NYU TANDON SCHOOL OF ENGINEERING

Do I Have to Stop Programming?

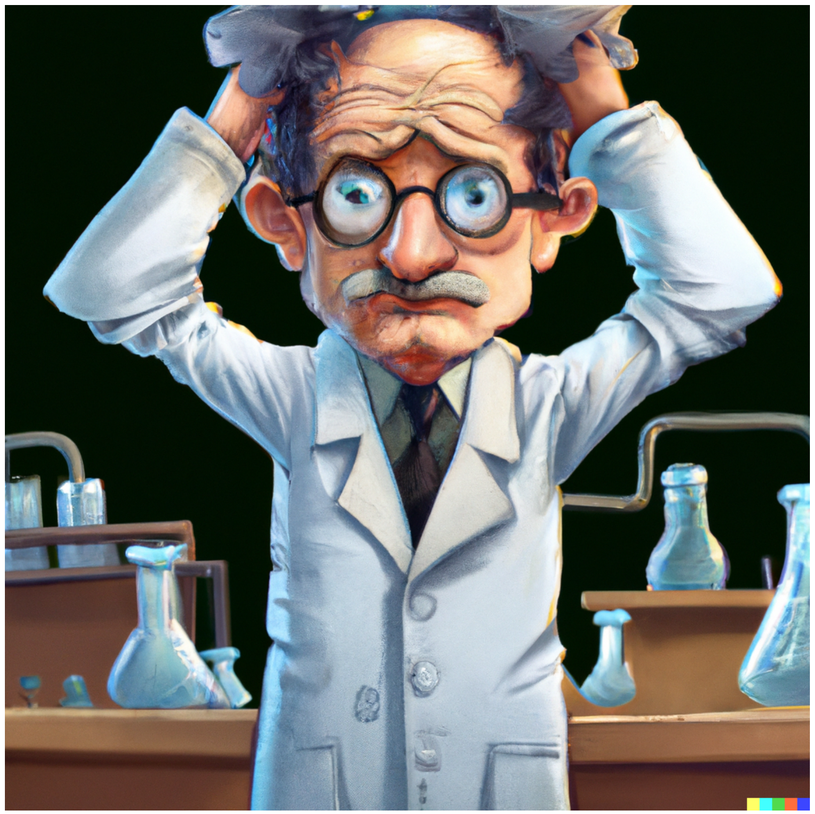
The Plight of the “Hackademic”

Brendan Dolan-Gavitt

ISSRE 2022 New Faculty Symposium

CENTER FOR CYBER SECURITY

My Worries As a Junior Prof
(Abridged)



- How do I find students?
- How can I pay them? How do I get grants?
- How do I learn to teach well?
- How many projects can I juggle at the same time?
- What internal/external service should I say yes to?
- Do I still get to have a life outside of work?
- How can I manage and mentor students effectively?
- **Am I going to have to give up hands-on technical work?**

The Usual Story

During PhD

- Develop world-class technical expertise
- In the software/systems world, this means you spend a *lot* of time gaining technical skills:
  - Programming
  - Debugging complex systems
  - Getting other people's code to run
- Success often depends on how well you, personally, can get things done

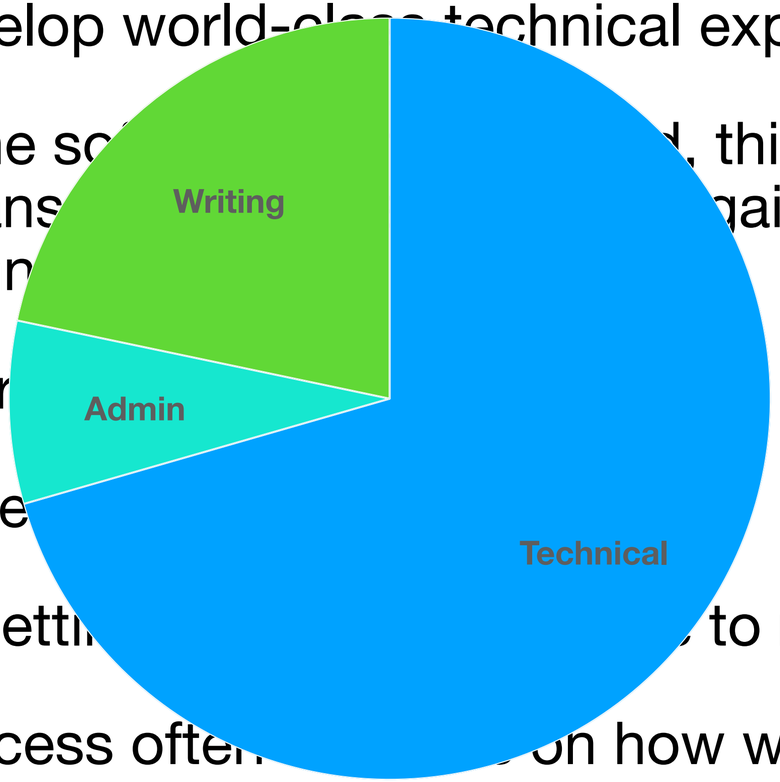
After PhD

- Build a strong group of PhD students
- Get and manage funding for your students
- Plan out multi-year research projects
- Mentor students, build their skills, make sure they're making good progress
- Prepare and teach compelling courses
- Participate in internal service and service to your academic community

The Usual Story

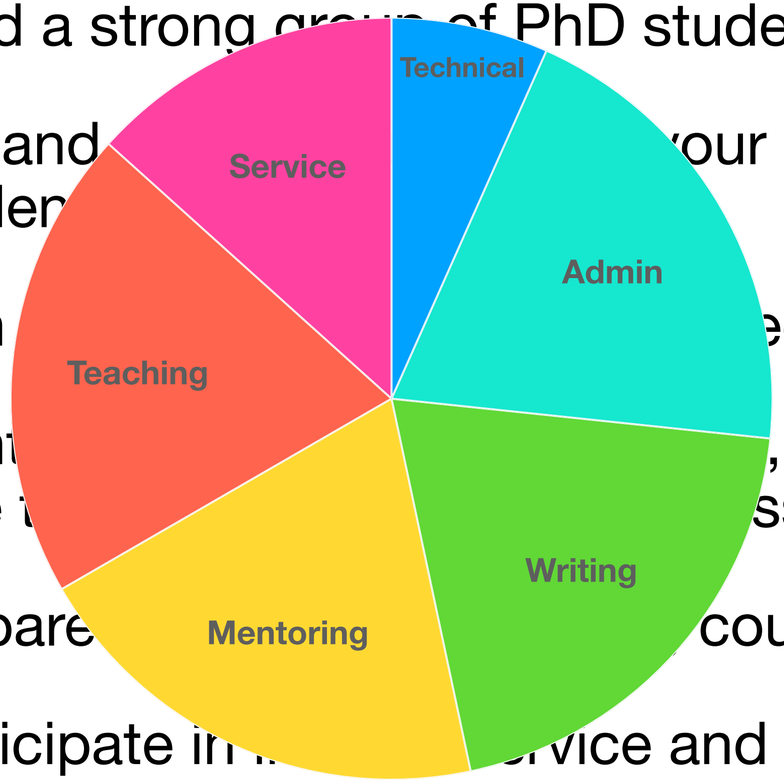
During PhD

- Develop world-class technical expertise
- In the sc d, this means gaining techn
  - Pr
  - De
  - Getti to run
- Success often on how well you, personally, can get things done



After PhD

- Build a strong group of PhD students
- Get and your studen
- Plan ects
- Ment , make sure t ss
- Prepare courses
- Participate in rvice and service to your academic community



A Confession

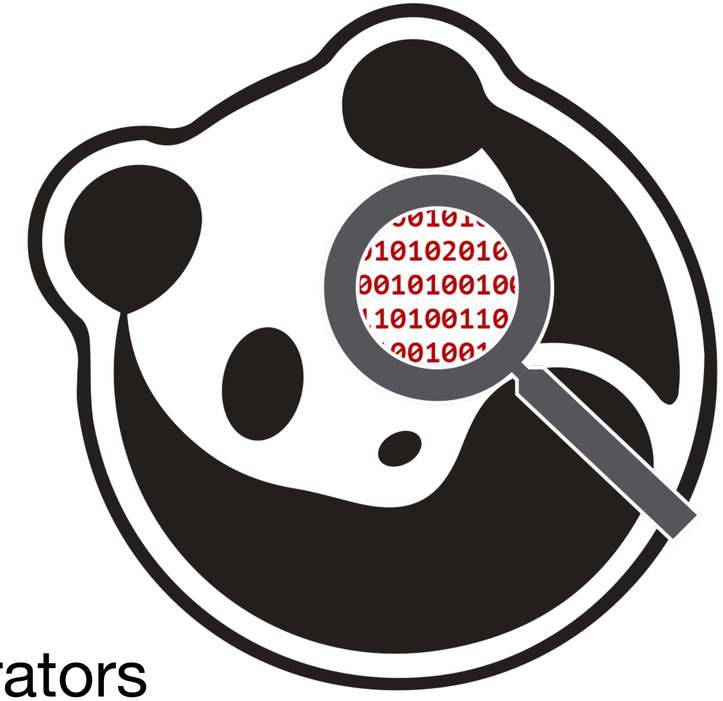
- Actually, I decided pretty early on that I would be miserable if I gave up technical work
- So most of this talk is about retroactively justifying my addiction to programming ;)
- But I do genuinely believe the arguments I'll make here!

The Case for Coding

- I will try to argue that there are good reasons to keep doing low-level technical work after the PhD:
  - To give meaningful help when students get stuck on tricky technical problems
  - To build **infrastructure** that can be used by many students over many years
  - To explore **new ideas** and research areas that may be too risky for a PhD student to take on yet
  - To help create **collaborations** and **outreach** as people use and build on your software

Case Study I: PANDA

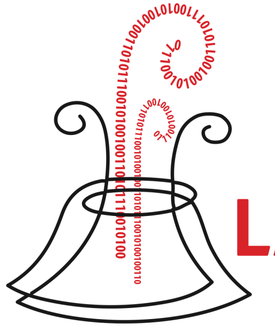
Programming as Infrastructure



- PANDA is a fork of QEMU designed for whole-system dynamic analysis + record/replay
- Started building it in 2013 (still in grad school) w/collaborators at MIT Lincoln Lab
- Fairly successful – 2K stars on GitHub, lots of community support
- Many projects from my lab & others' have been able to use PANDA as a base to investigate cool new research ideas!

Case Study I: PANDA

Research using PANDA



IRQDebloat: Reducing Driver Attack Surface in Embedded Devices

Zhenghao Hu
New York University
huzh@nyu.edu

Brendan Dolan-Gavitt
New York University
brendandg@nyu.edu

Drifuzz: Harvesting Bugs in Device Drivers from Golden Seeds

Zekun Shen
New York University
zekun.shen@nyu.edu

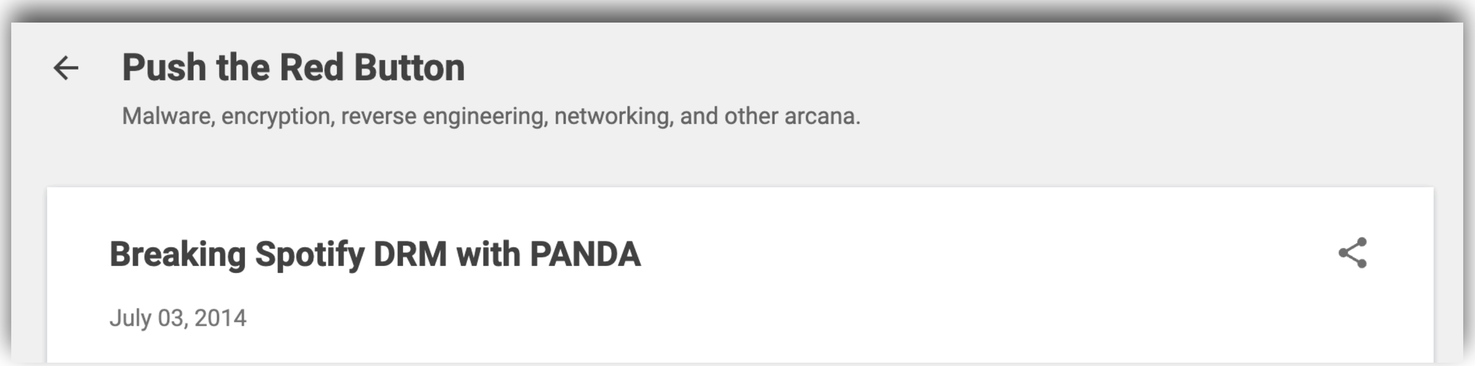
Ritik Roongta
New York University
ritik.r@nyu.edu

Brendan Dolan-Gavitt
New York University
brendandg@nyu.edu

Case Study I: PANDA

Lessons Learned

- Building *infrastructure* software can be a great accelerant for research
- Finding good collaborators in industry who find your software useful helps a lot – the bulk of PANDA maintenance is done by MIT LL folks
- Publicize what you've done! I gave talks on PANDA at industry conferences like REcon, wrote blog posts about it, talked about ongoing dev work on Twitter, etc.



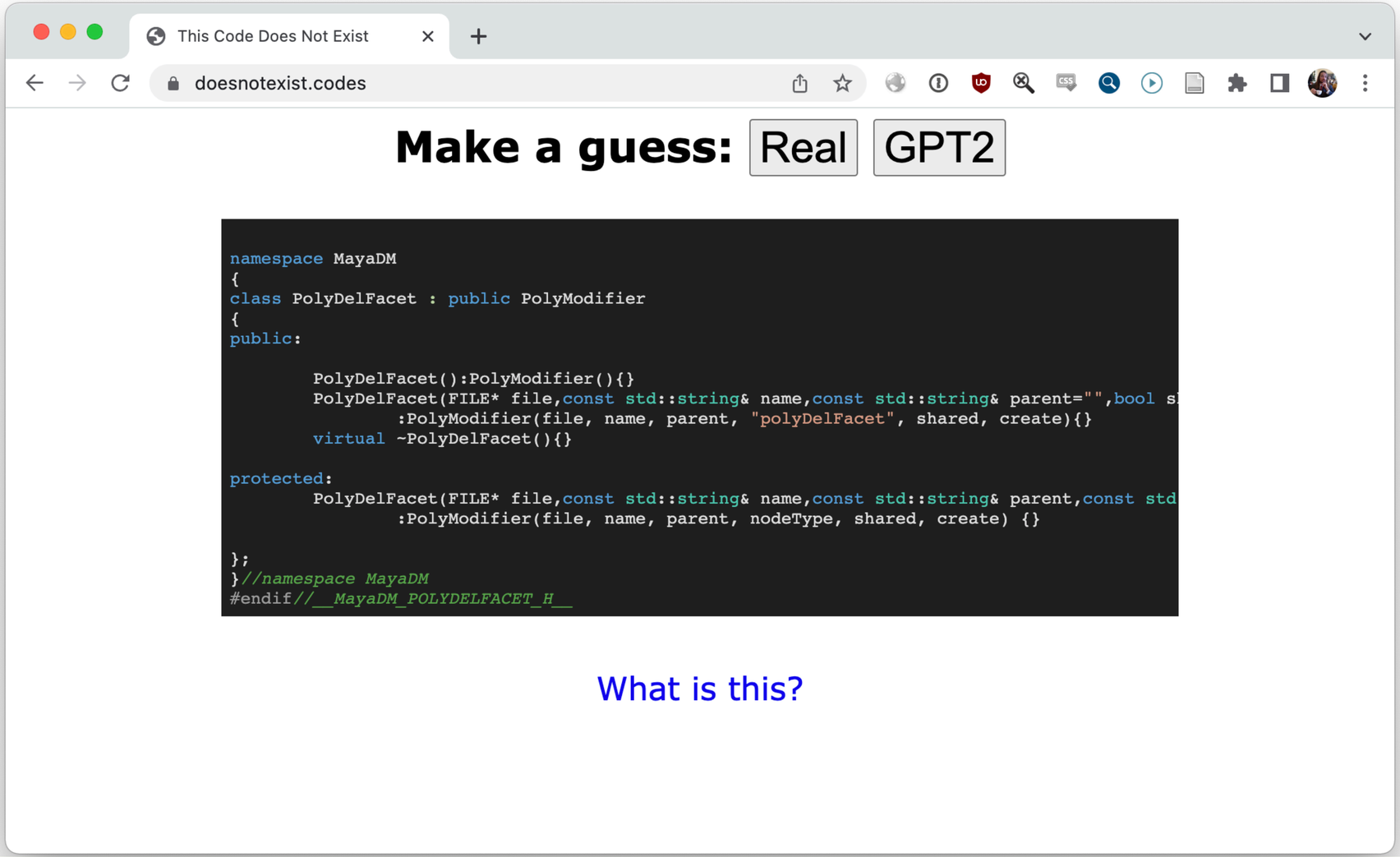
Case Study II: GPT-CSRC

Programming as Experimentation

- After GPT-2 was released I got interested in the possibility of using large language models for program synthesis (late 2020)
- ...but none of my students knew anything about ML!
- **Solution**: play around with building and training ML models like this on my own, explore feasibility
- Trained a 774M parameter GPT2 model on C/C++ source code over winter break
- Built a gamified “guess if code is human or GPT2” site
- Eventually became a key piece of my successful NSF CAREER proposal!

Case Study II: GPT-CSRC

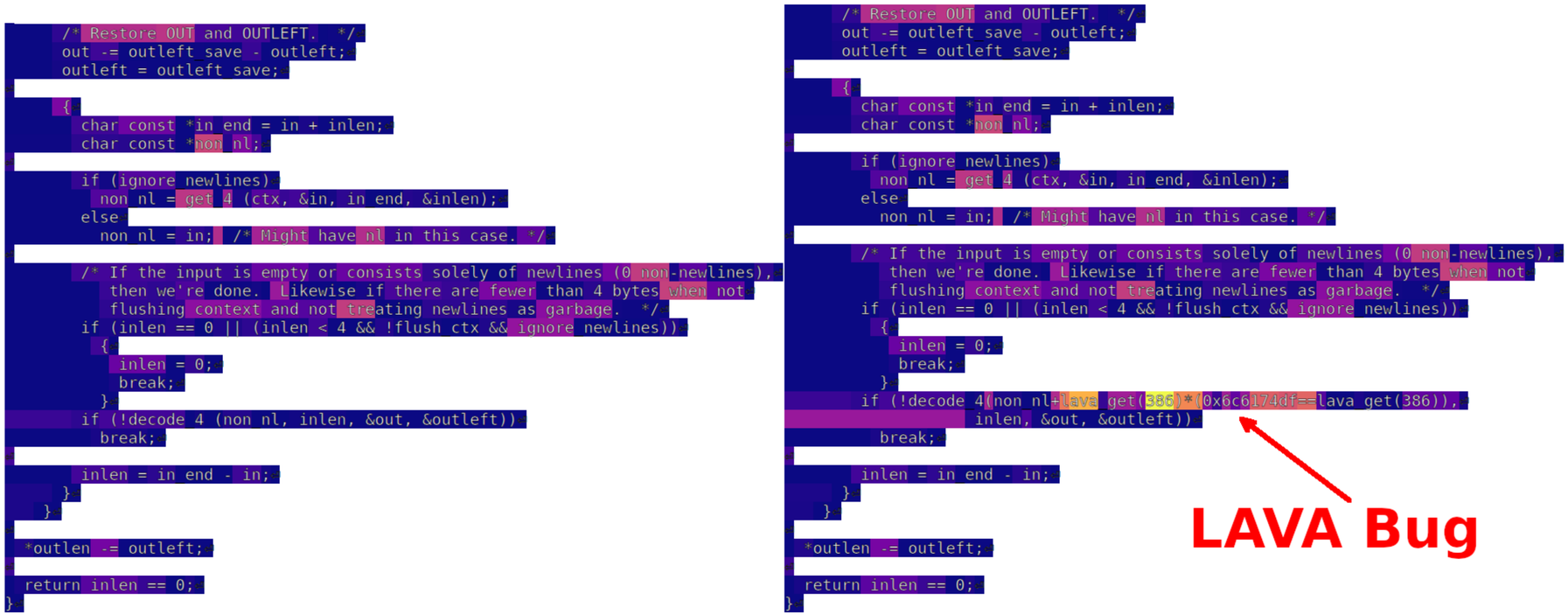
Prototyping



Case Study II: GPT-CSRC

New Research Ideas for Proposal

4 Research Thrust 2: Generality



(a) Original `base64` code.

(b) LAVA-injected `base64` code.

Figure 4: Visualization of the per-token likelihood of original and LAVA-injected code as assessed by a GPT2 model trained on C/C++ code. Brighter colors indicate tokens that are surprising to the model and may be unrealistic.

Case Study III: FauxPilot

Programming as Outreach

- When GitHub Copilot came out, we began doing research on it and the underlying Codex model
- Became clear we would need some way to plug in our own models and use them with
- Over summer 2022, I took some open code models from Salesforce, combined them with NVIDIA's Triton & FasterTransformer, and wrapped it in a Copilot-compatible proxy
- Way more popular than I expected: ~6K stars on GitHub in the first month!

Case Study III: FauxPilot

Programming as Outreach

- But what has been really exciting is how the project has contributed to *outreach*
- Lots of users and contributors to help with development and give feedback on functionality
- GitLab has decided to use it as the basis for their own AI autocompletion offering
- Has been a great way to start conversations with lots of folks in industry who can help answer research questions for us by e.g. sharing data!

Where to find the time?

Where to find the time?

- June

Where to find the time?

- June
- July

Where to find the time?

- June
- July
- August

Where to find the time?

- Just kidding
- There’s winter break too!
- Realistically: you are sometimes going to have to say no to other things
- One strategy I have found helpful is to dedicate a day or half a day each week to technical work, and defend it vigorously

Institutional Barriers & Incentives

How to get your tenure committee to care

- Building software is not rewarded/incentivized in academia the way top conf/ journal publications are
- You will have to keep track of and **quantify** impact to justify it
  - Is industry using your software? (Can you set up some way to let you know when someone is?)
  - Download counts, GitHub stars, other measures of “popularity”
  - What academic work is building on your software? (Make it easy to cite!)

Conclusions

- The adjustment from PhD to prof is not easy, and you won't be able to spend days at a time deep in code (at least during the school year)
- But it is really important to stay in touch with technical work; resist the urge to manage entirely at a high level
- Can pay off very well with new opportunities, big impact!
- Make sure to make these contributions **visible** and **legible** to folks who will be evaluating you!