



# ELEPHANT

IN THE LAB

OPINION

## Improving software infrastructure in multidisciplinary research

<b>Short title</b>	Improving software infrastructure
<b>Author</b>	Jacqueline M. Kory-Westlund <sup>1</sup>
<b>Author affiliation</b>	<sup>1</sup> Alumni of the MIT Media Lab
<b>Author bio</b>	Dr. Jacqueline M. Kory-Westlund is a mother, artist, writer, and scholar. She earned her PhD from the MIT Media Lab in 2019, where her research focused on social robotics, children's learning and relationships, and robot ethics. She blogs regularly at <a href="#">DeliberatedOwl</a> .
<b>Author social links</b>	<a href="#">Twitter</a> - <a href="#">LinkedIn</a> - <a href="#">GitHub</a> - <a href="#">Personal Website</a>
<b>Date published</b>	11 March 2020
<b>DOI</b>	10.5281/zenodo.3702301
<b>Cite as (APA)</b>	Kory-Westlund, J. Am.. (2020). Improving software infrastructure in multidisciplinary research. <i>Elephant in the Lab</i> . DOI: <a href="https://doi.org/10.5281/zenodo.3702301">https://doi.org/10.5281/zenodo.3702301</a>

## Juggling software, materials, and people

In my third week of grad school, I found myself metaphorically elbows-deep in my human-robot interaction lab's codebase. I was porting a robot teleportation interface from tablet to desktop. The goal: To run a psychology study exploring how [young kids learned language skills with social robots](#).

But comments were few and far between. Documentation consisted of asking older students how things worked. While I did eventually get it done, adding that teleportation interface took a lot longer than I had anticipated.

Unfortunately, this experience is all too common. Software-heavy research projects are frequently plagued with a lack of documentation, a lack of standardization, poor to no maintenance, legacy code, and/or reliance on expensive proprietary software. This may be especially true for projects in highly multidisciplinary fields like mine, since there are so many different balls to juggle. Dealing with infrastructure---such as code, robots, methodologies, assessments, domain knowledge, and more----was a huge part of the challenge.

Interdisciplinary and multidisciplinary researchers like me increasingly rely on a wide collection of software, materials, and people. And the more software, material, and people you rely on, the more inevitable it is that something will break.

## How do we improve our infrastructure?

First, we acknowledge that we *will* rely on other people and other people's software. There's no way I could build everything from the ground up (and also, you know, *graduate*). Science is built incrementally. We *must* build on the work of those who came before. As students in my lab group liked to say, "Human-robot interaction is a team sport."

But even acknowledging that point, the reality is that people move on and software breaks. At some point, maybe not tomorrow, maybe not this year, but at some point, critical code won't be maintained. We can put software into containers, such as Docker, or run the code in a virtual machine on an older operating system, to keep it working longer. But if you ever need to integrate that code with a new framework or new project, chances are, something will no longer be backwards compatible. The company maintaining proprietary software will fold. A library or some other dependency won't get updated. The senior PhD student who monitored a key repository will graduate and leave the project behind. So what do we do?

## Open Source Software

One answer is open source software: Common tools that you don't have to maintain yourself and that are available even if someone graduates or doesn't have funding to pay for a license. Think tools like R, numpy, scipy, QGIS, GIMP, ROS, LibreOffice, Zotero, Latex...

Problem is, that's a take-take-take model. Open source software still faces maintenance, compatibility, and documentation issues, and often lacks key features.

For example, the software I used in some early projects to code human behavior (such as eye gaze) was proprietary and expensive, only ran on one machine, only if the right USB dongle was plugged in ... etc. But it worked pretty well. At some point, we no longer had a license to that software. I still needed to code gaze. I tried out an open source variant that didn't work as well (weird bugs and missing features)---but at least my undergrad research assistants could install it on their laptops.

To make open source truly viable as a research tool, we need to give back---which doesn't mean you have to volunteer yourself. One easy idea is to add line items to your grant budgets for donations to critical open source projects. Or go a step farther: include funding to hire a software engineer who can directly contribute to critical projects. Investing in the projects we rely on means they'll still be around when we need them.

## Pipelines and Documentation

Documentation and pipelines for transmitting project knowledge from older graduate students to newer students can be critical. I was dependent on legacy code developed by earlier generations of grad students. Consider: If your group has a massive, largely undocumented codebase, at what point will newer graduate students throw it away (despite its purported many useful features) because they don't understand it and feel it would be easier to rebuild from scratch? And how do students learn the common methodologies in your lab group?

Faced with problems like these, part of my mission as a graduate student became leaving every project I worked on better documented than when I found it. Here are three key things I did:

- I gave every git repository an extensive readme explaining how to use the repo, install and run instructions, dependencies, design decisions, etc.
- I documented every process I followed that I felt others might need to replicate, step-by-step, on our group's wiki site. All students could edit the wiki. For example, I had one page explaining how I recorded audio for our robots, with

details about the software I used for recording, cleaning up the audio and noise removal, shifting the pitch, where to save the files, and how to play the files on the robot.

- I finished off every project with a public-facing blog post or a project page online, so people could get the gist without reading the whole academic paper. For example, here's a blog post about [kids' relationships and learning](#) with social robots, from my dissertation work. Here's a project page about my Master's thesis project, a [robot that played storytelling games](#) with kids.

I can't count the number of times I re-read my own documentation when trying to remember how or why I'd done different things.

Another idea: In one lab group I observed, every new student was assigned a "buddy" who had been in the group for a year or more. The more experienced student was responsible for getting the new student up to speed, answering all their questions about group methodologies, codecases, materials, and so forth.

I've also seen groups hire someone part-time to do documentation and maintenance work. Sometimes this kind of stuff is rolled into a lab manager position.

## Collaborate

I benefited from collaborations with people who had relevant expertise I lacked---such as domain knowledge in developmental psychology and education. How do you find collaborators? My first stop was always to ask friends, colleagues, or my advisor. My advisor knew many people and was excellent at connecting students with the right folks.

Find people in other fields who seem to be excited by the same kinds of questions as you. You can find these people at conferences (browse the talk listing and attend the ones that pique your interest), on Twitter, by reading papers and looking up the authors.

Then introduce yourself. Ask questions. I've found that most people in academia are very happy to talk about their own research field and their area of expertise. I mean, it makes sense: they wouldn't be in that field if they weren't passionate about it! This is to your advantage. Be friendly. Take notes.

Be willing to share your expertise as well. Research works better when we help each other out.

## Open source Academia?

I've seen a trend lately toward increasingly open science. Frequently, the conversation is about open source software and open datasets. We should also talk about open methodologies, open assessments, and open whatever-else-is-used-in-your-research. I've made many of my experimental study protocols, assessments, and code repositories publicly available. I'm excited about the questions my work tries to answer. If someone else is excited, too, and also wants to work on these questions, then I would love to be able to help them.