Making software FAIR

Slides: https://doi.org/10.6084/m9.figshare.22347154

28 March 2023, Academic Training Lectures, CERN (virtual).
Neil Chue Hong (@npch), Software Sustainability Institute / EPCC
ORCID: 0000-0002-8876-7606 | N.ChueHong@software.ac.uk
Research is not a competition against each other
Our aim is knowledge for society
The pandemic showed us what we gain from collaboration
Open Science needs software

“Re-use and value of data can depend on the availability of relevant metadata, algorithms, code, and software, together with information on workflows and the computational environment used”
- OECD Recommendation on Access to Research Data from Public Funding (2020)

“In the case of open source software, a community-driven process for contribution, attribution and governance is required to enable reuse, improve sustainability and reduce unnecessary duplication of effort.”
- UNESCO Recommendation on Open Science (2021)
“Arman Bilge, a 10th grader at Lexington High School in Massachusetts, was a newbie to phylogenetics when a science teacher there organized an after-school phylogenetic tree club. In the club, Bilge learned how to use a variety of software applications, including one well known to systematic biologists called BEAST.”

Slide courtesy of Nancy Wilkins-Diehr
BEAST software licensed under LGPL
In 2011 **Science changed its editorial policies**: “We require that all computer code used for modeling and/or data analysis that is not commercially available be deposited in a publicly accessible repository upon publication.”

“Normally we do not provide this kind of information to people we do not know. It might be that you want to check the data analysis, and that might be of some use to us, but only if you publish your findings while properly referring to us.”

“Thank you for your interest in our paper. For the [redacted] calculations I used my own code, and there is no public version of this code, which could be downloaded. Since this code is not very user-friendly and is under constant development I prefer not to share this code.”

“I have to say that this is a very unusual request without any explanation! Please ask your supervisor to send me an email with a detailed, and I mean detailed, explanation.”

“When you approach a PI for the source codes and raw data, you better explain who you are, whom you work for, why you need the data and what you are going to do with it.”

---

Stodden, Seiler, Ma. An empirical analysis of journal policy effectiveness for computational reproducibility

https://doi.org/10.1073/pnas.1708290115
Research relies on software

Do you use research software?

- Yes: 92%
- No: 8%

What would happen to your research without software?

- Would be impossible: 68%
- Possible, but difficult: 21%
- Develop their own software: 56%
- Have no formal software training: 71%
- No effect: 11%

2014 UK Research Software Survey. DOI: 10.5281/zenodo.14809
Software and Research

**Software is essential to research**

- 5 - Essential: 76
diagram showing distribution of importance

**Most researchers develop software**

- 80
diagram showing distribution of developers

**Most important software**

<table>
<thead>
<tr>
<th>Software</th>
<th>N</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>76</td>
<td>19</td>
</tr>
<tr>
<td>matlab</td>
<td>46</td>
<td>11.5</td>
</tr>
<tr>
<td>r</td>
<td>39</td>
<td>9.75</td>
</tr>
<tr>
<td>latex</td>
<td>11</td>
<td>2.75</td>
</tr>
<tr>
<td>mathematica</td>
<td>11</td>
<td>2.75</td>
</tr>
<tr>
<td>stata</td>
<td>10</td>
<td>2.5</td>
</tr>
<tr>
<td>git</td>
<td>9</td>
<td>2.25</td>
</tr>
<tr>
<td>pytorch</td>
<td>9</td>
<td>2.25</td>
</tr>
<tr>
<td>amber</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>vasp</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>overleaf</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>imagej</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>gaussian</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>fiji</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>paraview</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>excel</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>javascript</td>
<td>8</td>
<td>2.02</td>
</tr>
<tr>
<td>rust</td>
<td>7</td>
<td>1.76</td>
</tr>
<tr>
<td>csharp</td>
<td>5</td>
<td>1.26</td>
</tr>
<tr>
<td>cuda</td>
<td>5</td>
<td>1.26</td>
</tr>
<tr>
<td>c#</td>
<td>4</td>
<td>1.01</td>
</tr>
<tr>
<td>php</td>
<td>3</td>
<td>0.76</td>
</tr>
</tbody>
</table>

**Most used languages**

<table>
<thead>
<tr>
<th>Language</th>
<th>N</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>235</td>
<td>59.19</td>
</tr>
<tr>
<td>fortran</td>
<td>98</td>
<td>24.69</td>
</tr>
<tr>
<td>c++</td>
<td>92</td>
<td>23.17</td>
</tr>
<tr>
<td>c</td>
<td>65</td>
<td>16.37</td>
</tr>
<tr>
<td>matlab</td>
<td>57</td>
<td>14.36</td>
</tr>
<tr>
<td>r</td>
<td>52</td>
<td>13.1</td>
</tr>
<tr>
<td>bash</td>
<td>28</td>
<td>7.05</td>
</tr>
<tr>
<td>javascript</td>
<td>8</td>
<td>2.02</td>
</tr>
<tr>
<td>java</td>
<td>26</td>
<td>6.55</td>
</tr>
<tr>
<td>perl</td>
<td>10</td>
<td>2.52</td>
</tr>
<tr>
<td>idl</td>
<td>8</td>
<td>2.02</td>
</tr>
<tr>
<td>pytorch</td>
<td>9</td>
<td>2.25</td>
</tr>
<tr>
<td>amber</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>vasp</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>overleaf</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>imagej</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>gaussian</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>fiji</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>paraview</td>
<td>7</td>
<td>1.75</td>
</tr>
<tr>
<td>excel</td>
<td>7</td>
<td>1.75</td>
</tr>
</tbody>
</table>

UKRI Software and Skills Survey (2022)
Research software?

Code Lifespan
https://xkcd.com/2730/

From xkcd.com
by Randall Munroe
CC-BY-NC licensed
Software Sustainability Institute

A national facility for cultivating better, more sustainable, research software to enable world-class research

- Software reaches boundaries in its development cycle that prevent improvement, growth and adoption
- Providing the expertise and services needed to negotiate to the next stage
- Developing the policy and tools to support the community developing and using research software

Supported by all seven UK Research Councils through grants EP/H043160/1 + EP/N006410/1 + EP/S021779/1
Software

Helping the community to develop software that meets the needs of reliable, reproducible, and reusable research

Training

Delivering essential software skills to researchers via CDTs, institutions & doctoral schools

Outreach

Exploiting our platform to enable engagement, delivery & uptake

Collecting evidence on the community’s software use & sharing with stakeholders

Policy

Bringing together the right people to understand and address topical issues

Community

Exploiting our platform to enable engagement, delivery & uptake

Would be impossible 68%

Possible, but difficult 21%

No effect 11%
Research Software Tiers

- **Analysis Code**
  - One-off “me” research
  - Often not revised after publication

- **Prototype Tools**
  - Research need “professorware”
  - Often best-effort maintenance

- **Research Software Infrastructure**
  - Professionalised product

Adapted from Tom Honeyman, ARDC, after Konrad Hinsen
Good code takes practice

- Writing good code is not easy
- But there are things that make it easier over time
- The key is applying them and practicing their use
- Saves you time in the future

Xkcd: Good Code by Randall Munroe
https://xkcd.com/844/
FAIR Principles

- Findable
- Accessible
- Interoperable
- Reusable

10.1038/sdata.2016.18

The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807
Towards FAIR software

- A joint RDA Working Group, FORCE11 Working Group, and Research Software Alliance (ReSA) Taskforce.
  - 250 members, 80 active contributors.
- Coordinating of a range of existing community-led discussions on:
  - How to define and effectively apply FAIR principles to research software,
  - How to achieve adoption of these principles.

Introducing the FAIR Principles for research software (Scientific Data)
FAIR Principles for Research Software (FAIR4RS Principles) v1.0 (RDA)
• **Findable:** Software, and its associated metadata, is easy for both humans and machines to find.

• **Accessible:** Software, and its metadata, is retrievable via standardized protocols.

• **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).

(key differences from FAIR data principles in *italics*)
FAIR4RS Principles

F: Software, and its associated metadata, is easy for both humans and machines to find

F1. Software is assigned a globally unique and persistent identifier.
   F1.1. Components of the software representing levels of granularity are assigned distinct identifiers.
   F1.2. Different versions of the software are assigned distinct identifiers.

F2. Software is described with rich metadata.
F3. Metadata clearly and explicitly include the identifier of the software they describe.
F4. Metadata are FAIR, searchable and indexable.

A: Software, and its metadata, is retrievable via standardized protocols.

A1. Software is retrievable by its identifier using a standardized communications protocol.
   A1.1. The protocol is open, free, and universally implementable.
   A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
A2. Metadata are accessible, even when the software is no longer available.

I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.

I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.
I2. Software includes qualified references to other objects

R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).

R1. Software is described with a plurality of accurate and relevant attributes.
   R1.1. Software is given a clear and accessible license.
   R1.2. Software is associated with detailed provenance.
R2. Software includes qualified references to other software.

## FAIR4RS Principles

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Findable</td>
<td>Use persistent identifiers and descriptive metadata</td>
</tr>
<tr>
<td>Accessible</td>
<td>Make software and metadata easily retrievable</td>
</tr>
<tr>
<td>Interoperable</td>
<td>Use APIs, standards and references</td>
</tr>
<tr>
<td>Reusable</td>
<td>Document, license, and follow community good practice</td>
</tr>
</tbody>
</table>

**Software Sustainability Institute**
FAIR enough principles?

Write code to be readable, reusable & testable

1. Use a code repository and version control
2. License your software
3. Document for your future self
4. Split your code into small, modular parts
5. Use libraries for common functionality
6. Share your code with others
Licenses – key questions

• What is your objective? What impact do you seek?
  ▪ Disseminate research / research outcomes
  ▪ Supporting reproducibility
  ▪ Widespread usage / build community
  ▪ Commercial revenue / sell related services + infrastructure
  ▪ Social or cultural change

• Do you care whether changes made by others are made available?
• Do you care if certain people / organisations use your work?
• Does your work depend on / incorporate other works?
• Is there common practice in use already in your community?
Types of software license

<table>
<thead>
<tr>
<th>Type of license</th>
<th>Closed source / Proprietary</th>
<th>Academic / Non-commercial</th>
<th>Freeware</th>
<th>Copyleft</th>
<th>Permissive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides copyright protection</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can be used for commercial applications</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows redistribution</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows reuse / modification (including in commercial products)</td>
<td>No¹</td>
<td>No¹</td>
<td>No¹</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows reuse in closed source projects</td>
<td>Depends on license</td>
<td>No (normally)¹</td>
<td>No (normally)¹</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Requires changes to be shared</td>
<td>No (normally)¹</td>
<td>No (normally)¹</td>
<td>No (normally)¹</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Ability to restrict categories of users</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Examples of license</td>
<td>Matlab end user license</td>
<td>CASTEP license, OpenCarp license</td>
<td>Adobe Acrobat Reader license</td>
<td>GPL, LGPL, AGPL</td>
<td>BSD, MIT, Apache</td>
</tr>
</tbody>
</table>

Notes:
1. Unless license specifically allows it
2. Subset of Closed Source licenses
Why license? Protection

Protect suppliers and users
- “We used your software and it wiped our astronomy data”
- “We used your software, our lab burnt down and someone died”

Warranty
- Commitment to remedy defects

Liability
- Extent to which supplier is liable to provide remedies e.g. repairs, replacements, compensation
- Subject to fairness criteria

Indemnity
- Commitment by supplier to compensate user
Why license? Exploiting work

Commercialising your work

• A license allows you to set out the conditions of use
• Can use to define users rights when selling software commercially
  • Note: you can sell you software and have an open source license (more later)
• Choosing the right license will help you exploit your software outside the university, e.g. if you want to setup a company based on the software you developed

Getting more users and contributors

• A license can help users to choose software, or contribute back
• The right license can be used to build a community or encourage others to build additional functionality or tools that work with your software
Use community standards

- FAIR Principles for Research Software advocate for following community standards
  - Open formats for data
  - Choose common licenses, programming languages, libraries, style guides
- Improves both interoperability and reusability
  - You may need to help facilitate standardisation
Rich metadata description

• Document your software, ideally in a machine readable way
  ▪ README, LICENSE, CONTRIBUTION
  ▪ Dependencies
  ▪ APIs
  ▪ Tests
Why researchers should share their source code

• Methods do not produce results, source code does
  ▪ Results are produced by the implementation of a method
  ▪ Method may be scientifically valid, but its implementation flawed

• Allow others to
  ▪ Validate what has been done and to determine whether conclusions are sound
  ▪ Replicate, reproduce and reuse research

• Preserve historical record
  ▪ Source code has a value even if it no longer can be compiled or run
  ▪ Programmatic description of the research that was done

• Improve quality and trust
• Conform to requirements of funders and publishers
Why researchers don’t share their source code

- Web/disk space limitations 20%
- Competitors may get an advantage 30%
- Potential loss of future publications 30%
- Legal barriers, such as copyright 33%
- Possibility of patents 40%
- Code may be used without citation 44%
- Handle questions from users 51%
- Time to clean up and document 77%

Sharing your code

Don’t be afraid to share your code with others

• Get feedback – best way of finding bugs
  ▪ Get a colleague to use it
  ▪ Ask a collaborator to contribute

• Publish your code (and data)
  ▪ Deposit in a repository
  ▪ Cite in your papers, have a clear preferred citation

• If you use someone else’s code, contribute
  ▪ But you shouldn’t expect anything directly in return
# Software publishing options

<table>
<thead>
<tr>
<th>Code repository</th>
<th>Deposit in digital repository</th>
<th>Produce runnable version</th>
<th>Register in catalogue / registry</th>
<th>Paper in software journal</th>
<th>Paper in domain-specific journal</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td>Source code is in GitHub, GitLab or BitBucket with open license</td>
<td>Source code deposited in <a href="https://zenodo.org">Zenodo</a>, <a href="https://figshare.com">Figshare</a> or an institutional repository</td>
<td>Jupyter Notebook in <a href="https://binder.org">Binder</a>, Capsule in <a href="https://codeocean.com">CodeOcean</a>, <a href="https://www.docker.com">Docker</a> or <a href="https://singularity.lbl.gov">Singularity</a> container, <a href="https://www.nextflow.io">NextFlow</a> workflow. Package for <a href="https://cran.r-project.org">CRAN</a>, <a href="https://pypi.org">PyPI</a>, etc</td>
<td>Create an entry in a community registries e.g. <a href="https://ascl.net">ASCL</a> (astronomy), [CIG](<a href="https://www.cig">https://www.cig</a> gebruik) (geodynamics), <a href="https://www.reproducibilityregister.org">RRID</a> or <a href="https://swmath.org">swMath</a> (mathematics). <a href="https://nlesc.rsd.org">NLeSC RSD</a></td>
<td>Publish software paper in <a href="https://journals.aps.org/jors/">JORS</a>, <a href="https://www.learnedsources.org/journals">JOSS</a>, <a href="https://www.sciencedirect.com/journal/software-x">SoftwareX</a> etc. Publish executable research article in <a href="https://www.gigabyte-journal.org">GigaByte</a></td>
</tr>
</tbody>
</table>

**Advantages**
- Discoverable
- Fits with development workflow
- No waiting before available
- Archived
- Persistent identifier and metadata
- Little/no wait before available
- Enable direct reuse
- Can be given identifiers
- Makes available in location where users search
- Indexed
- Easier to find
- Often provides identifier
- May show citations
- Easily citable
- Peer reviewed
- Can describe software design
- Easier for developers to write
- Easily citable
- Easier to reach target audience
- Understood by promotion committees

**Disadvantages**
- Not archived
- Harder to cite
- Not easy to find if poorly described / documented
- Direct software citations not accepted by all journals
- Normally requires additional effort / resources
- Not available in every domain
- Many people just Google, so must be indexed
- Software not always archived
- Not as “prestigious” as domain-specific journal
- Software generally not archived.
- Longer time to publishing.
- Not easy to run.

---

**Software Sustainability Institute**
Referencing and citing content
You can use third-party tools to cite and reference content on GitHub.

Issuing a persistent identifier for your repository with Zenodo

To make your repositories easier to reference in academic literature, you can create persistent identifiers, also known as Digital Object Identifiers (DOIs). You can use the data archiving tool Zenodo to archive a repository on GitHub.com and issue a DOI for the archive.
Getting credit – Citation Files

- **CITATION.cff** files are plain text files with human- and machine-readable citation information for software.
- Include them in repositories to let others know how to correctly cite your software.

```plaintext
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/0000-0003-4925-7248
title: "My Research Software"
version: 2.0.4
doi: 10.5281/zenodo.1234
date-released: 2021-08-11
```
Software Citation Checklist for Authors

- Have I identified the software which makes a significant and specialised contribution to my academic work?
- Have I checked if the software has a recommended citation?
  - If this is to a paper, have I also cited the software directly?
  - If there’s no recommended citation, have I created as complete a citation as possible?
    - Who created the software
    - When it was created
    - Title of the software (and version if available)
    - Where the software can be accessed
- Have I referenced the software appropriately in my academic work, complying with any citation formatting guidelines?

Checklist for authors: https://doi.org/10.5281/zenodo.3479199

Software Sustainability Institute
Software Citation Checklist for Developers

- Have I assigned an *appropriate license* to my software?
- Have I *described my software* properly, using an appropriate metadata format, and included this metadata file with my software?
  - Have I given my software a clear *version number*?
  - Have I determined the *authors to be credited* for this release of my software, and included this in my metadata file?
- Have I procured a *persistent identifier* for this release of my software?
- Have I added my *recommended citation* to the documentation for my software?

*Checklist for developers: https://doi.org/10.5281/zenodo.3482769*
In summary

- Science depends on software being reusable
- FAIR, citable software leads to collaboration
- Share your software for yourself, and others

*Without data it’s difficult to validate results.*

*But without software, we waste the opportunity to advance science.*
Acknowledgements

The SSI team/alumni:
- Agata Dybisz
- Aleksandra Nenadic
- Aleksandra Pawlik
- Alexander Hay
- Ania Brown
- Anita Banerji
- Arno Proeme
- Carole Goble
- Caroline Jay
- Claire Wyatt
- Clem Hadfield
- Dave De Roure
- Denis Barclay
- Devasena Prasad
- Elena Breitmoser
- Giacomo Peru
- Graeme Smith
- Iain Emsley
- Ioanna Lampaki
- Jacalyn Laird
- James Graham
- Jenny Braidwood
- Johanna Walker
- John Robinson
- Kara Moraw
- Kathleen Glass
- Kirsty Pringle
- Les Carr
- Lucia Michielin
- Malcolm Atkinson
- Malcolm Illingworth
- Mario Antonioletti
- Mark Parsons
- Mike Jackson
- Olivier Philippe
- Philly Broadbent
- Pip Grylls
- Priyanka Singh
- Rachael Ainsworth
- Raniere Silva
- Rob Baxter
- Robin Wilson
- Sam Manghan
- Selina Aragon
- Shoalb Sufi
- Simon Hettrick
- Stephen Crouch
- Tim Parkinson
- Toni Collins
- Plus the SSI Fellows and RSE community

Research software:
- Abby Cabunoc-Reyes
- Arfon Smith
- Carlos Martinez
- Dan Katz
- Heather Piowowar
- James Hetherington
- James Howison
- Jeff Carver
- Jennifer Schopf
- Kaitlin Thaney
- Karthik Ram
- Kirstie Whittaker
- Martin Fenner
- Michelle Barker
- Shelley Stall
- Stephan Druskat
- Victoria Stodden
- Von Welch
- WSSSPE community
- ReSA and FAIR4RS

The Carpentries
- Greg Wilson
- Tracy Teal
- Kari Jordan
- Instructor Community

Software Preservation
- Daina Bouquin
- Digital Preservation Coalition
- Software Preservation Network

Supported by the UK Research Councils through grants EP/H043160/1, EP/N006410/1 and EP/S021779/1.
Additional project funding received from Jisc, Horizon Europe, UKRI ExCALIBUR, NERC Short Courses, UKRI DaSH.
This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). See: https://creativecommons.org/licenses/by/4.0/

You are free to:

• **Share** — copy and redistribute the material in any medium or format

• **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

Under the following terms:

• **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

• **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Copyright © 2021, The University of Edinburgh as lead partner of the Software Sustainability Institute