Occupy Hard Drives:
Making your work more valuable by giving it away

A Polemic based on Anecdata, a.k.a. Experience

Benjamin Weiner (U. Arizona)

Software is an integral part of contemporary astronomy.

Unless you're buying really expensive hardware, most of a grant is spent on salary. What do astronomers do with their research time?

- Make things in a lab (only a few of us), and use hardware (observe) for a few nights
- Write software - data reduction, processing, simulations, catalog handling, making plots ...
- By-hand processing that uses software and could/should be semi-automated (data reduction, quality control, classification ...)
- Write papers and proposals
- Send email
- Look at pictures of cats

That software time is an enormous sunk cost that nobody talks about.

How do we
1) Recoup these costs and make software re-usable?
2) Make sure that people get credit for their work?

Software workers of the world, unite!

No one spends 50% of a grant on hardware and then discards it at the end of the 3 year grant cycle. Yet most software is not shared or re-used. We don't value software work at its true cost, and we don't value it as "doing science."

Widely used software packages (e.g. SExtractor, DAOphot, Galfit) have produced at least as much science as a big telescope/instrument, but are largely the work of a few dedicated data scientists or volunteers.

Why do we let this state of affairs persist?

Because our community doesn't value software as real work or a real expense: we hand it off to students and postdocs, or at the opposite extreme, we hand it off to teams of software engineers and get giant, polished, expensive machines that none of us can service or maintain by ourselves.

How do we improve this situation?

Well, we can try to make the people at the top recognize the problem, and wait for a solution to be handed down. Or we can adopt a cooperative strategy that values our work, and try not to lose these values when we get kicked upstairs to positions of influence.

How do we cooperate and change the system?
Encourage everyone to engage in software release and collaboration.

Freedom's just another word for nothing left to lose

Release your software: what are the downsides?
Just make a webpage some day when you have writer's block. Even if it's only a guide to installing some package, you may save someone else a week of hassle and make their life better. And how often do we get a chance to make someone else's life better?

Arguments against release: it's not perfect yet, loss of proprietary advantage, maintenance burden, support requests, what if someone misuses the software and writes a wrong paper.
These pseudo-problems mostly turn out to be not worth worrying about.

"No good deed goes unpunished"
No matter how useful you make something, there will always be someone for whom it's not perfect enough. Don't be put off by this.

Remember, for every person who emails you to say either "Thanks" or "It didn't work", there are probably 50 people who used it.

Releasing your software means that YOU get credit for the work and are exposed to the outside world, rather than hoping and wishing for your collaborators, advisors, etc. to properly acknowledge you and credit you outside the group.

What Is To Be Done?

Make it easier to release and publish software:
Reward it with credit, recognition as “doing science,” and fund it.
Change outdated publishing ideas, e.g. the AJ policy discourages software methods papers. We need carrots for software release, not sticks.

Words of advice for young people**

Write and document all code as if you were going to release it.
That means as if somebody else were going to use it. That somebody might be a fellow student, or your future student or collaborator, or you a year from now after you've forgotten how it works.

Work for PIs who can code, or at least make their own plots.
If they don’t, they may not see the amount of work you put in or the sunk costs. If you have to work for these people, make sure to negotiate the division of labor and how you’ll be properly credited beforehand. This forestalls resentment and benefits everyone.

Resist the pundit-technician divide.
Don't aspire to a track that culminates when your job becomes sending emails all day and telling other people what to do. If you get to that position, remember where you came from and make time to get your hands on the bits.

**William S. Burroughs, google it