Research Statement

Sam Lau (lau@ucsd.edu)
October 2022

In today's interconnected world, people rely on data to make important decisions: for companies designing their websites, for cities deciding where to build new roads, and for scientists figuring out how to stop the spread of a disease. Because of this, many people are motivated to learn data science: the study of drawing conclusions from data using programs. However, learning data science is difficult for novices because code is opaque—current programming tools don't show how code transforms the data. **How can we facilitate learning by making code and data more visible?**

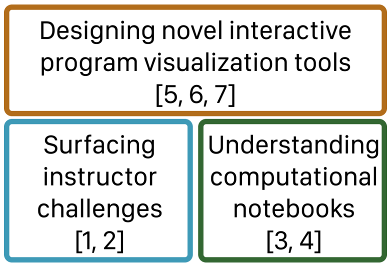

Designing novel interactive program visualization tools [5, 6, 7]
Surfacing instructor challenges [1, 2]
Understanding computational notebooks [3, 4]

Figure 1: My research goal is to design, implement, and evaluate novel tools for program visualization. This research is guided by challenges surfaced from studies of instructor challenges and computational notebooks.

To address this question, my research uses methods from **human-computer interaction (HCI)**. This research agenda is summarized in Figure 1. One line of work examines instructors of technical courses, surfacing previously unreported logistical challenges [1] and pedagogical negotations [2] that arise as courses are developed. Another line of research studies computational notebook systems, including the first survey and proposed design space of computational notebooks [3], and a novel tool for using notebooks to create interactive webpages [4]. Studying the people and existing tools involved in computing and data science courses enables me to pursue my overarching research goal: **designing interactive program visualization tools that demonstrate how code changes data**. For example, the Data Theater system enables instructors to quickly prototype interactive instructional animations using a concise domain-specific language [5]. TweakIt lets data analysts use code directly within their existing spreadsheets [6]. The Pandas Tutor system automatically generates explanatory diagrams from `pandas` code [7]. This statement explains Pandas Tutor, TweakIt, and describes future directions for my research.

Pandas Tutor: Step-by-Step Diagrams for Data Science Code

Instructors use diagrams to help explain code. For example, instructors for introductory programming courses hand-draw diagrams to visualize the program state during execution. Data science instructors also use diagrams to explain what table manipulation operations do. For example, when filtering rows, an instructor might cross out rows of a data table. Instructors currently make these diagrams manually, by displaying the data and adding annotations using a drawing tool, as depicted in Figure 2. However, this process doesn't repre-

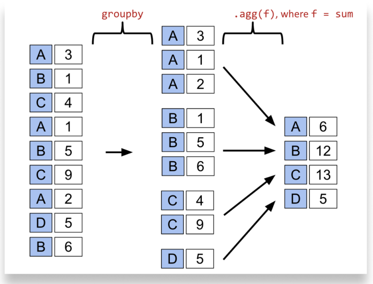
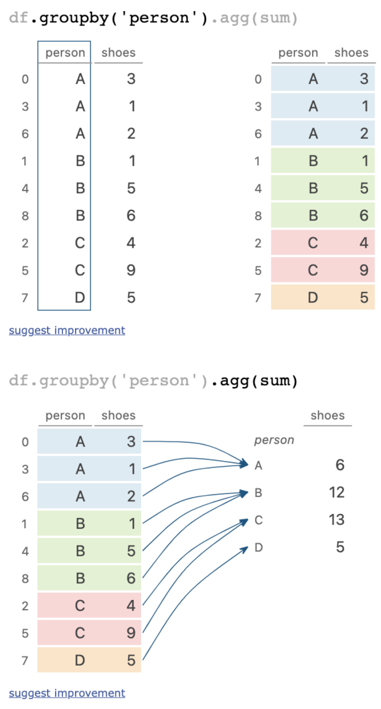


Figure 2: In this lecture slide on grouping and aggregating, the instructor manually displayed data and drew annotations.

sent the link between code and data—whenever the data change, the instructor needs to redraw the annotations. Drawing diagrams manually is also time-consuming. Although instructors may wish to use diagrams to help students debug their code, it is impractical to do so when there are hundreds of students who need help with many different bugs.

In response, I developed Pandas Tutor, a system that automatically creates step-by-step diagrams for `pandas` code [7]. `pandas` is the standard library for data manipulation in Python, so it is commonly taught in introductory data science courses. When a user enters `pandas` code into Pandas Tutor, the system uses a combination of static and runtime analysis to display both data tables and explanatory annotations, similar to what an expert data science instructor would draw, as shown in Figure 3.

Pandas Tutor is available for public use as a web application and as a Jupyter extension. Instructors can use Pandas Tutor to prepare diagrams for their lecture slides, and to explain code that they write live during class. Students can use Pandas Tutor to understand and debug code as they work on assignments. In the ten months since its launch in December 2021, Pandas Tutor has served over 38,000 users across 166 countries and is now actively used in large introductory data science courses, including DSC 10 at UC San Diego.



```
df.groupby('person').agg(sum)
```

	person	shoes
0	A	3
3	A	1
6	A	2
1	B	1
4	B	5
8	B	6
2	C	4
5	C	9
7	D	5

	shoes
person	
A	6
B	12
C	13
D	5

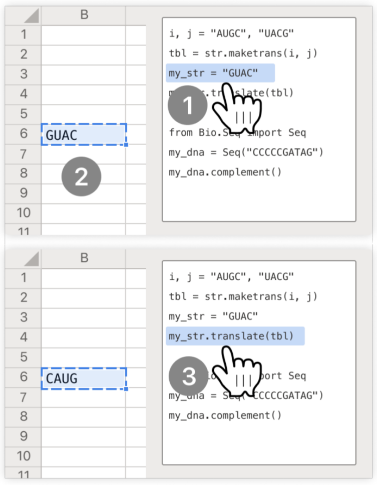
Figure 3: Pandas Tutor automatically draws diagrams similar to ones that expert instructors would manually create. This example recreates the grouping and aggregation operations shown in Figure 2.

TweakIt: Supporting Programmers Who Reuse and Tweak Code

Data analysts across a variety of diverse disciplines—chemists, material scientists, and cognitive psychologists—routinely wish to use code to transform data, explore hypotheses, and evaluate findings. Although these analysts use code, they don't see themselves as professional programmers. Their coding workflows consist less about *writing* code and more about *tweaking* it: in a typical workflow, data analysts cobble together various snippets of code from colleagues or online sites like Stack Overflow and make small edits to these snippets with trial-and-error incantations. Rather than asking users to change how they behave, **how can we design tools that support the highly goal-oriented coding workflows they prefer?**

To address this need, I developed TweakIt, a system that supports analysts as they reuse and tweak code [6]. TweakIt uses live previews to let analysts preview the effects of their code directly inside their usual spreadsheet environments. For example, an analyst who wants to use the BioPython package in their spreadsheet can open the TweakIt sidebar, paste in example code from the package documentation, and click on code expressions to preview the code outputs directly in the spreadsheet, as shown in Figure 4.

In a usability study with data analysts, I found that analysts valued TweakIt's support for their preferred guess-and-check coding workflows. TweakIt encouraged code exploration and analysts' confidence without decreasing their effectiveness and efficiency. As a whole, analysts reported many day-to-day tasks that they felt could be better addressed using code ranging from extracting values from loan applications to summarizing cell phone chip performance results.



```
i, j = "AUGC", "UACG"
tbl = str.maketrans(i, j)
my_str = "GUAC"
my_str.translate(tbl)

from Bio.Seq import Seq
my_dna = Seq("CCCCCGATAG")
my_dna.complement()
```

Figure 4: With TweakIt, analysts can reuse code examples from the web directly in their spreadsheet. 1) When an analyst clicks on a code expression, 2) TweakIt puts the output of the expression directly into the spreadsheet. 3) When an analyst clicks on a different expression, TweakIt updates the spreadsheet automatically.

Future Research Agenda

By continuing to pursue my research interests, I can not only advance state-of-the-art tools for program visualization but also directly help students. I'm especially interested in expanding the vocabulary of visualizations that systems can automatically create. For example, different instructors draw different diagrams—where one instructor might prefer to use arrows, another instructor might instead rely on color-coding. I want to develop a specification language for Pandas Tutor visualizations so that instructors can customize the visualizations for their own classrooms. This would enable instructors to automatically generate code visualizations in the way that they prefer, rather than requiring instructors to use the visualization styles that I built into Pandas Tutor.

Developing this grammar of data table visualizations could also make Pandas Tutor's approach useful in more contexts. For example, the current version of Pandas Tutor is difficult to use for professional data scientists who need to work with many large datasets at once, which would cause Pandas Tutor to display too many annotations to make sense of. Allowing users to specify how Pandas Tutor draws annotations would also enable them to reduce the number of annotations in more complex analyses.

I envision a future where both instructors and learners can fluently create explanatory diagrams through program visualization tools. Just as data visualizations enable people to understand complex data, I believe that program visualizations have the potential to help people understand complex programs. Through my research, I seek to make this vision a reality for instructors and learners everywhere.

References

[1] Sam Lau, Justin Eldridge, Shannon Ellis, Aaron Fraenkel, Marina Langlois, Suraj Rampure, Janine Tiefenbruck, and Philip J. Guo. The Challenges of Evolving Technical Courses at Scale: Four Case Studies of Updating Large Data Science Courses. In *Proceedings of the Ninth ACM Conference on Learning@ Scale*, pages 201–211, 2022.

[2] Sam Lau, Deborah Nolan, Joseph E. Gonzalez, and Philip J. Guo. How Computer Science and Statistics Instructors Approach Data Science Pedagogy Differently: Three Case Studies. In *Proceedings of the 53rd ACM Technical Symposium on Computing Science Education*, Providence, RI, USA, March 2022. Association for Computing Machinery.

[3] Sam Lau, Ian Drosos, Julia M. Markel, and Philip J. Guo. The Design Space of Computational Notebooks: An Analysis of 60 Systems in Academia and Industry. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–11. IEEE, 2020.

[4] Samuel Lau and Joshua Hug. Nbinteract: Generate interactive web pages from Jupyter notebooks. Master's thesis, Master's thesis, EECS Department, University of California, Berkeley, 2018.

[5] Sam Lau and Philip J. Guo. Data Theater: A live programming environment for prototyping data-driven explorable explanations. In *Workshop on Live Programming (LIVE)*, 2020.

[6] Sam Lau, Sruti Srinivasa Srinivasa Ragavan, Ken Milne, Titus Barik, and Advait Sarkar. TweakIt: Supporting End-User Programmers Who Transmogrify Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2021.

[7] Sam Lau and Philip J. Guo. Pandas Tutor - visualize Python pandas code. https://pandastutor.com/.