Focus group 3 - Software workshop 16 Jan 2017

Problem(s) that the group wants to solve:
Legacy code and dealing with insufficient documentation

Facilitator: Kirstie Whitaker

Participants:
- Kirstie - researcher from the Dept. of Psychiatry, brain imaging data, network analysis of the brain; fellow of Mozilla Science lab
  - Re-uses code from other researchers
  - Figures in papers shouldn’t be just made by screenshoting or from Powerpoint - they should be reproducible
  - So many mistakes in code - debunking rubbish is sometimes so difficult that creating things from scratch...
- Marta - Research Data Services at Cambridge:
  - More and more questions about software
- Inge - MRC Epidemiology, data manager there
  - Helps researchers manage and share their data and software, run databases, document them
- Martin - Head of Data Management at Clinical Trials unit @UEA
  - Databases for medical researchers
- Neil - Software Sustainability Institute
  - Worked with a fairly well-known computational chemistry legacy code. Wanted to do sth with it. Researchers are still there, use version control, but still the same problems - because the code was inherited from someone else; the documentation wasn’t sufficient. The code was composed from several linked parts, very difficult to understand. So one can still have the developers available, but not understandable
- Ruoyun - PhD in Genetics; analysing genomic data.
  - How to find good statistical algorithms? Sometimes packages developed decades ago, from a different field, sometimes difficult to add new functionalities
  - Sometimes easier to r-write the code than to re-use it
- Poonam - Computational Scientists, researcher in Experimental Economics; maintaining online platform for running the experiments (code + data available to repeat); codebase difficult to maintain a lot of dynamic elements which change from experiment to experiment.
  - Before Poonam arrived, no version control available; components not labelled (annotated); would like to move to GitHub to support version control
- Paul - University Information Services, ex Stem Cell Institute COmputer Officer
Often gets requests for help for software installation for end users, but packages not explained well how to do it

Quick wins:

- When writing a new piece of code, write a user manual to go with that. And it needs to clarify it well. Sometimes comments don’t go into code. But these recommendations go around the team and is not always adapted well by researchers.
  - This should also apply to little scripts for example for HPC processing and for a short bioinformatics process
- Computational chemistry projects - talk with people who would want to use it. Test it out and also create several test examples - you can always try to re-run them to know when exactly you have broken the code. And ask researchers if this looks OK.
  - Perhaps testing and documentation should be the key requirements for anyone who write code?
- Have multiple people with different skillsets and experience to look at the code. Even people who don’t know programming - is the end result sensible? Are the values within range?
- Perhaps asking a colleague to review your code (similarly as it is done with grant application or writing paper)
  - In Neil’s team there is always a technical reviewer assigned for a project
    - Difficult to achieve with projects where there is only 1 developer, So perhaps a code club? Who could help each other with discussion about the code they are writing and sharing advice.
      - Though it is difficult to get used to the idea (trust people to share the code) and also it is time consuming
      - So needs to be carefully set up - for example, specify the programming language and the time to be spent on review - e.g. max. 1h - find what you can in 1h
  - Poonam: it is useful to have two developers working on the same code - this improves the code, as well as knowledge transfer
- Promote the importance of comments and documentation. If code is shared, and even if someone doesn’t feel comfortable changing someone else's code, perhaps the person who borrows the code could return their extra annotation/comments. So if I re-use and annotate will re-using, I will share my notes with you. Sometimes it is simply difficult to write documentation for own code - might seem too obvious.
  - This also contributes to long-term goals - to implement this is quick to achieve; can be started today, but has long-term positive effect on software annotation
    - But perhaps some users will find uncomfortable suggesting changes
- If you are trying out a piece of code, write a blog post about your experience - a very short. If it didn’t work, would you do? Did it work? Did it not? Make it publicly searchable!
Long-term solution(s):

- Ensure that senior managers understand the importance of testing and annotation of code. Until managers understand how important it is to have proper documentation and testing set up, this is going to be challenging.
- Perhaps journals (and funders) should require code sharing? Should it be mandated? (certain level of testing and comments within the code?) But sometimes mandates are ineffective.
  - Sometimes senior managers allow the time to test the actual software, but don’t allow the time to create tests for end users. Perhaps mandates could help here

Summary: