Software Citation in HEP: Current State and Recommendations for the Future

Matthew Feickert
(University of Wisconsin-Madison)
matthew.feickert@cern.ch
Daniel Katz, Mark Neubauer, Elizabeth Sexton-Kennedy, Graeme Stewart

International Conference on Computing in High Energy and Nuclear Physics (CHEP) 2023

May 8th, 2023
Co-conveners

Daniel Katz
University of Illinois at Urbana-Champaign/NCSA

Mark Neubauer
University of Illinois at Urbana-Champaign

Elizabeth Sexton-Kennedy
FNAL

Graeme Stewart
CERN
Software Citation and Recognition Workshop

- 2022 HSF/IRIS-HEP Blueprint Process Workshop

This meeting aims to provide a community discussion around ways in which HEP experiments handle citation of software and recognition for software efforts that enable physics results disseminated to the public.

- Had representation from:
  - **Experiments**: ATLAS, CMS, LHCb
  - **Software project communities**: ROOT Team, Scikit-HEP, MCnent, IRIS-HEP
  - **Publishers**: INSPIRE, Elsevier, Journal of Open Source Software (JOSS)

This work was supported by the National Science Foundation under Cooperative Agreement OAC-1836650.
Principles of Software Citation

As established by FORCE11 Software Citation working group (2016, DOI 10.7717/peerj-cs.86)

1. Importance
2. Credit and Attribution
3. Unique Identification
4. Persistence
5. Accessibility
6. Specificity

Software Citation Principles

image credit: Data Cite
Currently have (in 2023)

- Software citation principles
- Policies from publishers
- Modern open source tooling
- Beginning of movement among developers, paper authors, journal reviewers and editors

Software Citation Principles

image credit: Data Cite
Current State of Software Citation in HEP: ATLAS & CMS

ATLAS

- Use a "catch-all" citation for ATLAS software tools
- For statistical analysis and ML generally cite the papers for the methods, but not the tools and software
  - Have seen some changes when the tools explicitly ask to be cited

CMS

- Endorses large CMS software projects having peer reviewed papers that would be cited in physics papers
- Have expressed positive views on additional papers being written and published
Current State of Software Citation in HEP: LHCb

- Following recommendations of Daniel Katz's CHEP 2018 presentation
- Most papers aim to cite all high-level software used in the analysis
  - Cite the software, and if there's a paper for the software cite that too
- Analysts are still adopting this habit and need reminding
  - Though open to idea: "We should cite software more"

<table>
<thead>
<tr>
<th>Description</th>
<th>Ref.</th>
<th>cite code</th>
</tr>
</thead>
<tbody>
<tr>
<td>PYTHIA</td>
<td>[Sjostrand:2007gs]</td>
<td></td>
</tr>
<tr>
<td>EVTGEN</td>
<td>[Lange:2001uf]</td>
<td></td>
</tr>
<tr>
<td>PHOTOS</td>
<td>[davidson/2015 photons]</td>
<td></td>
</tr>
<tr>
<td>GEANT4</td>
<td>[Allison:2006ve,*Agostinelli:2002hh]</td>
<td></td>
</tr>
<tr>
<td>LHCb simulation</td>
<td>[LHCb-PROC-2011-006]</td>
<td></td>
</tr>
<tr>
<td>RapidSim</td>
<td>[Cowan:2014mrr]</td>
<td></td>
</tr>
<tr>
<td>DIRAC</td>
<td>[Tsangorodtsev:2010zz,*BelleDIRAC]</td>
<td></td>
</tr>
<tr>
<td>sPlot</td>
<td>[Pivk:2004y]</td>
<td></td>
</tr>
<tr>
<td>sFit</td>
<td>[Xie:2009yka]</td>
<td></td>
</tr>
<tr>
<td>BDT</td>
<td>[Breiman]</td>
<td></td>
</tr>
<tr>
<td>BDT training</td>
<td>[AdaBoost]</td>
<td></td>
</tr>
<tr>
<td>TMVA</td>
<td>[Hocker:2007ht,*TMVA4]</td>
<td></td>
</tr>
<tr>
<td>RooUnfold</td>
<td>[Adye:2011gm]</td>
<td></td>
</tr>
<tr>
<td>scikit-learn</td>
<td>[Scikit-learn-paper]</td>
<td></td>
</tr>
<tr>
<td>LAURA++</td>
<td>[Bash:2017aq]</td>
<td></td>
</tr>
<tr>
<td>hep_ml</td>
<td>[Ragozhnikov:2018lde]</td>
<td></td>
</tr>
<tr>
<td>root_numpy</td>
<td>[root-numpy]</td>
<td></td>
</tr>
<tr>
<td>GammaCombo</td>
<td>[GammaCombo]</td>
<td></td>
</tr>
<tr>
<td>TENSORFLOW</td>
<td>[tensorflow2015-whitepaper]</td>
<td></td>
</tr>
<tr>
<td>ROOT</td>
<td>[Brun, Rademakers]</td>
<td></td>
</tr>
<tr>
<td>RooFit</td>
<td>[Verkerke, Kirkby]</td>
<td></td>
</tr>
<tr>
<td>scikit-hep</td>
<td>[Scikit-HEP]</td>
<td></td>
</tr>
</tbody>
</table>
Current State of Software Citation in HEP: Software Projects

• Community views vary widely
  ◦ **ROOT team**: Explicitly not interested in software citation *for ROOT* (view is too little impact)
    ▪ "ROOT's opinion likely cannot be extrapolated"
  ◦ **Scikit-HEP**: Adopting software citation recommendations from broader open source world community norms — interested in more citations
  ◦ **MCnet**: Find Monte Carlo generators are broadly well-cited (point to LHC experiments regular citations) — current system working well

• Agreement on importance of technical solutions
  ◦ Programmatic discovery of citations important
Recommendations from Journals and Publishers

- **INSPIRE**
  - Currently handles software **papers**, but has plans to add support for **Data and Software**
  - Citations would be tracked and counted by DOI

- **Elsevier**
  - Community needs to reach consensus on how to cite software, and share outcome with publishers (won't take lead)
  - Publishers can better instruct editors and referees what publishers expect from them

- **Journal of Open Source Software**
  - In addition to incentivizing high quality software, JOSS can help bridge the gap
  - "recognize that for most researchers, papers and not software are the currency of academic research"
Software in the field might advertise citation/copyright information with runtime banners

Conventions were not firmly established in the broader scientific computing community

Interrupting user logs is now avoided given modern tooling. Consider using APIs.

# CLI API
$ mytool --citation
$ mytool --cite

# Python API
import mytool
mytool.utils.citation()
Adopt the Citation File Format as a common standard and add a CITATION.cff to project repository

- Human- and machine-readable file format in YAML
- Has well defined, versioned schema
- Convertible to other citation formats (BibTeX, CodeMeta, EndNote, RIS, schema.org, Zenodo, APA)

Supported by GitHub, Zenodo, and Zotero!

**Web tool initializer** for easily creating first CITATION.cff

**Tooling for validation**

```bash
$ python -m pip install cffconvert
$ cffconvert --validate
```

Citation metadata are valid according to schema version 1.2.0.
Recommendations: Zenodo

Versioned archive of everything: code, documents, data products, data sets
(See Lars Holm Nielsen's CHEP 2023 talk)

Why use Zenodo?

- **Safe** — your research is stored safely for the future in CERN’s Data Centre for as long as CERN exists.
- **Trusted** — built and operated by CERN and OpenAIRE to ensure that everyone can join in Open Science.
- **Citeable** — every upload is assigned a Digital Object Identifier (DOI), to make them citable and trackable.
- **No waiting time** — Uploads are made available online as soon as you hit publish, and your DOI is registered within seconds.
- **Open or closed** — Share e.g. anonymized clinical trial data with only medical professionals via our restricted access mode.
- **Versioning** — Easily update your dataset with our versioning feature.
- **GitHub integration** — Easily preserve your GitHub repository in Zenodo.
- **Usage statistics** — All uploads display standards compliant usage statistics.
The easiest, but least robust way: If you have a particular citation that you want people to use, put it **everywhere**

- Version control repository README
- Online software documentation (landing page, how to cite page)
- Package distribution websites (e.g. PyPI)

- **Have single source of truth** for citations: version control repository that all other sources derive from.

- **Make your citation preferences clear** to the world and SEO. Do not rely on people emailing to ask (they shouldn't have to).

---

**Use and Citations**

Citation

The preferred BibTeX entry for citation of `pyhf` includes both the Zenodo archive and the JOSS paper:

```bibtex
@software{pyhf,  
  author = {Lukas Heinrich and Matthew Fiebichert and Giordon Stark},  
  title = "pyhf: v0.7.1",  
  version = {v0.7.1},  
  doi = {10.5281/zenodo.1169739},  
  url = {https://doi.org/10.5281/zenodo.1169739},  
  note = [https://github.com/scikit-hep/pyhf/releases/tag/v0.7.1]  
}
@article{pyhf_joss,  
  doi = {10.21105/joss.02823},  
  url = {https://doi.org/10.21105/joss.02823},  
  year = {2021},  
  publisher = {The Open Journal},  
  volume = {6},  
  number = {38},  
  pages = {2021},  
  author = {Lukas Heinrich and Matthew Fiebichert and Giordon Stark and Kyle Cranmer},  
  title = "pyhf: pure-Python Implementation of HistFactory statistical models".  
  journal = {Journal of Open Source Software}  
}
```

pyhf's "Use and Citations" page in documentation
Revisiting Software Citation Principles in HEP

- **Importance:** As a field HEP understands software is important, but improvements could be made on views towards research products.

- **Credit and Attribution:** Improving in HEP, but can leverage software friendly journals (i.e., JOSS) to help this.

- **Unique Identification:** Zenodo DOIs are common to HEP. CITATION.cff files can help as well.

- **Persistence:** Long term archival through Zenodo is common practice.

- **Accessibility:** HEP is becoming more **FAIR** focused. CITATION.cff provides common framework for metadata. Further support of INSPIRE by field could provide strong database access.

- **Specificity:** Include version numbers in CITATION.cff.
Summary

- Software citation is an **ongoing process** that is not straightforward (for any scientific field)

- Differing community **processes and standards** exist in HEP (we've never been homogeneous)

- Agreement that more citation is probably useful and **programatic discovery** of citations is important

- With modern tools and standards have the opportunity to **expand and standardize**

- Final summary paper from workshop forthcoming
Backup
CITATION.cff: Which DOIs?

Personal choice, but given specific release DOIs are generated *after* the release is made to have the versioned and distributed CITATION.cff be correct at release would recommend the Zenodo project level ("cite all versions") DOI.

See valid CITATION.cff in talk repository (examples/CITATION.cff):

cff-version: 1.2.0
message: "Please cite the following works when using this software.

type: software
authors:
  - family-names: "Feickert"
    given-names: "Matthew"
    orcid: "https://orcid.org/0000-0003-4124-7862"
    affiliation: "University of Wisconsin-Madison"
title: "mylibrary: v1.2.3"
version: 1.2.3
# This is the _project_ DOI ("cite all versions" DOI on Zenodo page)
doi: 10.5281/zenodo.1234567
repository-code: "https://github.com/myorg/mylibrary/releases/tag/v1.2.3"
url: "https://mylibrary.readthedocs.io/en/v1.2.3/"
keywords:
  - example
  - software

CITATION.cff: How to keep up to date?

- As plain text, very easy to update version information when cutting a release
- Can use tool control of version update to make it easier
  - Example: `tbump`
  - `$ tbump <version target>`
- Also possible to have automated version bump workflows using continuous integration
- (Jumping ahead a slide) What about the Zenodo DOI?
  - For simplicity, use the project level DOI and not the version level DOI

```yaml
cff-version: 1.2.0
message: "Please cite the following works when using this software."
type: software
... 
title: "mylibrary: v1.2.3"
version: 1.2.3
doi: 10.5281/zenodo.1123456
repository-code: "https://github.com/myorg/mylibrary/releases/tag/v1.2.3"
url: "https://mylibrary.readthedocs.io/en/v1.2.3/"
```
Zenodo: DOI minting made easy

- Everything on Zenodo has a DOI
  - Provides both a **project** DOI (resolves to latest) and **version specific** DOI
- Enable it to **automatically preserve work from GitHub** (can also directly upload, but lose out on automation)
  - Benefit from having a DOI for **every version** regardless of software paper landscape state
- Once you have a DOI, put it **everywhere** (again)
  - Recommend sharing the project DOI and letting users select a specific version if they want it

---

**GitHub Repositories**

Get started

1. **Flip the switch**
   
   Select the repository you want to preserve, and toggle the switch below to turn on automatic preservation of your software.

2. **Create a release**
   
   Go to GitHub and create a release. Zenodo will automatically download a .zip ball of each new release and register a DOI.

3. **Get the badge**
   
   After your first release, a DOI badge that you can include in GitHub README will appear next to your repository.

[Example DOI: 10.5281/zenodo.8475]
Zenodo + CITATION.cff

CITATION.cff used by Zenodo importer to fully define Zenodo archive metadata
The end.