

Software citation today and tomorrow

Daniel S. Katz

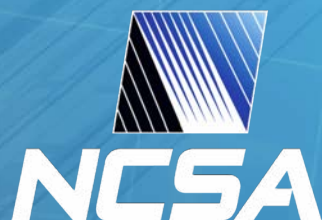
Assistant Director for Scientific Software & Applications, NCSA

Research Associate Professor, CS

Research Associate Professor, ECE

Research Associate Professor, iSchool

dskatz@illinois.edu, d.katz@ieee.org, [@danielskatz](https://twitter.com/danielskatz)



National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign

Software in research

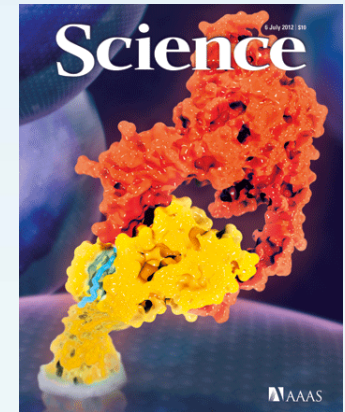
- Claim: software (including services) essential for the bulk of research
- Evidence from surveys
 - UK academics at Russell Group Universities (2014)
 - Members of (US) National Postdoctoral Research Association (2017)
 - My research would not be possible without software: 67% / 63% (UK/US)
 - My research would be possible but harder: 21% / 31%
 - It would make no difference: 10% / 6%

S. Hettrick, "It's impossible to conduct research without software, say 7 out of 10 UK researchers," Software Sustainability Institute, 2014. Available at: <https://www.software.ac.uk/blog/2016-09-12-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers>

S.J. Hettrick, M. Antonioletti, L. Carr, N. Chue Hong, S. Crouch, D. De Roure, et al, "UK Research Software Survey 2014", Zenodo, 2014. doi: 10.5281/zenodo.14809.

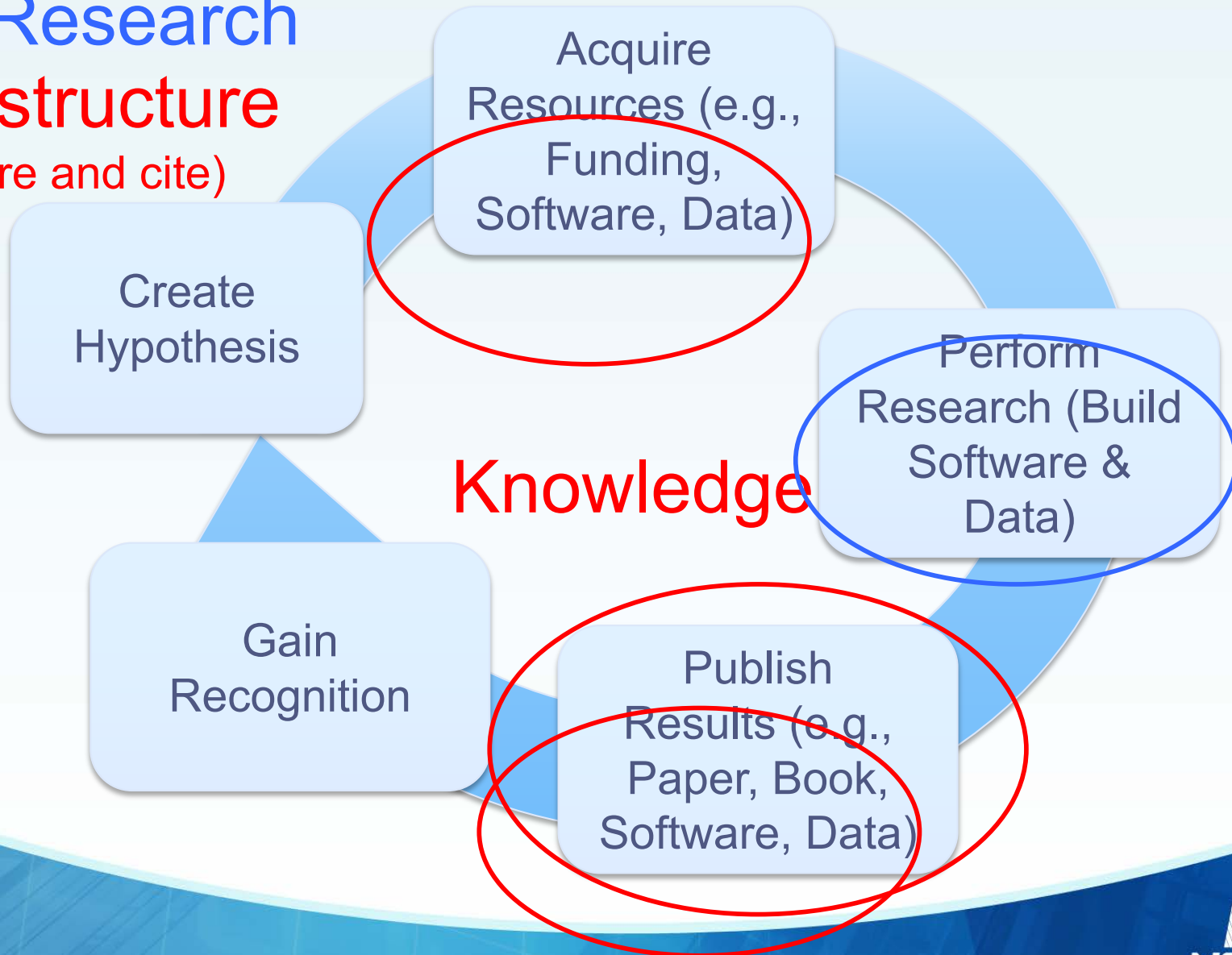
Software in scholarship

- Claim: software (including services) essential for the bulk of research
- Evidence from journals:
 - About half the papers in recent issues of Science were software-intensive projects
 - In Nature Jan–Mar 2017, software mentioned in 32 of 40 research articles
 - Average of 6.5 software packages mentioned per article



Software in research cycle

Research
Infrastructure
(share and cite)



Software Citation Motivation

- Scientific research is becoming:
 - More open – scientists want to collaborate; want/need to share
 - More digital – outputs such as software and data; easier to share
- Significant time spent developing software & data
- Efforts not recognized or rewarded
- Citations for papers systematically collected, metrics built
 - But not for software & data
- Hypothesis:
 - Better measurement of contributions (citations, impact, metrics)
 - > Rewards (incentives)
 - > Career paths, willingness to join communities
 - > More sustainable software

To better measure software contributions

- Citation system was created for papers/books
- We need to either/both
 1. Jam software into current citation system
 2. Rework citation system
 - Focus on 1 as possible; 2 is very hard.
- Challenge: not just how to identify software in a paper
 - **How to identify software used within research process**

Software citation today

- Software and other digital resources currently appear in publications in very inconsistent ways
- Howison: random sample of 90 articles in the biology literature -> 7 different ways that software was mentioned

Mention Type	Count (n=286)	Percentage
Cite to publication	105	37%
Cite to users manual	6	2%
Cite to name or website	15	5%
Instrument-like	53	19%
URL in text	13	5%
In-text name only	90	31%
Not even name	4	1%

- Studies on data and facility citation -> similar results

Software citation principles: people & process

- FORCE11 Software Citation group started July 2015
- WSSSPE3 Credit & Citation working group joined September 2015
- ~55 members (researchers, developers, publishers, repositories, librarians)
- Working on GitHub <https://github.com/force11/force11-scwg> & FORCE11 <https://www.force11.org/group/software-citation-working-group>
- Reviewed existing community practices & developed use cases
- Drafted software citation principles document
 - Started with data citation principles, updated based on software use cases and related work, updated based working group discussions, community feedback and review of draft, workshop at FORCE2016
 - Discussion via GitHub issues, changes tracked
 - Contents: 6 principles, discussion, use cases, ...
- Submitted, reviewed and modified (many times), published
 - Smith AM, Katz DS, Niemeyer KE, FORCE11 Software Citation Working Group.(2016) Software Citation Principles. *PeerJ Computer Science* 2:e86. DOI: [10.7717/peerj-cs.86](https://doi.org/10.7717/peerj-cs.86) and <https://www.force11.org/software-citation-principles>
 - Also includes reviews and responses

Principle 1. Importance

- **Software should be considered a legitimate and citable product of research.** Software citations should be **accorded the same importance** in the scholarly record **as citations of other research products**, such as publications and data; they should be included in the metadata of the citing work, for example in the reference list of a journal article, and should not be omitted or separated. Software should be cited on the same basis as any other research product such as a paper or a book, that is, authors should cite the appropriate set of software products just as they cite the appropriate set of papers.

Principle 2. Credit and attribution

- **Software citations should facilitate giving scholarly credit and normative, legal attribution to all contributors** to the software, recognizing that a single style or mechanism of attribution may not be applicable to all software.

Principle 3. Unique identification

- A software citation should include a method for identification that is machine actionable, globally unique, interoperable, and recognized by at least a community of the corresponding domain experts, and preferably by general public researchers.

Principle 4. Persistence

- **Unique identifiers and metadata describing the software and its disposition should persist – even beyond the lifespan of the software they describe.**

Principle 5. Accessibility

- **Software citations should facilitate access to the software itself and to its associated metadata, documentation, data, and other materials necessary for both humans and machines to make informed use of the referenced software.**

Principle 6. Specificity

- **Software citations should facilitate identification of, and access to, the specific version of software that was used.** Software identification should be as specific as necessary, such as using version numbers, revision numbers, or variants such as platforms.

Discussion: What to cite

- Importance principle: “...authors should cite the appropriate set of software products just as they cite the appropriate set of papers”
- What software to cite decided by author(s) of product, in context of community norms and practices
- POWL: “Do not cite standard office software (e.g. Word, Excel) or programming languages. Provide references only for specialized software.”
- i.e., if using different software could produce different data or results, then the software used should be cited

Discussion: What to cite (citation vs provenance & reproducibility)

- Provenance/reproducibility requirements > citation requirements
- Citation: software important to research outcome
- Provenance: all steps (including software) in research
- Software citation principles cover minimal needs for software citation for software identification
- Provenance & reproducibility may need more metadata

Discussion: Software papers

- Goal: Software should be cited
- Practice: Papers about software (“software papers”) are published and cited
- Importance principle (1) and other discussion: **The software itself should be cited on the same basis as any other research product; authors should cite the appropriate set of software products**
- Ok to cite software paper too, if it contains results (performance, validation, etc.) that are important to the work
- If the software authors ask users to cite software paper, can do so, *in addition to citing to the software*

Discussion: Derived software

- Imagine Code A is derived from Code B, and a paper uses and cites Code A
- Should the paper also cite Code B?
- No, any research builds on other research
- Each research product just cites those products that it directly builds on
- Together, this give credit and knowledge chains
- Science historians study these chains
- More automated analyses may also develop, such as transitive credit

Discussion: Software peer review

- Important issue for software in science
- Probably out-of-scope in citation discussion
- Goal of software citation is to identify software that has been used in a scholarly product
 - Whether or not that software has been peer-reviewed is irrelevant
- Possible exception: if peer-review status of software is part of software metadata
- Working group opinion: not part of the minimal metadata needed to identify the software

Discussion: Citations in text

- Each publisher/publication has a style it prefers
 - e.g., AMS, APA, Chicago, MLA
- Examples for software using these styles published by Lipson
- Citations typically sent to publishers as text formatted in that citation style, not as structured metadata
- Recommendation: **text citation styles should support:**
 - a) a label indicating that this is software, e.g. [Computer program]
 - b) support for version information, e.g. Version 1.8.7

Discussion: Citation limits

- Software citation principles
- → more software citations in scholarly products
- → more overall citations
- Some journals have strict limits on
 - Number of citations
 - Number of pages (including references)
- Recommendations to publishers:
 - **Add specific instructions regarding software citations to author guidelines to not disincentivize software citation**
 - **Don't include references in content counted against page limits**

Discussion: Unique identification

- **Recommend DOIs for identification of published software**
- However, identifier can point to
 1. a specific version of a piece of software
 2. the piece of software (all versions of the software)
 3. the latest version of a piece of software
- One piece of software may have identifiers of all 3 types
- And maybe 1+ software papers, each with identifiers
- Use cases:
 - Cite a specific version
 - Cite the software in general
 - Link multiple releases together, to understand all citations

Discussion: Unique identification (cont.)

- Principles intended to apply at all levels
- To all identifiers types, e.g., DOIs, RRIDs, ARKS, etc.
- Though again: **recommend when possible use DOIs that identify specific versions of source code**

Discussion: Types of software

- Principles and discussion generally focus on software as source code
- But some software is only available as an executable, a container, or a service
- Principles intended to apply to all these forms of software
- Implementation of principles will differ by software type
- **When software exists as both source code and another type, cite the source code**

Discussion: Access to software

- Accessibility principle: “software citations should permit and facilitate access to the software itself”
- Metadata should provide access information
- Free software: metadata includes UID that resolves to URL to specific version of software
- Commercial software: metadata provides information on how to access the specific software
 - E.g., company’s product number, URL to buy the software
- If software isn’t available now, it still should be cited along with information about how it was accessed
- Metadata should persist, even when software doesn’t

Discussion: Identifier resolves to ...

- Identifier that points directly to software (e.g., GitHub repo) satisfies Unique Identification (3), Accessibility (5), and Specificity (6), but not Persistence (4)
- Recommend that **identifier should resolve to persistent landing page that contains metadata and link to the software itself, rather than directly to source code**
- Ensures longevity of software metadata, even beyond software lifespan
- Point to figshare, Zenodo, etc., not GitHub

Example 1: Make your software citable

- Publish it – if it's on GitHub, follow steps in <https://guides.github.com/activities/citable-code/>
- Otherwise, submit it to zenodo or figshare, with appropriate metadata (including authors, title, ..., citations of ... & software that you use)
- Get a DOI
- Create a CITATION file, update your README, tell people how to cite
- Also, can write a software paper and ask people to cite that (but this is secondary, just since our current system doesn't work well)

Example 2: Cite someone else's software

- Check for a CITATION file or README; if this says how to cite the software itself, do that
- If not, do your best following the principles
 - Try to include all contributors to the software (maybe by just naming the project)
 - Try to include a method for identification that is machine actionable, globally unique, interoperable – perhaps a URL to a release, a company product number
 - If there's a landing page that includes metadata, point to that, not directly to the software (e.g. the GitHub repo URL)
 - Include specific version/release information
- If there's a software paper, can cite this too, but not in place of citing the software

Problem: software citation vs paper citation

- Three relevant steps for paper citation
 1. Creator (aka author) submits paper to “publisher”
 2. [review+], then publisher publishes paper & assigns identifier, often DOI
 3. To refer to paper within another work, cite paper metadata, often including DOI
- Fixed order, discrete steps
- For software today
 - Creator develops software on GitHub, released at different stages (versions) during its development
 - Someone who uses that software will likely not cite it, but if they do, they will cite the repository
 - No step 2
 - Partial step 3, because there is no clear metadata or identifier for the software that was used
- Software citation principles inserts step 2

Software citation vs paper citation (cont.)

- Software citation principles guidance may not work
 - Adds a step to the software developer's workflow
 - They may not care enough to implement it
- Even if we do get to a future time in which developers routinely published their software releases, what happens until then, or for existing software?
- Real problem:
 - Steps (create, publish, cite) don't match how open source is developed and used
 - Software is more fine-grained and iterative
 - Open source development mostly occurs in the open
 - No natural need for publish step, other than marketing and credit, which are not primary concerns in all projects

Software citation vs paper citation (cont.)

- Back to papers: what happens if the citer wants to refer to something that has not been published?
 - Students initially taught to avoid this situation, later taught to cite as “personal communication”
 - APA Publication Manual distinguishes between recoverable and unrecoverable data.
 - Recoverable data (that which can be accessed by the reader via the citation information) should be cited as a formal citation
 - Unrecoverable data should be referred to within the text as “(*author*, personal communication, *date*)”
- This distinction between recoverable (published & available) and unrecoverable (not published & not available) doesn’t work for software
- All versions of software on GitHub, even if never published, are recoverable by default
 - Unless project is deleted from GitHub; could still be recovered from a local copy
- Regarding credit, Software Citation Principles paper: “It is not that academic software needs a separate credit system from that of academic papers, but that the need for credit for research software underscores the need to overhaul the system of credit for all research products.”
- The fact that the three-step model of distinct creator, publisher, and citer doesn’t really fit modern open source practices is another argument for that overhaul



Software Heritage **as a partial solution?**

- Mission: to collect, preserve, and share **all** software that is publicly available in source code form
- Source code stored in a Merkle tree, with hashed pointers
- To cite unpublished software, need to find hash to version used
 - Tool to go from GitHub commit to SH hash could be developed
- Need ID for SH hash
 - Maybe <https://softwareheritage/id>
- What about metadata?

Software metadata

- Two types of metadata:
 - *Software creation metadata*: describe properties of software itself as source code, such as: authors, language, license, version number, location, etc.
 - *Software usage metadata*: describe how the code is being used, possibly including how it is built, such as: compiler version, operating system, parallel computing platform, command-line options, etc.
- Software Citation Principles
 - Software creation metadata needed for citation; software usage metadata needed for provenance and reproducibility
 - Metadata for provenance > metadata for citation
- For code, software user can determine software usage metadata, but not software creation metadata
- Software Heritage cannot provide the metadata needed for software citation
- Software creation metadata can only be determined by software creators

Software creation metadata & CodeMeta

- If authors of software want to be cited, they can fill this gap fairly easily: just need to create single metadata file in repo root
- As suggested by Martin Fenner, based on work done in CodeMeta (which has goal to create minimal metadata schema for science software and code, in JSON and XML)
 - Example: codemeta.json in <https://github.com/datacite/maremma> repo
 - DataCite can generate DOI and citation from this
- Code developers create codemeta.json in their repo when they start a project, then keep it up to date, like README or CONTRIBUTORS
- Software Heritage keeps all versions of codemeta.json with corresponding versions of software code
- Can use this to retrospectively build proper citation metadata for any version of the software

But this is not quite enough

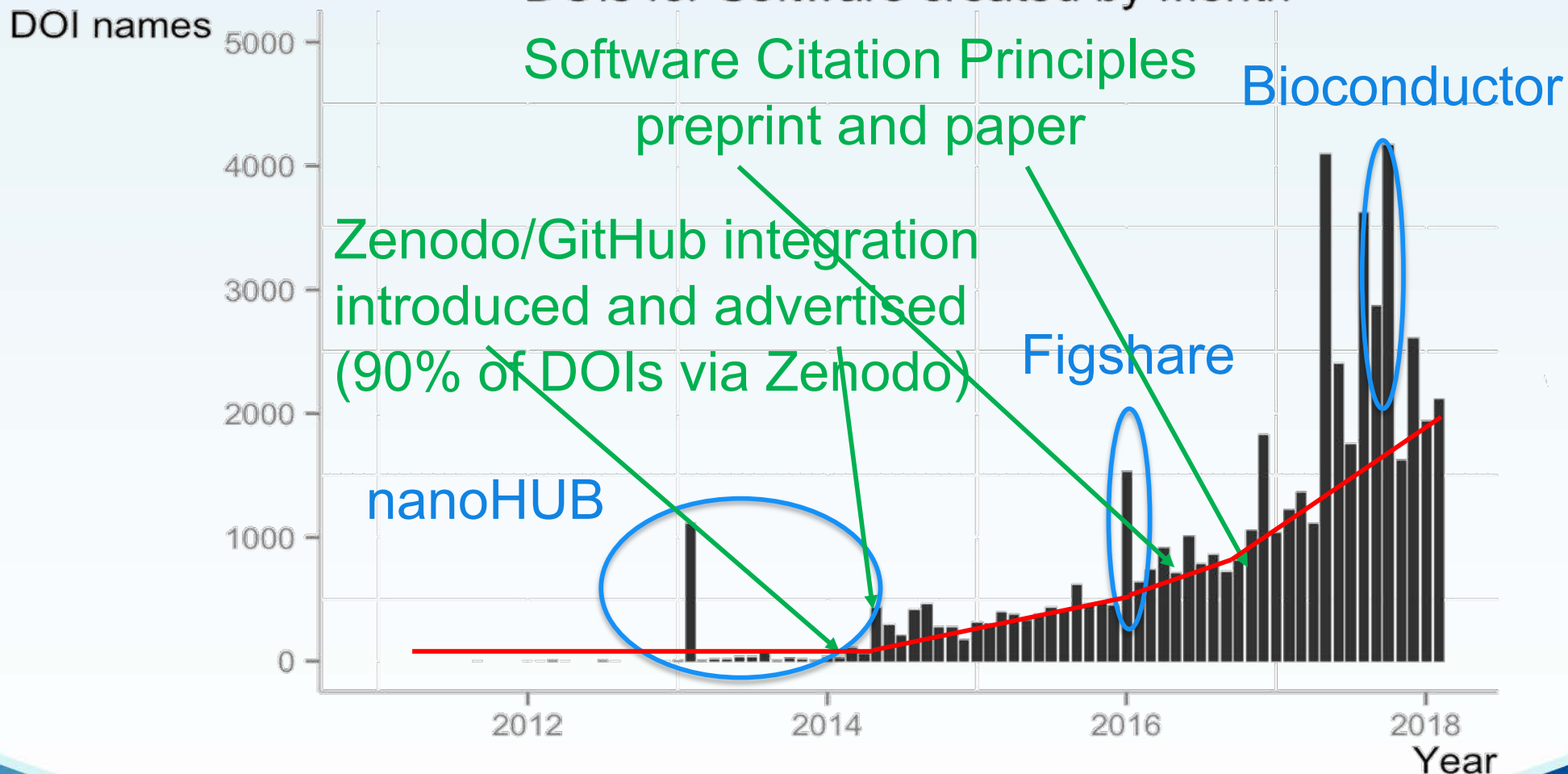
- How to build a software citation when authors don't care about credit or have not provided codemeta.json file?
- And, how to cite older software?
- Most software creation metadata can be extracted from a repo directly
- Except authors
- Authors \neq contributors to repo
- Some project contributors do not contribute to the repo
- Some contributors to the repo should not be authors
- Best thing: identify the project as the authors
 - e.g., authors = "CodeMeta Project"

Compact identifiers to the rescue?

- A method for referring to a stored object in a registered repository
 - `pdb:2gc4` is compact identifier for Protein Data Bank accession (item) number 2gc4
 - Compact identifier metaresolvers (identifiers.org & N2T.net) can resolve a compact identifier, for example: <http://identifiers.org/pdb:2gc4>.
- Compact identifier metaresolvers could work with Software Heritage to define compact identifiers for all the repositories that Software Heritage works with,
- And a means to resolve them to a pointer in the Software Heritage archive. For example, `github:repo_name/commithash`
- With Software Heritage, CodeMeta, and compact identifiers working together, users could easily and effectively cite any archived software

Software DOIs registered at DataCite

DOIs for Software created by Month



Journal of Open Source Software (JOSS)

- In the meantime, there's JOSS
- A developer friendly journal for research software packages
- “If you've already licensed your code and have good documentation then we expect that it should take **less than an hour** to prepare and submit your paper to JOSS”
- Everything is open:
 - Submitted/published paper: <http://joss.theoj.org>
 - Code itself: where is up to the author(s)
 - Reviews & process: <https://github.com/openjournals/joss-reviews>
 - Code for the journal itself: <https://github.com/openjournals/joss>
- JOSS papers are issued Crossref DOIs
- First paper submitted 4 May 2016
 - 31 May 2017: 111 accepted papers, 41 under review, ~15 pre-review
 - 5 Apr 2018: 258 accepted papers, 42 under review, ~15 pre-review

Working group status

- Principles document published in PeerJ CS
- Software Citation Working Group ended
- Software Citation Implementation group now in progress
 - Goal is implementing software citation
 - Working with institutions, publishers, funders, researchers, etc.
 - Planning subgroups in to study use cases with different roles
 - Find and advertise gaps
 - Suggest and try solutions
 - Some discussion of joining with Research Data Alliance (RDA) sw group
 - Want to join? Sign up on FORCE11 group page
 - <https://www.force11.org/group/software-citation-implementation-working-group>
- Examples ...

DataCite Schema Extensions

- DataCite Metadata Schema v4.1
 - <http://schema.datacite.org/meta/kernel-4.1/>
 - <https://blog.datacite.org/metadata-schema-4-1/>
- Support for software citation
 - Facilitate software discovery, sharing and citation
 - Two new relation types
 - (HasVersion, IsVersionOf) – relate versions of the same software e.g. relate an un-versioned code repository to one of its specific software versions
 - (IsRequiredBy, Requires) – software dependencies
 - Additions and modifications to documentation to clarify use for software citation

Community-specific implementation

- High Energy Physics
 - Work with conferences (ACAT & CHEP) so that proceedings authors cite software; software developers make their software citable
 - https://indico.cern.ch/event/567550/papers/2696191/files/6046-Katz_Citation.pdf
 - Could generalize for other conferences, and other communities
- Astronomy: handling software citations between publisher (AAS), indexer (ADS), software repository (Zenodo)
 - Trying to implement this with ADS in the next couple of months
 - Model is generic
- Geosciences (AGU): related effort focused on data; also starting to consider software
 - Enabling FAIR Data: <https://osf.io/jy4d9/>

Better Scientific Software (BSSw)



<https://bssw.io>
So your code will see the future.

- BSSw is supporting my work via a fellowship
- My goal: make scientific software more sustainable by providing credit to its developers via software citation
- I'm working with disciplinary communities
- More information
 - <https://danielskatzblog.wordpress.com/2018/02/08/better-scientific-software-fellowship/>

Using codemeta.json

- CaltechDATA software preservation
 - GitHub software repositories automatically preserved, with DOI
 - Extended to use codemeta.json file to incorporate full author list, keywords and license metadata in CaltechDATA landing page and associated with DOI
 - <https://www.library.caltech.edu/news/enhanced-software-preservation-now-available-caltechdata>
- DataCite support in new DOI registration web service
 - DOI for repo, not for specific version
 - Provide GitHub URL to service to register DOI with metadata automatically populated
 - If there's a codemeta.json file in root of repo
 - Will also give error if required metadata missing
- Need to coordinate with Citation File Format
 - “Human and machine readable citations”
 - <https://citation-file-format.github.io/>

Conclusions

- Software
 - Important today, essential tomorrow
- Credit
 - A known problem for papers, worse for software
- Citation
 - We know what to do (mostly), now need to do it
 - Explained in <https://cite.research-software.org/>
 - Need to build support for use codemeta.json in repos
 - Need to build up tools around Software Heritage
- What you can do
 - Cite the software you use, make it easy for others to cite the software you write (and see the “I solemnly pledge” manifesto – http://ceur-ws.org/Vol-1686/WSSSPE4_paper_15.pdf)
 - Organize/edit/review taking software citation into account

Credits

- Thanks to Arfon Smith and Kyle Neimeyer for co-leadership in FORCE11 Software Citation WG
- And to Neil Chue Hong & Martin Fenner for co-leadership in FORCE11 Software Citation Implementation WG
- And to the BSSw project (<http://bssw.io>) for the fellowship to pursue some parts of this
- More of my thinking
 - Blog: <http://danielskatzblog.wordpress.com>
 - Tweets: @danielskatz