Notes from Research software management, sharing and sustainability workshop - 16th January 2017

Neil Chue-Hong, Software Sustainability Institute - Is my research right? Surviving in a post-expert world

http://dx.doi.org/10.6084/m9.figshare.4541236

Scientists - 4th most trusted profession

Everyone can make mistakes; don't need to be an 'expert'

Trust in your expertise is based on your reputation, hard when everyone makes mistakes

Mentioning the paper by Reinhart and Rogoff - 'Growth in the time of debt' - results couldn't be re-created (during a student workshop on reproducibility) - they were surprise that data cannot be reproduced, but shared the data and the code and acknowledged the mistake - there was an error in the spreadsheet.

- Important error as this paper was the basis of most of the austerity policies imposed by Western governments

Famous 5 retractions - http://science.sciencemag.org/content/314/5807/1856

- A result of flipping two columns in the data spreadsheet
- Reputation and the quality of other work questioned

Zoology example - https://doi.org/10.1126/science.aad2879 - paper looking at migration out of Africa was undermined by a small bioinformatics error which completely changed the results.

- Making mistakes are not fraud, but how often do they happen?

Reproducibility problems are common across many disciplines, but the problem is not as bad as much of the mainstream press has made out (and it's got better in the last 30 years!).

Good software management and software sharing helps YOU - your research becomes reproducible.

Best practices are often intimidating and impossible to follow. Better suggestion - adapt to good enough practices: https://arxiv.org/abs/1609.00037

Researchers do not get credit for sharing code and data formally at the moment but, publishing code and data increases research impact. And there is evidence for this: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6200247

Get and give credit for software

- Publish software papers and cite software papers
- Use research identifiers ike ORCID
- Be a better reviewer (ask for code raw data and software) and be constructive

Also - it is important that everyone has an ORCID to be properly attributed. Tools integrate with ORCID - e.g. ImpactStory or altmertrics - they also show the non-traditional impact of your story.

You are not the only one out there with a problem so collaborate!

- Everyone, including experts make mistake. The difference - experts know that everyone makes mistakes - Croucher's Law (https://t.co/1NZvyBCCii)

Most research involves a lot of repetitive tasks - automation help reducing the errors and also to find mistakes. It is also easier to explain your pipeline to someone - you can just share your script.

- The end goal is that anyone can take your code, reproduce your results. They can also contribute to your code, and then you can reproduce and re-run their analysis.

For self-taught programmers - keep on getting training. Software development is constantly moving on - new tools are being developed and they are useful.

The purpose of using version control - good if it is for you to prevent you from messing up with your data, better if it allows you to better collaborate with others. Tools: github, gitlub, bitbucket.

Interesting stats: 92% researchers rely on software in their research

The goal of @SSI - getting people together. What are your ideas for improving practice in your community? SSI is willing to help. Also remember about local champions that might be able to help you.

Stephen Eglen, University of Cambridge - Towards standard practice for managing and sharing code

Slides at http://bit.ly/eglen2017-1

Research sharing as an 'inverse problem' - when you've got the paper (PDF) at the end it's very hard to get back to the data, code etc.
The paper is often just 'advertising' of the scholarship which has taken place

Stephen's selfish reasons to share

1. Funding mandates
2. Credit through data papers (and software papers)
3. Leads to further collaborations
4. Fixes data bugs/errors in collaborations - it is embarrassing but it happens to everyone
5. Prevent data loss (when students leave etc) - you should set up a github repo from the start
6. Your future self will benefit from your decision to share and document your code

Original paper by Florian Markowetz about selfish reasons for sharing:
https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0850-7

Q: What's your advice for people who are not established and no one yet follows them on social media? WHat's their benefit to share? How do they get traction?
A: Now is a good time for sharing. You help your future self. Not so many people do this - people will notice you! Stephen submitted a data paper a couple of years ago (couldn't publish that dataset elsewhere). Made all the paper fully reproducible. One reviewer loved it and signed his review openly. Re-analysed the data as well. Improved the figure! SHARING NOW gives the advantage that you will be noticed - it is still a rare enough practice to share.

https://dx.doi.org/10.1101/045104 - Stephen's paper on 'Towards standard practices for sharing computer code and programs in neuroscience'

Recommendations:

1. Include enough code to at least reproduce key figures and results
2. Provide toy examples if the whole analysis is too large/too time consuming to re-run
3. Version control in github
4. Appropriate licence (MIT)
5. Provide data
6. Provide tests (and write tests!) - ensure that things are working and provide people with confidence about your code
7. Use standards

8. Use permanent URLs (Zenodo/figshare) - this is very important. Makes it easier for people to access your code. They can do it via self-service. Stephen admitted that several years ago he still used to say ‘email me’ - not anymore :)

Q: Why do you recommend MIT licence for code? Why not a copy-left licence (e.g. GPLv3)? What is a copy-left licence?
A: Copy-left licence is a “viral” licence - any derivative work needs to be licenced under the same licence. So this impairs mixing outputs under various licences.

New tools:

- Docker makes it much easier to rerun code in papers
-

Docker will not ensure that your software will be available forever:
http://ivory.idyll.org/blog/2017-pof-software-archivability.html

- Forever is a long time, if you make your research reproducible for a couple of years that’s great (or ‘good enough’)

Advice from Stephen (Old Tools):

- Find a code buddy - someone who would be happy to test out the code for you
- Make sure your github repo has several things:
    - Readme file
    - Makefile - learn Make if you don’t know it already!
        - Some researchers from the audience recommend not to use Make - there are better tools.

Practical tips

- Lobby journals and funders about code-sharing
- When reviewing articles, ask for code to be made available
- Write your code as though it will be made public

Q: Is there any standard infrastructure to be used to reproduce figures in the paper?
A: R. And all the data and code sitting in Github. Or python. Or Matlab. They can all generate figures, and you can have a script to make figures.

Focus groups

All notes available:

- http://bit.ly/16JanGroup1
- http://bit.ly/16JanGroup2
- http://bit.ly/16JanGroup3
- http://bit.ly/16JanGroup4
- http://bit.ly/16JanGroup5
- http://bit.ly/16JanGroup6

Closing remarks

By Neil Chue Hong - small changes make a huge difference. Everyone can make a positive contribution to software management practice change at workplace. Start today.