the primary and secondary visualizations. The user can hover on a data mark to reveal additional information related to that data mark. For example, hovering over a point in a scatter plot displays additional information about that point, and hovering over a bar in a bar chart reveals the number of items represented by that bar.

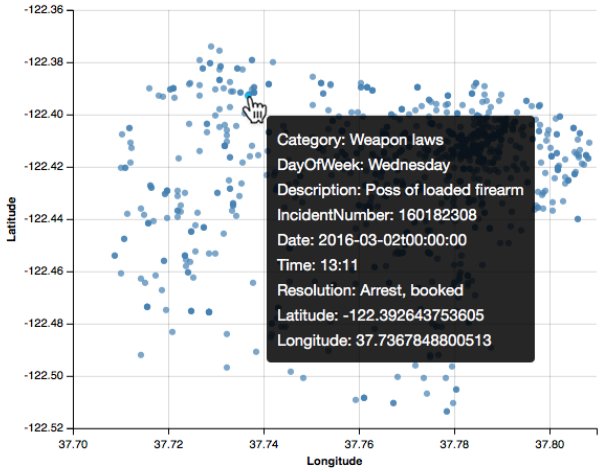

Figure 5. Hovering shows detailed information

Furthermore, the secondary visualization is "linked" to the primary visualization, in that clicking on a data mark in the secondary visualization highlights the corresponding data mark(s) in the primary visualization (See Figure 6). This enables the user to visually connect related data fields to each other even across different charts.

Originally, we planned to integrate additional interaction techniques such as pan and zoom, and region selection in a chart via click-and-drag. While such interaction is possible with Vega, the Vega-Lite API does not expose some of this underlying functionality. We attempted to workaround Vega-

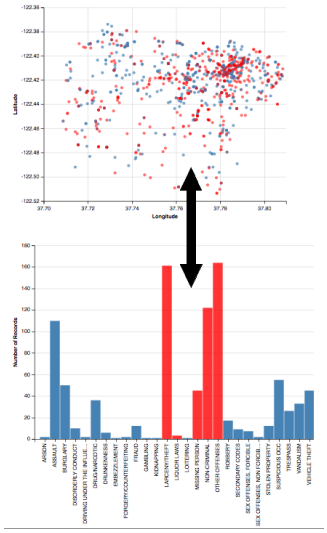

Figure 6. Cross filtering selects item in one viz. using items from another.

Lite's restrictions, but this ended up being delegated to future work because it costs too much time and effort.

Filter Panel

If the user assigns the data fields to the filter panel, Blitz takes those data fields and maps them individually to an appropriate GUI element. The following mapping is defined by Heer et al. [7]:

Temporal	Range slider
Quantitative	Range slider
Ordinal	Check box or search bar
Nominal	Check box or search bar

Figure 7. Data type to GUI mappings

These interaction widgets in Blitz are custom built to handle a variety of edge cases with data types. For example, time data is typically given as a string. Blitz first detects time formats, and then parses fields into a quantitative values like floats before mapping them to a range slider.

For the ordinal and nominal data types, check boxes work well if the amount of categories is easily enumerable. If there are hundreds of categories then some sort of keyword search for filtering would be better from a user experience perspective, but due to time constraints, this was also delegated to future work. Blitz's current solution is to bind the check boxes within a scrollable rectangular area to avoid a visual overflow of data items.

As mentioned in the introduction section, range sliders work well with continuous distributions to filter out data, however they involve limitations as well. If the the data is distributed unevenly then large portions of the slider might not filter anything out, which can confuse the user. This problem can be resolved by displaying a small histogram next to the slider that shows the data distribution, so that the user can make more informed decisions.

Another problem with sliders are unions. If a user wants to specify multiple ranges, this is currently not possible with our system. In the future we would have to add additional control mechanisms like multiple sliders to support this functionality.

RESULTS

In figures 8-10 we have some example visualizations generated by Blitz for three different datasets:

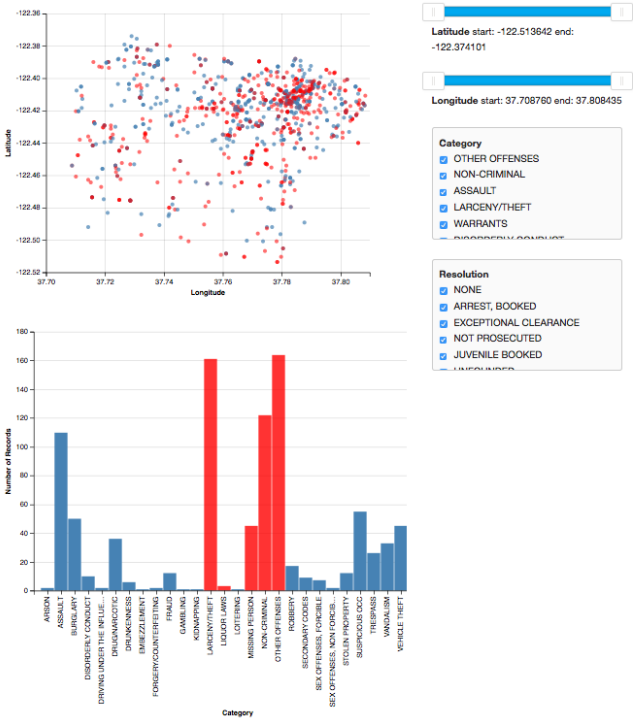

Figure 8. Police incidents in SF, 2016

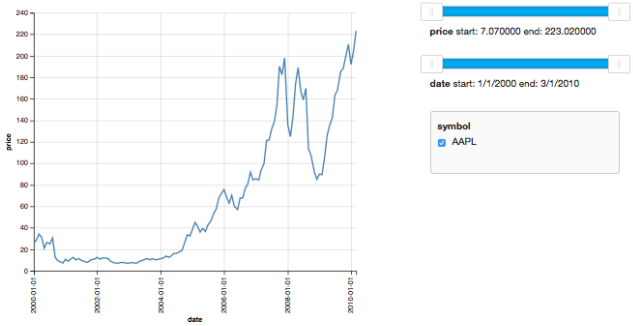

Figure 9. Apple stock, 2000-2010

Usability Tests

We presented Blitz to journalists, web designers, and developers with intermediate to professional experience with data visualization. Many people commented that Blitz fulfilled a need for them (short design cycle) and that they would be

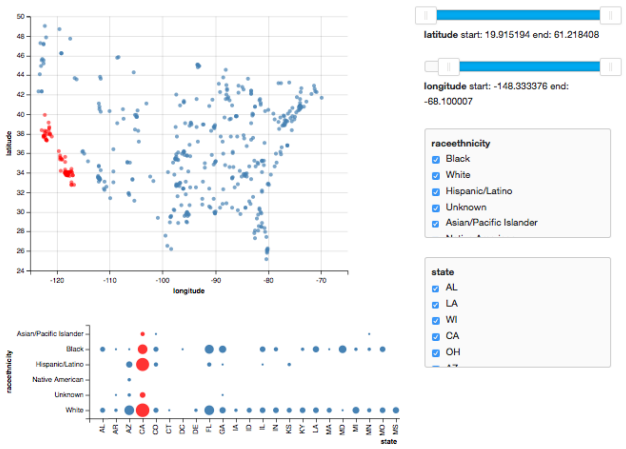

Figure 10. Police killings in U.S., 2016

interested in using it if it were to be refined into an actual product. The feedback (and critique) that we received can be broken down into the main categories of effectiveness and efficiency:

Effectiveness

We define effectiveness as how well the generated interaction paradigms fit a given data set and how useful they are for exploratory analysis.

Participants expressed that the interaction paradigms suited the datasets (Figure 8: crime in San Francisco [1] and Figure 10: US police killings in April 2015 [2]) and remarked how Blitz would help them in their own data exploration projects. For example, one participant observed that Blitz's interaction choices 'made sense intuitively'.

In addition to their positive feedback, participants had ideas about other interaction possibilities that Blitz should support. One participant suggested accompanying filter widgets – like range sliders and checkboxes – with histograms to display the underlying distribution. Showing the distribution would help users decide what to filter. Another participant wished Blitz could support multiple linked views, which would especially handy for large datasets with multiple data fields. A different participant wanted to be able to visualize and interact with multiple datasets at once; currently Blitz would require users to first merge the datasets into a single input.

Other recommendations included being able to rescale the axes of the primary and secondary visualizations with sliders. A user commented that doing so is unintuitive in the current system. Also it was mentioned that Blitz could better filter out bars in bar charts. To date, when filtering a bar chart, Blitz completely removes the bar and its associated axis label from the bar chart as if that data item never existed. Instead, Blitz should zero out the bar, but leave the associated label visible.

Efficiency

We define efficiency as quickness in development time or even no development time.

Using Blitz, participants were able to generate an interactive visualization within five minutes. Participants commented that it was much easier and faster to generate an interactive visualization with our system than with Tableau. The UI was simple and intuitive, and it was clear to them how to configure their dataset to match their needs. As a result, they were not bogged down by the process of creating a visualization and able to jump quickly to the subsequent process of analyzing and understanding a dataset (with the visualization).

DISCUSSION

Similar to how Bertin [4] presents a mapping from data types to visual elements, we explore the possibility of mapping data types to interaction paradigms. We discovered two insights that will aid in further development of not only Blitz, but other automated interaction system as well. The first insight is that certain types of data have a clear mapping to interaction paradigms. Secondly, the basic set of tools for exploratory data analysis often remain consistent between data sets. Actions such as filtering, selecting, hovering, are not particular to one data set. With these two insights we were able to prototype a system that algorithmically produces both visualizations and interactivity. A user should be able to rapidly explore data sets, without having to repeatedly produce the same boilerplate that is typically needed in interactive visualization.

Some people may question how Blitz differs from Tableau. To them, our response is that Tableau does not provide the immediate interaction advantages of Blitz. Although Tableau can provide a rapid visualization, changing visualization parameters requires reloading the visualization altogether, using SQL queries or other relatively unintuitive menu interactions. Our system both provides the visualization and interaction simultaneously, keeping them as closely linked as possible. Although there is some support for interactivity in Tableau, the process to adding interactivity is not that easy - standard actions like filtering should really be baked in without requiring extra effort from the user.

FUTURE WORK

Building a robust visualization generator is not a trivial task. Blitz is best considered as a prototype rather than a fully-featured product. As already mentioned in the methodology section, Blitz can be extended to support a more comprehensive set of interactions. Additionally, Blitz can implement the feedback given from usability studies, particularly multiple linked visualizations and histogram distributions accompanying filter widgets (sliders). Finally, there is the practical aspect of making Blitz more error resistant - for example, being able to handle badly formatted data more gracefully.

In the future, we envision the design process being entirely automated. For example, it should be possible to perform natural language processing (NLP) on a data columns name to extract is meaning, and use the meaning to deduce the

the appropriate data type tag (ordinal, nominal, etc.) for that column. With a better level of understanding of a dataset, it should even be possible to perform machine learning to figure out not only the correct way to plot the data, but which data columns are most salient and most important to visualize. Then we could replace our restrictive 3-part layout with one that has been designed expressly for the needs of a given dataset.

CONCLUSION

In summary, most existing visualization tools lie at an undesirable spot on the interactivity/productivity spectrum. One end is limited interactivity with fast development, and the other end is robust interactivity and slow development. We presented a new system, Blitz, which tries to find a better trade off between interactivity and development time by automating the process of interaction design, using mappings from the visualized data types to common paradigms. Feedback from users show that there is a real need for a product like Blitz and that further work on systems for generating interactive visualizations could benefit many people.

REFERENCES

1. Crime data for San Francisco, May 2016. `http://web.stanford.edu/class/cs448b/cgi-bin/wiki/images/0/0c/Scpd_Incidents.json.zip`.
2. Police killings in US (2015), March 2016. `https://github.com/fivethirtyeight/data/blob/master/police-killings/police_killings.csv`.
3. Vega-lite: A high-level Visualization grammar, May 2016. `https://vega.github.io/vega-lite/`.
4. J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.
5. J. Choi, D. Park, Y. Wong, E. Fisher, and N. Elmqvist. Visdock: A toolkit for cross-cutting interactions in visualization. In *IEEE Transactions on Visualization and Computer Graphics*, pages 1087–1100. IEEE, 2015.
6. J. Heer and M. Agrawala. Generalized selection via interactive query relaxation. In *Proceedings ACM CHI*, pages 959–968. ACM, 2008.
7. J. Heer, B. Shneiderman, and C. Park. A taxonomy of tools that support the fluent and flexible use of visualizations. In *ACM Queue 10(2)*, pages 1–26. ACM, 2012.
8. Y. JS, Y. Kang, S. J, and J. J. Toward a deeper understanding of the role of interaction in information visualization. In *IEEE Transactions on Visualization and Computer Graphics 2007; 13*, pages 1224–1231. IEEE, 2015.
9. J. Mackinlay. Automating the design of graphical presentations of relational information. In *Acm Transactions On Graphics (Tog),5(2)*, pages 110–141. ACM, 1986.
10. J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. In *IEEE TVCG, 13*, pages 137–1144. ACM, Nov/Dec 2007.
11. A. Satyanarayan, K. Wongsuphasawat, and J. Heer. Declarative interaction design for data visualization. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 669–678. ACM, 2014.