How Novices Use Program Visualizations to Understand Code that Manipulates Data Tables

Ylesia Wu*
xw001@ucsd.edu
University of California San Diego
La Jolla, CA, USA

Qirui Zheng*
q7zheng@ucsd.edu
University of California San Diego
La Jolla, CA, USA

Sam Lau
lau@ucsd.edu
University of California San Diego
La Jolla, CA, USA

Abstract

As data science and artificial intelligence continue to impact society, more and more people are learning how to manipulate data with code. To support these learners, program visualization tools automatically generate diagrams to show how code transforms data, in contrast to tools based on large language models (LLMs) that primarily focus on textual explanations. Although program visualization tools are popular among instructors, do novices find these tools usable and useful for data science programs that often manipulate datasets with many rows? To address this, we evaluate a popular, publicly available tool that generates diagrams for Python `pandas` code through a randomized, in-lab usability study with 17 data science novices. Despite minimal instruction on how to use the tool, novices found that program visualizations increased their confidence in comprehending and debugging code. In addition, even though the tool sometimes produced diagrams with many visual elements, participant performance on the study tasks was not negatively impacted. These findings suggest design guidelines for program visualization tools to help manage cognitive load for data science novices. To our knowledge, this is the first empirical study that investigates how novices use program visualization tools to understand code that manipulates data tables, and suggests a future where novices can use automatically generated diagrams as a complement to LLM tools for effectively understanding unfamiliar programs in data science.

CCS Concepts

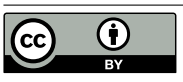
• **Human-centered computing → Empirical studies in HCI**.

Keywords

data science education, program visualization tools, novice programmers

ACM Reference Format:
Ylesia Wu, Qirui Zheng, and Sam Lau. 2025. How Novices Use Program Visualizations to Understand Code that Manipulates Data Tables. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641554.3701959

*Ylesia Wu and Qirui Zheng contributed equally to this work as co-first authors.



SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-0531-1/25/02
https://doi.org/10.1145/3641554.3701959

1 Introduction

As data science programs become more common, instructors and researchers are developing a deeper understanding of the differences between introductory data science and introductory computer science (CS1) curricula. One noticeable difference is that unlike many CS1 courses, introductory data science courses teach students to use software libraries like the `pandas` library for Python to work with datasets containing thousands and even millions of rows [27]. Although many data science courses introduce basic programming skills like loops and conditionals, they typically spend much more time covering data manipulation using `pandas` dataframes, which represent tables of data [1]. Unlike in CS1 where primitive operations typically transform single values, common operations on dataframes like sorting, grouping, and aggregating often transform multiple data values at once. In cases like making a pivot table, a single operation can sometimes even transform the entire dataframe structure itself. Because of this, data science instructors face the challenge of teaching novices how complicated software libraries manipulate datasets that are far too large to display in entirety.

To support introductory data science courses, program visualization tools can automatically generate explanatory diagrams of data table transformations, following a long history of program visualization tools for introductory programming courses [42]. These tools are popular among both instructors and learners. For example, the Pandas Tutor tool is a publicly available program visualization tool for data science and it is already used by 20,000 people across 160 countries per year [22]. While these tools have been designed to support instructors who teach introductory data science courses, there is a lack of understanding about whether data science novices are actually able to use these tools. This leads to the overarching research question for this paper: **How do data science novices use program visualization tools like Pandas Tutor to understand code that transforms data tables?**

To address this question, we conducted a within-subject, think-aloud study with 17 data science novices. Participants were presented with a series of `pandas` snippets demonstrating common data manipulation tasks in a computational notebook and were asked to explain what each snippet did. Each participant completed tasks using both the Pandas Tutor tool and a baseline condition with a standard Jupyter notebook environment. Participants were more confident in their responses using the Pandas Tutor tool without decrease in task accuracy or time to completion. Interviews with our participants revealed that novices found the tool usable and useful for verifying their understanding of code that transforms data tables. Participants also highlighted features of the tool that enabled them to more easily reason about large datasets and manage the additional cognitive load of interpreting diagrams about

code. From these findings, we derive design recommendations for program visualization tools that support data science learners: tools should be embedded within learner workflows; tools should reduce the effort required for novices to compare intermediate and final states of data tables; and tools should reduce cognitive load through interactions that allow users to selectively filter visual marks. In sum, this work contributes:

- The first empirical evaluation of a program visualization tool for data science novices (to our knowledge).
- Insights into how data science novices can use program visualization tools like Pandas Tutor to understand code that transforms data tables.
- Design recommendations for future program visualization tools that support data science learners.

2 Related Work

To meet the ever-increasing demand for studying resources in computer education, especially for beginning programmers, visualization tools have been developed to serve as a complement to traditional classroom teaching or self-studying of computer programming. These tools tackle different aspects of computer education and serve the purposes of instruction, illustration, explanation, clarification, or debugging [42].

Many visualization tools intended to facilitate the learning of general programming are designed in the format of a visual debugger. Some examples of such tools are Online Python Tutor [11], UUhistle [43], Ville [35], The Teaching Machine [4], Jeliot 2000 [26], Jeliot 3 [30, 31], Evizor [29], and VIP [19, 44]. Among these tools, Ville and VIP are especially focused on pedagogy by supporting teacher-defined examples, and Jeliot 2000 and Jeliot 3 create visualizations that make use of animation in addition to diagram generation. Evaluations of these tools come from anecdotes, experimental studies, qualitative studies, and surveys. Anecdotal evaluations are used to suggest ease-of-use [11], likability [5, 6], and usefulness for self-study [5, 6]. Experimental studies are generally focused on improvement on a specified set of tasks [2, 36]. Qualitative studies, including observational and comparative, examine performance [18, 26], ways of usage [13, 34], helpfulness [16], increase in motivation [40], effect on collaboration [32], and student preference [20]. Surveys mostly collect feedback regarding overall impression [14, 21], likability [5, 6], usefulness for self-study [5, 6], and helpfulness for answering questions [29]. For example, evaluations of UUhistle are specifically tied to visual program simulation exercises and were performed through observational and experimental studies [41].

Many visualization tools are focused on a specific subarea of computer science. For example, MemStep is dedicated to visualizing memory layout and interactively involves the user in the creation of visualizations [24]. CFlow is a tool developed for instructors to view aggregated semantics of code written by students [45]. CS-mart offers practice opportunities by showing visualization first and then prompting the user to type in code [10]. PlanAni creates role-based animation, specifically focusing on the types of variables [37]. Evaluation of these tools are done through experimental studies, qualitative studies, and surveys. Experimental studies focus on performance and understanding improvement [7, 24, 37, 38]. Qualitative studies concern accuracy and reliability [45]. Surveys concern metrics such as user experience, overall impression, effectiveness in helping understanding, correcting misconceptions, and increasing user preparedness [10, 24].

As the significance of data science has been increasingly recognized and enrollment in data science undergraduate and graduate programs has escalated, visualization tools specifically designed to facilitate the learning of tabular data manipulation have also been developed alongside computer programming. Such tools or projects work with tabular data in spreadsheets, R, or SQL and either create intermediate visualizations for data manipulations or create representations of aggregated data information. Examples include TweakIt [23], Unravel [39], Wrangler [15], Data Tweening [17], Datamations [33], QueryVis [25], SQLVis [28], Taggle [9], Digestable [3], and VisuLab [12]. Evaluations found that these tools increase perceived usefulness [15, 23, 39], perceived helpfulness [28, 39], user confidence [23], understanding and retaining knowledge [17, 33], and task performance [15, 17, 25, 28].

These previous tools are either intended for users with substantial amounts of programming experience or require users to switch to a different application to visualize their programs. In contrast, Pandas Tutor is embedded inside Jupyter notebooks which enables our study to investigate how novices make use of program visualization tools in their standard programming environment and workflows.

3 Overview of Pandas Tutor System

Pandas Tutor is a visualization tool for the Python `pandas` library that executes user-defined data manipulation code. Since Pandas Tutor was already introduced in previous work [22], we will only provide an overview of its most relevant features for this paper and introduce additional features that were added to enable the work of this paper. When a user enters Python `pandas` code into Pandas Tutor, the tool will automatically generate before-and-after diagrams for the last expression of the user's code snippet. If the expression contains multiple function calls, which is common for `pandas` code, Pandas Tutor will visualize each function call separately, with a diagram for each call. Pandas Tutor diagrams display the dataframe before a transformation (on the left), the dataframe after the transformation (on the right), and annotations that depict how the data was transformed. For example, in Figure 1b, Pandas Tutor adds colors to each row to show how rows are grouped together. Pandas Tutor diagrams also support a basic form of interactivity: if a user hovers their cursor over a row or column of a dataframe, Pandas Tutor only displays the annotations for the hovered data.

Pandas Tutor was originally released as a standalone webpage where users needed to copy-paste their code into a code editor embedded on the page. However, if a user had a dataset they wanted to explore, they would need to copy-paste the entire dataset as a string at the top of the Pandas Tutor code editor in order to import the data, since the Pandas Tutor website doesn't allow network access. This barrier proved too difficult for novices to overcome in pilot studies and made the tool difficult to evaluate. To address this, we implemented an integration of Pandas Tutor into the Jupyter notebook programming environment. This allows users to add a

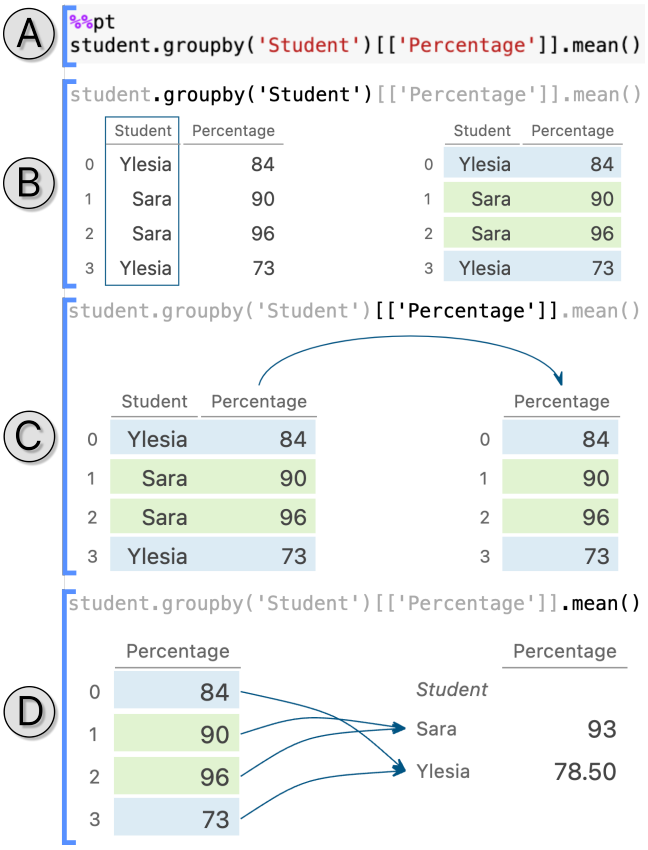


Figure 1: To support the user study in this paper, we implemented an extension to Pandas Tutor to render diagrams directly inside of Jupyter notebooks by adding a single line of code to the top of a code cell (A). Since the pandas expression in this example contains three dataframe operations, Pandas Tutor displays three diagrams (B, C, D).

single line of code to the top of a code cell (`%%pt`) to render a Pandas Tutor diagram directly in their notebook, depicted in Figure 1.

4 Methods

With interest in understanding how novices use such a tool, a user study with after-session interviews was conducted to explore their experiences, challenges, and perceptions.

Participants. Participants were undergraduate students aged 18-22 recruited from data science courses at our university, a public research-focused university in North America. To recruit participants, we posted email announcements for students who were enrolled in lower-division introductory data science courses. Participants received a $20 gift card for their participation. A total of 17 participants (n=17) were recruited. Most participants had minimal experience with Python, with 11 participants having only one year of Python programming experience, 5 participants having two years of experience, and 1 participant having more than two years of experience. To control for background knowledge of pandas, we filtered for participants who had only taken one course that used the pandas library prior to the study.

Tasks. Participants completed tasks in a Jupyter notebook which contained two example data analyses using different datasets. Each data analysis contained three code snippets that performed common data manipulation steps in Pandas, such as sorting values in a DataFrame, aggregating rows to compute summary statistics, merging DataFrames, compting pivot tables, and filtering rows. For each task, participants read one of the code snippets and were asked to explain the code in plain English. Since there were two analyses with three snippets each, participants completed a total of six tasks. Participants were given tasks both with Pandas Tutor and without Pandas Tutor (the baseline condition). We randomized the order in which participants used Pandas Tutor and the baseline. We also randomized the task order to mitigate learning effects.

After completing the code description tasks, participants were given an open-ended code writing task where they were given a question to answer about the data and were asked to write code to produce the desired result. For this task, we provided skeleton code, but participants were not required to use it. Participants were able to use Pandas Tutor to complete the code writing task.

Since all participants already had taken one course that used pandas, we deliberately designed the tasks to make use of functions and pandas idioms that participants had not previously before. During the study, participants were asked to explain their thought process when approaching and understanding code. Participants were also allowed to search the Web and documentation during the study. Our complete task code is available on GitHub[1].

Protocol. The user study was conducted via 1-hour in-person interviews with participants. At the start of the study, the experimenter explained the study to the participant and helped the participant to open the Jupyter notebook for the study. Then, participants were given 40 minutes to complete the code description tasks, with 20 minutes allocated for each data analysis. For each task, participants were asked to write their final description of the code and record their confidence on a scale from 1-7 using an online form. Then, participants were given 10 minutes to complete the code writing task. For the final part of the user study, the experimenter conducted a semi-structured interview with the participant, asking them to reflect on how they used Pandas Tutor and what they found easy or difficult to use about the tool. We recorded participant screens and audio.

5 Quantitative Results

This section presents quantitative results from our user study. Because the primary objective of our study was to develop deeper understanding of how novices use program visualization tools, we did not recruit a large enough sample size to obtain statistically significant differences between conditions. We report these preliminary quantitative findings here in order to point out potential measures for a future larger-scale study.

Participant responses to the code description task were scored for accuracy on a scale of 1-5 by two members of our research group who are not authors of this paper. The two research group members scored each response, then discussed discrepancies until

[1] https://github.com/dstl-lab/Novice-Program-Visualizations-Usage

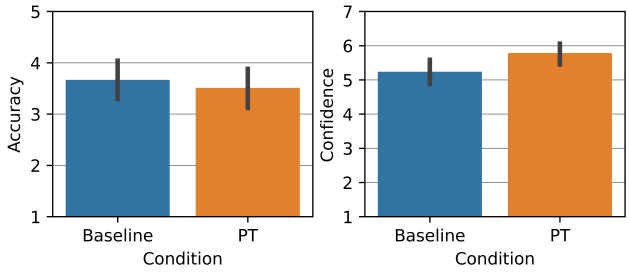


Figure 2: Participants were approximately equally accurate in baseline and Pandas Tutor conditions (left), and reported more confidence in their responses when using Pandas Tutor (right). Error bars show 95% confidence intervals.

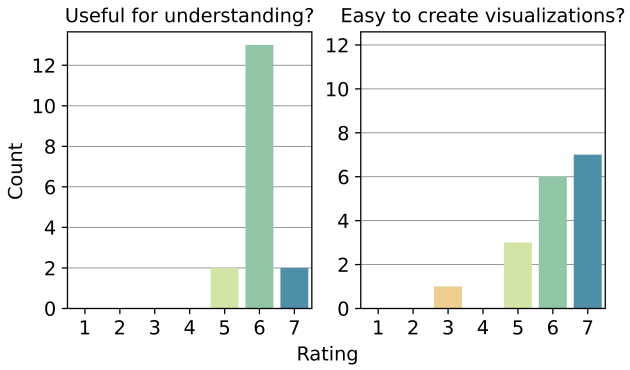


Figure 3: Participants generally found Pandas Tutor useful and easy to use.

agreement was reached. In the Pandas Tutor condition, participants obtained an average accuracy score of $\mu = 3.51$, ($\sigma = 1.35$), while participants in the baseline condition obtained an average accuracy score of $\mu = 3.67$, ($\sigma = 1.44$), as shown in Figure 2 (left).

Participants also took approximately the same amount of time to complete tasks in both conditions. Baseline with $\mu = 4.75$ ($\sigma = 2.44$) minutes per task, and a $\mu = 4.91$ ($\sigma = 3.32$) minutes per task for Pandas Tutor.

When using Pandas Tutor, participants reported an average confidence of $\mu = 5.78$ ($\sigma = 1.14$), which was higher than their confidence in the baseline condition $\mu = 5.24$ ($\sigma = 1.29$), shown in Figure 2 (right). Participants also found Pandas Tutor useful and easy to use overall, as shown in Figure 3.

6 Qualitative Results

This section presents findings from participants' verbal responses as they worked on tasks as well as from post-task interviews. To derive themes from the interviews, the paper authors met regularly to watch participant videos and discuss notes together. Throughout this process, we iteratively came up with a set of themes, using grounded theory and an inductive approach [8]. Our findings are summarized in Table 1 and elaborated below.

6.1 In-notebook visualizations were usable and useful

Overall, participants found the in-notebook visualizations to be easy to use. Although Pandas Tutor also exists as a webpage, users in a previous pilot study found it challenging to transfer their data from their notebooks into Pandas Tutor. In contrast, novices were able to create visualizations in their notebooks with minimal instruction and without any prior experience using the tool (P1-P17).

Participants also found the visual annotations provided by the tool to be helpful. For example, the groupby-aggregate operation is relatively trickier for novices to understand, as a pure Python implementation typically requires a `for` loop over all unique values of a column. One participant commented: "My thought process here was since just looking at the colors, we can just assume that they're going to be groups given how the visualization is." (P7). Another participant noted, "Yeah, since you actually do [the groupby and aggregate] step by step, I'm like, oh, I get it." (P4). Pivot tables are also often challenging for novices to understand since they restructure the entire dataframe. Participants mentioned that Pandas Tutor was helpful "especially for pivot table [...] since I could see the arrows for each argument, it has a different arrow and I can understand it better from that." (P3).

6.2 Using visualizations to verify assumptions

Participants used the visualizations generated by Pandas Tutor to confirm their understanding of the code. They appreciated that Pandas Tutor displayed a visualization for each step within an expression with multiple function calls (e.g., both function calls in `df.groupby(...).sum()`), stating that the "step-by-step process" (P1, P4, P8-P10, P13, P16) helped them understand the code. One participant remarked:

> "Actually I think the reason why I feel more confident is because of the visualization. Good visualization gives me more hints and gives me a clear understanding of what's going on in this defined function. But with pure code, I have to imagine a graph and the changes to that graph in my mind." (P1)

This example points to a tension for novices: even though novices felt that seeing the intermediate outputs would help them understand the program, they did not choose to do so unprompted in the baseline conditions and only examined intermediate outputs when Pandas Tutor visualized them (P1-P17). Even when novices knew the steps they should take to develop confidence in their understanding of the code, they perceived that the effort required to do so was too high. Here, tools like Pandas Tutor can lower the effort needed for novices to double-check their assumptions about the code.

6.3 Beyond the default table output

When the `pandas` library displays a dataframe, it shows the first few rows and the last few rows which is similar to other dataframe libraries (e.g. `data.frame` in R). Pandas Tutor does the same, but also attempts to selectively show rows from the middle of the dataframe by identifying rows with annotations. Additionally, Pandas Tutor allows users to interactively show rows that were originally hidden

Finding	Description	Representative Quote
In-notebook visualizations were usable and useful (Section 6.1)	Participants appreciated that Pandas Tutor visualized code within their familiar notebook environment, and found the tool easy to use and useful with minimal instruction.	"Yeah, since you actually do [the groupby and aggregate] step by step, I'm like, oh, I get it." (P4)
Visualizations helped to verify assumptions (Section 6.2)	Pandas Tutor lowered the effort required to visualize intermediate program results, which gave novices more confidence in their understanding of code.	"Good visualization gives me more hints and gives me a clear understanding of what's going on." (P1)
Revealing normally-hidden table rows can highlight salient information (Section 6.3)	Participants pointed out that the default pandas output only shows the top and bottom rows of a dataframe, when the middle rows of the dataframe can contain relevant information.	"It's not that clear what has happened within each step since there might be 1,010,000 rows of information inside [hidden in the dataframe output]." (P8)
Visualization tools can generate complicated diagrams, but interaction can help users manage cognitive load (Section 6.4)	In more complex cases, Pandas Tutor generated diagrams with many visual elements (e.g. arrows), which novices found overwhelming. However, they found that interactions that enabled selective filtering of visual elements were helpful for understanding.	"I see when you put it like this, it looks very complicated, but as soon as you put the allow hovers, the hovers make it a lot easier." (P11)
Limitations of the tool (Section 6.5)	Pandas Tutor was reported by novices to be better for understanding code rather than writing code because the tool lacked direct support for debugging.	"I think it's better for checking instead of writing." (P4)

Table 1: An in-lab usability study with 17 data science novices found that Pandas Tutor was usable and useful. This work highlights how novices used program visualization tools to understand code and suggests guidelines for future tool developers.

in the dataframe output by dragging a handle. Participants specifically mentioned this feature as useful (P2, P8, P11). One participant stated, "It is nice and intuitive this thing to show, because a lot of the times when I'm doing code, I try to get the rows, but I'm not sure if I'm right in the middle. Just seeing the top and bottom [of the dataframe] doesn't guarantee I'm right." (P11). Another participant mentioned that the default behavior of pandas reduced their self-confidence in their own ability to write code:

> "I think I am not that confident. I went to office hour every week before I submit the homework, even if my results seem reasonable. It's not that clear what has happened within each step since there might be 1,010,000 rows of information inside [hidden in the dataframe output]. So you don't know what is within the middle of the row because you can't read all of the data," (P8)

All participants knew how to use pandas to view the middle rows of a dataframe. However, no one in the baseline conditions chose to do so, even though they perceived that this would help them develop understanding of the code and data. This is another example of how tools like pandas can enable novices to check their understanding of code using methods that they already prefer.

6.4 Managing cognitive load with interaction

Although participants generally liked using Pandas Tutor, they also pointed out that its output could be overwhelming with many visual marks. For example, one participant noted, "And also for, I think sorting is kind of confusing. It gets really cluttered with all the arrows" (P4). This issue was especially evident for operations that manipulated all the dataframe values like making a pivot table, as another participant mentioned, "It looks very complicated with many arrows" (P11). Visualization tools like Pandas Tutor strive for completeness, which means their generated diagrams can have many visual marks and thus incur a high cognitive load for novices.

However, participants also noted that the interactions implemented in Pandas Tutor made it easier to understand the code. For example, participants liked hovering over rows of the dataframes to only display the arrows associated with those rows (P5, P7, P8, P11, P17). One participant commented, "I see when you put it like this, it looks very complicated, but as soon as you put the allow hovers, the hovers make it a lot easier because when I saw this for the first time, I was like, it's a lot" (P11). In general, novices found interaction useful for filtering the number of visual elements on screen.

6.5 Limitations of Pandas Tutor

Although participants generally found Pandas Tutor helpful, it can only generate visualizations for code that does not have syntax or runtime errors. Due to this limitation, participants found Pandas Tutor was more useful for debugging code rather than writing code from scratch (P3, P4, P8, P10). Syntax and runtime errors are

common when writing code, and these errors prevent Pandas Tutor from generating a diagram.

Other participants pointed out that Pandas Tutor did not show all the steps they wanted for certain operations. This was most commonly noted when using `groupby` with a custom function (P4) since Pandas Tutor does not step into function bodies.

7 Discussion

In this section, we describe the limitations of our study design and reflect on our findings to draw more general guidelines for program visualization systems for data science.

7.1 Study Limitations

The conclusions we draw from this work are limited by the tasks and sample of participants that took part in the study. The `pandas` library is extensive, and our study was limited to a relatively small subset of its functionality. It is likely that the usefulness of Pandas Tutor will vary depending on the specific function being visualized. When a function is not directly supported by Pandas Tutor, the tool still provides a side-by-side display of the dataframe before and after the transformation but does not add annotations such as arrows or colors. Although we did not study the usefulness of this display with our participants, some features are still available, including the ability to reveal more rows in the dataframe by dragging a handle. We speculate that Pandas Tutor output for unsupported functions will be perceived as more useful than the baseline output but less useful than the output for supported functions. Lastly, our study participants were all undergraduates, whereas actual users of the tool might also include adult learners, graduate students, and K-12 students.

7.2 Design guidelines for program visualization tools in data science

Our study highlights general design guidelines for tools that assist data scientists through program visualization.

Tools should be embedded within existing user workflows. Pilot studies leading up to this work used Pandas Tutor as a standalone webpage. This approach proved too difficult for novices, as they needed to export their data from their notebooks and then import it into the online tool to visualize their programs. By integrating Pandas Tutor outputs directly into participant notebooks, usability improved substantially, making this study feasible.

Tools should enable comparison of intermediate results. The novices in our user study generally believed that visualizing intermediate results would be helpful for understanding longer expressions of code. However, to our surprise, they almost never chose to visualize intermediate results in the default Jupyter notebook environment. They perceived this process as too effort-intensive – it would require them to comment out lines of code, create temporary variables, print out the intermediate table outputs, and then undo all these changes to revert to the original, unmodified code. One of the main reasons novices felt Pandas Tutor increased their confidence was that Pandas Tutor automatically visualized these intermediate results without extra effort and displayed the entire step-by-step process on the screen at once without requiring additional interactions from the user. Therefore, we recommend that future tools designers make it easy for users to see and compare intermediate results.

Tools should include features to manage cognitive load. The hover feature in Pandas Tutor was initially designed to help instructors present diagrams to students. However, novices in our study repurposed this feature to reduce the number of visual marks on the screen at one time, a use case we had not anticipated. We attribute the use of this feature to our finding that the output of tools like Pandas Tutor can be overwhelming, which highlights the importance of allowing users to selectively filter and abstract visual outputs. Therefore, we recommend that tool designers carefully consider scenarios where their tools may display many visual elements simultaneously and provide interactions that enable users to manage cognitive load effectively.

7.3 Complementing LLM tools

Tools like Pandas Tutor can serve as useful complements to LLM tools such as ChatGPT and Copilot, which can already generate accurate code for data analysis. However, one limitation of LLM tools is that their code can be incorrect in subtle ways. This limitation is particularly relevant in data analysis, where code can often run without errors but lack semantic meaning. For example, it is not uncommon for datasets to encode missing values with dummy values. Calculating the mean of a column with this encoding will not raise an error, but such a calculation should not be used for analysis.

Data scientists who want to use an LLM for programming still need to verify that the LLM output is correct and makes sense in the context of their data. Here, tools like Pandas Tutor can complement the output of an LLM by enabling users to generate code and then check that the outputs make sense in the context of the data analysis.

Additionally, program visualization tools can be further augmented with explanations generated by an LLM. Such a tool could not only provide novices with helpful diagrams for their code but also add annotations to these diagrams using an LLM. This would further help users manage cognitive load by offering both visual and textual explanations of the code.

8 Conclusion

This paper presents an evaluation of a program visualization tool for data science called Pandas Tutor, which visualizes code using the Python `pandas` library. A first-use, in-lab usability study found that Pandas Tutor was both usable and useful for data science novices as they worked with unfamiliar data cleaning code. Overall, using Pandas Tutor increased novices' confidence in their understanding of code without decreasing their task performance. Our study suggests that program visualization tools can be valuable for novices working with data science code written by others. We envision a future where novices will use program visualization tools in tandem with LLM tools to not only generate code but also verify its correctness in the context of their data analysis.

References

[1] Ani Adhikari, John DeNero, and Michael I Jordan. 2021. Interleaving computational and inferential thinking: Data science for undergraduates at. *Harvard Data Science Review* 3, 2 (2021).

[2] Tuukka Ahoniemi and Essi Lahtinen. 2007. Visualizations in Preparing for Programming Exercise Sessions. *Electronic Notes in Theoretical Computer Science* 178 (07 2007), 137–144. https://doi.org/10.1016/j.entcs.2007.01.043

[3] David Borland and David Gotz. 2022. Digestable: Condensed Views of Tabular Data. https://api.semanticscholar.org/CorpusID:252991357

[4] M.P. Bruce-Lockhart and Theodore Norvell. 2000. Lifting the hood of the computer: program animation with the Teaching Machine, Vol. 2. 831 – 835 vol.2. https://doi.org/10.1109/CCECE.2000.849582

[5] Michael Bruce-Lockhart and Theodore Norvell. 2007. Developing Mental Models of Computer Programming Interactively Via the Web. S3H–3. https://doi.org/10.1109/FIE.2007.4418051

[6] Michael Bruce-Lockhart, Theodore Norvell, and Yiannis Cotronis. 2007. Program and Algorithm Visualization in Engineering and Physics. *Electr. Notes Theor. Comput. Sci.* 178 (07 2007), 111–119. https://doi.org/10.1016/j.entcs.2007.01.040

[7] Pauli Byckling and Jorma Sajaniemi. 2005. Using Roles of Variables in Teaching: Effects on Program Construction. In *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group*. University of Sussex, University of Sussex, Sussex, UK. http://www.cs.joensuu.fi/~pbyckli/ P.O.Box 111, 80101 Joensuu, Finland.

[8] Juliet Corbin and Anselm Strauss. 2015. *Basics of qualitative research*. Vol. 14. sage.

[9] Katarina Furmanova, Samuel Gratzl, Holger Stitz, Thomas Zichner, Miroslava Jaresova, Alexander Lex, and Marc Streit. 2019. Taggle: Scalable Visualization of Tabular Data through Aggregation. *Information Visualization* 19, 2 (2019), 114–136. https://doi.org/10.1177/1473871619878085

[10] Roger Gajraj, Margaret Bernard, Malcolm Williams, and Lenandlar Singh. 2011. Transforming Source Code Examples into Programming Tutorials.

[11] Philip J. Guo. 2013. Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE 2013)*. ACM. https://doi.org/10.1145/2445196.2445368

[12] Hans Hinterberger. 2010. *The VisuLab: An Instrument for Interactive, Comparative Visualization*. Technical Report 682. ETH Zurich, Department of Computer Science. https://doi.org/10.3929/ethz-a-006869932 ETH Bibliography.

[13] Essi Isohanni and Maria Knobelsdorf. 2010. Behind the curtain: students' use of VIP after class. In *Proceedings of the Sixth International Workshop on Computing Education Research* (Aarhus, Denmark) *(ICER '10)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/1839594.1839610

[14] Erkki Kaila, Teemu Rajala, Mikko-Jussi Laakso, and Tapio SALAKOSKI. 2009. Effects, Experiences and Feedback from Studies of a Program Visualization Tool. *Informatics in Education* 8 (04 2009). https://doi.org/10.15388/infedu.2009.02

[15] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3363–3372. https://doi.org/10.1145/1978942.1979444

[16] Osku Kannusmaki, Andrés Moreno, Niko Myller, and Erkki Sutinen. 2004. What a Novice Wants: Students Using Program Visualization in Distance Programming Course. *Proceedings of the Third Program Visualization Workshop* (01 2004).

[17] Meraj Khan, Larry Xu, Arnab Nandi, and Joseph M. Hellerstein. 2017. Data tweening: incremental visualization of data transforms. *Proc. VLDB Endow.* 10, 6 (feb 2017), 661–672. https://doi.org/10.14778/3055330.3055333

[18] Mikko-Jussi Laakso, Teemu Rajala, Erkki Kaila, and Tapio Salakoski. 2008. The Impact of Prior Experience in Using a Visualization Tool on Learning to Program. *IADIS International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2008* (01 2008).

[19] Essi Lahtinen and Tuukka Ahoniemi. 2007. Annotations for Defining Interactive Instructions to Interpreter Based Program Visualization Tools. *Electron. Notes Theor. Comput. Sci.* 178 (jul 2007), 121–128. https://doi.org/10.1016/j.entcs.2007.01.041

[20] Essi Lahtinen, Tuukka Ahoniemi, and Anniina Salo. 2007. Efiectiveness of Integrating Program Visualizations to a Programming Course. (11 2007).

[21] Essi Lahtinen, Hannu-Matti Järvinen, and Suvi Melakoski-Vistbacka. 2007. Targeting program visualizations. *ACM SIGCSE Bulletin* 39, 256–260. https://doi.org/10.1145/1268784.1268858

[22] Sam Lau, Sean Kross, Eugene Wu, and Philip J. Guo. 2023. Teaching Data Science by Visualizing Data Table Transformations: Pandas Tutor for Python, Tidy Data Tutor for R, and SQL Tutor. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research* (Seattle, WA, USA) *(DataEd '23)*. Association for Computing Machinery, New York, NY, USA, 50–55. https://doi.org/10.1145/3596673.3596972

[23] Sam Lau, Sruti Srinivasa Srinivasa Ragavan, Ken Milne, Titus Barik, and Advait Sarkar. 2021. TweakIt: Supporting End-User Programmers Who Transmogrify Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 311, 12 pages. https://doi.org/10.1145/3411764.3445265

[24] Michelle Le Pham, Anna Nguyen, and Rebecca Schreib. 2024. MemStep: An Interactive Tool for Constructing and Visualizing the Run-Time Memory Layout of Java Programs. In *Proceedings of the 29th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2024)*. ACM, Milan, Italy. https://doi.org/10.1145/3649217.3653532

[25] Aristotelis Leventidis, Jiahui Zhang, Cody Dunne, Wolfgang Gatterbauer, H.V. Jagadish, and Mirek Riedewald. 2020. QueryVis: Logic-based Diagrams help Users Understand Complicated SQL Queries Faster. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2303–2318. https://doi.org/10.1145/3318464.3389767

[26] Ronit Levy, Mordechai Ben-Ari, and Pekka Uronen. 2003. The Jeliot 2000 program animation system. *Computers & Education* 40 (01 2003), 1–15. https://doi.org/10.1016/S0360-1315(02)00076-3

[27] Wes McKinney and PD Team. 2015. Pandas-Powerful python data analysis toolkit. *Pandas—Powerful Python Data Analysis Toolkit* 1625 (2015).

[28] Daphne Miedema and George Fletcher. 2021. SQLVis: Visual Query Representations for Supporting SQL Learners. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–9. https://doi.org/10.1109/VL/HCC51201.2021.9576431

[29] Jan Moons and Carlos Backer. 2013. The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education* 60 (01 2013), 368–384. https://doi.org/10.1016/j.compedu.2012.08.009

[30] Andrés Moreno and Niko Mylle. 2003. PRODUCING AN EDUCATIONALLY EFFECTIVE AND USABLE TOOL FOR LEARNING, THE CASE OF JELIOT FAMILY. *ACM Transactions on Programming Languages and Systems - TOPLAS* (01 2003).

[31] Andrés Moreno, Niko Myller, Erkki Sutinen, and Mordechai Ben-Ari. 2004. Visualizing programs with Jeliot 3 *(AVI '04)*. Association for Computing Machinery, New York, NY, USA, 373–376. https://doi.org/10.1145/989863.989928

[32] Niko Myller, Roman Bednarik, Erkki Sutinen, and Mordechai Ben-Ari. 2009. Extending the Engagement Taxonomy: Software Visualization and Collaborative Learning. *TOCE* 9 (01 2009).

[33] Xiaoying Pu, Sean Kross, Jake M. Hofman, and Daniel G. Goldstein. 2021. Datamations: Animated Explanations of Data Analysis Pipelines. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 467, 14 pages. https://doi.org/10.1145/3411764.3445063

[34] Teemu Rajala, Erkki Kaila, Mikko-Jussi Laakso, and Tapio Salakoski. 2009. Effects of Collaboration in Program Visualization. https://api.semanticscholar.org/CorpusID:202706336

[35] Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, and Tapio Salakoski. 2007. VILLE: a language-independent program visualization tool. https://api.semanticscholar.org/CorpusID:58796028

[36] Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, and Tapio Salakoski. 2008. Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. *Journal of Information Technology Education: Innovations in Practice* 7 (01 2008). https://doi.org/10.28945/195

[37] Jorma Sajaniemi and Marja Kuittinen. 2003. Program animation based on the roles of variables. In *Proceedings of the 2003 ACM Symposium on Software Visualization* (San Diego, California) *(SoftVis '03)*. Association for Computing Machinery, New York, NY, USA, 7–ff. https://doi.org/10.1145/774833.774835

[38] Jorma Sajaniemi and Marja Kuittinen. 2005. An Experiment on Using Roles of Variables in Teaching Introductory Programming. *Computer Science Education* 15 (2005), 59 – 82. https://api.semanticscholar.org/CorpusID:14724609

[39] Nischal Shrestha, Titus Barik, and Chris Parnin. 2021. Unravel: A Fluent Code Explorer for Data Wrangling. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 198–207. https://doi.org/10.1145/3472749.3474744

[40] Kimmo Sivula. 2005. *A Qualitative Case Study on the Use of Jeliot 3*. Master's thesis. University of Joensuu, Department of Computer Science. https://cs.uef.fi/pub/Theses/2005_MSc_Sivula_Kimmo.pdf

[41] Juha Sorva. 2012. *Visual Program Simulation in Introductory Programming Education*. Ph. D. Dissertation.

[42] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education* 13, 4 (Nov. 2013), 15:1–15:37. https://doi.org/10.1145/2490822 Article 15.

[43] Juha Sorva and Teemu Sirkiä. 2010. UUhistle - A software tool for visual program simulation. (10 2010), 49–54. https://doi.org/10.1145/1930464.1930471

[44] Antti T. Virtanen, Essi Lahtinen, and Hannu-Matti Järvinen. 2005. VIP, a Visual Interpreter for Learning Introductory Programming with C++. In *Kolin Kolistelut - Koli Calling 2005 Conference on Computer Science Education*. Koli, Finland, 125–130.

[45] Ashley Ge Zhang, Xiaohang Tang, Steve Oney, and Yan Chen. 2024. CFlow: Supporting Semantic Flow Analysis of Students' Code in Programming Problems at Scale. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale* (Atlanta, GA, USA) *(L@S '24)*. Association for Computing Machinery, New York, NY, USA, 188–199. https://doi.org/10.1145/3657604.3662025