VizLinter: A Linter and Fixer Framework for Data Visualization

Even though automated visualization tools are in market demand, it is not uncommon for current systems to deliver outputs that do not necessarily match the input data or satisfy visualization requirements. In most real-world use cases more specification modifications are still needed. Since most users lack the ability or visualization expertise, manual modifications are challenging. In this paper, the authors offer a framework called VizLinter with the objective to assist users identify visualization errors and modify them to their correct and desirable outputs. The framework is made up of two parts: a visualisation linter that checks the legitimacy of immediate visuals using established principles, and a visualisation fixer that automatically fixes any violations that are found.

Some previous studies take the visualization recommendation system approach where it automatically generates and recommends visualization for a dataset input. For example, Abbas et al. presented ClustMe which ranks cluster patterns of monochrome scatterplots based on perceptual image quality measurement [5]. A novel directed-graph model for visualization depending on char similarity by Kim *et al.* [4]. On the contrary, VizLinter leaves the choice to the users; to choose or not to choose recommended adjustment and have more instructive method.

Moreover, data preparation for optimized visualization since data quality plays a major role on the quality of visualization. For example, Ellis et al. researched on clutter reduction using sampling techniques [2]. Byron et al. optimized the overall shapes of stacked graphs [3]. However, these previous works are not able to optimize inaccurate visualizations with violated design rules. Does not address how violations prevent graphs from compiling. VizLinter's aim is to present an ideal framework for common solutions to various types of visualizations and rectify design errors.

A VizLinter prototype is created as an online Vega-Lite editor to demonstrate how developers can utilise VizLinter to produce flawless visualisation. A prompt panel for the linting and fixing operations of VizLinter in addition to the basic code editing panel and the chart display panel. "Inspect Specs" and "Suggest Revision" buttons are used to display any rules of the present visualisation specification that are broken, and users can examine suggested actions or make their own corrections to the specification in accordance with the descriptions of the broken rules respectively. The "Preview" button allows users to preview the alternative output. The code panel will display the corresponding updated specification. Users can accept then apply using the "Accept" and "Reject" buttons.

An in-lab user study to assess VizLinter's advantages and efficiency was also carried out. The outcomes demonstrated that VizLinter could assist users in locating and correcting problems. It consisted of twenty people with prior visualizing expertise. Data for participant modifications, acceptance decisions for each VizLinter proposal, completion times for each question, and the percentage of errors were all collected. The average time it took the participants to identify and correct faults in each visualisation was 97.7 seconds (SD = 28.1). The overall questions' average correction was 77%. And the average acceptance rate was 90% (SD=14.2%), meaning that most of the recommendations were preferred.

VizLinter can eventually embrace other well-known visualisation grammars like D3.js, ggplot2, and Matplotlib. The authors also intend to apply the present framework in a wider range of application scenarios and to expand the framework's rule coverage, system configurations, and user interactions. VizLinter's further goal is to show the faults and the connections between the rules, codes, and graphics.

References

[1] Q. Chen, F. Sun, X. Xu, Z. Chen, J. Wang and N. Cao, "VizLinter: A Linter and Fixer Framework for Data Visualization," in IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 206-216, Jan. 2022, doi: 10.1109/TVCG.2021.3114804.

[2] G. Ellis and A. Dix, "Enabling Automatic Clutter Reduction in Parallel Coordinate Plots," in IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 5, pp. 717-724, Sept.-Oct. 2006, doi: 10.1109/TVCG.2006.138.

[3] L. Byron and M. Wattenberg, "Stacked Graphs – Geometry & Aesthetics," in IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 6, pp. 1245-1252, Nov.-Dec. 2008, doi: 10.1109/TVCG.2008.166.

[4] Y. Kim, K. Wongsuphasawat, J. Hullman and J. Heer, "GraphScape: A Model for Automated Reasoning about Visualization Similarity and Sequencing", *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems CHI'17*, pp. 2628-2638, May 2017.

[5] M. M. Abbas, M. Aupetit, M. Sedlmair and H. Bensmail, "ClustMe: A Visual Quality Measure for Ranking Monochrome Scatterplots based on Cluster Patterns", *Computer Graphics Forum*, vol. 38, no. 3, pp. 225-236, 2019.

Report by:
Tahsin Saira
2152766
MSc, Data Science