Reimagined-Vis: Project Part 3

Kathryn (Katy) Koenig **Lucy Li** **Belen Michel Torino**

PROJECT DESCRIPTION

Most data visualizations are not created using best practices for accessibility. Specifically, while there are tools to assist low vision viewers in understanding plots, these tools put the burden of translating plots more accessible visualizations on the low vision consumer instead of the data visualization designer. We seek to standardized the creation of data visualizations that are accessible for low vision users, resulting in plots that are the same for viewers with and without visual impairments.

To foster this new status quo, we have created v_a11y_lint: a lint software that flags issues regarding accessibility in data visualizations. Our software is to be used by programmers to facilitate easier created of accessible charts.

REQUIREMENTS SUMMARY

Our most important goal is allow people with low vision to be able to glean more information from data visualizations than they previously could. With respect to our prototype, we hope that low vision viewers can internalize more information from a plot created incorporating v_a11y_lint and it's accessibility flags than a plot created without our accessibility lint software. We also want low vision viewers to understand this information more quickly and using fewer additional add-on tools than previously needed.

We chose to move forward with our accessible data visualization linter prototype as it allows people with low vision to have the same experience as viewers without visual impairments: low vision viewers do not have to download an additional application or buy additional tools to understand charts created with v_a11y_lint. Instead, the data visualization designer should address accessibility issues directly, thus avoiding any degradation of experience for low vision viewers.

As many of the participants in our study reflected the desire more self-reliance in interpreting data visualizations, charts made with v_a11y_lint should not require mediation through a person without a visual impairment for understanding and will therefore allow low vision viewers to feel more empowered and self-reliant.

From the result of our user study, we will address issues regarding data visualization accessibility through integrating the following best practices and guidelines:

- Increasing text font size.
- Use of white space to further distinguish between aspects of plot.
- Dual encodings of data (pattern and color) for clarity regarding similar colors.
- Increasing contrast between text color and background color.
- Adding descriptive titles for each plot.
- Increasing threshold for perceptible differences between colors in a color scheme.

Finally, as we want to ensure use our software by programmers, it is important the v_a11y_lint is easy to use and incorporate into the workflow of those creating data visualizations.

PROTOTYPE DESCRIPTION

Overview

Looking at different options for our prototype, we decided to go with creating something that would allow creators of data visualizations to make their visualizations accessible while still giving them the ultimate decision on how they want to present their data. In order to do this, we created a linter for Altair-create plots in Python called v_a11y_lint. Lint software reviews code for bugs, errors and stylistic issues has been and notifies the user aspects of their code should be fixed. Our linter looks through code for an Altair chart object and notifies the user of what aspects of their data are not accessible by outputting a JSON-like object. Because most of interviewees in our user study experience low vision, v_a11y_lint specifically flags issues associated with accessibility for low vision users, including font size, font to background contrast, data to background contrast, etc.

Description

Our accessible visualization lint program v_a11y_lint flags accessibility issues for charts created using Python's visualization library Altair. To create a plot in Altair, a programmer can either set a theme and then create their plot from a pandas dataframe, as show in Figure 1, or use the default theme for Altair and create a visualization in the same manner. In the first panel of Figure 1, our editor of choice is open, specifying our theme, which is then called and created in the second panel. The third panel of Figure 1 show the code for the creation of our data and bar chart object, which is then called and rendered in Jupyter Notebook in the subsequent panels.

In Figure 2, we call our lint software's main function, run_lint with the chart object as the argument. In this figure, we also see the output of our run_lint which is a dictionary (a JSON-like type in Python) of any accessibility issue. Note issues are divided by issue type. In our example, we see the issues with this bar chart are broken into two categories: font and color.

Figure 1: Initial Bar Chart Created

In Figure 3, we see updates to our theme noted in green to address the accessibility flags that were outputted by v_a11y_lint. In this figure, we also see that our plot is updated and provide

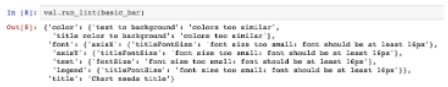

Figure 2: Linting with v_a11y_lint

Figure 3: Theme Updates Made Highlighted in Green

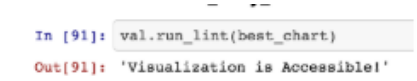

Figure 4: No Accessibility Issues Found

an image of our updated chart. We then call v_a11y_lint's run_lint on our updated chart in figure 4. We see that there are no additional issues that need to be addressed. The output is then saved and used as any other data visualization would be used.

In its current state, v_a11y_lint is available at https://github.com/katykoenig/reimagined-vis/tree/master/v_a11y_lint. This is source code created in Python and is not hard coded

for a specific visualization. Therefore, it should run on all one layer chart objects created in Altair. While Altair chart objects do not render when called in ipython3 directly (so it is preferred to use a notebook, like Jupyter or Google Colab, when creating the visualization), a user can use v_a11y_lint to lint the chart object in ipython3 or in a notebook. Currently, the output of the accessible linting processes is JSON-like object, but we are working to create a more easily interpret able output.

Regarding flags for v_a11y_lint: we use WCAG guidelines for accessible webpages[1] to check for accessible contrast between text and backgrounds, using the RGB color space, and minimum accessible font size. We incorporate the idea of "just noticeable differences" (JND) in color[2] to check if color palettes varying in perceivable ways using the CIEL*a*b* color space. Because the JND threshold is found through testing with people without visual impairments, we double this threshold as v_a11y_lint should assist in creating data visualizations that are accessible for people with low vision. Our lint software also flags issues regarding lack of title and length of title as during our user study, many of our interviewees noted a desire for a description to accompany each data visualization.

Scenario

Alex, a data scientist, wants to create data visualization to show the results of their research regarding population change in Illinois and the Chicagoland area over time using their preferred programming language, Python. They choose to use Python's Altair visualization library to create a stacked bar chart as they are most familiar creating data visualizations using the grammar of graphics [5]. Alex quickly creates a stacked bar chart with year on the x-axis and population in on the y-axis.

Alex understands that this visualization may not be accessible, even with magnification, to people with low vision and decides to to use a new linting software v_a11y_lint. Alex downloads the software and imports it into the Jupyter Notebook they are using to create the visualization. Alex then calls the v_a11y_lint function, "run_lint" with their Altair chart object as the argument. In the output table, Alex sees that their chart their chart is not fully accessible to people with low vision: the chart title is not descriptive, the font size for the axes is too small and the color of the bars in the chart is too similar to the background color. Alex takes a couple minutes to update the chart's theme to include larger fonts sizes and change the fill color of the bars. They then reset the chart object's title to include more information regarding the information in the plot. Alex then reruns v_a11y_lint's "run_lint" with the updated chart object, and the program tells them that the chart is now accessible.

Alex saves the chart with a high dpi value and now uses this chart in their paper and subsequent Medium.com post regarding population change in Illinois and Chicago. All visitors see the same chart and viewers with low vision comment on how useful and easily understandable this chart feels in comparison with other plots regarding the subject.

Rationale

While we were excited by our three prototype ideas of data sonification, an enhanced visualization mobile application and our accessible visualization lint software, ultimately, we proceed with the third option. We chose to create the accessible visualization linter as we believe it is a more concrete contribution towards the normalization of accessible visualization: with our coding skills, we are able to truly create the product instead of conducting Wizard of Oz prototyping. At the time of writing, we are already linting any given chart's global configuration, or theme, for title presence, title length, font size, font to background contrast, fill color to contrast and ensuring perceptible differences[2] between all colors in color schemes. We hope to add more plot-specific features before the final due date, e.g. checking white space between bars in a bar chart. As our software is publicly available on GitHub, anyone could use our software to make their visualization more accessible. Therefore, our v_a11y_lint can truly make an impact as opposed to our other prototypes of which, due to our skills and resources, we would be unable to create a high-fidelity prototypes.

With v_a11y_lint, all viewers see the same accessible visualization. Low vision viewers are provided with the exact same experience as viewers with no visual impairments. This is essential for two reasons:

1.) There is no degradation of information for people with low vision nor does a low vision viewer need to use additional tools to be able to glean the same information from the visualization as user with no visual impairments. While other systems create a separate product for people with visual impairments to use, e.g. data sonification, or make people with visual impairments add yet another tool to their already quite large toolkit, e.g. the enhanced visualization mobile application, v_a11y_lint puts the onus on the data visualization designer and provides a low vision viewer the freedom of being just a viewer.

2.) Our linting software creates a standardization of accessible data visualizations. Currently, it is not the norm to incorporate accessibility into data visualizations, and for programmers that wish to do so, there is no easy way to check for accessibility. V_a11y_lint provides a simple method to ensure accessibility, therefore, encouraging designers to make all visualizations more accessible.

Conversely, as v_a11y_lint forces the data visualization designer to incorporate any suggested changes for making a data visualization more accessible, the linter is only useful if programmers actually use it. Therefore, we cannot ensure that any data visualizations will be made more accessible through v_a11y_lint. Additionally, while there are some guidelines to make web pages more accessible [1], there are no available guidelines explicitly regarding data visualization accessibility and low vision viewers. We use the Web Content Accessibility Guidelines [1] where applicable as well as commonly disseminated recommendations available online [3, 4] to create v_a11y_lint's rules. We recognize the limitations of our rules and hope that future research and users regarding best practices from accessible data visualization as completed, which we can then incorporate into v_a11y_lint.

RESPONSE TO IN-CLASS EVALUATIONS

Discussion of Successful Aspects

Throughout user testing, many users told us they liked the design and goal of our prototype. Furthermore, the output of running our program clearly showed issues regarding accessibility in a given data visualization. The result also made clear which aspects of the data visualizations needed to be changed.

It was also a fairly high-functioning prototype for the time restraints we had, allowing us to explore more detailed and realistic options and changes to be made in the future. Additionally, the lack of hard-coding or Wizard of Oz testing allowed our prototype testers to apply our program to a variety of visualizations to further check functionality.

Discussion of Issues

Through our user testing, we were able to reveal a few issues that needed to be resolved. While our prototype in its current state is relatively high-fidelity as it is able to lint most Altair chart objects, due to time constraints, we have only incorporated linting of font and color issues. Therefore, there are many other accessibility flags that need to be added for a more useful and relevant product. However, at the time of testing, we were able to have it mostly working.

Throughout testing, it was apparent that there were issue in the output style. As mentioned previously, our linter currently outputs a JSON-like object to note issues in accessibility. While we originally thought this would be easily understand because it is a common programming output, this style resulted in readability issues that could have been avoided using a different output format for issues. We also noticed a small bug in our code for linting for text to background contrast that must be addressed in future iterations of v_a11y_lint.

Furthermore, our current prototype currently searches the global theme set for visualizations instead of only the chart-specific issues. This had to be explained and caused some confusion as issues that were flagged were not always relevant to the chart that was created, e.g. v_a11y_lint states that the legend font is too small but the chart that was linted does not have nor needs a legend.

Additionally, while our Juptyer Notebook offered exploration via the Altair examples page, because our lint software is Python and Altair-specific, we had to spend a fair amount of time explaining how to build a chart and the inner-workings of Altair prior to demonstrating our prototype.

Discussion of Tools & Materials

For programming languages, as a group, we are most familiar with Python. Therefore, we programmed v_a11y_lint using Python. As there are low barriers to entry to learning Python (as compared to other programming languages) and Python was the third most popular programming language last year according to GitHub, demonstrating the potentially large impact of our program. Conversely, many data visualizations are created using JavaScript or more GUI-friendly software like Tableau, for which v_a11y_lint is incompatible.

We chose to lint chart objects created in specifically in Python's Altair library. Altair is a declarative visualization library whose code is automatically generated from Vega-Lite[1]. Because Altair implements Wilkinson's grammar of graphics [5], v_a11y_lint provides a foundation for linting for visualization accessibility that can be extrapolated into other grammar of graphics visualization libraries including but not limited to R's ggplot2, Vega-lite and Vega.

Conversely, we must note the following limitations in our choice of linting Altair chart objects:

1.) Altair is not Python's most commonly used visualization library and that Python's Matplotlib (and its wrapper library, Seaborn) visualization library does not follow a grammar of graphics. Therefore, v_a11y_lint cannot be used to flag accessibility issues for low vision users in matplotlib figure object.

2.) Because Altair is automatically generated from Vega-lite code, and both libraries are relatively new, there appear to be some issues that have been fixed for other libraries. Most relevantly for our research, we found that charts become more accessible when data is encoding in more than just color, i.e. bars in bar chart that would be distinguished using different colors are also differentiated using different patterns to fill each bar. Currently, Vega-Lite, and therefore Altair, do not offer this feature. We have therefore filed an issues on the Vega-Lite GitHub page to add this feature and will add flags for dual encodings to v_a11y_lint subsequently.

Future Changes

Through our user testing, we got many useful suggestions for improvements and additions. Many users recommended changing the format of the output for clarity. One user specifically recommended a verbose option when running the software that could allow the user to have more detailed results and recommendations for fixing accessibility issues if they preferred. In the future, we hope to test with different options for the output such as tables, multiple columns, and verbose options, in order to make the results more digestible.

Another suggestion multiple users had was to have an option to detect for color-blindness. Originally, we wanted our prototype to focus on use for low-vision users, but with our current prototype, we hope to fairly easily add a flag for detecting colorblind accessibility of visualizations allowing us to maximize the use for the prototype we design.

Incorporating both of the comments above allows for more flexibility for a user. It was also suggested that the lint software be customizable. While future iterations of the software may allow for users to modify flags based on personal preferences, we may be unable to accommodate this request before the end of the course.

As previously mentioned, our current prototype flags for issues regarding accessibility for the entire theme set in Altair, showing all accessibility problems of a theme even if they are irrelevant to a specific chart. We want to refine our flags

[1] Vega-Lite is a high-level grammar of interactive graphics. It was created and is maintained by the University of Washington's Interactive Data Lab as the high-level counterpart to Vega.

to only show relevant issues as to avoid confusion and overwhelming our users as well and hope to complete this task prior to submission of our final prototype.

REFERENCES

[1] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, and Gregg Vanderheiden. 2008. Web content accessibility guidelines (WCAG) 2.0. *WWW Consortium (W3C)* (2008).

[2] Connor C Gramazio, David H Laidlaw, and Karen B Schloss. 2016. Colorgorical: Creating discriminable and preferable color palettes for information visualization. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 521–530.

[3] Guidea. 2019. Enterprise Data Viz: Rules and Best Practices. `http://www.storytellingwithdata.com/blog/2018/6/26/accessible-data-viz-is-better-data-viz`. (2019). Accessed: 2020-02-16.

[4] Cole Nussbaumer Knaflic. 2018. Accessible Data Viz is Better Data Viz. `http://www.storytellingwithdata.com/blog/2018/6/26/accessible-data-viz-is-better-data-viz`. (2018). Accessed: 2020-02-16.

[5] Leland Wilkinson. 2012. The grammar of graphics. In *Handbook of Computational Statistics*. Springer, 375–414.