



Making Science Interactive: Enhancing Synthesis and Collaboration

Qualifying Exam
Caroline Berger
May 2025

1

Hello, today I will talk about my PhD project, making science interactive, enhancing synthesis and collaboration. This project lies at the intersection of human-computer interaction, and programming languages. It is supervised by Clemens Klokmoose and Magnus Madsen.

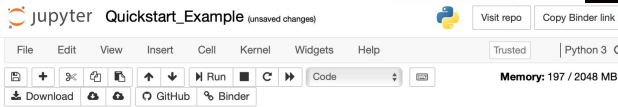
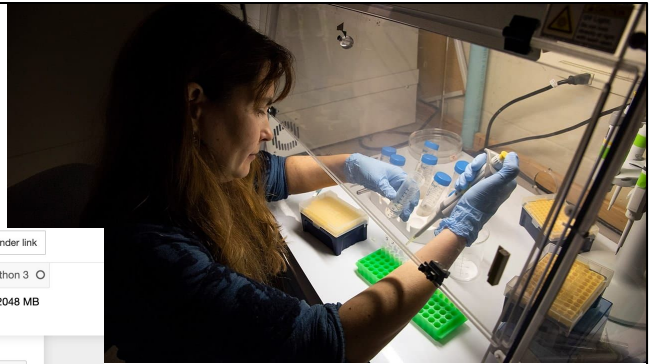


The way we experience the world matters, and our experience is influenced by technology and tools. With our bare eyes, when we look up at the moon, it's an amorphous white circle.

But, through the lens of a telescope, the moon's craters become apparent, and we can discern valleys and edges.

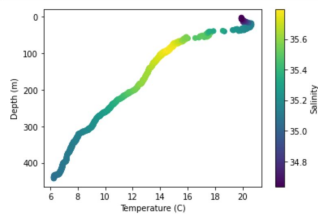
In my work, I aim to develop tools that act as cognitive extensions—enabling scientists to see and reason in new ways. I want to provide the building materials and tools that allow a scientist to build their own telescopes, that allows them to further their scientific discoveries.

Scientific Work

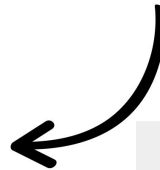


```
In [25]: import matplotlib.pyplot as plt

In [27]: # Temperature vs. Depth
ax = plt.scatter(x=data['Temperature'], y=data['Depth'], c=data['Salinity']);
plt.gca().invert_yaxis(); # Flip the y-axis
plt.colorbar(label='Salinity')
plt.xlabel('Temperature (C)')
plt.ylabel('Depth (m)');
```



Also Scientific Work



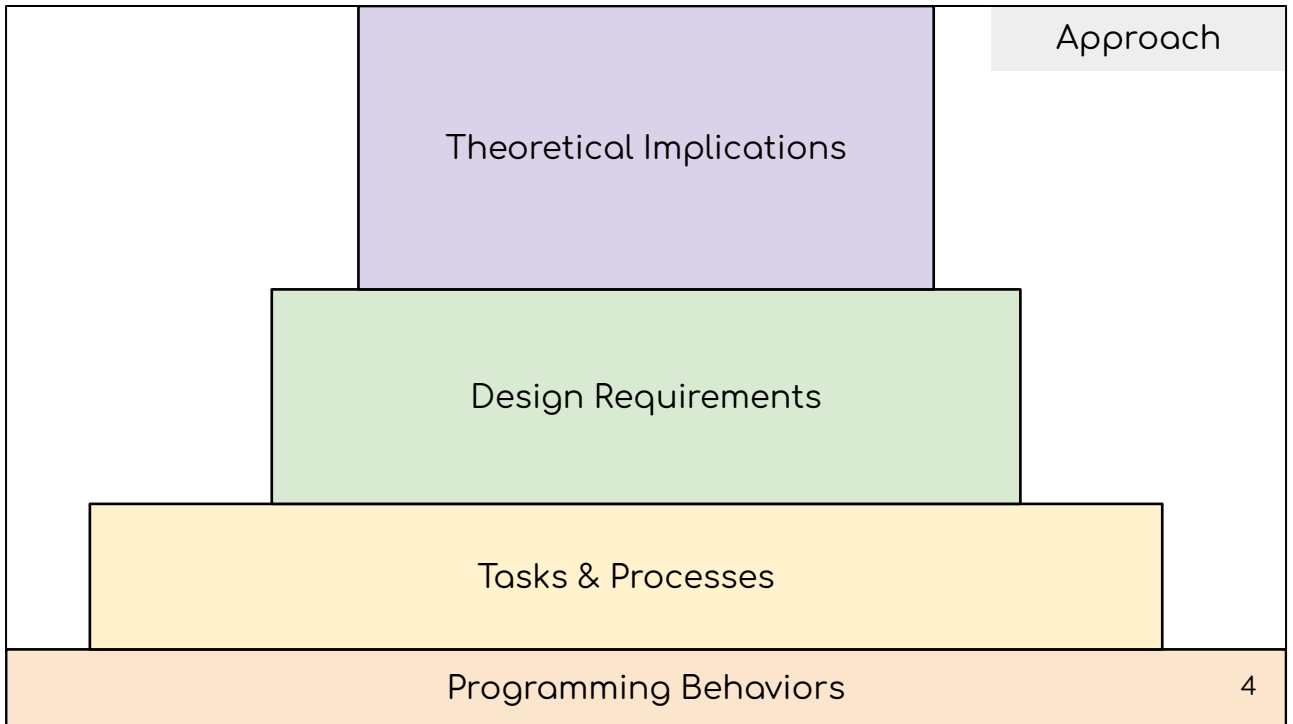
Motivation

3

Nowadays, scientific work is not confined to a lab bench, but it is increasingly reliant on programming for simulations and data analysis.

Yet, many scientists lack formal training in computer science, and confidence in programming.

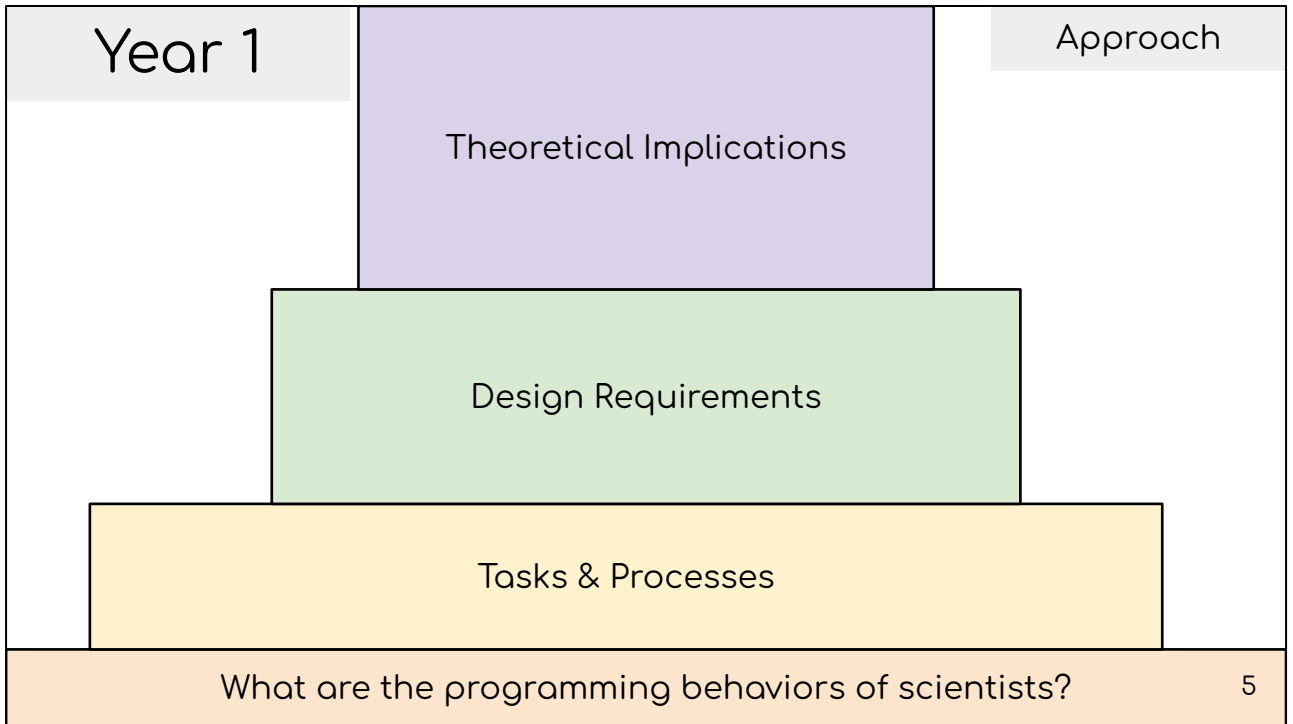
But, the computational tools they use require a high level of technical fluency.



Broadly, my work is in the domain of programming environments. Computational notebooks are one example of a programming environments, and research in this area includes the interface where programming takes place, the accompanying tools, and the experience from a user's point of view.

I approach this from a human-centred lens, with special focus on non-software engineer users.

The different parts of my research build upon each other. Beginning with programming behaviors



My first research question that I explored during my first year was what are the programming behaviors of scientists.

This question investigates how scientists—often self-taught or informally trained in programming—engage with code in their day-to-day work. It seeks to uncover patterns, challenges, and contextual influences that shape their computational practices. The question examines the social practices around computing, as well as the programming behaviors, and stylistic and structural tendencies observed in the code they write. Later in the talk, I will present findings from this first research question with scientists.

Year 1 & 2	Theoretical Implications	Approach
	Design Requirements	
	What scientific tasks and processes could be supported by extending interaction capabilities?	
	What are the programming behaviors of scientists?	6

While there are many possible avenues to study after learning about programming behaviors, we chose to scope and focus on interactivity.

We look at the workflows and pain points in scientific inquiry that could benefit from enhanced interactive capabilities, such as interactive visualizations, responsive simulations, or direct manipulation of data and models.

I hypothesize that interactivity could facilitate communication, through the extension of artifacts, and could enhance analysis, helping scientists to uncover patterns, anomalies, and features of their data. I have partially answered this question, and will present early findings related to this question later on.

Year 2 & 3	Theoretical Implications	Approach
	What are the design requirements for tools that support scientists authoring interactive elements?	
	What scientific tasks and processes could be supported by extending interaction capabilities?	
	What are the programming behaviors of scientists?	7

Based on insights into scientific practice, this question aims to derive practical and technical requirements for programming environments that empower scientists to create interactive artifacts—without requiring deep software engineering expertise. Sliders, zoom, and filters are examples of interactive elements.

To better answer this question, I focus my research on one type of scientist: oceanographers. I have done a bit of work aiming to answer this question, and plan to drive forward in this direction this summer.

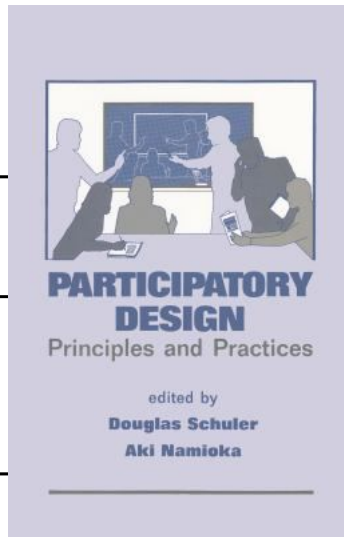
Year 3		Approach
	What are the theoretical implications associated with scientists authoring interactive elements?	
	What are the design requirements for tools that support scientists authoring interactive elements?	
	What scientific tasks and processes could be supported by extending interaction capabilities?	
	What are the programming behaviors of scientists?	8

As a final research question, shifting to a higher level of abstraction, I plan to look into literacy, and propose a framework for how programming, interaction, and visualization literacy is related for scientists.

Next, I'll explain my methodological approach and theoretical basis for answering these questions.

	What are the theoretical implications associated with scientists authoring interactive elements?
	What are the design requirements for tools that support scientists authoring interactive elements?
	What scientific tasks and processes could be supported by extending interaction capabilities?
<i>Contextual Inquiry</i>	What are the programming behaviors of scientists?
	9

I use contextual inquiry to learn about the programming behaviors of scientists and tasks and processes that could be supported by extending interaction capabilities.



9 Contextual Inquiry: A Participatory Technique for System Design

Karen Holtzblatt
InContext Enterprises, Inc. Sudbury, MA

Sandra Jones
Digital Equipment Corporation, Nashua, NH

- Observation & interview in-situ
- Heavily tied to context
- Rich & detailed insights (thick data)

Contextual Inquiry

What are the programming behaviors of scientists?

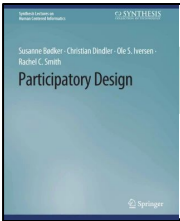
10

Contextual inquiry involves observing and interviewing people in their real work environments. It helps to reveal challenges and workflows that might not show up in surveys or lab studies.

While it provides rich, detailed insights grounded in real practice, the results can be hard to generalize since they're tied to specific contexts and individuals.

	What are the theoretical implications associated with scientists authoring interactive elements?
	What are the design requirements for tools that support scientists authoring interactive elements?
<i>Participatory Design</i>	What scientific tasks and processes could be supported by extending interaction capabilities?
<i>Contextual Inquiry</i>	What are the programming behaviors of scientists?

In addition to contextual inquiry, I use participatory design, to help us learn about tasks and processes as well as the design requirements for authoring interactive elements.

	<p>CHAPTER 2</p> <h2>What Is Participatory Design?</h2>	<ul style="list-style-type: none"> • Envisioning possible futures • Human Actors not Human Factors
		<p>interactive elements?</p>
<p><i>Participatory Design</i></p>	<p>What are the design requirements for tools that support scientists authoring interactive elements?</p>	
<p><i>Contextual Inquiry</i></p>	<p>What scientific tasks and processes could be supported by extending interaction capabilities?</p> <p>What are the programming behaviors of scientists?</p>	<p>12</p>

Participatory design is a collaborative method that involves end-users directly in the design process, making sure their needs and experiences shape the final product. Participatory design includes performing activities that encourage participants to envision possible or alternative futures.

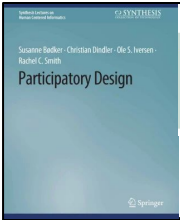
It helps ensure that tools fit real-world workflows by engaging users.

This approach not only leads to more relevant and effective solutions, but also gives users a sense of ownership, making them more likely to adopt the tool.

Instead of viewing humans as factors in technology design, it mobilizes them into active participants in technology development.

	What are the theoretical implications associated with scientists authoring interactive elements?
<i>Prototyping</i>	What are the design requirements for tools that support scientists authoring interactive elements?
<i>Participatory Design</i>	
	What scientific tasks and processes could be supported by extending interaction capabilities?
<i>Contextual Inquiry</i>	What are the programming behaviors of scientists? 13

In addition to working with people, I use prototypes as key tools for exploring and refining design ideas.

	<p>CHAPTER 6</p> <p>What Are the Tools and Materials of Participatory Design?</p>	<ul style="list-style-type: none"> • Externalization of ideas • Provoke ideation
<p><i>Prototyping</i></p>	<p>interactive elements?</p>	
<p><i>Participatory Design</i></p>	<p>What are the design requirements for tools that support scientists authoring interactive elements?</p>	
<p><i>Contextual Inquiry</i></p>	<p>What scientific tasks and processes could be supported by extending interaction capabilities?</p>	
	<p>What are the programming behaviors of scientists?</p>	

External prototypes offer new perspectives, stimulate creative thinking, and serve as a source of inspiration for possible design directions.

By presenting these existing tools to participants, we invite them to engage critically with ideas that have already been explored, encouraging them to offer insights and suggest modifications or alternatives to fit tools to their own use cases.

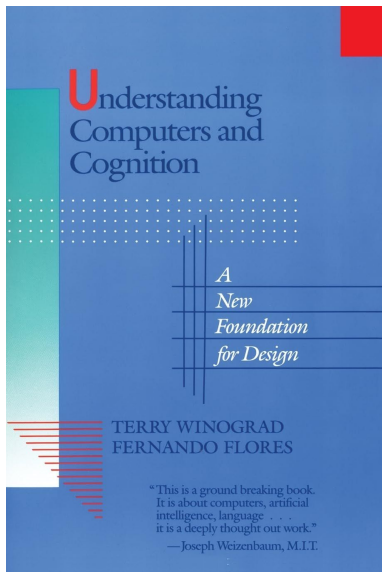
By sharing prototypes during participatory design, mutual learning takes place, participants get the opportunity to teach researchers about their practice, and learn about the possibilities.

This process not only enriches our understanding of the participants' needs but also allows for a broader exploration of design possibilities in the context of our project.

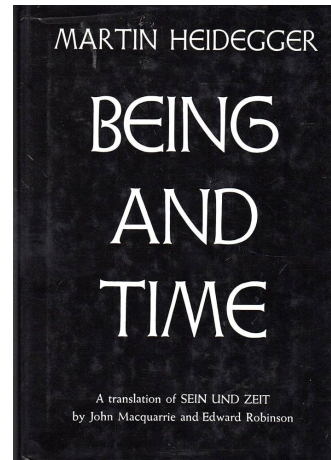
<i>Framework Development</i>	What are the theoretical implications associated with scientists authoring interactive elements?
<i>Prototyping</i>	What are the design requirements for tools that support scientists authoring interactive elements?
<i>Participatory Design</i>	What scientific tasks and processes could be supported by extending interaction capabilities?
<i>Contextual Inquiry</i>	What are the programming behaviors of scientists?

In the future, I plan to create a framework connecting literacies.

Designing for Appropriation: A Theoretical Account



Pierre Tchounikine
University of Grenoble, France



Specifically, theory has shaped my research questions and explanations of findings, drawing from thinkers like Winograd and Flores, Heidegger, and chonikine.

These theories emphasize how tools—whether physical or digital—impact scientists' actions, shaping their understanding and practices.

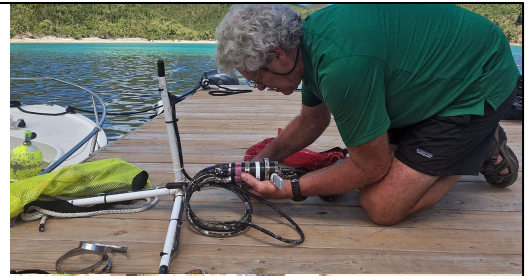
Approach

Toolness of things: Significance derived from use, not just form

Ecology of things: Element's importance derived from whole

hammer

gamel



Here we see a gamelan and a hammer, although they are similar in form, their usage distinguishes one from the other. Informed by Balinese culture, musicians play the gamelan with a gamelan.

The Gamelan is apart of traditions like the Galungan holiday festival, and dancers perform to the Gamenlan's music at local temples.

For Balinese musicians, the frame of reference and context for what they understand and experience is personal, and specific to their lifeworld.

Further, the gamelan exists in the shared practice of a culture as part of an equipmental nexus, and gets its significance through its relation to the gamelan.

Each element derives its importance from this whole, referred to in HCI as an ecology of things.

A scientist's use of technology is deeply influenced by their discipline, lab culture, and individual experiences. Cognition, the way scientists think and learn, is socially constructed, situated, and culturally dependent. Heidegger's concept of "lifeworld" refers to the personal and cultural context through which scientists engage with phenomena. The "toolness of things" emerges through the ways scientists interact with tools, revealing their essence beyond their form.

Tools are part of a larger ecology, where their significance arises from how they're used in specific contexts.

Appropriation: Tools being used for another purpose than their originally intended/designed purpose



18

Appropriating tools is key, as scientists adapt and recontextualize both physical and digital tools to fit their needs. In the image, I see a physical object being appropriated, by using the earring post to pop open a sim card tray of a phone.

The earring is designed as a decorative accessory.

However, the user is opening their SIM card tray in order to fulfill a need, like to change the SIM card, for example. The toolness of the earring is made apparent through its appropriated use.

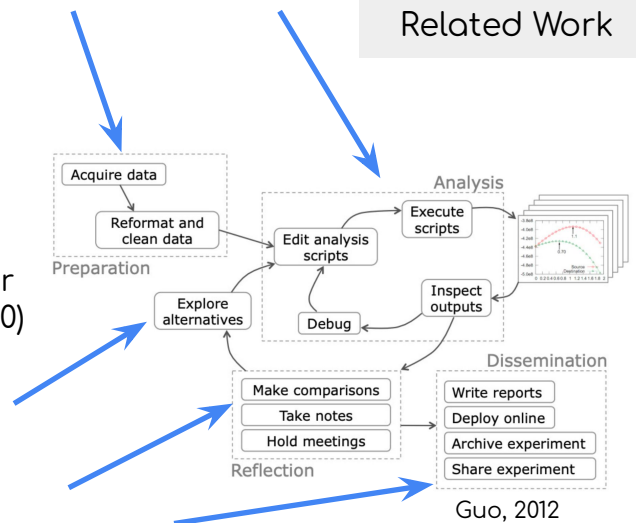
Designing tools for appropriation means understanding how they are personally adapted by users to resolve challenges, like swapping sim cards. This process mirrors how digital tools like programming tools are integrated into scientists' workflows.

Scientific Programming

Scientific output > Programming performance (Basili et al., 2008)

Code comes from online resources or colleagues' code (Nouwens et al., 2020)

Resulting scripts and software are difficult to maintain due to low reliability and low comprehensivity (Morris et al., 2009)



Basili et al., "Understanding the High-Performance-Computing Community: A Software Engineer's Perspective," in IEEE Software, vol. 25, no. 4, July-Aug. 2008, doi: 10.1109/MS.2008.103.
Nouwens et al., "Between Scripts and Applications: Computational Media for the Frontier of Nanoscience," in CHI '20, doi: 10.1145/3313831.3376287.
Morris et al., "Some challenges facing scientific software developers: The case of molecular biology," in IEEE International Conference on e-Science, doi: 10.1109/e-Science.2009.38.
Guo, 2012. *Software tools to facilitate research programming*. Stanford University.

19

Now that I've told you about the research questions and my methodological and theoretical approach, I'd like to tell you about work that has been done in this area. Luckily I am not the only researchers working on this topic, others have studied the scientific programming.

For scientists, the scientific value that comes from the software or scripts, is much more important to them than the programming performance.

The code comes from online resources or colleagues code, and is often in several languages.

The resulting scripts and software are difficult to maintain due to low reliability and low comprehensivity.

The process of scientific programming starts with preparation,

then during analysis scripts are edited and executed iteratively.

Scientists then make comparisons, take notes, and hold meetings during reflection,

they will explore alternatives, then might go back to analysis. After iteration, they might disseminate findings via reports or sharing with their colleagues.

Since we are looking specifically at producing interactive artifacts, we are interested in learning how the process might change. For example, if making comparisons is more

coupled into the programming process.

Related Work

Tools to Author Interactivity

Anywidget: authoring via code - cross platform (Manz et al.)

DynaVis: authoring via prompts (Vaithilingam et al.)

Lyra2: authoring via demonstration (Zong et al.)

Manz et al., anywidget, anywidget.dev/
Vaithilingam et al., "DynaVis: Dynamically synthesized UI widgets for visualization editing," in CHI '24 doi: 10.1145/3613904.3642639
Zong et al., 2020. Lyra 2: Designing interactive visualizations by demonstration. *IEEE Transactions on Visualization and Computer Graphics* doi: 10.1109/TVCG.2020.3030367.

anywidget

reusable interactive widgets made easy

Get Started

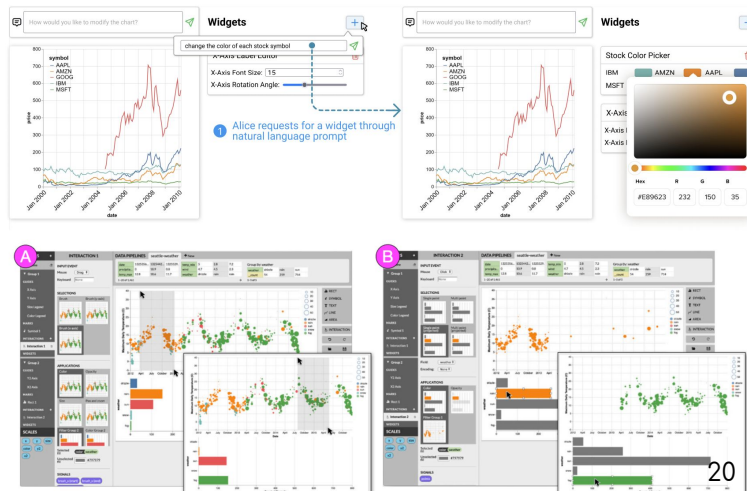
> pip install anywidget

```
import anywidget
import traitlets

class MarineBiologyWidget(anywidget.AnyWidget):
    _esm = "src/widget.js"
    value = traitlets.Int(0).tag(sync=True)

# Create and display the widget
widget = MarineBiologyWidget()
widget
```

Depth Range: 0



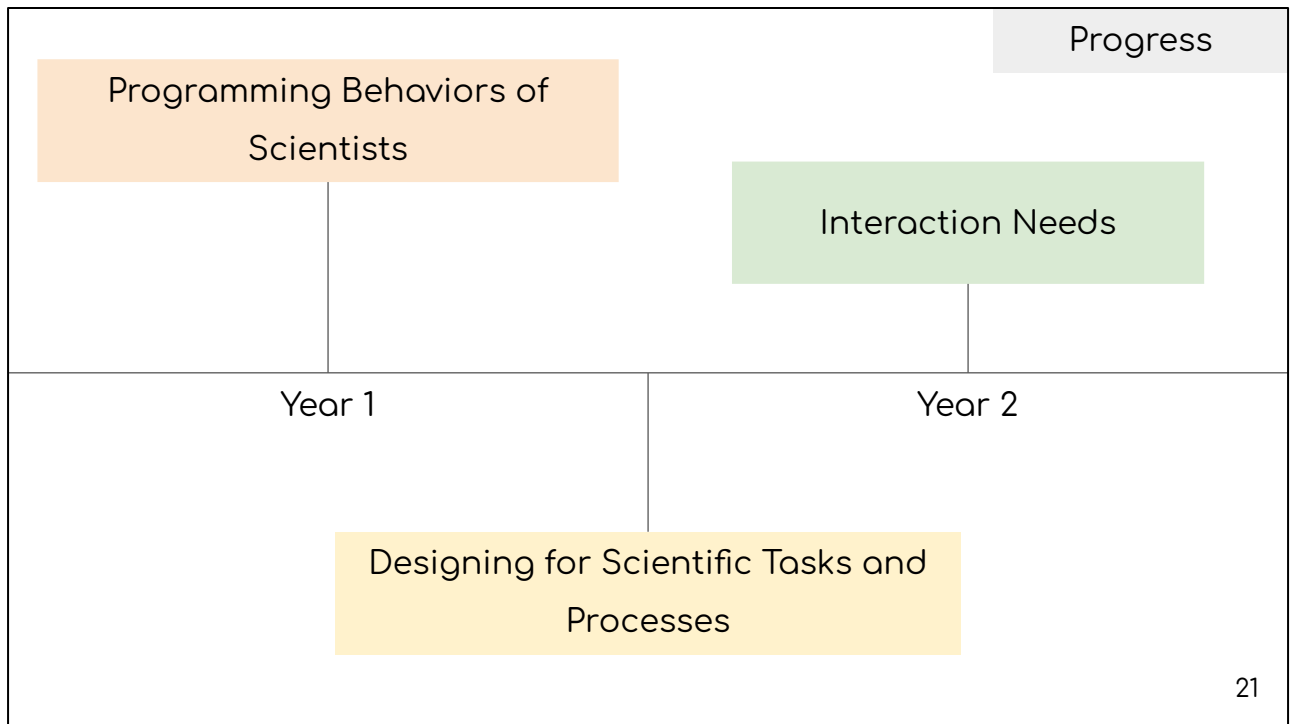
In terms of tools, there are excellent examples of tools aimed at making authoring interactivity easier, and more expressive.

Within the space of computational notebooks, where scientists often write code, Anywidget offers a code-based approach to creating interactive Jupyter widgets, which can be seamlessly integrated into various computational environments. By enabling the authoring of custom widgets through Python code, Anywidget allows scientists to craft interactive elements that are both flexible and portable across different platforms. This method retains the power and customization potential of coding while aiming to simplify the process of creating reusable, interactive components.

DynaVis on the other hand presents an innovative approach by enabling the creation of dynamic UI widgets based on natural language commands from users

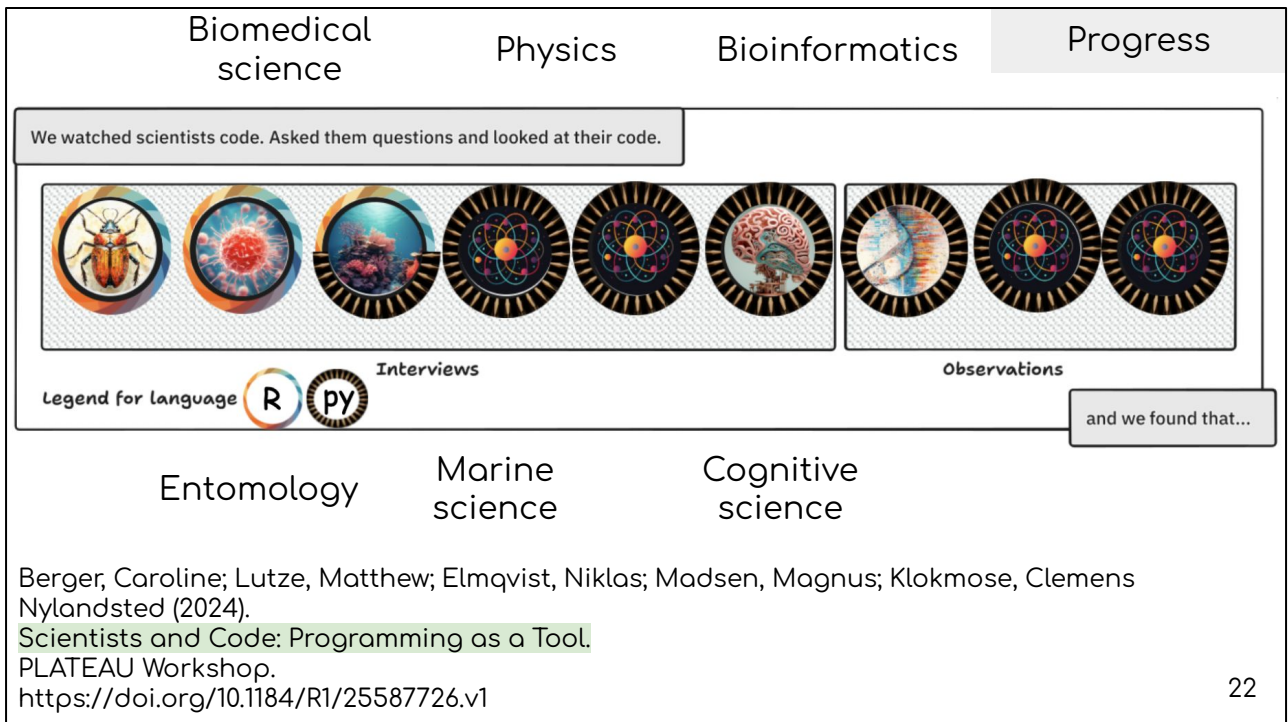
Lyra 2 is an environment which enables people to author interactions by demonstration.

These tools serve as background and inspiration for my work.



During the first half of my PhD, I have made progress on three of my research questions. I used contextual inquiry, participatory design, and prototyping to learn more about the programming behaviours of scientists, the tasks and processes that could be better supported with interactivity, and began learning about how to design for authoring interaction.

Now, I will talk through some of the progress I have made.



My first project involved nine scientists from diverse fields including physics, entomology, and bioinformatics. I employed a mixed-methods approach, conducting interviews with six scientists, and directly observing three scientists while they were coding, and analyzing one participants' code repository. The project was small scale in terms of participants, but included depth and variety by including observation and code review, and scientists from different disciplines.

The study aimed to answer three central questions: What programming practices do scientists actually use in their work? How do current tools support those practices? And importantly, how do these tools sometimes hinder scientific progress? Our findings offer valuable insights into the computational ecosystem of scientists and highlight opportunities for better tool design. Our work was reviewed and accepted, and we presented it at the workshop on the intersection between human-computer interaction and programming languages last year.

They translate hand drawn derivations...

$$\begin{aligned}
 dE &= \frac{\sigma \cdot 2\pi r dr}{4\pi\epsilon_0(x^2+r^2)^{3/2}} = \frac{\sigma \cdot x}{2\epsilon_0} \cdot \frac{r dr}{(x^2+r^2)^{3/2}} = \frac{\sigma x}{2\epsilon_0} \frac{(r dr)}{(x^2+r^2)^{3/2}} \\
 E &= \int_0^R \frac{\sigma x}{2\epsilon_0} \frac{r dr}{(x^2+r^2)^{3/2}} \\
 E &= \frac{\sigma x}{2\epsilon_0} \int \frac{r dr}{(x^2+r^2)^{3/2}} = \frac{\sigma x}{2\epsilon_0} \int \frac{1}{t^{3/2}} \frac{dt}{2} \\
 E &= \frac{\sigma x}{2\epsilon_0} \left[-\frac{1}{t^{1/2}} \right]_x^{\sqrt{x^2+R^2}} \\
 E &= \frac{\sigma x}{2\epsilon_0} \left[\frac{1}{x} - \frac{1}{\sqrt{x^2+R^2}} \right] \\
 E &= \frac{\sigma}{2\epsilon_0} \left[\frac{x}{x} - \frac{x}{\sqrt{x^2+R^2}} \right] \\
 E &= \frac{\sigma}{2\epsilon_0} \left[1 - \frac{x}{\sqrt{x^2+R^2}} \right]
 \end{aligned}$$

```

# Baseline
# README with instructions for content
# - outward-facing
# - no Github CI
# contains:
# - jupyter notebooks
# - scripts folder
# - tests folder
# - no semantic commit messages
# - most commits are adding models/data
# - print debugging
# - print inside function
# - function closes over top-level def that comes from data
# - published with error reported in notebook
# - (execution record is security issue (leaks path etc.))
# - inline string manipulation, should be helper
# - no error handling
# invariants assume nonempty files
# effort made to ad-hoc fiddle with display
# dead comment

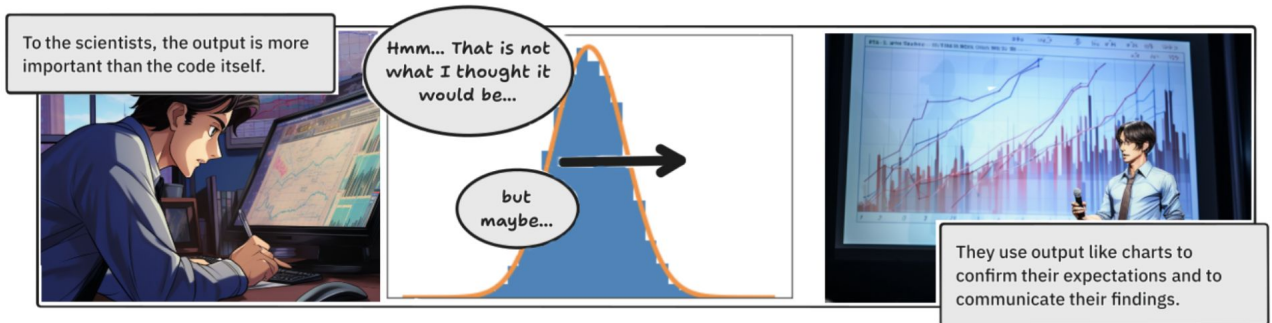
# Simple Analysis
# lots of uses of 'copy': aliasing is an important challenge
"observations" 511 16018

```

...to mathematical code that is experimental and deviates from software engineering best practices.

The most significant finding from this research is that scientists fundamentally view programming as a tool rather than an end product. They prioritize what the program produces over the quality of the code itself. This means traditional software engineering best practices are often set aside in favor of getting results.

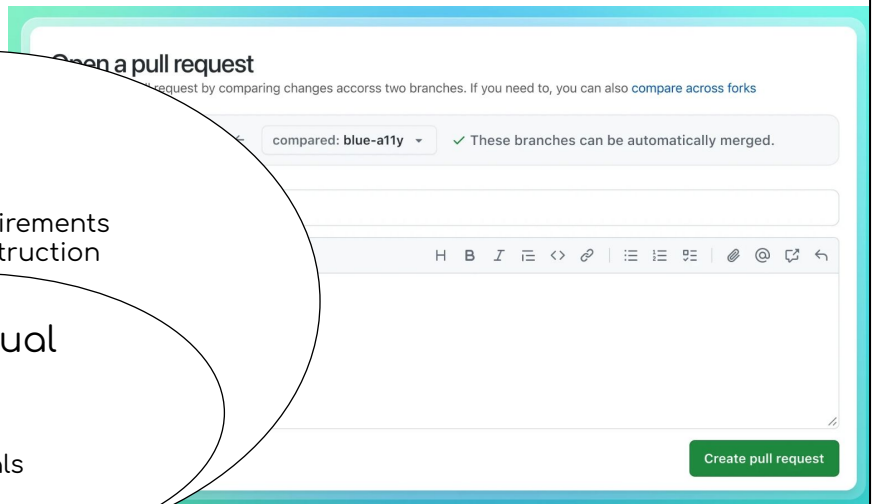
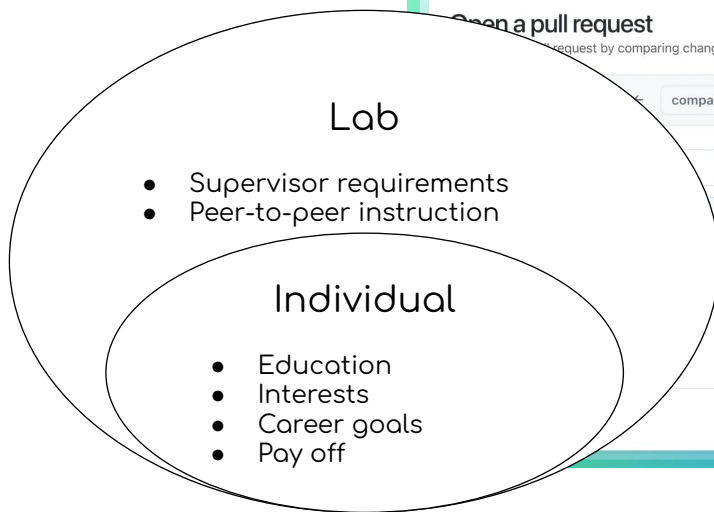
I observed two common programming approaches. First, scientists frequently rely on copy-paste programming, borrowing code from documentation, Stack Overflow, and increasingly, ChatGPT. Second, they manually translate mathematical formulas to code, which can be tedious and error-prone. Overall, scientists take an experimental approach to coding, focused on whether the output meets their expectations.



Visualizations play a dual role in scientific computing. First, they serve as tools for confirmatory analysis, where scientists produce quick, iterative visualizations to verify their expectations. In one observation, a physicist generated multiple plots in rapid succession, discarding those that didn't match expected outcomes.

Second, visualizations are essential communication tools. I found what I call "audience-driven visualization quality" - meaning the fidelity and time invested in visualizations scale with their intended visibility. A quick plot for personal use might be rough, while a visualization for publication receives extensive refinement.

This visualization process is deeply iterative. Scientists create visualizations, compare them to expectations, discard unhelpful ones, and continually refine those that will be used to communicate findings to colleagues or in publications.



Despite the collaborative nature of science, programming remains largely an individual effort. Code sharing does happen among scientists, but typically requires significant rework to adapt to new contexts. The study found a notable absence of formal code review practices.

I observed interesting patterns in how scientists appropriate tools. GitHub, for instance, was perceived quite differently from its intended purpose. Some viewed it as a publishing platform rather than a version control system. Others saw it as an elusive but potentially useful tool they hadn't quite figured out how to incorporate into their workflow.

Tool adoption among scientists is influenced by several factors: personal considerations like interest and career goals; social factors such as lab culture and supervisor requirements; and a practical assessment of whether the learning investment will pay off in their work.

Copy-and-Paste + LLMs → improved code quality

Handwritten equations → code

code



Scientific programming ≠ software engineering

So, scientific programming tools should be different.

26

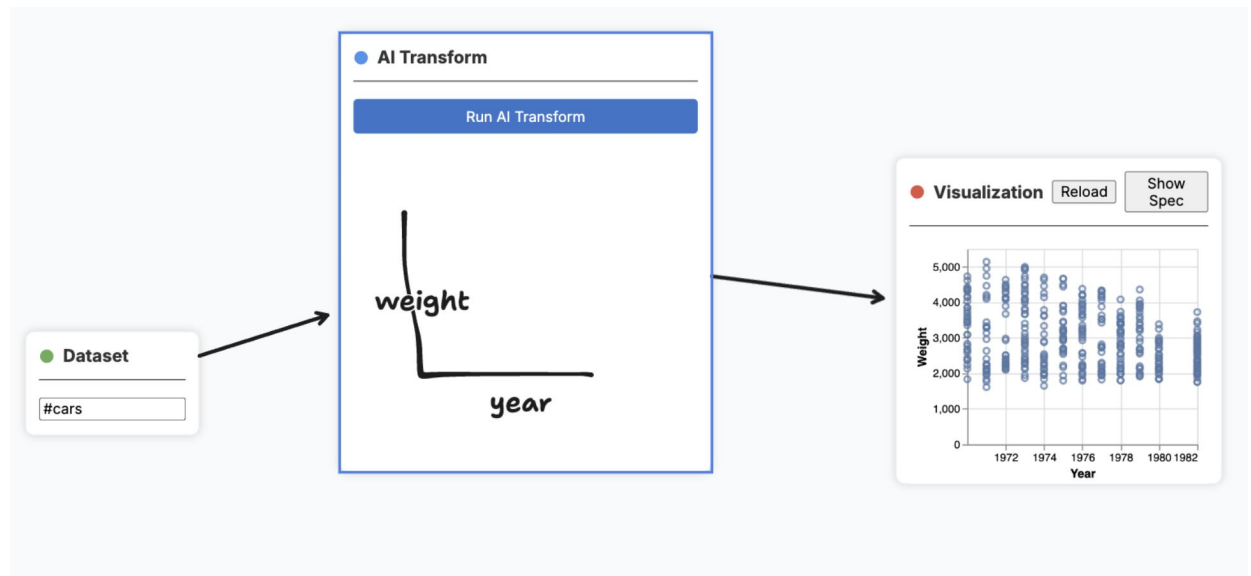
Based on our findings, I identified several promising design opportunities. First, since copy-paste programming is prevalent, tools could better support this workflow by integrating LLMs to help evaluate code quality or providing accessible static analysis tools.

Second, given the challenges of translating mathematical formalism to code, new tools could assist this process - perhaps using computer vision combined with LLMs to convert handwritten equations directly to efficient code.

Third, I suggest creating intermediaries between code and visualization that support collaborative editing of outputs while preserving reproducibility.

Finally, there's a clear need for simplified version control alternatives to Git and GitHub that integrate directly into scientific workflows and support tracking of ephemeral plots and outputs.

These opportunities highlight how programming tools could better align with scientists' actual practices rather than imposing software engineering standards that may not serve their primary goals of scientific discovery.



Following the study with scientists, I created prototypes in collaboration with my colleague, Marcel to explore how computational tools could better support scientific tasks.

One key insight from our contextual inquiry was that scientists use programming mainly as a means to create explanatory visualizations.

So, I set out to design a tool that would lower the barrier to creating charts—while still allowing customization through code.

The result was *QuickCharts*, built upon a tool called *tlDraw*.

QuickCharts is meant to be accessible to non-programmers, support fast iteration, exploratory analysis, and comparative views.

Users add a Dataset panel, AI Transform panel, and Visualization panel to a canvas, connecting them with arrows. They can draw or write in the AI panel, run the transform, and get a chart—while also being able to edit the Vega-Lite spec under the hood.

Originally, AI wasn't part of the project, but as the landscape changed, we started integrating AI into our prototypes to enhance flexibility and ease of use. I used *QuickCharts* as a thinking tool, but ultimately chose not to present it to the scientists during the workshop as I didn't want to constrain their thinking by showing them a tool that too closely corresponded to their needs.



PARTICIPANT	ROLE	TOPIC
P1	PhD Student	Meroplankton
P2	PhD Student	Fisheries
P3	Post-Doc	Marine Snow
P4	PhD Student	Bacteria in Coral Reefs
P5	PI	Microbiology
P6	PhD Student	Bacteria and Ecology
P7	Post-Doc	Marine Predators

Building on our contextual inquiry and prototyping, the next step was to explore what tools *could* be—specifically, what’s needed to help scientists author interactive elements. This led to a participatory design project with ocean scientists.

In March 2025, I conducted a workshop using the Future Workshop method to explore the needs of ocean scientists around data visualization. Seven marine biologists from the Woods Hole Oceanographic Institution participated, including PhD students, postdocs, and a principal investigator. The workshop was co-facilitated by members of the MIT Visualization Group.

The session lasted about three hours and progressed through critique, fantasy, and realization phases. Participants began by identifying their biggest data visualization challenges, imagined ideal solutions, and finally engaged with working prototypes. I closed with a hands-on tutorial using Plotly Express, chosen for its popularity among Python users and strong support for map visualizations.



(P7, Post-doc researching Marine Predators)

"How do I show data for 80 species without watering down the story?" (P2, Fisheries PhD Student)

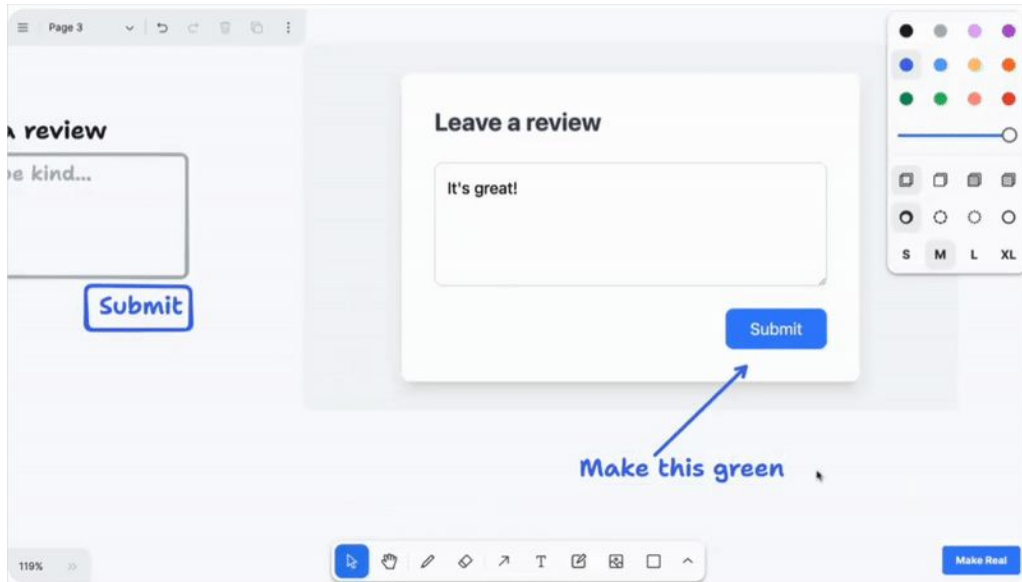
29

Participants expressed a range of motivations for visualizing data—some focused on understanding complex ecological patterns, like Arctic community composition or microbial diversity in coral reefs. Others aimed to communicate findings to stakeholders, such as protected area managers.

A common challenge was integrating multidimensional data, such as combining spatial, depth, and species information into a single visual. Grouping large datasets—sometimes involving dozens of species—was another key difficulty. Scientists noted how these grouping decisions can obscure the narrative they're trying to tell.

During the critique phase, P7 draws the problem she is having with combining depth data, environmental variables, and position information.

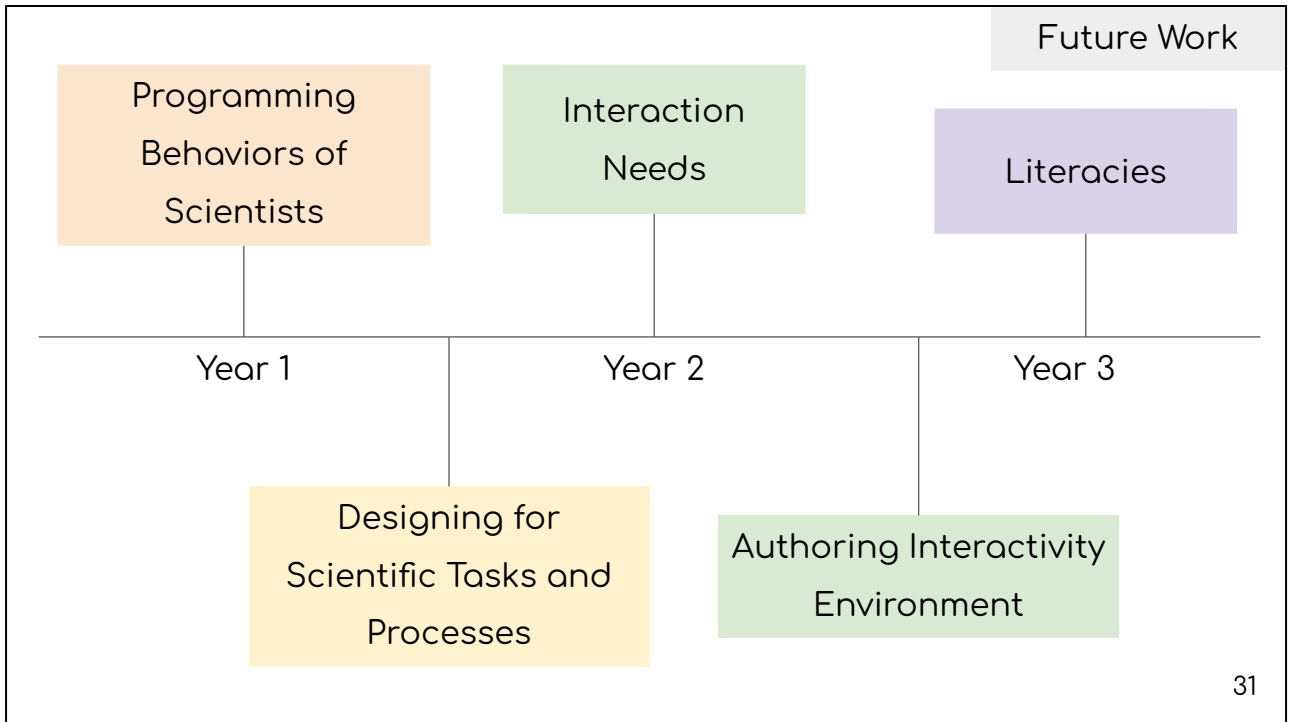
As one participant put it: 'How do I show data for 80 species without watering down the story?' This tension between detail and clarity was a recurring theme.



Here is *tldraw*, a tool that supports users in creating interfaces using annotations and designs on a canvas. It has an LLM under the hood to transform user's prompts and images into UIs. We showed *tldraw* to the participants to inspire further brainstorming.

When shown *tldraw*, participants responded positively—especially because it allowed visual editing without coding. One scientist shared frustrations with tools like ggplot2, where even small tweaks required extensive code.

Moving forward, these insights point us toward building systems that reduce coding overhead, support multi-layered data exploration, and make it easier for researchers to share and explain their visuals. The next phase of this project will analyze these needs in more depth to guide the design of new authoring environments.

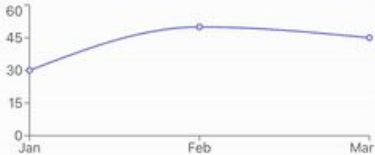


I plan to continue answering the research questions around the environment where authorship of interactivity takes place, and literacies.

Example Gallery

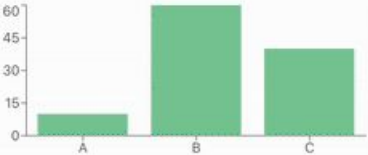
Future Work

Select AllClear All



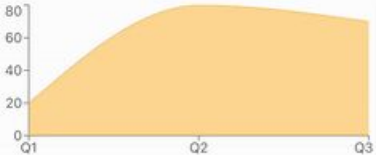
Line Chart

☐ Select for RemixAnnotate



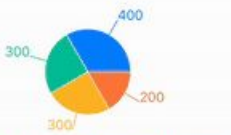
Bar Chart

☐ Select for RemixAnnotate



Area Chart

☐ Select for RemixAnnotate



Pie Chart

☐ Select for RemixAnnotate

<https://may-20-prototype.vercel.app/>

32

To do this, I plan to document design ideas from the participatory design workshop and develop a prototype based on these ideas. I will deploy the prototype to oceanographers involved in the workshop and gather their feedback on the tool. I have begun to look at environments that already exist, and have tried to replicate the scientists use cases within these environments. I have also started to prototype out ideas, such as a remixed example gallery.

First the user would select a chart and indicate what it is they like about the chart. Then they would remix a few of them, to be able to customize and further iterate.

Programming Literacy

- Way to learn about everything else (Guzdial, 2022)
- Insight generation opportunities through new representation (Disessa, 2001)
- Programming as a medium (Kay, 1995)

Visualization Literacy

- Ability to extract goal directed information from visual display (Boy et al., 2014)
- Understand patterns, trends, and correlations (Börner et al, 2016.)
- Moving "beyond the data" (Curcio, 1987)

Interaction Literacy

- Understanding functional principles (Carolus et al., 2023)
- Control over appearance, behavior, and transitions of visual elements and in response to actions

34

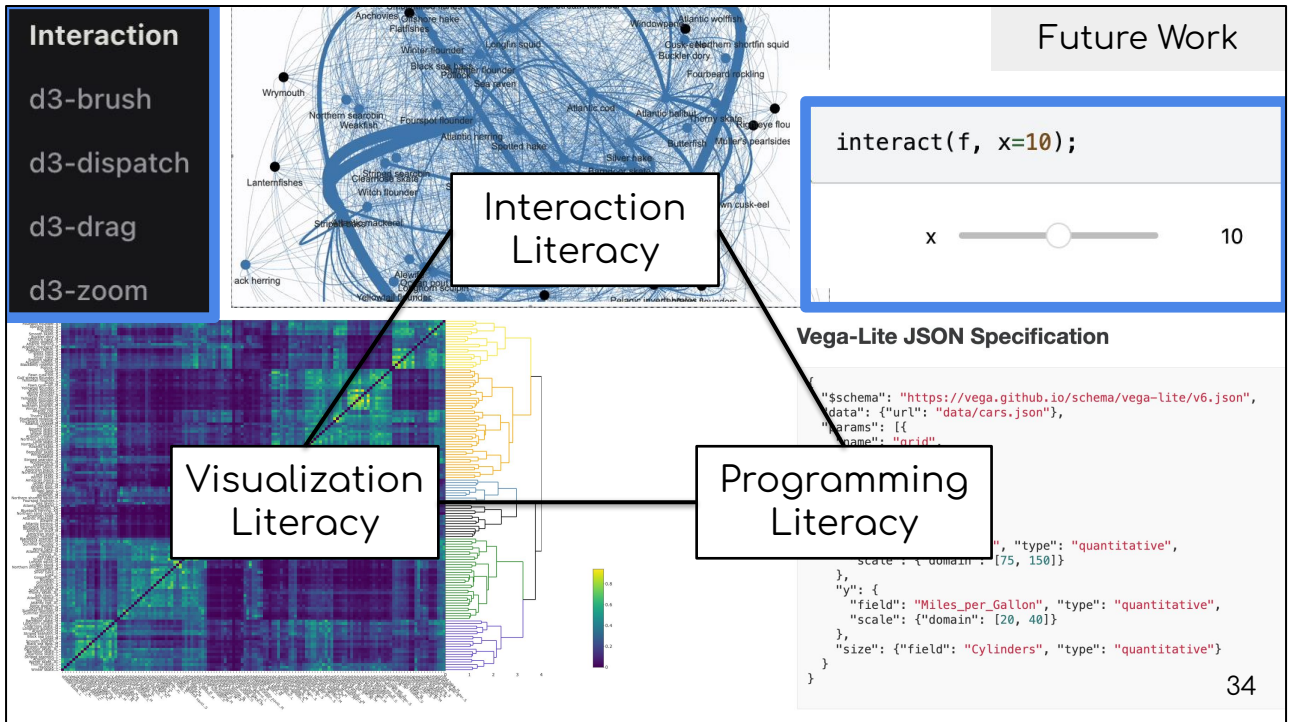
I plan to work on understanding the relationship between programming literacy, visualization literacy and interaction literacy for scientists.

Programming is a way to learn about everything else, including a way to dive deeper into science, those that are literate in programming have the opportunity to generate novel insights through new representation. Gaining a fluency in programming helps someone to use programming as a medium, akin to writing.

Visualization literacy, on the other hand has to do with extracting information from a visual display based on one's goals, this includes making sense of patterns, trends and correlations. Importantly, high literacy means going beyond the data to making inferences, predictions, or drawing conclusions.

Interaction literacy is about understanding the functional principles of a technology as a user, but in our case, it also includes control over the appearance, behavior, and transitions of a visual elements

and in response to a user's actions.



Take D3, a highly expressive javascript visualization library. Using D3 requires fluency in all three forms of literacy. In another dimension, Jupyter interact, while programmatic is limited in terms of customization.

By learning about the relationship between these literacies I may be able to create tools that have a low barrier of entry and are expressive.

Publications

Completed

1. Berger, Caroline; Lutze, Matthew; Elmqvist, Niklas; Madsen, Magnus; Klokmoose, Clemens Nylandsted (2024). *Scientists and Code: Programming as a Tool*. PLATEAU Workshop.
<https://doi.org/10.1184/R1/25587726.v1>

Planned

2. Outcomes of work with Oceanographers - CHI 2026
3. Theoretical Contribution - ACM Transactions on Computer-Human Interaction

I have one publication so far, and plan to have two more. One I plan to submit to CHI, that includes outcomes from the participatory design work with the marine biologists. For the theoretical contribution, I would like to submit this work to TOCHI.

Semi-Formal Programming Designing for Semi-formal Programming with Foundation Models

Apr 7, 2025

Josh Pollock¹, Ian Arawjo², Caroline Berger³ and Arvind Satyanarayan¹

CHI 2025 Tools for Thought Workshop

Ryan Yen, Josh Pollock, Caroline Berger, Arvind Satyanarayan

¹MIT, Cambridge, USA

²Université de Montréal, Montréal, Canada

³Aarhus University, Aarhus, Denmark

collaborations

GPTerraform: A GPT-Based Conversational Website Editor

ANNA GYORFFY, Aarhus University, Denmark

CAROLINE BERGER, Aarhus University, Denmark

CLEMENS NYLANDSTED KLOKMOSE, Aarhus University, Denmark

Constructing Gingerbread Houses: Computer-Mediated Collaboration

CAROLINE BERGER*, Aarhus University, Denmark

ARVIND SRINIVASAN*, Aarhus University, Denmark

MEAGAN B. LOERAKKER, Chalmers University of Technology, Sweden

ALINA LUSHNIKOVA*, University of Luxembourg, Luxembourg

RESEARCH-ARTICLE | [OPEN ACCESS](#) |  

On-body Icons: Designing a 3D Interface for Launching Apps in Augmented Reality

Authors:  [Uliana Tsimbalistaia](#),  [Caroline Berger](#),  [Hans Gellersen](#),  [Pavel Manakhov](#) | [Authors Info & Claims](#)

CHI '25: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems • Article No.: 629, Pages 1 - 15
<https://doi-org.ez.statsbiblioteket.dk/10.1145/3706598.3713954>

36

In addition to the work pertaining to the project that will contribute to my final thesis, I have also been involved in collaborations with researchers from my institution, around Europe, and internationally.

With the MIT Visualization group, I have been working on semi-formal programming concepts, and have co-authored two workshop papers in this domain.

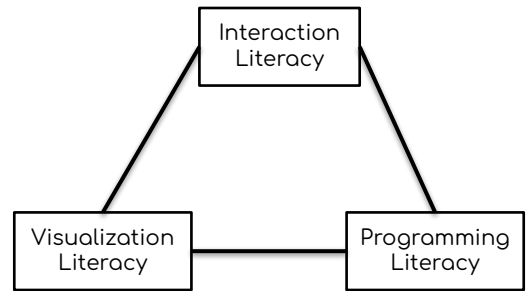
I have helped mentor a master's student in her research on LLM tools to support programming.

Following a winter course in Theory in HCI, I worked with PhD students from several universities around Europe on a manuscript about research-through-design and crafting as a way to interact with theory.

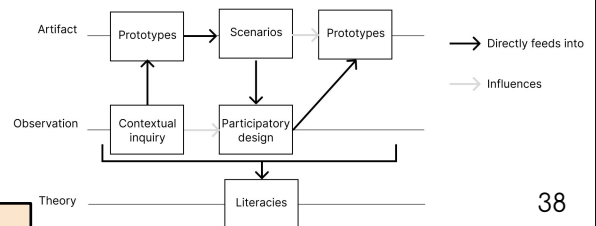
In collaboration with researchers at my institution and in Russia, I provided subject matter expertise on qualitative data analysis, and co-authored an article pertaining to augmented reality.



Before I conclude, I would like to highlight the best parts of my PhD so far. My colleagues, at Aarhus and MIT have been excellent office sharers and beer accompaniers along with thought partners. And of course DreamCake, a brand new discovery for me, and arguably one of my favorite parts of living in Denmark. The best one I have tried so far is served at the cafeteria of the main Aarhus library. This also has been a good reward for progress.



CAROLINE BERGER
CAROLINE.BERGER@CS.AU.DK



38

My PhD research investigates programming environments specifically designed for scientists, examining how scientists interact with programming tools in their daily work. Using theories of tools and situated action, I study the complex relationship between scientific workflows and the programming environments that support them.

Through contextual inquiry, I observed scientists in their natural work environments to gathered real-time data on programming behaviors. Complementing this, participatory design workshops engage scientists in collaboratively ideating features that would make programming tools more accessible and intuitive for scientific workflows.

Moving forward, I aim to develop a theoretical framework examining the intersection of visualization, programming, and interaction literacy among scientists. This framework will inform the design of tools that enable scientists to author interactive visualizations without requiring extensive programming expertise, ultimately creating programming environments better tailored to scientific research needs.

Now I am happy to take questions, and to discuss.

References

Slide 2

- Moon - https://www.reddit.com/r/space/comments/xtobwp/one_of_the_sharpest_moon_image_i_ever_captured/
- Telescope - <https://www.livescience.com/space/astronomy/types-of-telescope>

Slide 3

- Notebook - <https://datalab.marine.rutgers.edu/2020/10/how-to-share-and-run-python-notebooks/>
- Scientist - <https://www.whoi.edu/multimedia/dna-detective/>

Slide 17

- Gamel - <https://www.drumsforschools.com/product/gamelan-beater-gamel-premium/>
- Hammer - <https://en.lial.biz/ivan-shoe-hammer>
- Musicians - <https://www.teachsecondary.com/humanities/view/lesson-plan-ks3-4-music-introducing-gamelan>
- Scientist with tool - <https://www.whoi.edu/multimedia/soundscapes/>

Slide 18

- Phone - https://www.youtube.com/watch?v=Q_pKlsXA-Is

References cont.

Slide 19

- Basili et al., "Understanding the High-Performance-Computing Community: A Software Engineer's Perspective," in IEEE Software, vol. 25, no. 4, July-Aug. 2008, doi: 10.1109/MS.2008.103
- Guo, 2012. Software tools to facilitate research programming. Stanford University.
- Morris et al., "Some challenges facing scientific software developers: The case of molecular biology," in IEEE International Conference on e-Science, doi: 10.1109/e-Science.2009.38
- Nouwens et al., "Between Scripts and Applications: Computational Media for the Frontier of Nanoscience," in CHI '20, doi: 10.1145/3313831.3376287

Slide 20

- Manz et al., anywidget, anywidget.dev/
- Vaithilingam et al., "Dynavis: Dynamically synthesized UI widgets for visualization editing," in CHI '24 doi: 10.1145/3613904.3642639
- Zong et al., 2020. Lyra 2: Designing interactive visualizations by demonstration. IEEE Transactions on Visualization and Computer Graphics doi: 10.1109/TVCG.2020.3030367

References cont.

Slide 33

- Börner et al. Investigating aspects of data visualization literacy using 20 information visualizations and 273 science museum visitors. *Inf Visual* 2016; 15(3): 198-213.
- Boy, Jeremy, et al. "A principled way of assessing visualization literacy." *IEEE transactions on visualization and computer graphics* 20.12 (2014): 1963-1972.
- Carolus, Astrid, et al. "Digital interaction literacy model-Conceptualizing competencies for literate interactions with voice-based AI systems." *Computers and Education: Artificial Intelligence* 4 (2023): 100114.
- Curcio, F. (1987). Comprehension of mathematical relationships expressed in graphs. *Journal for Research in Mathematics Education*, 18 (5), 382-393.
- diSessa, Andrea A. 2001. *Changing Minds*. Cambridge, MA: MIT Press.
- Kay, Alan C. 1995. "Computers, Networks and Education." *Scientific American* 272 (3): 148-155.
- Mark Guzdiol, 2022. "Providing Students with Computational Literacy for Learning About Everything", *Computational Thinking Education in K-12: Artificial Intelligence Literacy and Physical Computing*, Siu-Cheung Kong, Harold Abelson

Slide 34

- Heat map - https://fwdp.shinyapps.io/tm2020/#4_DIET_OVERLAP_AND_TROPHIC_GUILDS
- Network visualization - https://heatherwelch.shinyapps.io/beyond_temperature/