Software in science and engineering research: the case for open source and proper citation

Kyle E Niemeyer
School of Mechanical, Industrial, & Manufacturing Engineering

Research Computing Seminar – 8 May 2017

Kyle.Niemeyer@oregonstate.edu  @kyleniemeyer  https://git.io/nrg
Topics

• Need for software citation

• Principles of software citation

• Venues for sharing and publication

• Best practices for reproducibility
Kyle Niemeyer
kyleniemeyer

Assistant Professor of mechanical engineering. Leads
@Niemeyer-Research-Group

Oregon State University
Corvallis, OR
kyle.niemeyer@gmail.com
http://kyleniemeyer.com

Organizations

Overview
Repositories 61 Stars 141
Followers 23 Following 15

Pinned repositories

- **SLACKHA/pyJac**
  Creates C and CUDA analytical Jacobians for chemical kinetics ODE systems
  Python ★ 8 ✔ 7

- **PyTeCK**
  Automatically test chemical kinetic models using experimental data
  Python ★ 2

- **irrev_mech**
  Python utility that converts Chemkin-format chemical reaction mechanism with reversible reactions into one with only irreversible reactions
  Python ★ 5 ✔ 2

- **WSSSPE/meetings**
  Products from Working towards Sustainable Software for Science: Practice and Experiences (WSSSPE) activities
  TeX ★ 15 ✔ 34

- **pr-omethe-us/PyKED**
  Python interface to the ChemKED database format
  Python ★ 1 ✔ 4

- **Niemeyer-Research-Group/pyMARS**
  Python-based (chemical kinetic) Model Automatic Reduction Software
  ✔ 2

1,596 contributions in the last year

Learn how we count contributions.
Modern science and engineering research depends on software.

2009 survey: 91% of scientists consider software “important” or “very important” to research\(^1\).


But, 40–70% of software used is not cited\(^2,3\).


Citing software & data is important.

Using different versions of software and data changes our answers.

Without proper citations, your work is not 
reproducible.

Also, academia relies on citations for 
credit. (for better or worse)
Software citation principles

Arfon M. Smith1,*, Daniel S. Katz2,*, Kyle E. Niemeyer3,* and FORCE11 Software Citation Working Group

1 GitHub, Inc., San Francisco, California, United States
2 National Center for Supercomputing Applications & Electrical and Computer Engineering Department & School of Information Sciences, University of Illinois at Urbana-Champaign, Urbana, Illinois, United States
3 School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, Oregon, United States
* These authors contributed equally to this work.

ABSTRACT

Software is a critical part of modern research and yet there is little support across the scholarly ecosystem for its acknowledgement and citation. Inspired by the activities of the FORCE11 working group focused on data citation, this document summarizes the recommendations of the FORCE11 Software Citation Working Group and its activities between June 2015 and April 2016. Based on a review of existing community practices, the goal of the working group was to produce a consolidated set of citation principles that may encourage broad adoption of a consistent policy for software citation across disciplines and venues. Our work is presented here as a set of software citation principles, a discussion of the motivations for developing the principles, reviews of existing community practice, and a discussion of the requirements these principles would place upon different stakeholders. Working examples and possible technical solutions for how these principles can be implemented will be discussed in a separate paper.
Software is a critical part of modern research... yet there is little support for its acknowledgement and citation.

**SOFTWARE CITATION PRINCIPLES**

**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; they should be included in the metadata of the citing work, such as a reference list. Software should be cited on the same basis as any other research product such as a paper or a book.

**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.

**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.

**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.

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

**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.
Principles

1. **Importance:** software as important as other research products

2. **Credit & attribution:** citations should facilitate scholarly credit and attribution to all contributors

3. **Unique identification:** citation should include machine actionable, globally unique, interoperable, and recognized identification method

4. **Persistence:** Unique identifiers and metadata should persist

5. **Accessibility:** Citations should facilitate access to software and associated metadata

6. **Specificity:** Citations should facilitate identification of, and access to, specific version of software used
When to cite?

Cite software/data if used directly and important to research results.
How to cite?

Name/description

Author(s)/developer(s)

DOI or other unique/persistent identifier

Location (e.g., GitHub repo)

(If there’s a paper describing it, cite that too)


Where to cite?

In the text with the references/bibliography.

The next three sections explain the implementation of each primary module. PyTeCK also includes the module detect_peaks, based on the work of Duarte [Duarte2015], for detecting peaks in targeted quantities (e.g., pressure, temperature) to determine the ignition delay time. Supporting modules in PyTeCK include exceptions for raising exceptions while reading YAML files, utils that initializes a single Pint-based unit registry [Grecco2016], and validation that provides quantity validation functions.


The Python [59] package pyJac implements the methodology for producing analytical Jacobian matrices described in the previous sections, which we released openly online [60] under the MIT license. pyJac requires the Python module NumPy [61]. The modules used to test the correctness and performance of pyJac are included in this release, and additionally require Cython [62], Cantera [54], PyYAML [63], and Adept [64]; however, these are not required for Jacobian/rate subroutine generation. In addition, interpreting Cantera-format models [54] requires installing the Cantera module for any purpose, while pyJac includes native support for interpreting Chemkin-format models [53].


Sharing software (and data) openly has clear benefits to others and yourself

A paper that isn’t accompanied by the software or data produced is just advertising.¹

People find reproducible results more trustworthy...

...and cite you more!²

Reduce duplicated effort and increase impact.


Kyle Niemeyer
Oregon State University Assistant Professor

Open Access  Top 10%
98% of your research is free to read online. This level of availability puts you in the top 2% of researchers.

Global Reach  Top 25%
Your research has been saved and shared in 39 countries. That’s high: only 14% of researchers get that much international attention.

Hot Streak  Top 10%
People keep talking about your research. Someone has shared your research online every month for the last 15 months. That’s a sharing streak matched by only 2% of scholars.

Timeline
- 595 Online mentions over 7 years
- 585 5 4 1

Publications
- Software citation principles
  2016 PeerJ Computer Science
  200

- The Journal of Open Source Software
  2017 Figshare
  97

- Software Citation Principles
  2016 PeerJ Preprints
  70
Sharing software (and data) openly is now really easy

Software/scripts: GitHub, GitLab
Data: Zenodo, Figshare
Papers: arXiv.org, engrxiv, Open ACCESS, PeerJ Preprints
Example: pyJac

Creates C and CUDA analytical Jacobian matrices for chemical kinetics ODE systems

- Source code at github.com/SLACKHA/pyJac has README with basic usage
- Full documentation website with API docs, installation guide, and examples: slackha.github.io/pyJac/
- Functional and performance testing suites built-in
- Software paper published with full theory details (doi.org/10.1016/j.cpc.2017.02.004)
- Full source of paper also available via niemeyer-research-group.github.io/pyJac-paper/
- Data, figures, and figure scripts from paper available openly via Figshare: https://doi.org/10.6084/m9.figshare.4578010
JOSS: Journal of Open Source Software

- “Developer-friendly journal for research software packages”
- Affiliate of Open Source Initiative
- Open access, no fees

“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.”
Make software available in repository with OSI-approved license:
https://opensource.org/licenses

Author short Markdown paper: paper.md

Submit to JOSS by filling out short form

Editor assigns ≥1 reviewers, who review submission

Reviewer(s) raise comments and issues following guidelines:
http://joss.theoj.org/about#reviewer_guidelines

Authors fix issues

Editor accepts paper, authors archive software

Paper published & receives JOSS DOI

JOSS 10.21105/joss.######
Submit software for review

Before you submit
Please make sure you've read the submission instructions before submitting. In particular please make sure there is a paper.rst present in your repository that is structured like this. We promise this will make things go much more quickly during the review process.

Title
What's the title of this paper?

Repository address
What's the URL of your software?

Software version
E.g. v1.0.0

Suggested editor
Suggested editor

Description
Please give short (1-2 line) description of your software.

☐ I certify that I am submitting software for which I am a primary author
☐ I confirm that I read and will adhere to the JOSS code of conduct

Submit paper

The Journal of Open Source Software is an affiliate of the Open Source Initiative.

JOSS paper submission
JOSS paper reviews
[REVIEW]: hdbscan: A high performance implementation of HDBSCAN* clustering. #205

Submitted by: @lmclines (Leland McInnes)
Repository: https://github.com/scikit-learn-contrib/hdbscan
Version: v0.8.8
Editor: @danielskatz
Reviewer: @zhaozhao
Archive: 10.5281/zenodo.401403

Status

JOSS 10.21105/joss.00205

Status badge code:

```
HTML: &lt;a href="http://joss.theoj.org/papers/b5c5d4b749180e771c0e6225dcd60e49d5">![status](http://joss.theoj.org/papers/b5c5d4b749180e771c0e6225dcd60e49d5/status.png)
```

Reviewers and authors:

```
```

JOSS paper review
Journal of Open Research Software (JORS)

- Open access, peer-reviewed journal for “software metapapers” describing research software with high reuse potential
- Also publishes papers on aspects of creating, maintaining, and evaluating open-source software
- Fee: £100 (waivable)

http://openresearchsoftware.metajnl.com/
We should all care about using best practices

- We all write code or use software (yes, even experimentalists).
- Good programmers are 10x more productive than average.¹
- Research software frequently has errors—perhaps 5–100% of the time.²

² Soergel DAW. Rampant software errors may undermine scientific results [version 2; referees: 2 approved]. F1000Research 2015, 3:303 (https://doi.org/10.12688/f1000research.5930.2)
³ https://software-carpentry.org/
Using good practices can help us...

- accelerate long-term development
- avoid duplicate work
- increase reproducibility
- increase verification
- avoid headaches
Writing documentation and commenting code helps users and co-developers

- All projects are collaborative, so be nice to your team. Your most common collaborator is your future self!

- Helpful code comments say why something happens, not just what happens.

Not helpful:

```
# Gets radius
get_radius(object)
```

More helpful:

```
# Calculate the radius, needed
# to determine surface area.
get_radius(object)
```

- Annotating your code reduces the number of bugs!
We all like to save versions of our work...

Version Control Systems let you record changes and recover old versions

Other benefits

- Lab notebook: a record of who changed what, when, and why
- Off-site backup: on remote server
- Collaborative tool: co-authors can share changes and detect or resolve conflicts

This is not new:

- Subversion (2000)
- Git (2005)
Git is a modern version control system that will change your life

• very fast
• distributed (everyone has entire history, locally)
• trivially easy branching,
• and merging,
• rebasing,
• bisecting...

“Git is a knife whose handle is also a knife”¹

¹Philip Guo http://ixd.ucsd.edu/home/f16/index.php
Branching

crazy-experiment

master

new-function
Merging

Combination of commits from both branches

master

new-function
Automated testing can prevent many catastrophes

- How do you know your code still works if you don’t test it again?
- Legacy code: “code without tests”\(^1\)
- Unit tests: individual pieces work
  Integration tests: the pieces work together
  Regression tests: results haven’t changed by mistake
- Easy to measure “code coverage” of test suite. Aim to always increase it.

Developers warn against having unit tests without any integration tests.

twitter/@ThePracticalDev
twitter/@withzombies
Continuous Integration automatically runs entire test suite after every proposed change. Easy and free to set up! 

Travis CI

circleci

AppVeyor
https://doi.org/10.1371/journal.pbio.1001745
WSSSPE

• Working Towards Sustainable Software for Science: Practices and Experiences

• https://wssspe.researchcomputing.org.uk/

• WSSSPE Workshop series:
  • WSSSPE5.2: 24–27 Oct 2017, Auckland, New Zealand
  • WSSSPE4: report on arXiv tonight
Acknowledgements

• FORCE11 Software Citation Working Group: https://www.force11.org/group/software-citation-working-group

• WSSSPE Community: http://wssspe.researchcomputing.org.uk/


• NSF SI2 grants: ACI 1535065 & 1702722
Thank you!
More References


• Software Carpentry: [software-carpentry.org/](http://software-carpentry.org/)


• Git for Humans: [https://speakerdeck.com/alicebartlett/git-for-humans](https://speakerdeck.com/alicebartlett/git-for-humans)