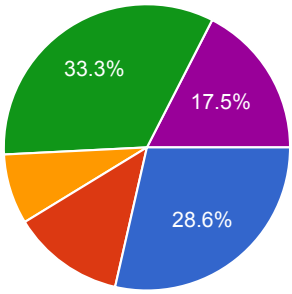
# 65 responses

View all responses

## Summary

### What is your job title?




| | | |
|---|---|---|
| PhD student | **18** | 28.6% |
| Research Associate | **8** | 12.7% |
| Software engineer | **5** | 7.9% |
| Academic | **21** | 33.3% |
| Other | **11** | 17.5% |

### What is your research area?

Computer Science

AI

Computer Vision

Genomics

Biology

Knowledge Representation

computer graphics

Text mining

Computational biology

Feature selection

End-User Service Mashup

machine learning application in education games

Ontologies

information management

Data Management

Computer Systems Research for many-cores

Web Engineering

natural language processing

Computational physics/maths

Ontological Engineering

Computer Networks

Formal Methods

Computer science

Machine Learning

Machine Learning/Biology

eScience

scientific open data

machine learning, data mining, bioinformatics

Integration of spintronic devices to CMOS

Computer vision

Ontology Reasoning

Computer Arithmetic, Neuromorphics, Formal Methods

hardware/architecture/SoC

Computer science

Web, HCI, Accessibility

nanoscale devices and materials

health informatics

medical statistics

Computational Neuroscience

Bioinformatics

Research software

Infectious diseases

biochemical simulators

Theoretical Physics

operations research

Sciences

mathematics

Oceanography / renewable energy

Research Software and Scientific Software

Materials modelling

online collaboration

Cognitive neuroscience

particle & nuclear physics

Climate Change

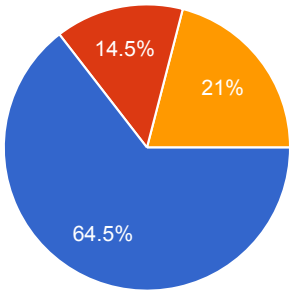
Geophysics

Professional software development

genomics

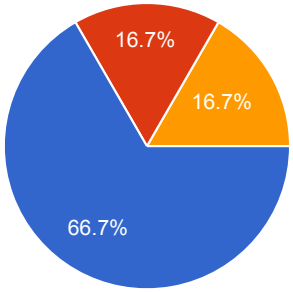
## Storing code in online repositories

### If repositories had a simple, easy to use GUI, alongside the current command line/GUI interfaces, would this encourage scientists to use them?



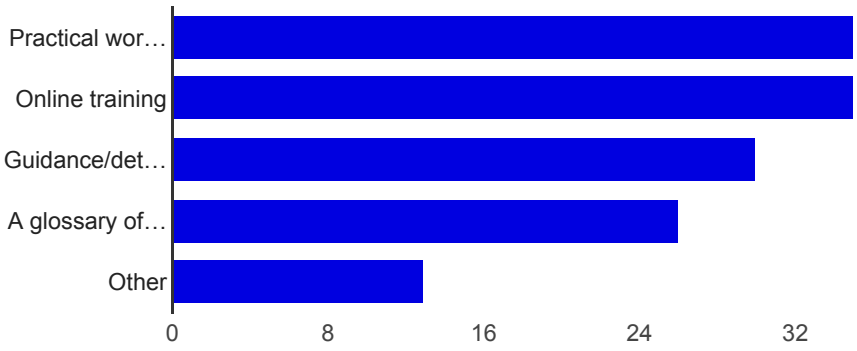

| | | |
|---|---|---|
| Yes | **40** | 64.5% |
| No | **9** | 14.5% |
| Other | **13** | 21% |

### Would it be helpful for the repository to 'track changes' to a script, so they don't have to be documented manually?




| | | |
|---|---|---|
| Yes | **40** | 66.7% |
| No | **10** | 16.7% |
| Other | **10** | 16.7% |

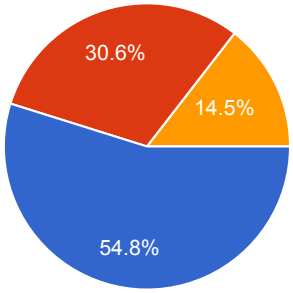
### Which of the following types of training or support would you find useful for learning to use repositories?




| | | |
|---|---|---|
| Practical workshops. | **35** | 60.3% |
| Online training | **35** | 60.3% |
| Guidance/details about the various types of repositories and their features | **30** | 51.7% |

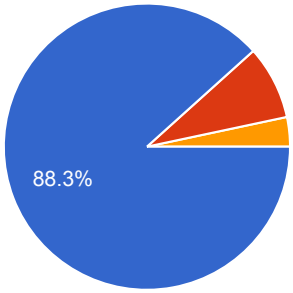
| | | |
|---|---|---|
| A glossary of the terminologies relating to repositories | **26** | 44.8% |
| Other | **13** | 22.4% |

**Would you like your institution to have a central repository, which could be used to store all data and code internally?**




| | | |
|---|---|---|
| Yes | **34** | 54.8% |
| No | **19** | 30.6% |
| Other | **9** | 14.5% |

**Would you like a central internal repository to be compatible with an external one, such as Github, so any code and data can be published externally when required, and anything updated in an external Github repository would automatically appear in the internal one?**




| | | |
|---|---|---|
| Yes | **53** | 88.3% |
| No | **5** | 8.3% |
| Other | **2** | 3.3% |

**Comments:**

Your last question is incredibly loaded... The issue of internal repository is rather complex. In principle we only need such a thing in order to have control over our own data (a good thing), but I predict that University Administration will soon want to get rid of the cost of maintaining it and will eventually outsource the whole thing (eg github, but could be a worse one...). Then the only issue is being able to have private repositories which the University would have to pay for. In the long run it may be better to propose using GitHub right away as long as the University pays for allowing us to have private repositories.

I released most of my PhD code via mloss.org and github. The main burden was in making the code cross platform and dealing with questions from users.

This all feels on the wrong track. The major issue is not wanting to deal with the pain of making the code sharable. Some people are embarrassed. Some people just don't want to take the time. It's not hard to publish code online. I don't think that's a barrier.

Any internal repository should absolutely definitely be implemented using an existing, widely used version control system. Then you get "track changes", massive amounts of documentation, and integration with external repositories "for free". Even better: researchers who currently use version control systems will barely need to change their

workflows.

The cost of space and the time required for code storage has always been a big block. In experimental environments code changes rapidly and any code storage facility should ideally be transparent to code developers.

Existing repositories have pretty good UI - both graphical and text-based. I don't think that using command line is much of a challenge for people who do programming already. Automated tracking of changes sounds pretty bad (at least as far as I can tell how would that work). I don't want to archive all the changes. I want to have control of branching, etc. Also, manual grouping changes to code in commits (annotated with messages) makes it easier to understand what was done when and why. Building (from scratch) and maintaining institutional repositories would require a lot of resources and I doubt it would be any better than existing external ones. Maybe using and contributing to some open source repo software would be viable.

you'll need settings to include an option to be asked whether you'd like to sync the two repositories automatically or manually.

It would be great to have a university-supported procedure to obtain infinite amounts of private repositories on GitHub.

Option does no harm.

Privacy during development is critical, which rules out github The code often doesn't run very well outside of our environment

The barriers are not technical, but social.

There already are several different GUI interfaces for git, but they only reduce the learning barrier.

The difficulty of using online repositories sometimes deters scientists from using them as places to deposit code." "If repositories had a simple, easy to use GUI, alongside the current command line/GUI interfaces, would this encourage scientists to use them?" Those questions make presumptions about the current state of affairs and the reasons for that state. E.g. how am I supposed to answer the first question if I think that repositories already have a simple (or simple enough) GUI? "Would it be helpful for the repository to 'track changes' to a script, so they don't have to be documented manually? " Is there a repository format that doesn't do that? "Would you like your institution to have a central repository, which could be used to store all data and code internally?" The main reason to have an internal repository is to have non-public projects. Without specifying some details what "internal" means, it's really hard to answer this question in any meaningful way. "Would you like a central internal repository to be compatible with an external one..." Again, what repository is not compatible with github? Everything that uses git is compatible. If you are talking about automatic syncing, then OK, but then the first part of the question makes no sense.

It would be helpful to define what you mean by "online reposistories" before these questions. I've answered assuming you mean github or similar.

We honestly do not have the capacity to support an institutional repository that would handle code, data, and publications. I am instead pushing the use of public internet

repositories like Figshare or Github.

In my mind, the main barrier to use of repositories for code is simply making sure that they're used from day one - I've found that the technical barriers for something like GitHub are now small enough that most researchers can set a repository up on their own (using the online guidance), however if they don't do it to start there's never enough time to do it later on. Time is the biggest barrier, along with perceived difficulty, and fear that their code is not good enough for sharing - even with colleagues.

I have been pushing scientists to publish their code for years; the barriers are not technical - they are emotional. Scientists are almost universally scared and embarrassed to share their code, for fear of 'looking dumb' or unprofessional.

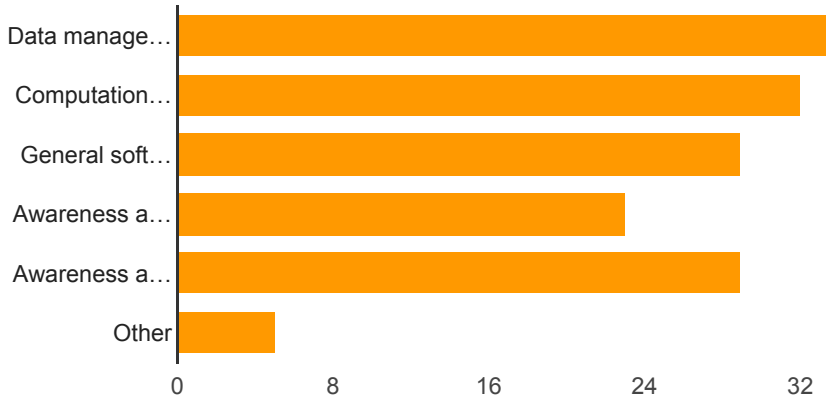
My experience working with scientists within our organization showed that people needed both sufficient motivation to adopt version control and the training from a peer. I did training early on in my tenure, but people weren't quite ready for it. Then I harped on them for literally years thereafter and slowly convinced people that it was something that was useful to them. But it wasn't until the second training (again, years later) that I saw a significant uptick in the number of people using version control.

It is worth noting that I already make my code available online as it is developed via publicly accessible version control systems.

Being a software engineer (my title says "research programmer" though), I may be a wrong person to answer these questions, but I do work with faculty and their PhD students on a daily basis, and it is very surprising that they can't seem to use version control system properly. I was wondering how to spread this practice, to no avail. What I noticed though: - PhD students do use repository if it is configured by someone else - Command line tools are horrible for non-CS majors, but I worked with a lady who finished her degree in Epidemiology and she use SourceTree to commit/push her stuff to a git repository without major problems.

## Training in computational research

### In which of the following areas would you find training/support helpful?



| | | |
|---|---|---|
| Data management planning | **35** | 67.3% |
| Computational research skills | **32** | 61.5% |
| General software engineering skills | **29** | 55.8% |
| Awareness about intellectual properties issues | **23** | 44.2% |
| Awareness about open source licensing systems | **29** | 55.8% |
| Other | **5** | 9.6% |

**Comments:**

I don't need training in the above as I have had already, but others may find any of the above useful

In general using academic code in industry is hard as it frequently lacks a license for either the code and/or the data.

I'm already adequately up to speed with all these.

My students/RAs would benefit from the above.

I've read a lot about the other topics already.

Software Carpentry provides great training on Git.

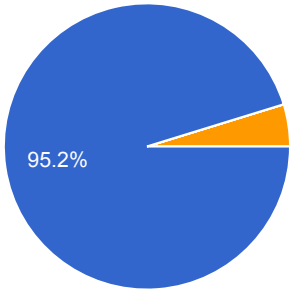
For my faculty, not for myself.

I'm speaking not on behalf of myself, but with respect to what I think would be useful for people in my organization without a computation background.

It is always scary to open code written by a scientist, especially when they say - just take and use it. Two or three weeks of refactoring guaranteed.

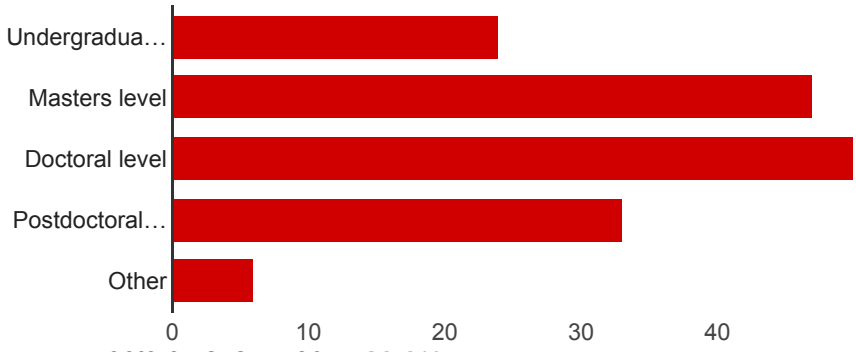
# Scientific Reproducibility

**Do you think it would beneficial to offer this type of experience at other universities?**



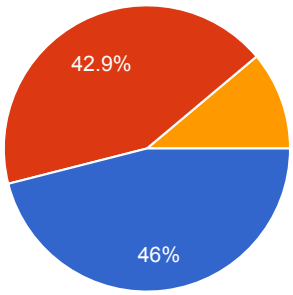

| | | |
|---|---|---|
| Yes | **60** | 95.2% |
| No | **0** | 0% |
| Other | **3** | 4.8% |

**What level should this kind of course should be provided at?**



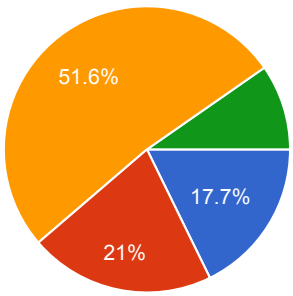

| | | |
|---|---|---|
| Postdoctoral level | **33** | 53.2% |
| Other | **6** | 9.7% |

**Should this kind of course be optional or compulsory?**




| | | |
|---|---|---|
| Compulsory | **29** | 46% |
| Optional | **27** | 42.9% |
| Other | **7** | 11.1% |

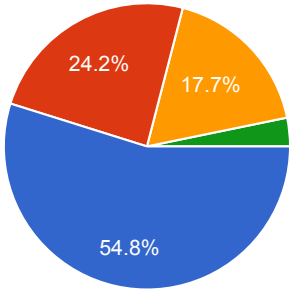
**Should the course to be provided internally, or by an expert body, such as Software Carpentry?**




| | | |
|---|---|---|
| Internally | **11** | 17.7% |
| By an expert body | **13** | 21% |
| Either | **32** | 51.6% |
| Other | **6** | 9.7% |

**Do you think the course should be short and intensive, or spread out over a**

**longer period?**




| | | |
|---|---|---|
| Short and intensive (over a few days) | **34** | 54.8% |
| Spread out over a few weeks | **15** | 24.2% |
| Completed over a time period suiting the student | **11** | 17.7% |
| Other | **2** | 3.2% |

**Comments:**

There needs to be ongoing support.

Spread out is better as more could be covered, but is probably less achievable due to tight availability of resources.

the analyses to be replicated should be pretty simple to not waste people's time.

Working towards Reproducibility should be a skill in which all PhD candidates should be well versed before starting their PhD. So getting those skills before hand would be the best in my opinion. However, to make the argument to include this type of courses into a undergrad or Master programme (other than e.g. Software Engineering) is, I believe, complicated.

i don't think there is enough material for a whole course. the idea of reproducibility is simple and there are just a few steps one needs to take to ensure once research is reproducible (as far as code is concerned). dedicating a whole course to it would be forced and most likely 'teaching' pretty obvious things.

Social system is set at professor level. Pointless to train lower.

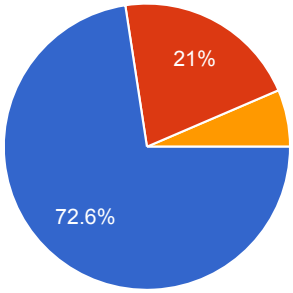
I am not familiar with Software Carpentry so I was not sure how to answer the 4th question. I'd like to point out that scientific reproducibillity is about much more than software.

This is a hard, fast moving problem. Collaboration between SWCarpentry style and institutions seems vital. Making it compulsory is always a way to kill enthusiasm.

It would be great if it could be spread out, but ultimately people don't have the time, and the subject is well formed enough that it could be done over one or two sessions.

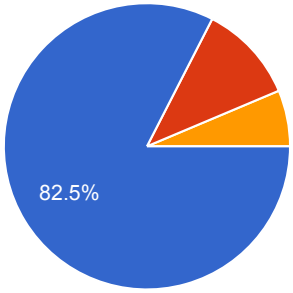
## Computing environments supporting reproducibility

**Would you be interested in using this type of technology for your own research?**



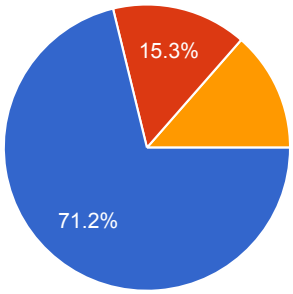

| | | |
|---|---|---|
| Yes | **45** | 72.6% |
| No | **13** | 21% |
| Other | **4** | 6.5% |

**Would you be interested in using this type of technology to rerun other people's research, or use their analysis code?**




| | | |
|---|---|---|
| Yes | **52** | 82.5% |
| No | **7** | 11.1% |
| Other | **4** | 6.3% |

**Do you think this will have any effect on scientific reproducibility?**




| | | |
|---|---|---|
| Yes | **42** | 71.2% |
| No | **9** | 15.3% |
| Other | **8** | 13.6% |

**Comments:**

You must be referring to the Docker approach, though other VM technology could also be useful. The only (big) problem with this approach is that the way in which it is implemented has an enormous security problem. Who is patching their Docker images when there is a new important security patch? Researchers in IT security have been talking about this, but the hype is blinding people...

I suspect it will result in many more bugs being found in programs that have learned their data and the results their users wish to generate. In the long run this may, if time pressures allow, lead to higher code quality but I doubt it.

Previously I have provided scripts to download, setup and run an experiment for reproducibility. One reviewer said that they would not trust a script that did this and wanted a step-by-step guide to allow them to do so themselves... it's impossible to please

everybody.

Existing repositories are good enough for that.

of course, if you can literally run other people's experiments. it means however that you just have to blindly trust their code. how useful this might be depends on how much you can look under the hood.

Most of my immediate 'research' work is development, not result taking; thus this is not applicable.

VM style snapshots can help with reproducibility, but may undermine modification and extension.

I think that everything (i.e. software, data, and text) should be version controlled, but I don't think that using a "specialist platform" is the right way to do it. I prefer normal popular tools like git, public hosting, etc.

I've tried a few - binder, docker, etc. For small projects they work well - I'm particularly impressed with binder's UI - but it's scaling it up to a "real" problem run on HPC resources that's the concern.

Depends on the accessibility of the software in question and if the "snapshot" it saves is in a long-term accessible format. For example, some backup software saves the backup as a proprietary format. If you lose your license to that software or it goes out of business you're SOL.

Need to be careful, though - loading up a container of all somebody else's code will reproduce bugs as well as analysis. Need to avoid confusing this with reproducing from the level of the algorithms.

Being able to run an identical copy of data and analysis software is not enough for scientific reproducibility on its own, though it does help because you have access to the tools. It is the lab bench equivalent of being given access to someone's laboratory setup - without instructions, sensible materials (input data), understanding of the setup (parameters) and trust in the experimental method, you don't get reproducibility.

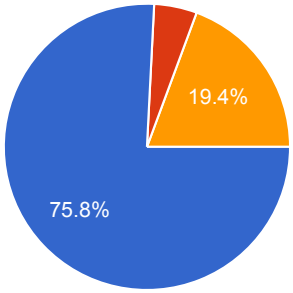
Not totally sure what you're referring to here, but I'm very wary of large, complicated frameworks; they are unmaintainable, inflexible and never universally adopted. I encourage people to use collections of smaller, simpler tools to achieve automated, reproducible results.

I do not believe it is possible to capture all dependencies. What if the system depends on access to private database running on campus only. There are some security concerns too (credentials/access keys in a snapshot)

## Other suggestions

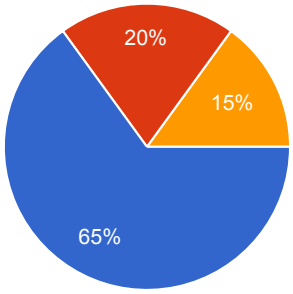
### Would having access to research software engineers (developers who specialise in scientific software) on campus be something that could help scientists to produce code that they would be happy to publish?

Yes **47** 75.8%



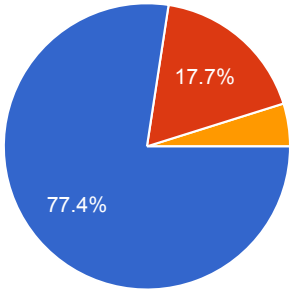

| | | |
|---|---|---|
| No | **3** | 4.8% |
| Other | **12** | 19.4% |

**Would you want to become part of a research software engineering community, where people could discuss code publishing and open science?**




| | | |
|---|---|---|
| Yes | **39** | 65% |
| No | **12** | 20% |
| Other | **9** | 15% |

**Would you be more inclined to focus on making analysis code and data available if more publishers required it?**




| | | |
|---|---|---|
| Yes | **48** | 77.4% |
| No | **11** | 17.7% |
| Other | **3** | 4.8% |

**Comments:**

This will only happen when publisher demand and enforce this requirement

In addition, publishing code for hardware (e.g. verilog) will prevent commercial exploitation

I publish some of my code, but not all. Main reasons for not publishing: a) Some elements licensed commercially with exclusivity clauses in a field, making it hard for us to publish that, as we can't police how it gets used. We are trying to avoid such clauses in future. b) Time to tidy up code and document it sufficiently for a general audience c) Code may be evolving quickly and never quite gets to a "finished" state d) If we publish core components, it makes it harder for us to make significant changes. Occasionally we have a "big bang" where we make significant revisions to our infrastructure - if groups elsewhere are using it such changes could impact on them in unexpected ways, so we are less likely to do that.

Only if the top conferences I'm targetting demand it. Otherwise, I will keep my code secret, but my theory and method public and published.

It would be important for the incentives to be there. Currently there are few incentives for

such things - it is rare for such analysis to be considered a 'contribution'. It is not only the publishers that need to change but the communities surrounding them and the measures by which research is evaluated.

There are many factors that influence code publishing. Consultations with software engineers may help with software quality to some extend, but require time investment, which is often the limiting factor. I think that people would publish their code more if they didn't have to rush / cut corners because of approaching conference deadlines. But switching from the conference-based to journal-based publishing system is not a likely change.

My experience is that publishers are more interested in the eight page limit, and not the reproducibility.

This is boring work, and not what I signed up to do

If publishing high quality papers required code to be made available, then I expect that this level of coercion would have an effect. That is not the same as saying that it is a good idea.

If publishers required it ... that's a no-brainer. Problem is that currently publishers hardly know what they (should) want/need.

Plenty of researchers already discuss code publishing and open science online and in person (some conferences are aligned with this)

Again, you presume too much about the current state. How do you know that your responders are not already publishing everything, and that they are not members of some communities, etc?

RSEs are great, and funders should make it explicit that money be allocated in grants to pay for them.

Having a dedicated scientific computing person on campus would be like a dream come true. Right now it's just me, and only one small part of my overall job description.

In the UK, there should be stronger encouragement to interpret the RCUK guidelines on research data to include software as an integral part of the research validation process - see for example: http://www.software.ac.uk/resources/guides/epsrc-research-data-policy-and-software

access to RSEs will help only if those RSEs take on responsibilities like training and making code legible and accessible - again, the real challenge is fear, and a programmer who just makes fun of everyone's code will make things worse, not better.

As the most senior/experienced computational person in our organization, I'm often saddled with the responsibility of evaluating, fixing, and/or improving software and code from other neighboring research groups. Most research code falls well shy of professional standards. Actually *having* professional standards for research code would enable much, much more collaboration (as it has in the Open Source community), and would undoubtedly help our fields and research to be pushed farther.

### Finally, do you have any other recommendations that may help scientists to publish code and data?

Do it. It's your duty.

If you decided to publish/release your code/data, try to make them stay available as long as possible. There are lots of cases that the early published source code/runnable application/data got cut off as soon as the related research is finished.

No.

Publishing code should be the decision of the research scientist. There are clear benefits for publishing the software code, but it should not be compulsory as there are clear cases (commercialisation and interaction with industry) where good science is done but it is not advisable and cannot be accepted to release the software.

Make it as straight forward and pain free as possible and provide appropriate training. Do this and people are more likely to use repositories/share their data.

To help younger researchers (such as myself) quickly come up to speed with the current state-of-the-art, a chance to read and run code would be fantastic.

Learn the tools BEFORE you need them! Usually when the time comes for publishing your code, you are on a tight schedule, and too stressed to learn the required skills!

Keep it short. I've managed to publish code in the past: http://dl.acm.org/citation.cfm?id=103741 http://dl.acm.org/citation.cfm?id=2228932 but no one's really interested. The biggest waste was 500,000 lines of maths and proof I contributed to the NASA/PVS library. It's unpublishable, even though it's the mechanized proof of the correctness of fourrier transforms.

You need to consider very carefully what you mean by data and code. Typically in applied physics experiments small sets of data are combine from a number of different stand alone instruments. This is very difficult to integrate into a "nice" package in a way that is suitable for workflow models not least because often different experiments will be chosen to investigate the problem at hand.

Help ensure institute IP policies do not undermine the efforts of scientists to share their software and data openly.

Go take a Software Carpentry course.

Don't be afraid of publishing your code and data - it's good enough. And the flip-side: don't criticise other scientists if the code they release isn't perfect for what you need it to do. Be grateful for them releasing the code, and provide constructive criticism or even contributions to help make it better!

Both code and data are valuable, so if we want scientists to publish them they need to be properly recognised as research outputs (including by the REF) and there needs to be a mechanism for commercialisation. Until then, the danger is that much of the code that would be published is of low quality.

need to develop a code and data citation framework so that publishing highly cited code and data is as valuable to career prospects as highly cited papers.

I don't think the difficulty is the tools or knowledge, it's the promotion and hiring culture inside academic departments and universities. If software was properly credited (e.g. via a REF return and the resulting HEFCE QR income) all code would be available quickly.

Stop funding project that do not publish the code and data.

## Number of daily responses