THE UNIVERSITY OF EDINBURGH | MANCHESTER 1824 | UNIVERSITY OF OXFORD | UNIVERSITY OF Southampton

www.software.ac.uk

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

Supported by:
UKRI Arts and Humanities Research Council
UKRI Biotechnology and Biological Sciences Research Council
UKRI Economic and Social Research Council
UKRI Engineering and Physical Sciences Research Council
UKRI Medical Research Council
UKRI Natural Environment Research Council
UKRI Science and Technology Facilities Council

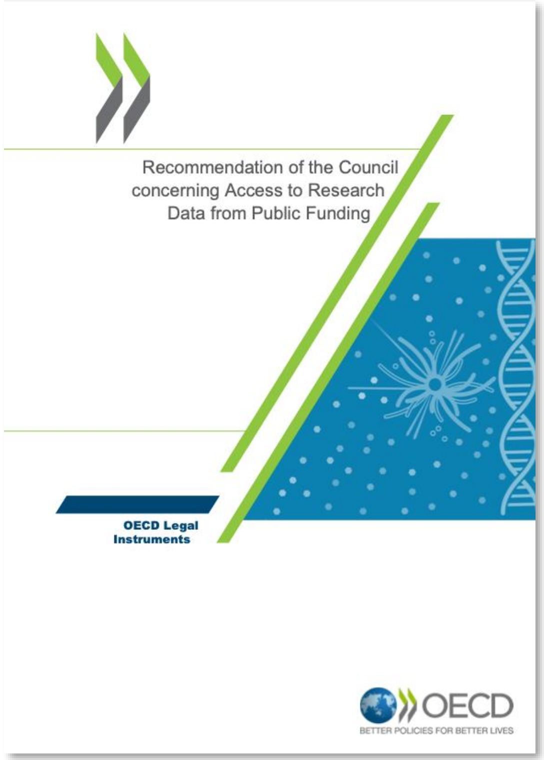
Research is not a competition against each other

Our aim is knowledge for society

The pandemic showed us what we gain from collaboration

Open Science needs software

www.software.ac.uk

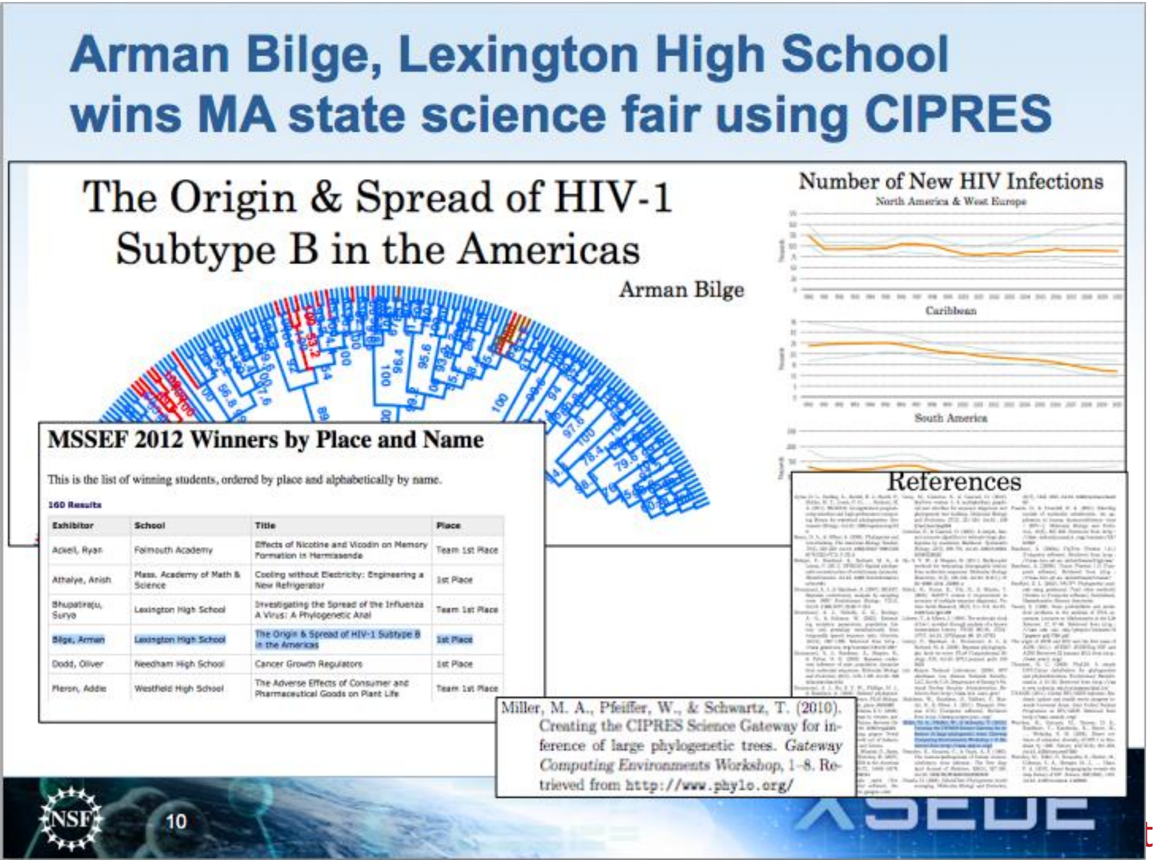


"Re-use and value of data can depend on the availability of relevant metadata, algorithms, code, and software, together with information on workflows and the computational environment used"
- *OECD Recommendation on Access to Research Data from Public Funding (2020)*

"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)*

Software lets others benefit

www.software.ac.uk



"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

Culture change is hard

www.software.ac.uk

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."

Table 1. Responses to emailed requests (*n* = 180)

Type of response	Count	Percent, %
Did not share data or code:		
Contact another person	20	11
Asked for reasons	20	11
Refusal to share	12	7
Directed back to supplement	6	3
Unfulfilled promise to follow up	5	3
Impossible to share	3	2
Shared data and code	65	36
Email bounced	3	2
No response	46	26

Stodden, Seiler, Ma. An empirical analysis of journal policy effectiveness for computational reproducibility
https://doi.org/10.1073/pnas.1708290115

"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."

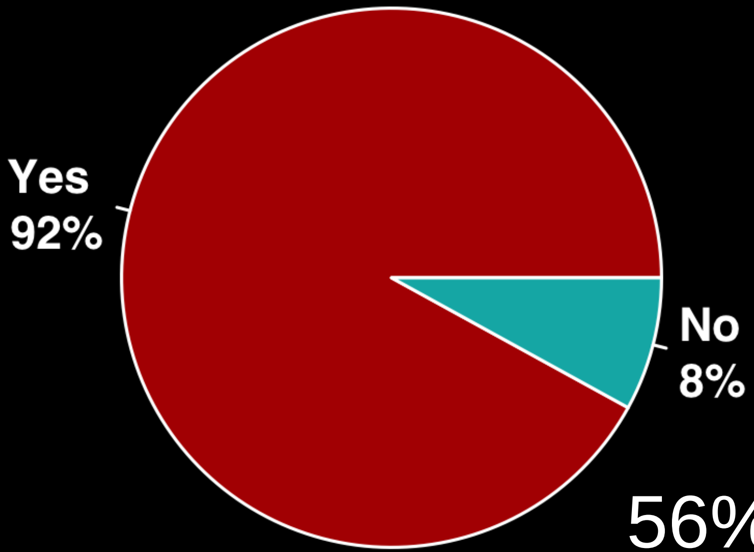
"I have to say that this is a very unusual request without any explanation! Please ask your supervisor to send me an email with a detailed, and I mean detailed, explanation."

"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."

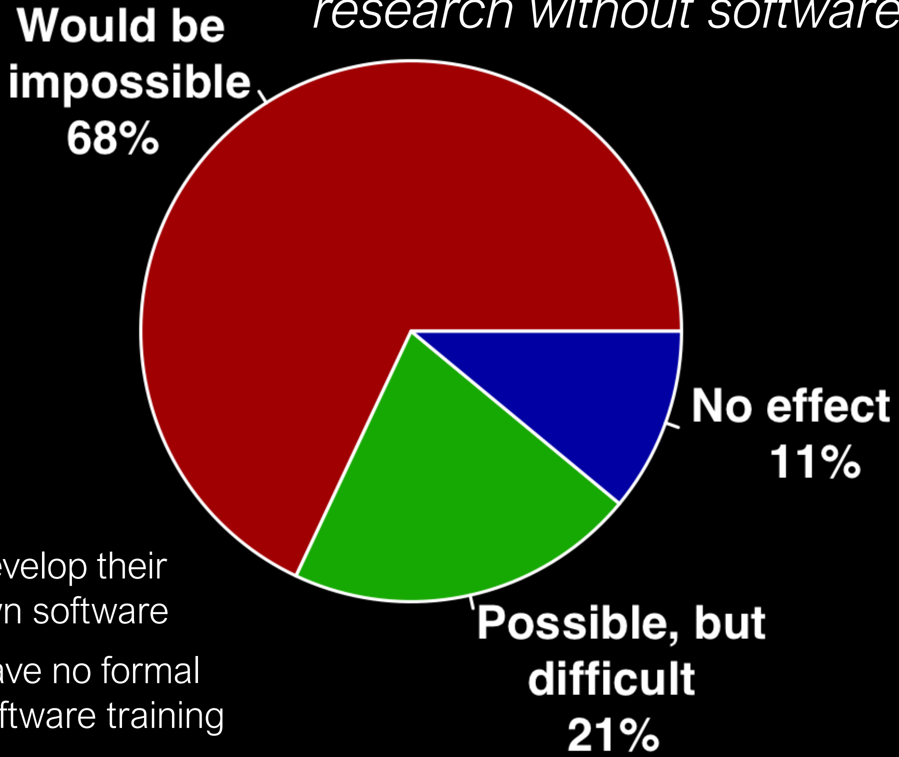
Research relies on software

www.software.ac.uk

Do you use research software?



What would happen to your research without software



56% Develop their own software

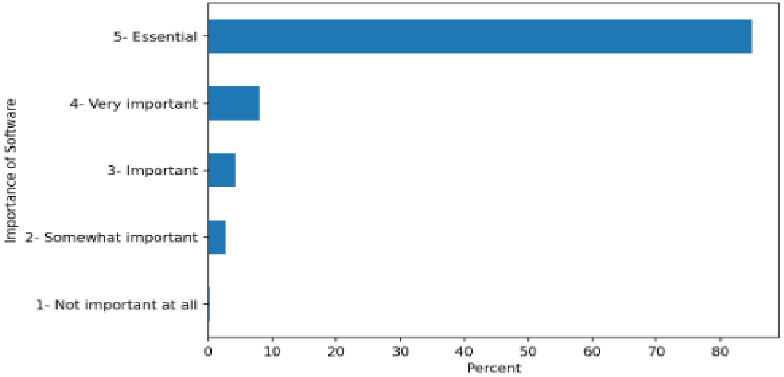
71% Have no formal software training

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

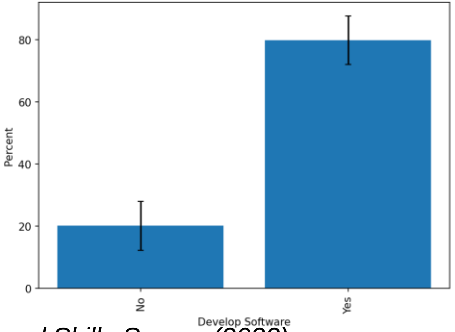
Software and Research

www.software.ac.uk

Software is essential to research



Most researchers develop software



Most important software

Software	N	%
python	76	19
matlab	46	11.5
r	39	9.75
latex	11	2.75
mathematica	11	2.75
stata	10	2.5
git	9	2.25
pytorch	9	2.25
amber	8	2
vasp	8	2
overleaf	7	1.75
imagej	7	1.75
gaussian	7	1.75
fiji	7	1.75
paraview	7	1.75
excel	7	1.75

Most used languages

Language	N	%
python	235	59.19
fortran	98	24.69
c++	92	23.17
c	65	16.37
matlab	57	14.36
r	52	13.1
bash	28	7.05
java	26	6.55
perl	10	2.52
idl	8	2.02
javascript	8	2.02
rust	7	1.76
cuda	5	1.26
julia	5	1.26
c#	4	1.01
php	3	0.76

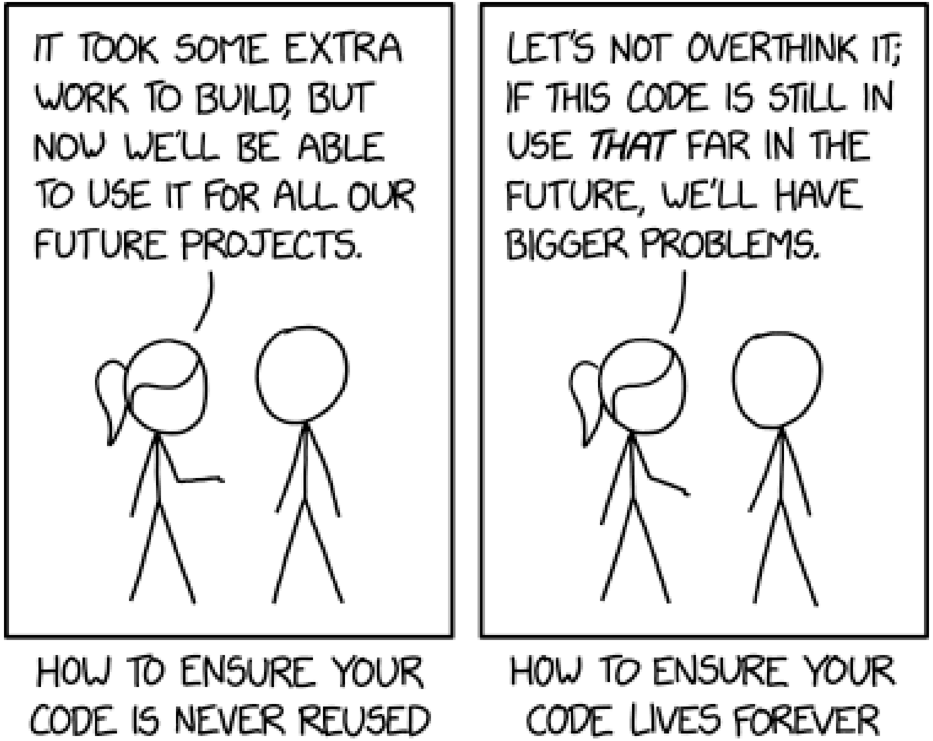
UKRI Software and Skills Survey (2022)

Research software?

www.software.ac.uk

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

From xkcd.com
by Randall Munroe



Software Sustainability Institute

www.software.ac.uk

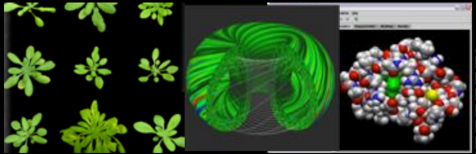
A national facility for cultivating better, more sustainable, research software to enable world-class research

- 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



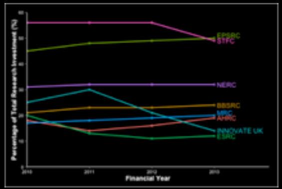
Training

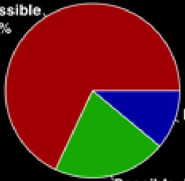
Delivering essential software skills to researchers via CDTs, institutions & doctoral schools

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

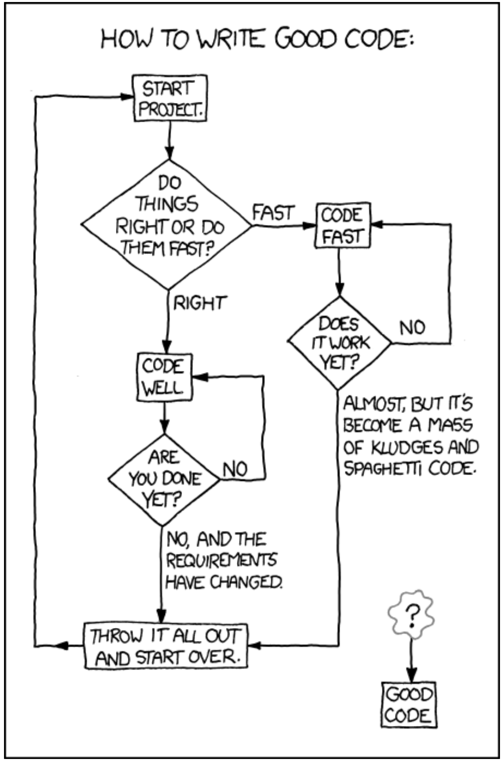
Community

Research Software Tiers

Adapted from Tom Honeyman, ARDC, after Konrad Hinsen

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

Good code takes practice



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

- 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

FAIR Principles

www.software.ac.uk

The Turing Way project illustration by Scriberia.

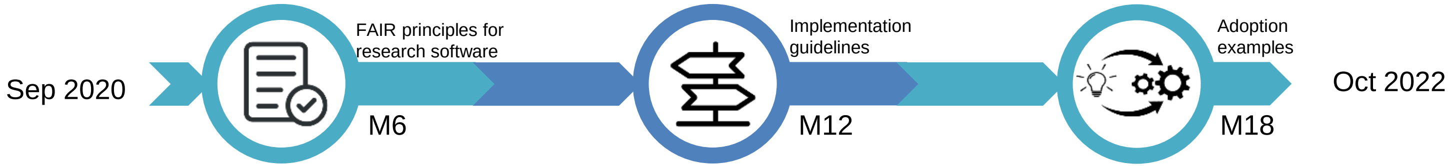
- Findable
- Accessible
- Interoperable
- Reusable

Wilkinson, M., *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3,** 160018 (2016). 10.1038/sdata.2016.18

itute

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)

FAIR4RS Principles

- 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

www.software.ac.uk

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.
R3. Software meets domain-relevant community standards.

Chue Hong, N. P., et al. (2022). FAIR Principles for Research Software version 1.0. (FAIR4RS Principles v1.0). Research Data Alliance. DOI: https://doi.org/10.15497/RDA00068

FAIR4RS Principles

www.software.ac.uk

Findable

Use persistent identifiers and descriptive metadata

Accessible

Make software and metadata easily retrievable

Interoperable

Use APIs, standards and references

Reusable

Document, license, and follow community good practice

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

www.software.ac.uk

Type of license	Closed source / Proprietary	Academic / Non-commercial[2]	Freeware[2]	Copyleft	Permissive
Provides copyright protection	Yes	Yes	Yes	Yes	Yes
Can be used for commercial applications	Yes	Yes	Yes	Yes	Yes
Allows redistribution	No	No	Yes	Yes	Yes
Allows reuse / modification (including in commercial products)	No[1]	No[1]	No[1]	Yes	Yes
Allows reuse in closed source projects	Depends on license	No (normally)[1]	No (normally)[1]	No	Yes
Requires changes to be shared	No (normally)[1]	No (normally)[1]	No (normally)[1]	Yes	No
Ability to restrict categories of users	Yes	Yes	Yes	No	No
Examples of license	Matlab end user license	CASTEP license, OpenCarp license	Adobe Acrobat Reader license	GPL, LGPL, AGPL	BSD, MIT, Apache

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%

Victoria Stodden, "The Scientific Method in Practice: Reproducibility in the Computational Sciences", 2010. DOI:10.2139/ssrn.1550193

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

www.software.ac.uk

	Code repository	Deposit in digital repository	Produce runnable version	Register in catalogue / registry	Paper in software journal	Paper in domain-specific journal
Example	Source code is in GitHub, GitLab or BitBucket with open license	Source code deposited in Zenodo, Figshare or an institutional repository	Jupyter Notebook in Binder, Capsule in CodeOcean, Docker or Singularity container, NextFlow workflow. Package for CRAN, PyPI, etc	Create an entry in a community registries e.g. ASCL (astronomy), CIG (geodynamics), RRID, swMath (mathematics). NLeSC RSD.	Publish software paper in JORS, JOSS, SoftwareX, etc. Publish executable research article in GigaByte	Many journals now accept papers about software – see bit.ly/softwarejournals
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.

zenodo

Search

Upload Communities

Log in Sign up

www.software.ac.uk

All versions

Access Right

- Open (2811911)
- Closed (59138)
- Restricted (11486)
- Embargoed (1311)

File Type

- Pdf (1240608)
- Html (455193)
- Png (422087)
- Jpg (379492)
- Zip (153714)
- Xlsx (37532)
- Txt (33121)
- Docx (29413)
- Csv (28635)
- Xml (19960)

Keywords

Found 97400 results.

< 1 2 3 4 5 6 7 8 9 >

Sort by: Most recent asc.

March 27, 2023 (v0.2.0-SNAPSHOT) Software Open Access

View

nasa-pds-engineering-node/pds-template-repo-java: pds-template-repo-java v0.2.0-SNAPSHOT

PDSEN CI Bot; Jordan Padams; Sean Kelly; Michael Joyce; Alex Dunn; Galen Hollins;

Template for new NASA PDS repositories. For Python software, see https://github.com/nasa-pds/pds-template-repo-python

Uploaded on March 27, 2023

54 more version(s) exist for this record

March 27, 2023 (v2023.03.0) Software Open Access

View

Geoscience Community Analysis Toolkit: GeoCAT-viz

Visualization & Analysis Systems Technologies;

What's Changed Move get_skewt_vars from geocat-comp to geocat-viz by @hCraker in https://github.com/NCAR/geocat-viz/pull/90 Add badges to README by @marodrig in https://github.com/NCAR/geocat-viz/pull/88 version-bump-v2023.03.0 by @hCraker in https://github.com/NCAR/geocat-viz/pull/97 Update badges

Uploaded on March 27, 2023

8 more version(s) exist for this record

March 27, 2023 (0.0.9,5) Software Open Access

View

Nikeshbajaj/spkit: 0.0.9.5

Nikesh Bajaj;

Following main functions are added in 0.0.9.5 version Processing for MEA - Multi-Electrode Array System for Electrophysiology Signal Differentiation, filter_smooth, filtering_pipeline Geometrical funstion Stats Also fixed a bugs wavelet_filtering_win doc string

Uploaded on March 27, 2023

6 more version(s) exist for this record

GitHub → Zenodo

Repositories / Archive a repository /

Referencing and citing content

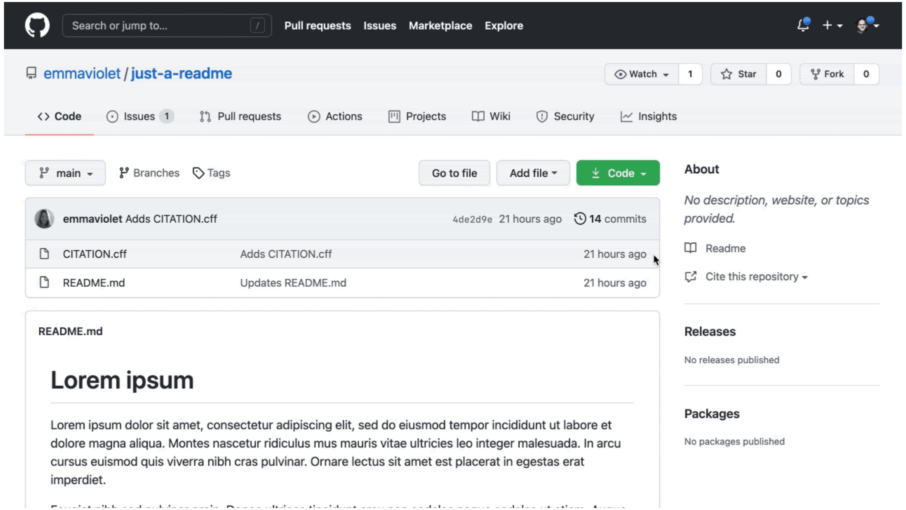
You can use third-party tools to cite and reference content on GitHub.

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.




```
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

www.software.ac.uk

- ❑ 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 Citation Checklist for Developers

www.software.ac.uk

- [ ] 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

www.software.ac.uk

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
- Shoaib Sufi
- Simon Hettrick
- Stephen Crouch
- *Tim Parkinson*
- *Toni Collis*
- *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.

Reusing these slides

www.software.ac.uk