Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows

Pin-Xuan Chen
基蛋所碩二
no ChatGPT or other AI Tool use

1 PROBLEM

In data science, researchers spend most of their time in data cleaning and visualizing, a process called EDA (exploratory data analysis). Nowadays, data scientists usually utilize dataframe libraries in Python, such as pandas, matplotlib, and ggplot, to achieve the goal. According to a survey from Kaggle, nearly 75% of data scientists specifically use pandas, which support higher API levels and manipulate through Jupyter Notebook efficiently [1]. A good visualization can help validate whether data sets have been operated desirably or are suitable for further analysis. However, two main issues confront during the visualizing process:

(1) **Cumbersome boilerplate code**. Because the data visualizing step is performed routinely, some specification codes are highly-templatized and utilized in a repeated process of copy-and-pasted across the notebook [2,3]. Besides, users tend to apply particular visualization code in the last step directly, which may cause data experimentation loss from other aspects.

(2) **Challenges in determining the next steps.** Although many visualization codes are available, users may need clarification to extract a valuable combination of attributes from the dataframe for further analysis. It is better to have automated assistance and user recommendations to advise further steps for data scientists.

PVLDB Reference Format:
Doris Jung-Lin Lee, Dixin Tang, Kunal Agarwal, Thyne Boonmark, Caitlyn Chen, Jake Kang, Ujjaini Mukhopadhyay, Jerry Song, Micah Yong, Marti A. Hearst, Aditya G. Parameswaran. Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows. PVLDB, 15(3): 727 - 738, 2022. doi:10.14778/3494124.3494151

2 PRIOR WORK

They summarized the pros and cons between previously published libraries/database management system and their developing tool Lux. They discuss the following three categories: data visualization, specification, and exploration.

https://doi.org/10.1145/1234567890
Proceedings of the VLDB Endowment, Vol. 15, No. 3 ISSN 2150-8097.
doi:10.14778/3494124.3494151

2.1 Visualization Recommendation

If users program the visualization code from scratch, they may confront a series of sub-setting attributes, mapping across multi-dataframes and then plotting graphs. Therefore, some open-source or commercial databases, such as Tabular [4], offer an easy-to-use interface for visualization construction. However, GUI-based tools are inconvenient for data scientists due to the need for customized programming queries and advanced integration with other related workflows [5]. Lux combines the advantages of auto-exploration in GUI-based tools and manual programming interaction for users.

2.2 Visualization Specification

During the visualizing process, different types of library design affect the degree to which users are required to specify low-level details related to visualization definition. *Imperative* visualization such as matplotlib [6] requires sophisticated computation for the data manually. In comparison, *declarative* visualization such as Altair [7] accelerates the visualizing efficiency by applying defaults to lower-level visualization details (ex., legend, axis). In between, a *partial* specification such as *CompassQL* [8] retains the user-defined specification but increases the coding burden for users on the other hand. In contrast, Lux utilizes *intent* language that only requires users to specify data aspects of interest without advanced visualization encodings.

2.3 Visualization Visual Data Exploration with Dataframe

Data exploration needs substantial programming follow-up and analytical know-how. Although some GUI-based tools, such as pandasgui [9], offer data transformation and summarized statistical output, Lux can offer visualization and Recommendation at all times.

3 SOLUTION

They introduce Lux, an extension to the pandas' package, to make data visualization automatically at any time point of the workflow. They constructed Lux based on the following advantages:

(1) Adopting an always-on-approach. In an existing workflow, users use the data and view table by default (Figure 1). To create other plots, they need to specify code to generate optimized visualizations. Conveniently, Lux generates a dashboard as a part of a multi-tiered framework driven by user-specific intent. Other

framework includes visualizations showing plots and action, which calculates the correlation coefficient for the plot. Users can change intent through this framework and get diversely recommended visualization outputs. (Figure 1).

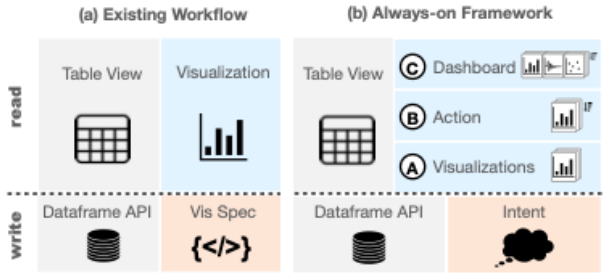

Figure 1: Conceptual framework for dataframe interaction.

(2) Use of the *intent* language. When the user specifies codes to analyze the dataframe, Lux will grep them as an intent. The intent comprises *clauses*, and each is decomposed into the *Axis* (the row) or *Filter* (the column) of interest. Next, Lux can use the *Vis* and *VisList* keyword to directly generate specific visualizations instead of attaching an intent to a dataframe. Finally, the intent is versatile and efficient in interpreting queries and thus reduces the specification given by users. The operation of each plot in Lux is listed in the table1.

Vis Type	Relational Operation
Scatterplot	Selection on 2 columns
Color Scatterplot	Selection on 3 columns
Line/Bar	Group-By Aggregation
Colored Line/Bar	2D Group-By Aggregation
Histogram	Bin + Count
Heatmap	2D Bin + Count
Color Heatmap	2D Bin + Count + Group-By Aggregation

Table 1: Table summarizing the relational operations performed for processing different visualizations.

(3) Implementing a novel recommendation system. Lux gives suggestions based on the data structure and the historical analysis of the same database.

4 RESULT

As for the performance evaluation, they select two large datasets (Airbnb and Communities) as input and measure the average time it takes to (1) execute every cell in the notebook and (2) print a dataframe or series in each cell across five conditions: one using **pandas** without Lux**, no-opt** (naïve implementation of Lux), and three optimization groups (**wflow**, **wflow+prune**, and **all-opt**). In the overall runtime, Lux with all optimizations speed-up up to 11X in overall runtime for the Airbnb dataset (and up to 345X for Communities) compared to the no-opt group (Figure 2). In addition, Lux with optimization can also reduce the overheads and accelerate the processing efficiently.

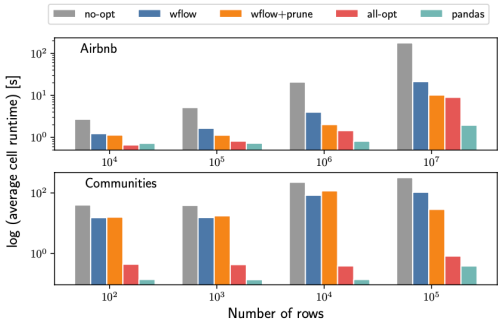

Figure 2: Average runtime of a notebook cell across the workload for different dataframe sizes and conditions.

5 CRITIQUE

Overall, Lux can help users initially explore data and establish the database know-how from the default visualizations. In addition, the recruitment of intent language can reduce the programming debugging and specification ability for the user, and simply the following extraction steps. Finally, the recommendation system gives directions for the following steps. Among these characteristics, intent language is an excellent concept similar to the Chatbot or AI tools. The visualization process includes multiple glue codes that only need to replace the input data and specific attributes. The intent method simplifies the code and outputs the same demand. However, I am worried about the diversity of the visualization type systemically. Besides simple plots, can Lux update the available plot type by users. Compared to other visualization libraries such as sweetviz [10] or autoviz [11], their runtime efficiency may differ by the implementation. They can be helpful in their simplified input code requirements and additional recommendation functions. In conclusion, Lux is applicable before the analytical process and

6 POSSIBLE EXTENSION

We can extend the bridge between the Lux and sequential analytical workflow. For example, Lux can extend to assign report function if the user wants all of the visualizations outputted in a pdf format. In addition, Lux can build an extraction function if they find some outliers or specific tuples from the current visualization results. Finally, users may

CITATION

[1] State of Data Science and Machine Learning 2020. Kaggle, 2020.
[2] S. Alspaugh, N. Zokaei, A. Liu, C. Jin, and M. A. Hearst. Futzing and moseying: Interviews with professional data analysts on exploration practices. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):22–31, 2019.
[3] A. P. Koenzen, N. A. Ernst, and M. A. D. Storey. Code duplication and reuse in jupyter notebooks. In 2020 *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–9, 2020.
[4] Tableau. https://www.tableau.com/, 2019. Accessed: 2019-09-11.
[5] S. Sarawagi. User-adaptive exploration of multidimensional data. *Proc of the 26th Intl Conference on Very Large*, pages 307–316, 2000
[6]. J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
[7]. J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert. Altair: Interactive sta-

tistical visualizations for Python. *Journal of Open Source Software*, 3(32):1057, 2018.
[8]. K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards a general-purpose query language for visualization recommendation. *In Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 4. ACM, 2016.
[9]. adamerose. PandasGUI. https://github.com/adamerose/pandasgui.
[10]. Fbdesignpro. sweetviz. https://github.com/fbdesignpro/sweetviz.
[11] AutoViML. Autoviz. https://github.com/AutoViML/AutoViz