-Better software for Better research- Introduction to the FAIR^2 for Research Software training Programme

Romain Thomas
Head of Research Software Engineering,
The University of Sheffield
rse.shef.ac.uk

FAIR24RS training programme Kick-off Seminar, October 2025

Access to the slides: here - **Source** Github repository

Useful Information:

- This talk is recorded.
- You can follow the slides ➡️➡️
- Slides are available freely on github.
- You can interupt me at any moment if you have a question.
- Every blue text is a hyperlink.

Access to the slides: here - **Source** Github repository

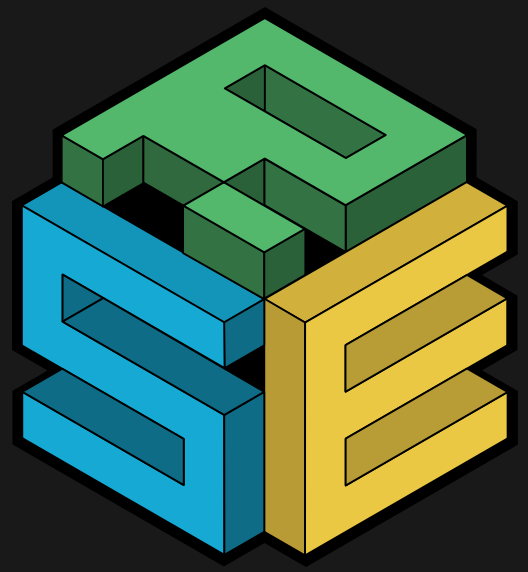
Who am I?

- Name : Romain Thomas
- Role : Head of Research Software Engineering
- Previously : Staff Astronomer and Software Project manager at the Very Large Telescope (Chile)
- Released/Published a few modules/software:
  - STON (submitted, JOSS)
  - dfitspy
  - SCUBA
  - SPARTAN

Access to the slides: here - **Source** Github repository

Who are we? The teams behind the programme

Research Software Engineering



The Research Software Engineering team is composed of 13 members and **collaborates with researchers across the University in building research software**. Areas of expertise within the group include: general software development, code optimisation and performance, reproducibility, GPU computing and Deep Learning, High Performance Computing, training, etc…

Who are we? The teams behind the programme

Data Analytics Service

The Data Analytics Service (IT Services) supports research excellence at the University of Sheffield by bridging technical and analytical gaps through consultation, delivering training, and long-term collaboration with research teams. DAS supports researchers with reproducible data analysis, data visualisation, data engineering, machine learning, statistics, big data, research software, web design, and much more.

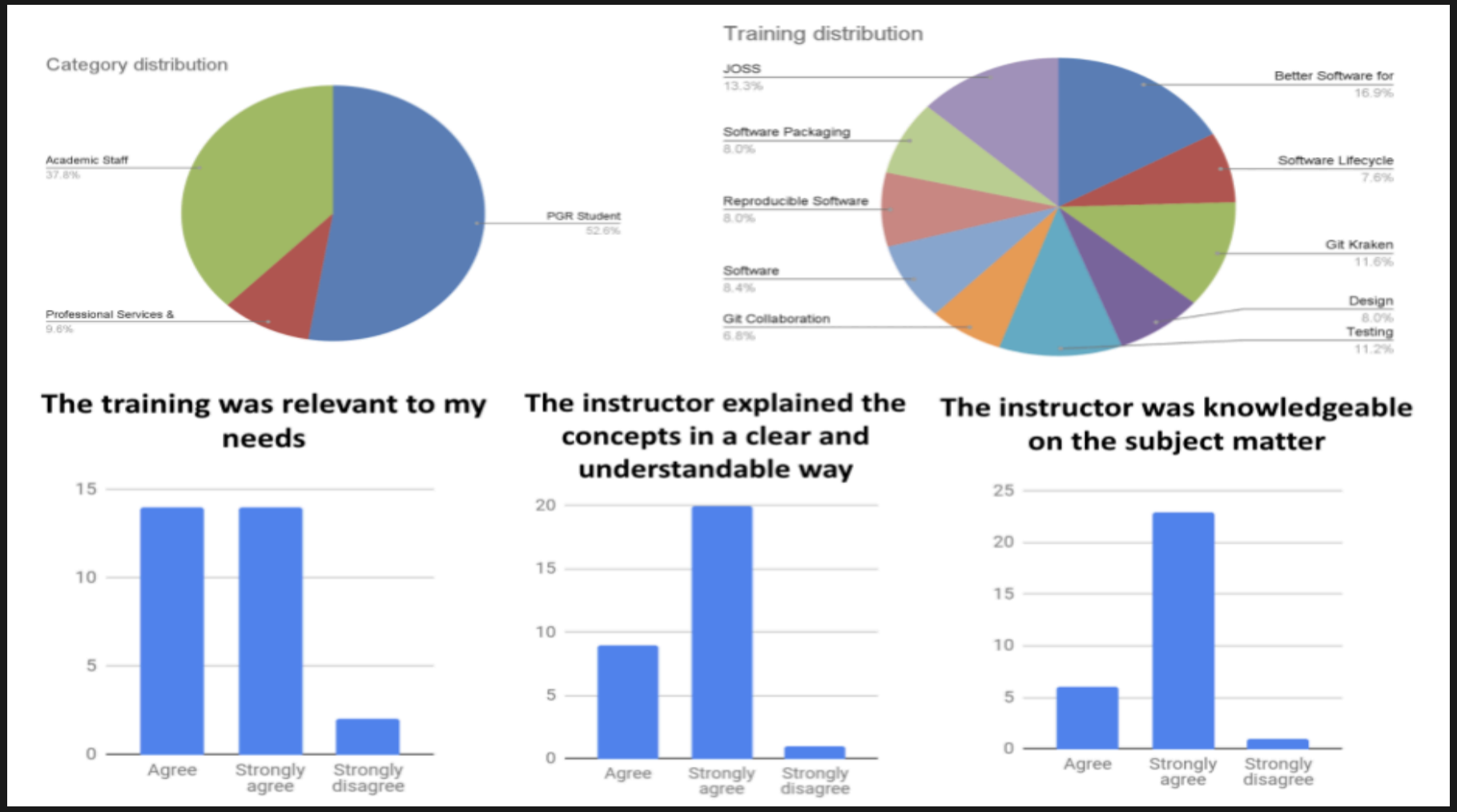
Who are we? The teams behind the programme

Library's Scholarly Communications

The Library's Scholarly Communications Team provides specialist services to support researchers at the University of Sheffield. They offer guidance on making your research outputs open access, and give support on good practice in research data management, copyright and licensing as well as open research more broadly.

This is our second year!

During our first delivery we have seen ~250 registrations to our training programme from all Faculties at the university.



Access to the slides: here - **Source** Github repository

Programme very well received !!!

- *“Very clear explanations, which was great because my depth of knowledge of software development is varied. It’s helpful to have the foundations explained clearly.”*
- *“I really liked having presentations from multiple different people from across the university within one session.”*
- *“I appreciated that time was set aside to put the learning into practice”*
- *“It reinforced many of the software development rules and ideas that will help with future development”*

Access to the slides: here - **Source** Github repository

We are starting teaching it now!

- This second year also sees us **teaching** a (shortened) version of our programme to the made4manufacturing CDT.
- Part of the Data Science and RSE module.

Why FAIR?

Why ~~FAIR~~?

Access to the slides: here - **Source** Github repository

Why OPEN?

Access to the slides: here - **Source** Github repository

Research is a continuous process

"The succession of researchers is comparable to a single person who learns indefinitely.

Pascal, Pensee, French Mathematician, Physicist, inventor, philosopher and theologian [1623-1662]

- That's very old….
- But still very valid…
- And becomes much more difficult with the complexity of modern research

Access to the slides: here - **Source** Github repository

Research creates knowledge…that is passed down

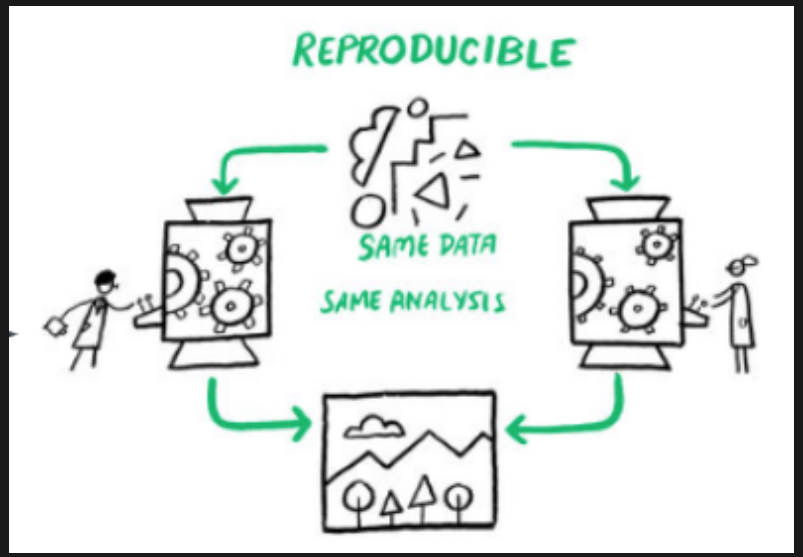
"Knowledge is humankind's most precious treasure. Everything that we accomplished has been done due to the capacity to create a transmissible heritage, which spares each new generation the task of starting from scratch." *B. Sirbey, le grand homme qui apprend.*

> If we are doing the research we are doing today, it is thanks to the work of previous generations that created the knowledge that we are using now.

Access to the slides: here - **Source** Github repository

And that can be trusted…

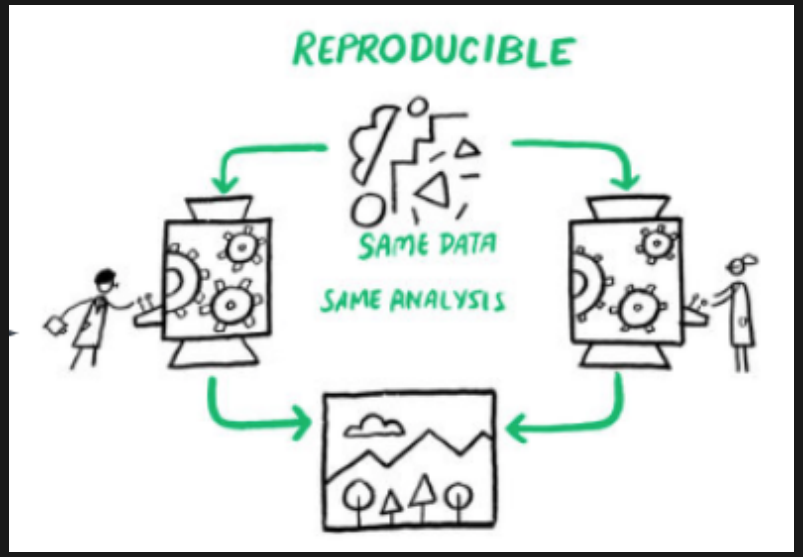
- Research relies on the ability to trust what has been done before.
- This means that a result has been tested, verified and could be reproduced ➡️➡️
- Tools and methods used for a particular result are known and shared…



Access to the slides: here - **Source** Github repository

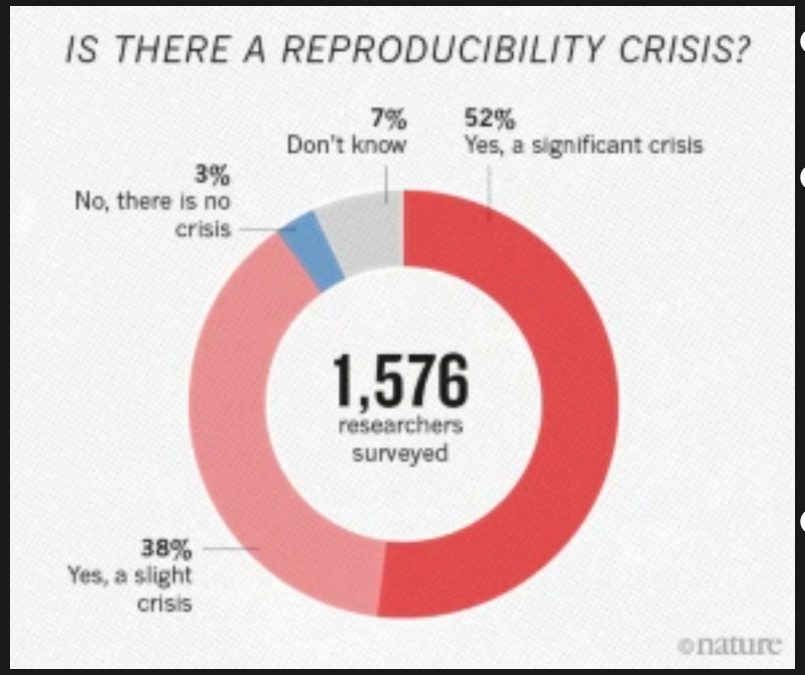
What if a generation of researchers stop doing this?

- Tools and methods used for a particular results are NOT known and shared…
- This means that a result can NOT be tested and verified and can NOT be reproduced.
- ➡️ It is harder to trust research



Access to the slides: here - **Source** Github repository

Are we far from reaching this situation?



- 9
- M
  t
  s
- A

er

reproduce their own experiments.

Source: Baker M., Nature, 2016

Access to the slides: here - **Source** Github repository

So how do we get better?

Let’s improve!

Continuum of practices in (data) science

A single folder for all projects where everyone can write
No backup strategy
No README files
No comments on code
No version control
No separation between raw/derivative data
No meaningful filenames
No history log
No way to replicate past analysis
No way for external person to know what’s going on

(Open) data archived with DOI
Data versioning with git-annex/DVC
Registered protocols for data collection
GIT version control + branches & tags
(Open) code on GitHub + archived w DOI
Unit tests and continuous integration
Singularity/Docker container
Automation with Make/Snakemake
Consistent folder structure across projects
Extensive documentation
Collaboration tools (slack, trello, etc)

Source: www.aalto.fi

Access to the slides: here - **Source** Github repository

And why not start with your software?

Access to the slides: here - **Source** Github repository

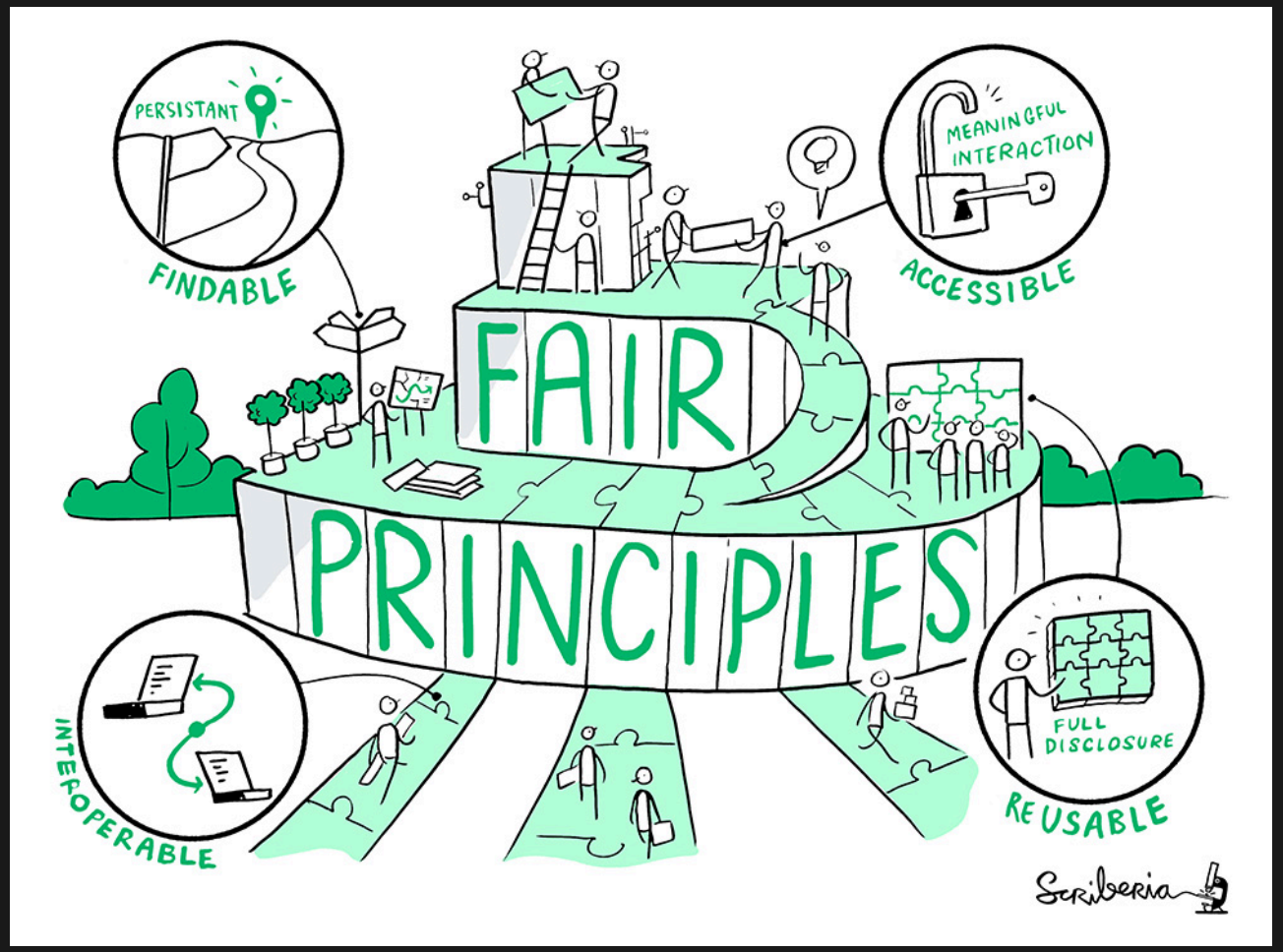
Let’s start by a definition: What is a software?

“Source code files, algorithms, scripts, computational workflows and executables that were created during the research process or for a research purpose.”

Barker et al. Scientific Data 9:622 (2022) “Introducing the FAIR Principles for research software”

What is FAIR?

The FAIR principles



A guideline for those wishing to enhance the reusability of their data holdings

–Wilkinson et al. (2016)–

Access to the slides: here - **Source** Github repository

The FAIR principles

“Many of the FAIR Guiding Principles can be directly applied to research software by treating software and data as similar digital research objects. However, specific characteristics of software — such as its executability, composite nature, and continuous evolution and versioning — make it necessary to revise and extend the principles.”

Chue Hong, Neil P. et al, FAIR Principles for Research Software (FAIR4RS Principles)

The FAIR principles: what do they say?

- Findable: Software, and its associated metadata, is easy for both humans and machines to find
- Accessible: Software, and its metadata, is retrievable via standardised protocols

Barker *et al. Scientific Data* 9:622 (2022) “Introducing the FAIR Principles for research software” DOI: 10.1038/s41597-022-01710-x

The FAIR principles: what do they say?

- Interoperable: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.
- Reusable: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software)

Barker *et al. Scientific Data* 9:622 (2022) “Introducing the FAIR Principles for research software” DOI: 10.1038/s41597-022-01710-x

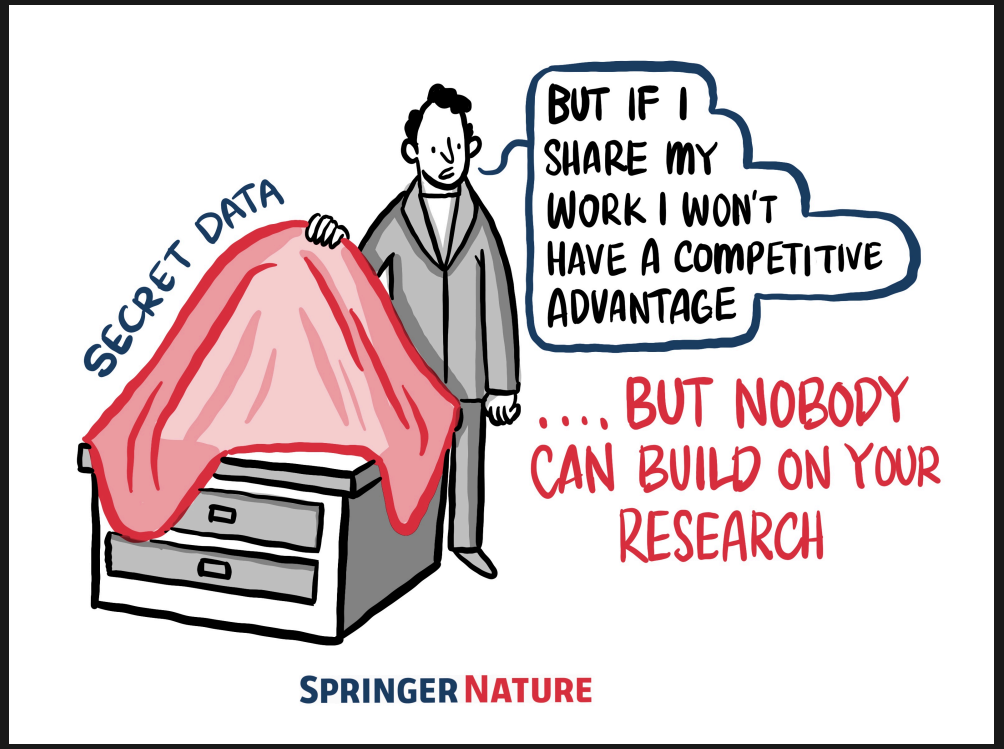
University’s position about FAIR

“We aspire to open research culture that values a diverse range of contributions and adheres to the FAIR principles to enable the results of our research to be of maximum benefit to society (findable, accessible, interoperable and reusable), whilst also respecting circumstances that limit data sharing (for example, due to issues of privacy, non-consent, contractual agreements, legislation or practicality).”
University of Sheffield, Statement on Open Research

“All researchers, including postgraduate research students, have a personal responsibility to manage effectively the data they create….. All researchers are expected to document research data and software in line with the FAIR principles…..”
University of Sheffield, Policy on good research and innovation practices

Barriers to FAIR^{2}4RS

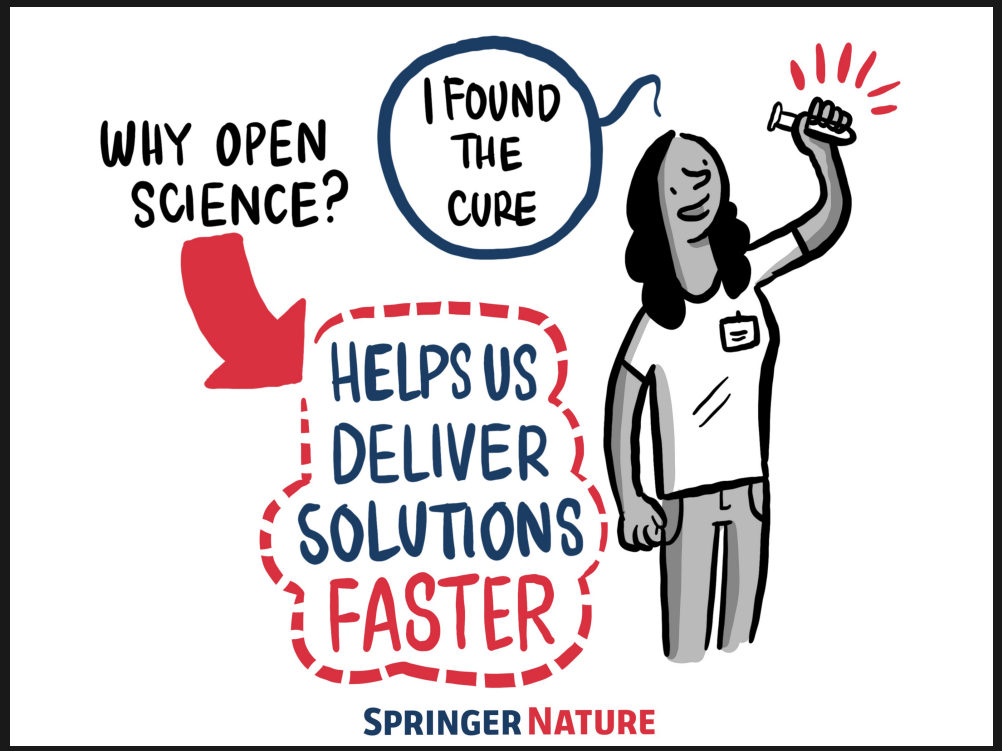
- fear of prejudice
  - important to create a positive culture
- fear of ‘theft’
  - licensing and citation
- technical and time barriers
  - support is available!
  - only need to learn once
- non-commercialisable?
  - open source and commercialisation are compatible
  - greater impact through open source



Better Science through Better Data 2017) scribe images.

Access to the slides: here - **Source** Github repository

Benefits of FAIR^{2}4RS



- Accelerate research
- Increase transparency of research
- Increase visibility, citation, reputation and impact
- Reduce duplication of effort

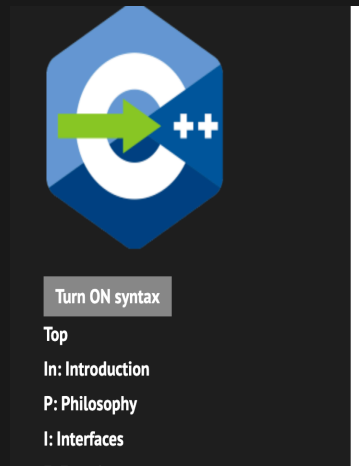
Better Science through Better Data 2017) scribe images.

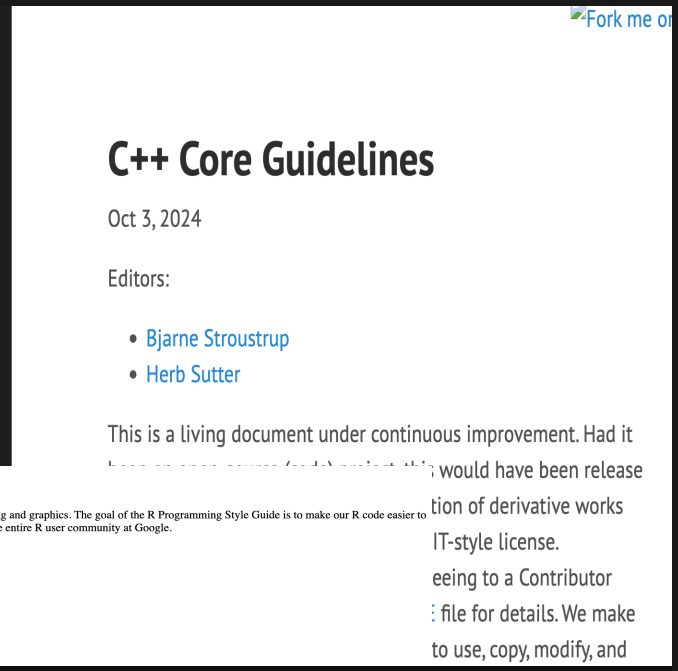
How to be FAIR?

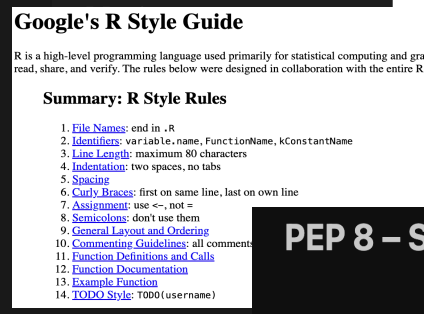
Access to the slides: here - **Source** Github repository

FAIR4RS: Think about how you are coding…

- Where possible, make your code modular.
- Comment your code to make it as clear as possible.
- Create and provide tests that others can use.
- Follow code standards



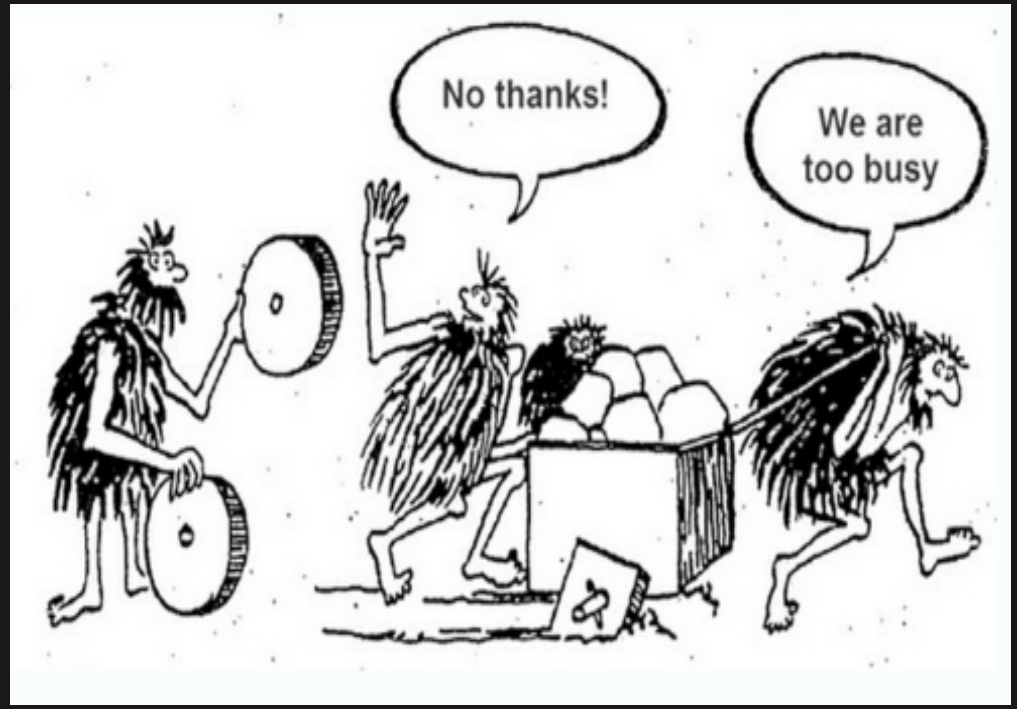




Access to the slides: here - **Source** Github repository

FAIR4RS: Be open even inside the code!

- Where possible and applicable, outputs (even between pieces of code) should use open and accessible data formats, which will help if other researchers only wish to use part of your code.
- But do NOT reinvent the wheel! In some research fields data format are standardized ➡️ if you want people to use your code, use [your] community standards!



FAIR4RS: Version your code!

- Using version control software platform such as Github/GitLab allows you to keep track of the changes you make to your code
- You can release version of your software/code/scripts directly from Github. While it should not be used a long term storage place, It gives a place where your code can be downloaded and where people can contribute.

Repository	Data supported	Examples	Benefits
Software	Software under development (eg software tools, libraries, scripts, packages, interactive notebooks). It is good software engineering practice to use a version controlled repository for development. Software releases (eg code associated with a specific publication or data set) should also be placed on a platform that supports long term storage, eg ORDA.	GitHub GitLab	Collaborative tools to facilitate good practice and sustainability. These do not guarantee long term storage or define metadata.

https://www.sheffield.ac.uk/library/research-data-management/repositories

Access to the slides: here - **Source** Github repository

FAIR4RS: Document your code!

A little poem from A beginner's guide to writing documentation:

- If people don't know why your project exists, they won't use it.
- If people can't figure out how to install your code, they won't use it.
- If people can't figure out how to use your code, they won't use it.

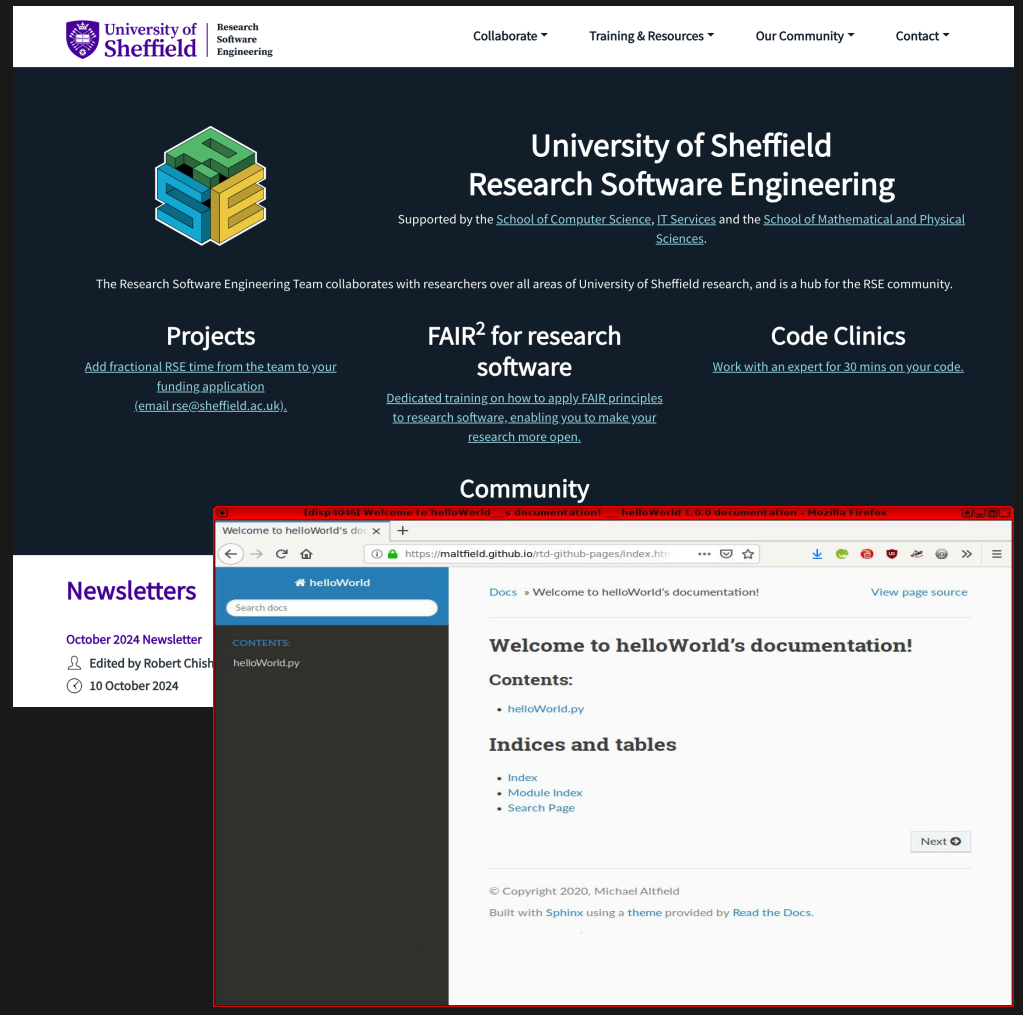
Access to the slides: here - **Source** Github repository

FAIR4RS: Document your code!

A little poem from A beginner's guide to writing documentation:

- If people don't know why your project exists, they won't use it.
- If people can't figure out how to install your code, they won't use it.
- If people can't figure out how to use your code, they won't use it.

In practice, Github can host documentation as website (and it is very easy to do!) ➡️➡️

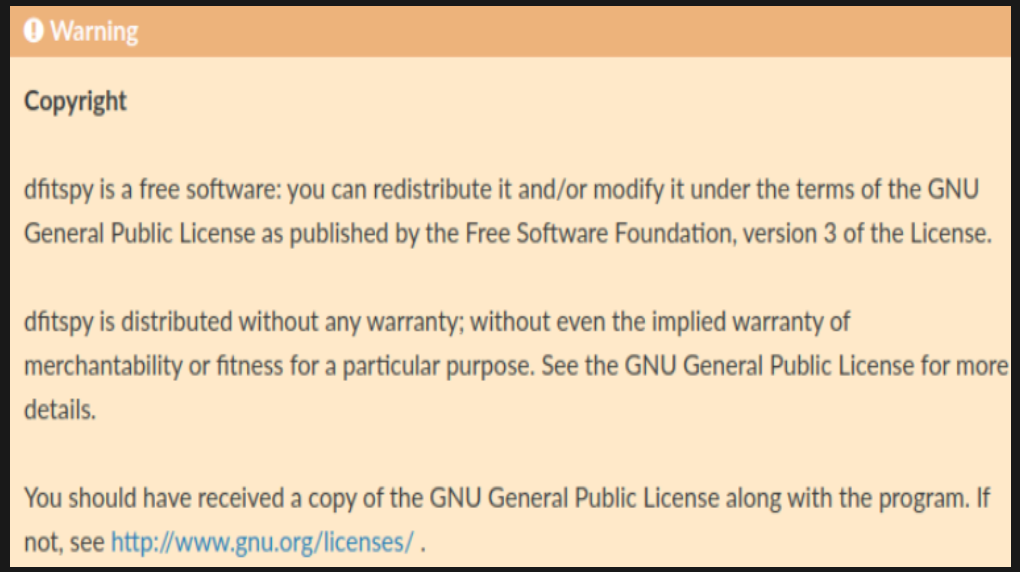


Access to the slides: here - **Source** Github repository

FAIR4RS: Licence your code!

You need to tell people how they can re-use your code.

- GPLv3 The GNU General Public License: a free, copyleft license for software and other kinds of works. It is intended to guarantee your freedom to share and change all versions of a software to make sure it remains free software for all its users
- MIT licence: is a permissive free software license. Without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,



Warning

Copyright

dfitspy is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 3 of the License.

dfitspy is distributed without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the program. If not, see http://www.gnu.org/licenses/ .

The licence must be made clear in the code repository and in the documentation.

Access to the slides: here - **Source** Github repository

FAIR4RS: Get credit for your work

If people are using your software you should get credit for it.

➡️ state how you want to be credited. You can add it in the documentation and/or create a citation.cff file that you can add with your code (tools. are available to generate them)

Acknowldgements and citation

The python wrapper of the CFITSIO library has been made for the fitsio python library (fitsio) and is used in dfitspy.

If you get to use dfitspy for your work, please quote the JOSS Paper [Journal of Open Source software]. Thanks!

```yaml
cff-version: 1.2.0
message: "If you use this software, please cite it as below."
authors:
  - family-names: Druskat
    given-names: Stephan
    orcid: https://orcid.org/1234-5678-9101-1121
title: "My Research Software"
version: 2.0.4
identifiers:
  - type: doi
    value: 10.5281/zenodo.1234
date-released: 2021-08-11
```

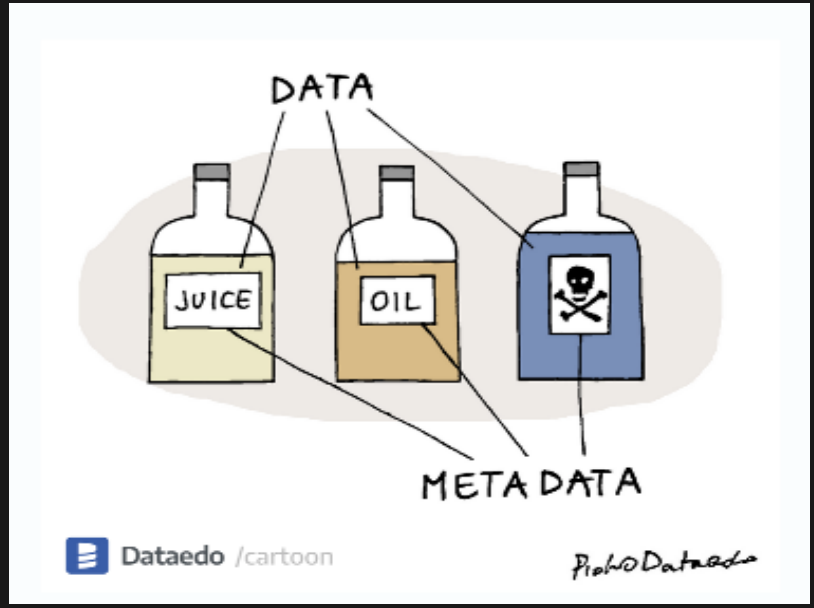
Access to the slides: here - **Source** Github repository

FAIR4RS: Share it!

- Create a description of your code with metadata [data about your software].
- Codemeta is a set of keywords used to describe code and how to structure them in a machine readable way

Examples:

- The citation.cff file contains metadata
- Description keywords
- Url to repository, to documentation
- List of contributors and affiliations
- etc…

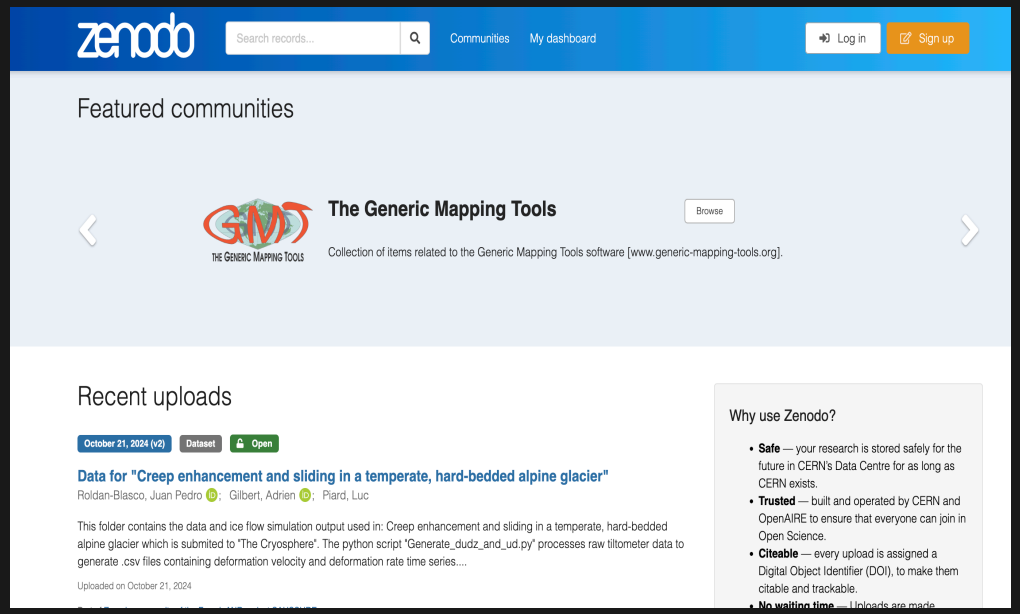


FAIR4RS: Share it!

In order to ensure that others can access and download your code, and that this access remains permanent over time, you should deposit your code in a repository. Two types:

- General purpose
- Domain Specific

Choose what makes more sense for you project!



Access to the slides: here - **Source** Github repository

FAIR4RS: Share it!

In order to ensure that others can access and download your code, and that this access remains permanent over time, you should deposit your code in a repository. Two types:

- General purpose
- Domain Specific

Choose what makes more sense for you project!

Numerous repositories give your content a DOI [Digital Object Identifier] It means it can be cited in publication and other communications in order to open up your research to others and invite collaboration, as well as ensuring a constant link to your code.

Access to the slides: here - **Source** Github repository

FAIR4RS: Publish it!

- Generalist software journals
  - JOSS: Journal of Open Source Software: Academic journal with a formal peer review process that is designed to improve the quality of the software submitted.
  - JORS: Journal of Open Research Software: Features peer reviewed Software Metapapers describing research software with high reuse potential.
  - Software Impacts: multidisciplinary, open access, peer-reviewed journal which publishes short, articles that describe software which addresses a research challenge.
- Some are domain specific:
  - Astronomy and computing
  - Journal of Artificial Societies and Social Simulation
  - Journal of Statistical Software
  - Science of Computer Programming
  - Computer Methods and Programs in Biomedicine

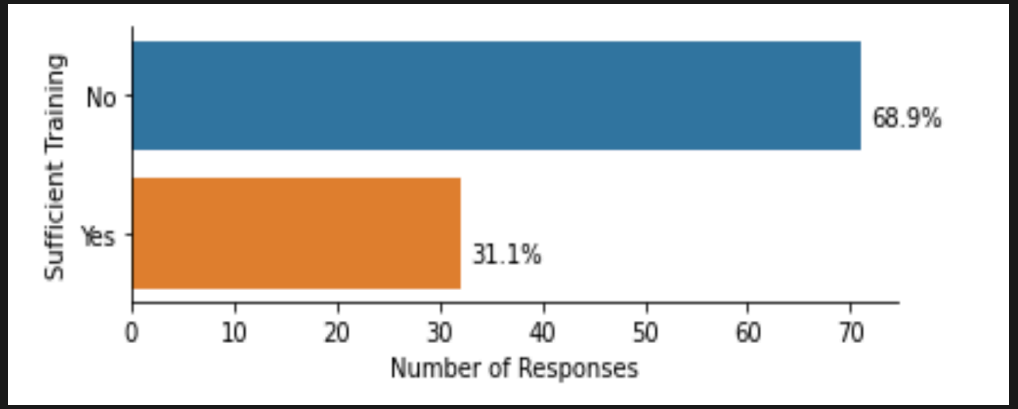
You can find a list of potential journals here

The FAIR^{2}4RS training programme at the University

Access to the slides: here - **Source** Github repository

Lack of skills for developing software

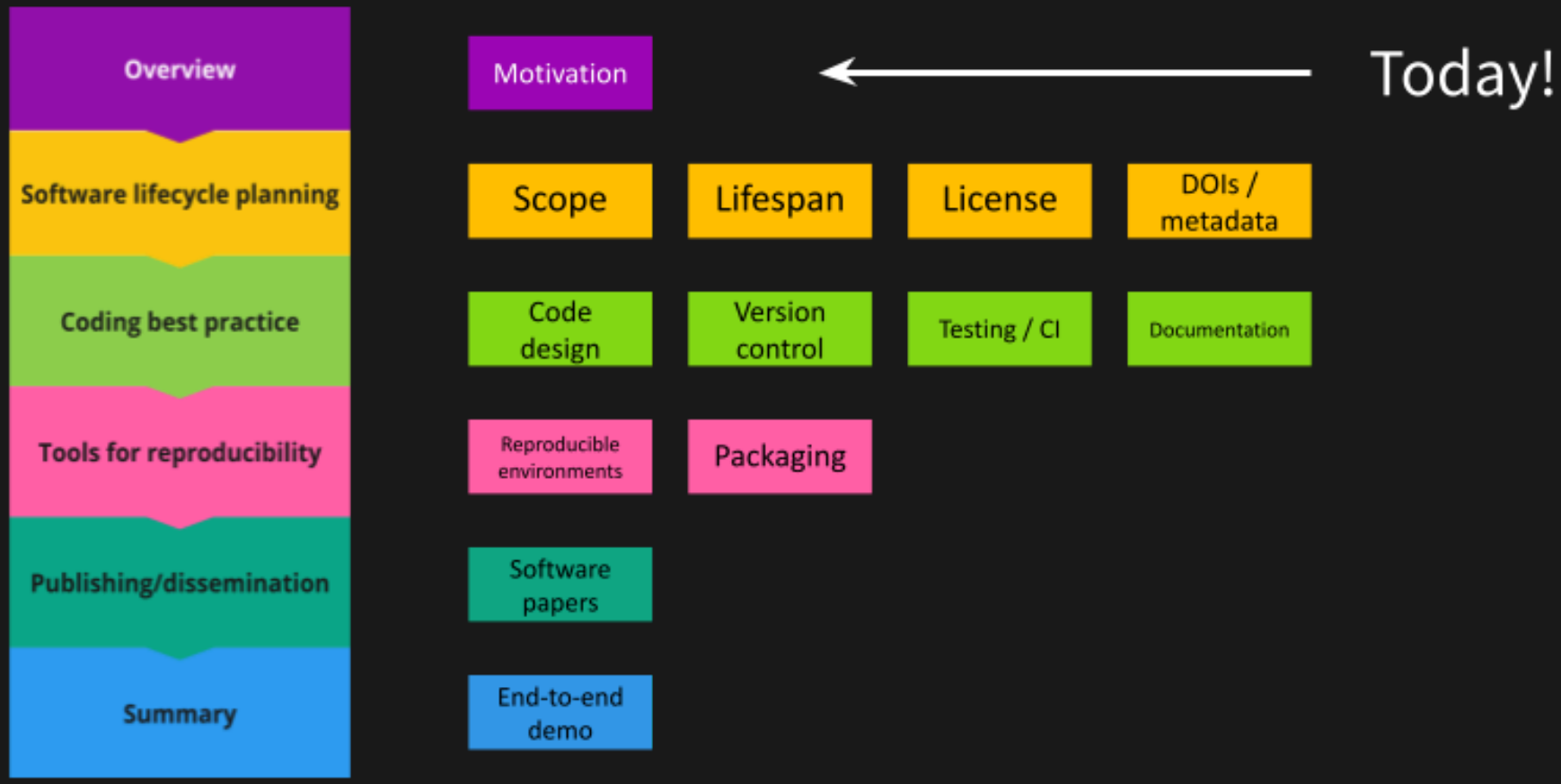
Do you feel that you have received sufficient training to develop reliable software?



Bob Turner & Paul Richmond, UoS, RSE team, github.com/RSE-Sheffield/sssurvey.

Access to the slides: here - **Source** Github repository

The FAIR^{2}4RS Programme: Overview



Access to the slides: here - **Source** Github repository

Version control: Git, GitHub and GitKraken - From Zero to Hero

Who: Michael Foster & Dan Brady

When/Length: 18/19 November (2 half days). Another session in February

Format: In person

Abstract:

If you've never heard of or used version control and Git before this is the course for you. We start by introducing version control and exploring how it can be beneficial to researchers, then we introduce some useful tools and get started with some basic workflow using these tools. We build on those foundations with collaborative exercises that introduce key concepts such as forks, pull requests and branches and give you the chance to *get some hands-on experience with using version control in a research setting*.

Access to the slides: here - **Source** Github repository

[Advanced] Version control: Git with it!

Who: Neil Shephard

When/Length: February

Format: In person

Abstract:

This course aims to help you develop a *deeper understanding of how Git works to facilitate collaboration*. It builds on foundations laid by the Git beginners course. The core idea around the course is that by improving your understanding of working with branches and how to make your commits tidier and neater it makes it easier to understand pull requests and Git history which in turn makes it easier to collaborate and work on code with others (including your future self!).

Access to the slides: here - **Source** Github repository

Design your code (and write less of it)

Who: Martin Dyer & Neil Shephard

When/Length: 26/11 & 03/12 (2 half days)

Format: In person

Abstract:

The way you write your code will have a massive impact on how easy it is to maintain in the long run. *This course on Code Design introduces essential principles and best practices for writing clean and maintainable code*. We will learn how we can write clean code, adhering to naming conventions, commenting, and following PEP 8 guidelines. We will then explore some fundamental principles such as DRY, KISS or YAGNI that are important to keep in mind when writing new code and see how we can spend less time touching the code by introducing configuration files and command line interface.

Access to the slides: here - **Source** Github repository

Software Lifecycle Planning

Who: R Thomas, R. Campbell, K. O'Neill

When/Length: 11/12, ~2.5h (afternoon)

Format: Online

Abstract:

When you start writing software it is often very useful to think about the development process and how you will make your software sustainable in the long term. In this module we will introduce important aspects of software development in research: *software management plan, licences and dissemination*. This module should allow you to ask yourself the right questions when starting a research software project.

Access to the slides: here - **Source** Github repository

Code Testing

Who: Sylvia Whittle & Michael Foster

When/Length: January (half day)

Format: In person

Abstract:

Does your code work? Are you sure? How do you ensure that it keeps working when you change it? Manually verifying is slow and tedious. Why not automate it? Software testing checks that your code works for you, and when it breaks, it can show you exactly where it broke, without you having to trawl through hundreds of lines of code manually.*This course aims to provide you with the tools you need to start automatically ensuring the reliability of your code.*

Access to the slides: here - **Source** Github repository

Documentation

Who: Joe Heffer

When/Length: February (half day)

Format: In person

Abstract:

Well-documented software promotes reproducibility, maintainability, and increased research impact through wider adoption and citation. *This course teaches researchers how to document their software effectively, making it accessible and understandable to others.* It covers topics such as writing readable code and usage instructions.

Access to the slides: here - **Source** Github repository

Reproducible Environments

Who: Dan Brady

When/Length: March [half day]

Format: In person

Abstract:

Ensuring that others are able to take your code, run it, and are able to produce the same (or equivalent) results is one of the key tenets of FAIR and reproducible research software. *This course will provide you with an overview of different ways to make your code reproducible* and then focus on virtual environments as a specific tool for computational reproducibility.

Access to the slides: here - **Source** Github repository

Packaging

Who: Chris Wild and Farhad Allian

When/Length: April [half day]

Format: In person

Abstract:

Packaging your software is one of the most important steps in a software project to make it both findable and accessible. *This course will provide you with an understanding of why and when packaging is useful*, what different standards exist to package Python projects and take you through each step of the packaging process.

Access to the slides: here - **Source** Github repository

Software Papers

Who: Romain Thomas

When/Length: April, 45min

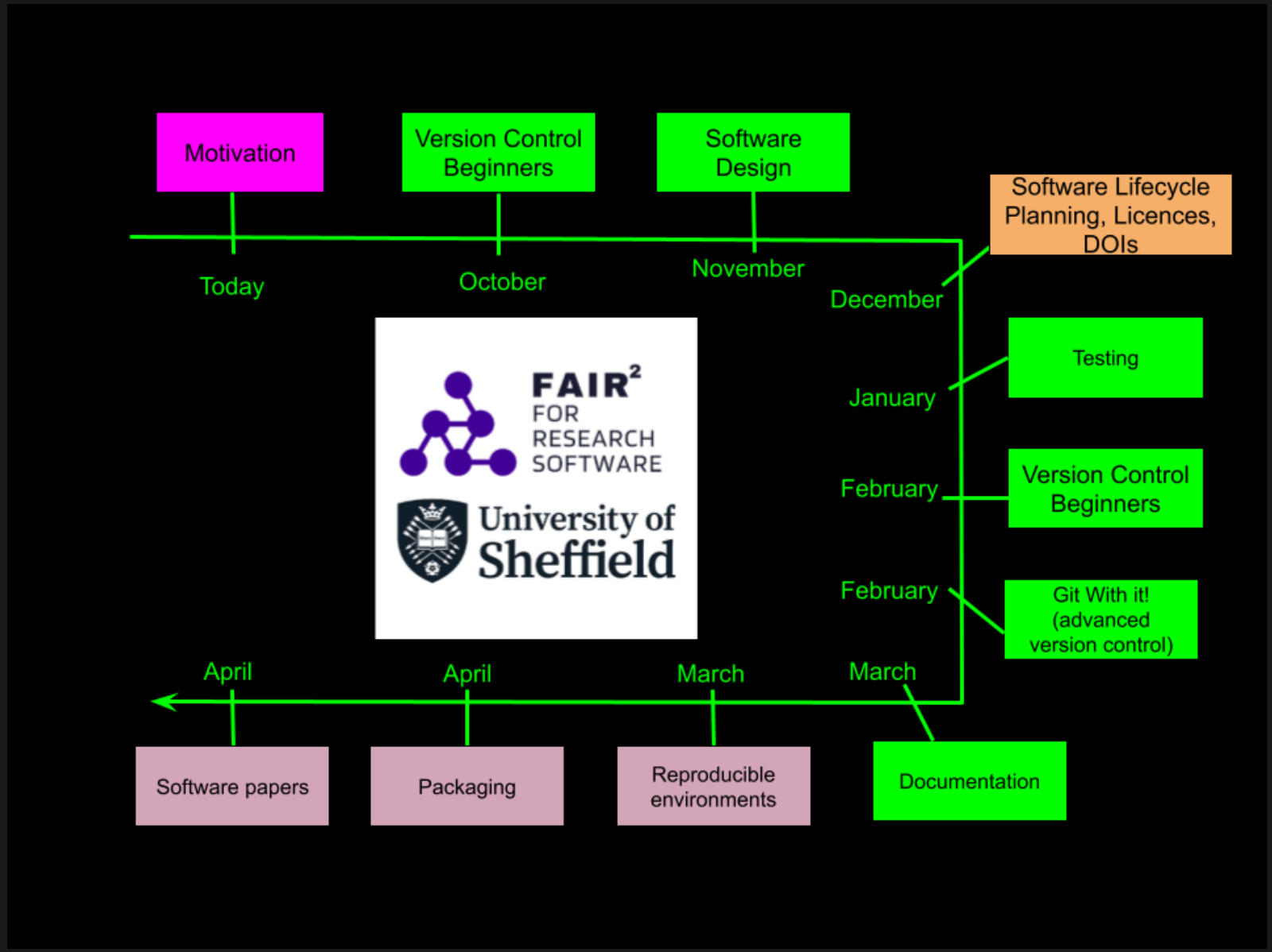
Format: Online

Abstract:

Did you know that you can actually publish a paper about your software? This is an ideal way to get recognition (and citation) for the software you have spent countless hours creating. *In this course we will walk you through an example of submission in the Journal of Open Source Software.* We will make an example software submission to the journal, and thanks to the collaboration of the Editor in Chief of JOSS (Arfon Smith), we will look at how the review process is done.

Access to the slides: here - **Source** Github repository

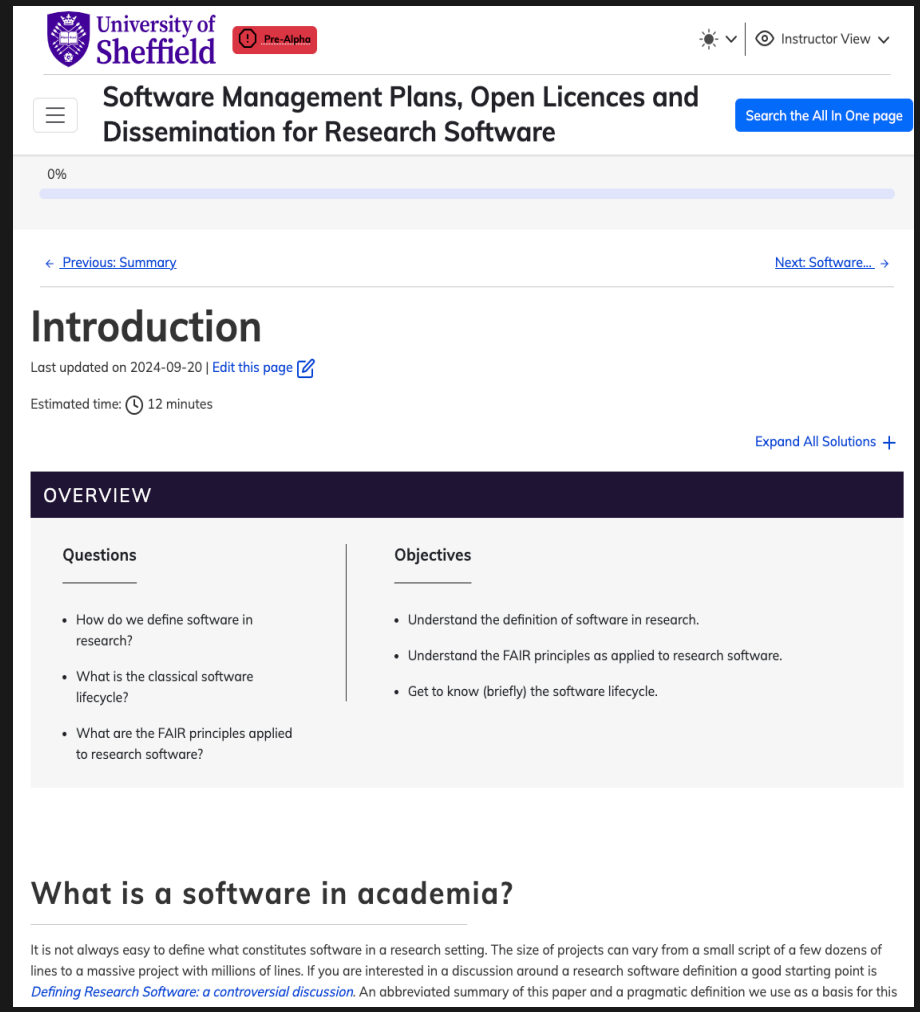
The FAIR²4RS Programme: Timeline



Access to the slides: here - **Source** Github repository

The FAIR^{2}4RS Programme: Material and dependencies

- All materials are designed using the same structure (Software Carpentry workbench) and are freely accessible on Github.
- **You can pick-and-choose the lecture** you will follow based on the skills you already have. Each lecture comes with a set of prerequisities that are clearly identified.
- A feedback form will be provided after each lecture.



Access to the slides: here - **Source** Github repository

the fair^{2}4rs programme: important notes

- Training are all Free of charge.
- BUT! they all need registration and in-person sessions have limited places!
- They will all be available on mydevelopment platform. The first 3 sessions are already open for registration:
  - Git/Github zero to Hero - Oct 28th & Nov 4th
  - Code Design - Nov 26th & Dec 3rd
  - Software Management plan - Dec 12th
- January onward sessions will be available for booking around December.

- Direct links are also available on the RSE website:

Access to the slides: here - **Source** Github repository

RSE website

Contacts:

- Tamora James -RSE and FAIR²4RS Programme Manager -(t.d.james@sheffield.ac.uk)
- Romain Thomas -Head of RSE-(romain.thomas@sheffield.ac.uk)

Access to the slides: here - **Source** Github repository

Acknowledgements & References

- Thank you to Tamora James for leading the development of this training programme
- Thank you to Christopher Wild, Ric Campbell, Farhad Allian, Daniel Brady, Kate O'neill, Joe Heffer, Jenni Adams, Neil Shephard, Sylvia Wittle and Arfon Smith for dedicating time to prepare all the material!

References

* D. Wilby Lunchbyte talk on the FAIR principles
* T. James, FAIR for research software, Talk OpenFest 2024
* The Turing Way
* B. Sirvey Le grand homme qui apprend
* Chue Hong, Neil P. et al, FAIR principles for Research Software

Thank you!

Help us improve!

Scan to give your feedback!

Access to the slides: here - **Source** Github repository