In a poll on the Astrostatistics Facebook group, 80% of people reported spending more than half (average around 70%) of their research time writing code/software on a computer.
SO WE MUST BE GREAT AT THIS, RIGHT? WELL…

- **log(Faster)**
- **log(Harder)**
- **log(Better)**
- **log(Stronger)**

- Code you might share with a senior colleague
- Code you might share with a junior colleague
- Excel spreadsheet
- Rough code you wrote yesterday
- Code you will use in a published paper
- Good
- Bad
- Ugly

Public code on GitHub

Most people aim to get here

CRAN

But overall it is worth it!

log(Better) + log(Faster) + log(Stronger)

It’s worth pushing past this point!
THE SOLUTION

- We (as a community) bootstrap the “implement” part wherever feasible.

- Even when we re-invent the wheel, knowing what wheels are available makes this:
  - A) worthwhile (need a roller-blade wheel in a world of cartwheels)
  - B) efficient (we can still borrow ideas - make it round!).

- To help our community this means the following:
  - Aggressively public code +
  - Default collaborative efforts =
  - Open-source analysis (statistical methods and software)
BUT REMEMBER!

Feral Code ≠ Open Source!

[www.placeiworkedonce.edu.au]
FTP
• src
  • temp.txt
  • temp2.f
• README
• BUTREADTHISFIRST
• auto.config.old&broken
• LICENSE

www.github.com/ne/code

Code | Commits | Issues
• src
• Man
• README.md
• LICENSE

This really matters!!!

GPL/MIT/BSA/APACHE
WHY WE SHOULD DO OPEN-SOURCE IN ACADEMIA?

- Because it is the new fun thing!
- I keep seeing talks about GitHub, maybe it will help me get a job...?
- My grant funding insists.
- Better than nothing for backing things up.
- Paying for things sucks.
- To make the world a better place...
- These are not the best reasons actually... we can do better by appealing to our more ruthless and selfish instincts!
WHY WE **REALLY** SHOULD DO OPEN-SOURCE IN ACADEMIA?

- Only way to efficiently collaborate on complex projects, and drops the bar to entry to almost nothing. If people cannot see it then they will not ask to help.

- Get a lot more credit and exposure to your work (you will start appearing all over the internet).

- It instills a much higher level quality of work: if you do not feel comfortable sharing it then you should not feel comfortable publishing it! As Russians say: “Trust, but verify”.

- It provides free (!) wide-coverage quality checking you cannot replicate alone.

- It allows you to work seamlessly on projects across multiple devices without penalty (I often show people snippets from GitHub on my phone - sad yes, but handy).

- It naturally enforces modern version control by any practical route (Git / GitHub).

- It helps less senior members of our community. What does this mean...?
A new PhD student in a highly mature field like astronomy (a few thousand years of research and counting) has a lot of catching up to do. They need:

- Solid coding skills (no matter what they do).
- Relatively sophisticated knowledge of data analytics and statistics.
- Broad understanding of the current state of the field, and a deep understanding of their own sub-field.
- Excellent project management.
- Functional people skills (ahem).
A CASE STUDY WITH HYPER-FIT

Astronomy is unusual within physics in how expensive it is to get observations, so we tend to have un-ignorable errors, and available techniques like PCA and SVM rarely support the treatment of errors properly.

Because of some mass-size data, I started wondering about hyperplane fitting with heteroscedastic covariant errors, and published a paper tackling it:

Hyper-Fit: Fitting Linear Models to Multidimensional Data with Multivariate Gaussian Uncertainties

\[
\ln L = -\frac{1}{2} \sum_{i=1}^{N} \left[ \ln \left( \sigma_i^2 + \frac{\mathbf{n}^T \mathbf{C}_i \mathbf{n}}{\mathbf{n}^T \mathbf{n}} \right) + \frac{(\mathbf{n}^T (\mathbf{x}_i - \mathbf{n}))^2}{\sigma_i^2 \mathbf{n}^T \mathbf{n} + \mathbf{n}^T \mathbf{C}_i \mathbf{n}} \right]
\]

This involves N-dimensional projections and careful treatment of the data covariance terms. This is all fiddly stuff that took me a lot of sanity checking to implement robustly (e.g. adapting for geometric corner cases). Now the question is, would you confidently trust a new PhD student to implement the above in code you need to use?
So a paper with an equation is not enough. How about an R CRAN package?

The bar for this is high (much higher than a simple code repo like SourceForge or PyPI)- it requires full code documentation (40 pages!) and a full suite of useable examples.
A CASE STUDY WITH HYPER-FIT

- Is that enough? Well lots of people do not use R, fear it, and will not read a manual (arguably they do not deserve my help!). Behold Shiny Apps:
A NEW PROSPECT

- A more complex recent example of combing modern statistics with traditional astronomy is the new SED tool ProSpect.

- There are far too many moving parts for a modern PhD student to make significant progress in three years.

- Fully opening the source has made it easy for students and junior postdocs to understand how this all fits together, and pick out the bit they really need.
THINGS TO CONSIDER

- Contributing software is a great way to achieve objective translation of what we do. This will start to matter a lot as translation and impact becomes more established in the Australian ERA process.

- Documentation is very useful even for your own purposes. Seems like a waste of time when everything is fresh in your head, but I read my own documentation for years old code all the time (and quicker than digging through source code).

- Examples. Lots of them. You always like seeing examples of things in practice right? So for the same reason you should write lots, and cover trivial -> complex.
THE DOWNSIDES

- I support a lot of packages in the R eco-system - around 12 proper ones that are mostly used by people who are not me. Here is my comment FAQ (or FSS):

  - The danger is you become a support desk!

    - Well yes, if you are writing bad code that is poorly documented - see this as positive motivation. For these stable packages, hassling emails are pretty rare.

  - All this code faffing isn’t “proper research”!

    - Vanishingly rare that anyone below the age of 65 seriously thinks like this now. And history will show that open-source research is the correct idea, and worth the effort.
Final takeaway message: you wouldn’t want to build a computer based on a poem, so don’t try to describe highly complex data analysis and software in a paragraph in a paper. The best solution it to share your working, and open-source your efforts.

In any case, open-source your research and methods for selfish reasons (free QC and more credit), and feel virtuous when it also helps new students join our community and make productive contributions in an efficient manner.

Consider co-producing professional code as a required by-product of any published work (up there with releasing data).

Not all version control systems are equal- basically use Git (and then plenty of decent online interfaces: GitHub, Bitbucket, GitLab).