



**Oregon State**  
University

# A project-based course on software development for (engineering) research

Kyle Niemeyer  
Oregon State University

13 June 2019

✉ [kyle.niemeyer@oregonstate.edu](mailto:kyle.niemeyer@oregonstate.edu)

🐦 @kyleniemeyer

🏠 [git.io/nrg](https://git.io/nrg)



# Motivation

The stats:

- 2009: **91%** scientists consider research software important/very important<sup>1</sup>
- UK academics in 2014: **90%** use software in research; **70%** of them would find research impractical without it<sup>2</sup>
- US postdocs in 2017: **95%** / **63%** same<sup>3</sup>

But: practically no graduate students / researchers receive formal training in this area.

*(Contrast with typical experimental methods courses)*

<sup>1</sup> JE Hannay, HP Langtangen, et al. SECSE 2009. Vancouver, BC, 1–8. <https://doi.org/10.1109/SECSE.2009.5069155>

<sup>2</sup> S Hettrick et al. *UK Research Software Survey 2014*. (2015) <https://doi.org/10.7488/ds/253>

<sup>3</sup> U Nangia & DS Katz, WSSSPE 5.1 (2017) <https://doi.org/10.5281/zenodo.814102>

# Computational Science?

# Computational Science?

software / computational skills : computational science

# Computational Science?

software / computational skills : computational science

::

# Computational Science?

software / computational skills : computational science

::

experimental skills : experimental science



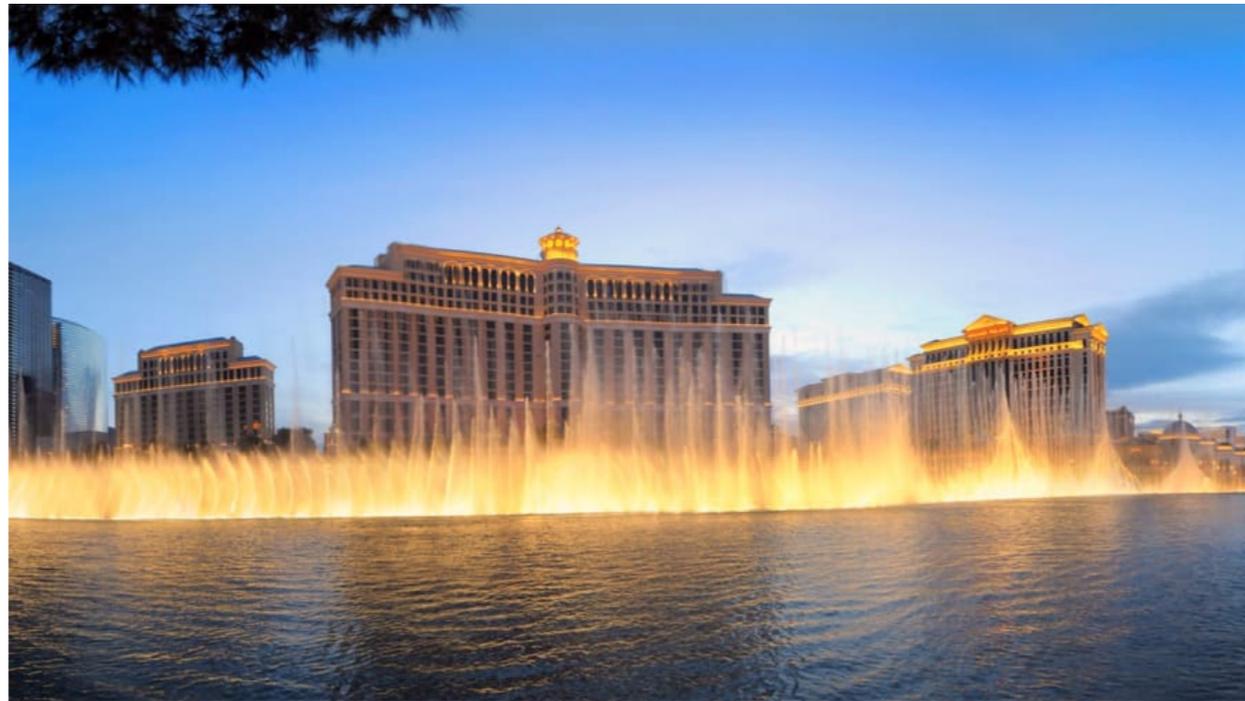


<https://edition.cnn.com/travel/article/spectacular-fountains/index.html>



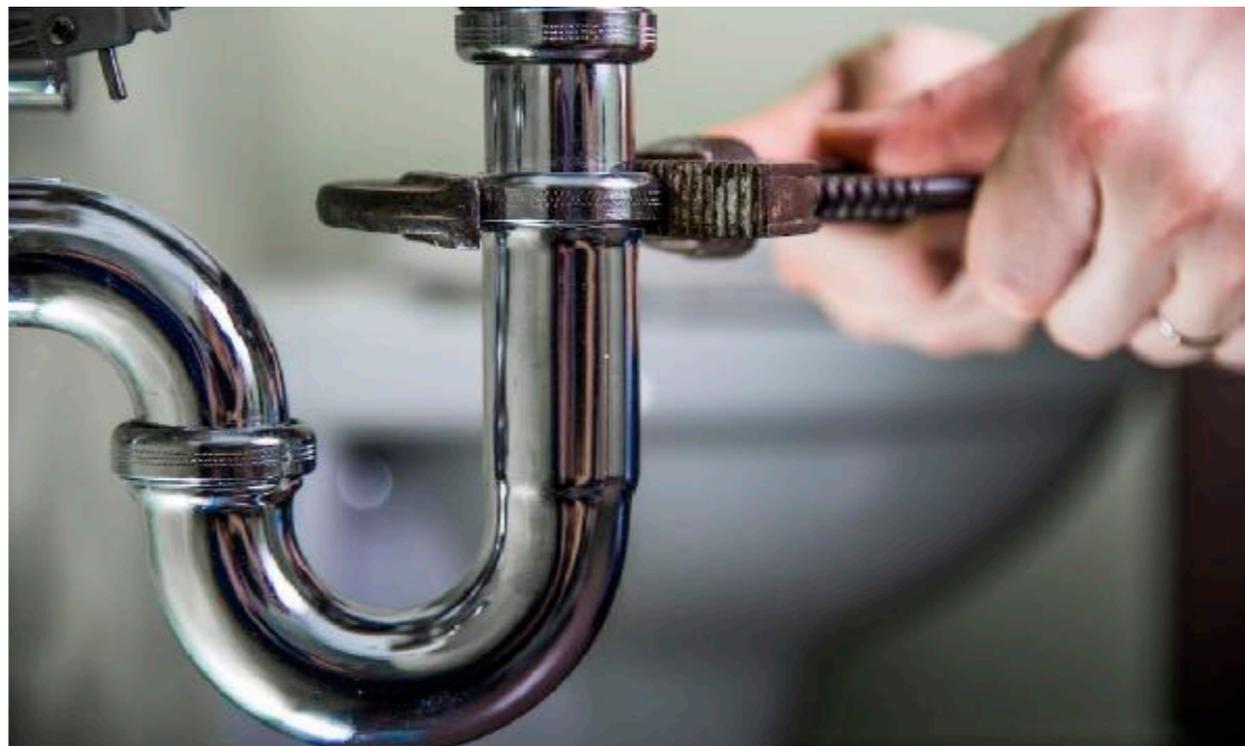
<https://edition.cnn.com/travel/article/spectacular-fountains/index.html>

needs



<https://edition.cnn.com/travel/article/spectacular-fountains/index.html>

# needs

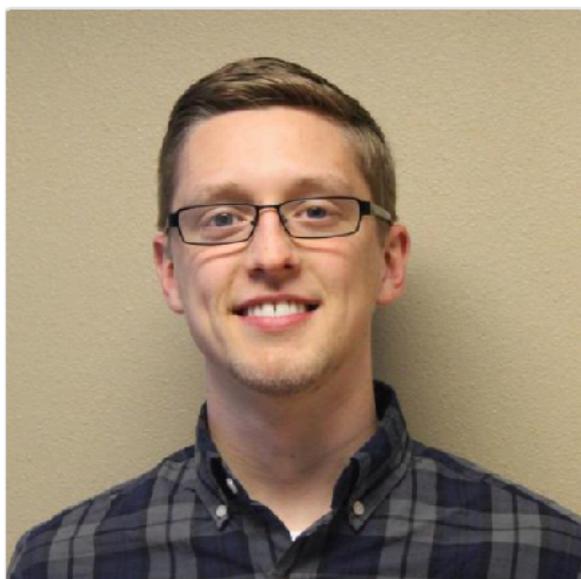


<http://artisanplumbing.ca>

# Who am I?

- Assistant Professor of Mechanical Engineering, Oregon State University
- Research focuses on simulations and modeling of reacting fluid flows and chemical kinetics
- Associate Editor-in-Chief, *Journal of Open Source Software*





at NCM 2019 in Pasadena!

**Kyle Niemeyer**  
kyleniemeyer

Developer Program Member

★ PRO

Edit profile

Assistant Professor of mechanical engineering. Leads

@Niemeyer-Research-Group

Oregon State University

Corvallis, OR

kyle.niemeyer@gmail.com

http://kyleniemeyer.com

Organizations

Overview Repositories 87 Projects 0 Stars 192 Followers 61 Following 16

Pinned

Customize your pins

SLACKHA/pyJac

Creates C and CUDA analytical Jacobians for chemical kinetics ODE systems

Python ★ 16 🍴 15

pr-omethe-us/PyTeCK

Automatically test chemical kinetic models using experimental data

Python ★ 5 🍴 6

irrev\_mech

Python utility that converts Chemkin-format chemical reaction mechanism with reversible reactions into one with only irreversible reactions

Python ★ 6 🍴 3

WSSSPE/meetings

Products from Working towards Sustainable Software for Science: Practice and Experiences (WSSSPE) activities

TeX ★ 16 🍴 35

pr-omethe-us/PyKED

Python interface to the ChemKED database format

Python ★ 8 🍴 12

Niemeyer-Research-Group/pyMARS

Python-based (chemical kinetic) Model Automatic Reduction Software

Python ★ 12 🍴 16

816 contributions in the last year

Contribution settings



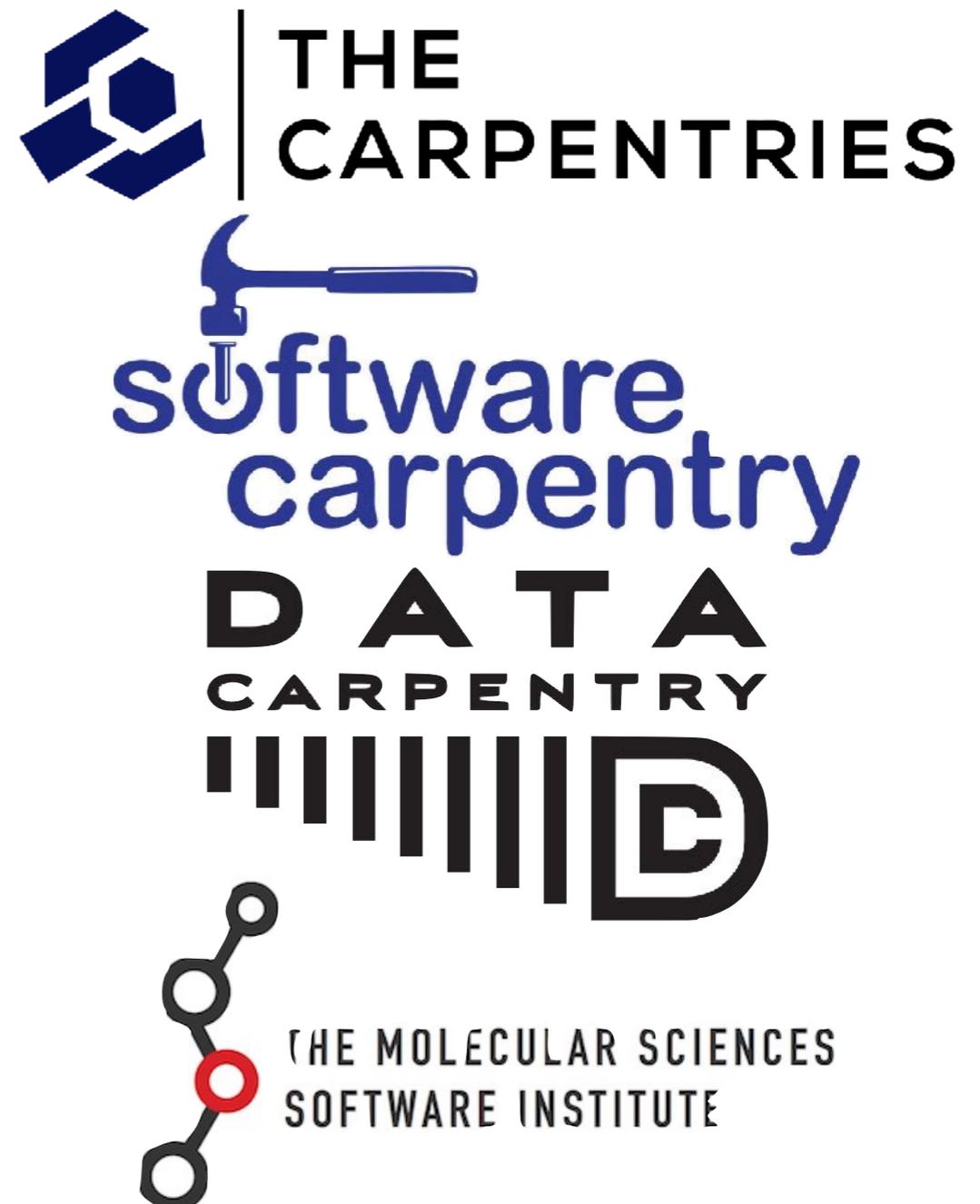
Learn how we count contributions.

Less      More

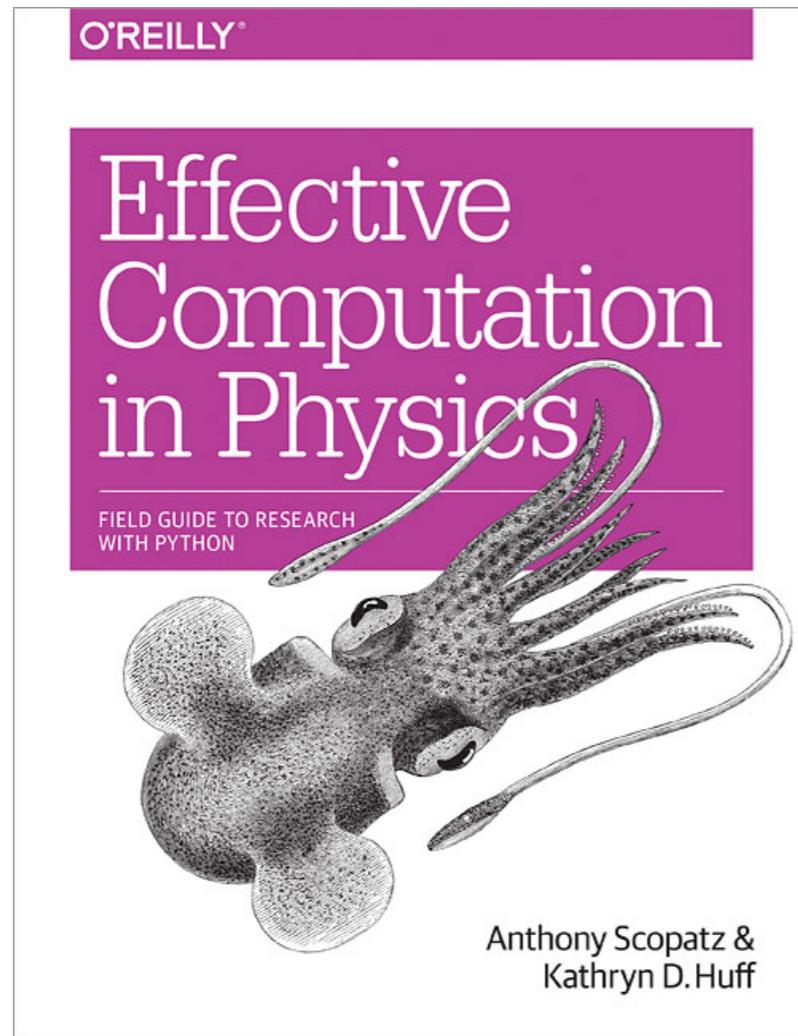
<https://github.com/kyleniemeyer>

# Existing Resources

- Software Carpentry workshops have become a popular way to teach fundamental skills: Unix/command line, version control with Git, and Python programming
- Similarly, Data Carpentry workshops for data science
- For domain scientists: MolSSI Software Summer School



# More (Static) Resources



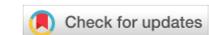
<http://physics.codes>



<https://doi.org/10.1371/journal.pcbi.1005510>

F1000Research

F1000Research 2017, 6:876 Last updated: 17 MAY 2019



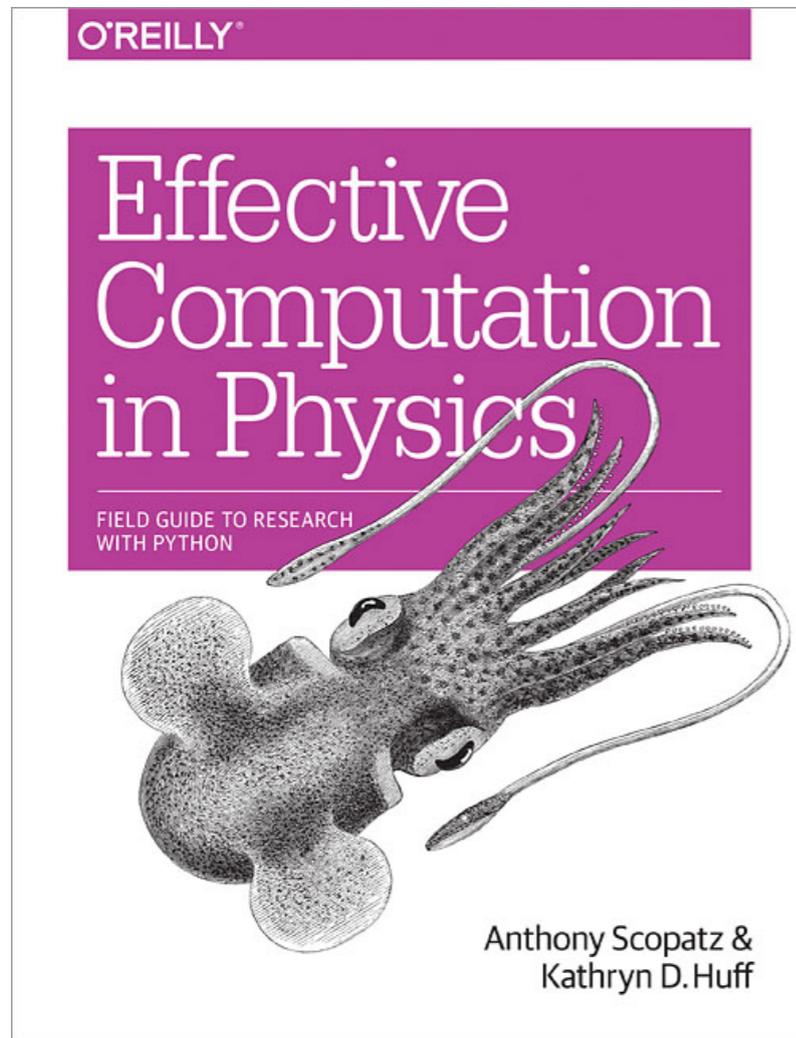
OPINION ARTICLE

**Four simple recommendations to encourage best practices in research software [version 1; peer review: 3 approved]**

Rafael C. Jiménez <sup>id</sup>1, Mateusz Kuzak<sup>2</sup>, Monther Alhamdoosh <sup>id</sup>3, Michelle Barker<sup>4</sup>, Bérénice Batut <sup>id</sup>5, Mikael Borg<sup>6</sup>, Salvador Capella-Gutierrez <sup>id</sup>7, Neil Chue Hong<sup>8</sup>, Martin Cook<sup>1</sup>, Manuel Corpas <sup>id</sup>9, Madison Flannery<sup>10</sup>, Leyla Garcia<sup>11</sup>, Josep Ll. Gelpí<sup>12,13</sup>, Simon Gladman<sup>10</sup>, Carole Goble<sup>14</sup>, Montserrat González Ferreiro<sup>11</sup>, Alejandra Gonzalez-Beltran <sup>id</sup>15, Philippa C. Griffin<sup>10</sup>, Björn Grüning <sup>id</sup>5, Jonas Hagberg <sup>id</sup>6, Petr Holub<sup>16</sup>, Rob Hooft <sup>id</sup>17, Jon Ison<sup>18</sup>, Daniel S. Katz <sup>id</sup>19-22, Brane Leskošek<sup>23</sup>, Federico López Gómez <sup>id</sup>1, Luis J. Oliveira<sup>24</sup>, David Mellor <sup>id</sup>25, Rowland Mosbergen<sup>26</sup>, Nicola Mulder <sup>id</sup>27, Yasset Perez-Riverol <sup>id</sup>11, Robert Pergl<sup>28</sup>, Horst Pichler<sup>29</sup>, Bernard Pope<sup>10</sup>, Ferran Sanz<sup>30</sup>, Maria V. Schneider<sup>10</sup>, Victoria Stodden<sup>20</sup>, Radosław Suchecki<sup>31</sup>, Radka Svobodová Vařeková<sup>32,33</sup>, Harry-Anton Talvik<sup>34</sup>, Ilian Todorov<sup>35</sup>, Andrew Treloar<sup>36</sup>, Sonika Tyagi<sup>10,37</sup>, Maarten van Gompel<sup>38</sup>, Daniel Vaughan<sup>11</sup>, Allegra Via<sup>39</sup>, Xiaochuan Wang<sup>40</sup>, Nathan S. Watson-Haigh<sup>31</sup>, Steve Crouch<sup>41</sup>

<https://doi.org/10.12688/f1000research.11407.1>

# More (Static) Resources



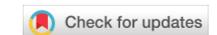
<http://physics.codes>



<https://doi.org/10.1371/journal.pcbi.1005510>

F1000Research

F1000Research 2017, 6:876 Last updated: 17 MAY 2019



OPINION ARTICLE

### Four simple recommendations to encourage best practices in research software [version 1; peer review: 3 approved]

Rafael C. Jiménez<sup>1</sup>, Mateusz Kuzak<sup>2</sup>, Monther Alhamdoosh<sup>3</sup>, Michelle Barker<sup>4</sup>, Bérénice Batut<sup>5</sup>, Mikael Borg<sup>6</sup>, Salvador Capella-Gutierrez<sup>7</sup>, Neil Chue Hong<sup>8</sup>, Martin Cook<sup>1</sup>, Manuel Corpas<sup>9</sup>, Madison Flannery<sup>10</sup>, Leyla Garcia<sup>11</sup>, Josep Ll. Gelpí<sup>12,13</sup>, Simon Gladman<sup>10</sup>, Carole Goble<sup>14</sup>, Montserrat González Ferreiro<sup>11</sup>, Alejandra Gonzalez-Beltran<sup>15</sup>, Philippa C. Griffin<sup>10</sup>, Björn Grüning<sup>5</sup>, Jonas Hagberg<sup>6</sup>, Petr Holub<sup>16</sup>, Rob Hooft<sup>17</sup>, Jon Ison<sup>18</sup>, Daniel S. Katz<sup>19-22</sup>, Brane Leskošek<sup>23</sup>, Federico López Gómez<sup>1</sup>, Luis J. Oliveira<sup>24</sup>, David Mellor<sup>25</sup>, Rowland Mosbergen<sup>26</sup>, Nicola Mulder<sup>27</sup>, Yasset Perez-Riverol<sup>11</sup>, Robert Pergl<sup>28</sup>, Horst Pichler<sup>29</sup>, Bernard Pope<sup>10</sup>, Ferran Sanz<sup>30</sup>, Maria V. Schneider<sup>10</sup>, Victoria Stodden<sup>20</sup>, Radosław Suchecki<sup>31</sup>, Radka Svobodová Vařeková<sup>32,33</sup>, Harry-Anton Talvik<sup>34</sup>, Ilian Todorov<sup>35</sup>, Andrew Treloar<sup>36</sup>, Sonika Tyagi<sup>10,37</sup>, Maarten van Gompel<sup>38</sup>, Daniel Vaughan<sup>11</sup>, Allegra Via<sup>39</sup>, Xiaochuan Wang<sup>40</sup>, Nathan S. Watson-Haigh<sup>31</sup>, Steve Crouch<sup>41</sup>

<https://doi.org/10.12688/f1000research.11407.1>

Big influence on my class!

# **My Solution:**

**A 10-week (primarily) graduate course to fill this gap, using existing resources as inspiration/building blocks.**

# Course Philosophy

# Course Philosophy

- **Main goal:** teach students best practices and practical skills to develop useful, tested, and (potentially) sustainable research software.

# Course Philosophy

- **Main goal:** teach students best practices and practical skills to develop useful, tested, and (potentially) sustainable research software.
- Rather than give standalone assignments, all assigned work relates to a quarter-long project, which is intended to support their thesis research.

# Course Philosophy

- **Main goal:** teach students best practices and practical skills to develop useful, tested, and (potentially) sustainable research software.
- Rather than give standalone assignments, all assigned work relates to a quarter-long project, which is intended to support their thesis research.
- Also try to instill open science as the default—relatively early in students' careers.

# Course Philosophy

- **Main goal:** teach students best practices and practical skills to develop useful, tested, and (potentially) sustainable research software.
- Rather than give standalone assignments, all assigned work relates to a quarter-long project, which is intended to support their thesis research.
- Also try to instill open science as the default—relatively early in students' careers.
- Classes mix a typical presentation-based lecture approach with interactive/hands-on work and discussion.

# Topics Taught

- Version control with Git
- Remote and collaborative version control with GitHub
- Licensing and copyright
- Testing software, test coverage, and continuous integration
- Structuring Python packages
- Writing documentation
- Objects and classes in Python
- Working with files
- Building command-line interfaces
- Packaging and distributing your Python software
- Introduction to parallel programming
- Open science, software citation, and reproducibility best practices
- Documentation with Sphinx

# **Example: module on testing**

<https://softwaredevengresearch.github.io/lecture-packaging-testing/>

# Testing your code

How do you know your code gives the right answers?

... what about after you make changes?

**Legacy code:** "code without tests" (Michael Feathers, *Working Effectively with Legacy Code*)

# Testing your code

when to test: **always.**

where to test: **external test suite.**

Example: `tests` subdirectory inside package.

Perfect is the enemy of good; a basic level of tests is better than nothing. But a rigorous test suite will save you time and potential problems in the long run.

# Why test?

Testing is a core principle of scientific software; it ensures results are trustworthy.

Scientific and engineering software is used for planes, power plants, satellites, and decisionmaking. Thus, correctness of this software is pretty important.

And we all know how easy it is to have mistakes in code without realizing it...

# What and how to test?

```
def kepler_loc(p1, p2, dt, t):  
    '''Use Kepler's Laws to predict location of celestial body'''  
    ...  
    return p3
```

```
def test_kepler_loc():  
    p1 = jupiter(two_days_ago)  
    p2 = jupiter(yesterday)  
    exp = jupiter(today)  
    obs = kepler_loc(p1, p2, 1, 1)  
    if exp != obs:  
        raise ValueError("Jupiter is not where it should be!")
```

# What is a test?

*Tests compare expected outputs versus observed outputs for known inputs. They do not inspect the body of the function directly. In fact, the body of a function does not even have to exist for a valid test to be written.*

```
def test_func():  
    exp = get_expected()  
    obs = func(*args, **kwargs)  
    assert exp == obs
```

# Good idea: test through assertions

For exactness:

```
def test_kepler_loc():  
    p1 = jupiter(two_days_ago)  
    p2 = jupiter(yesterday)  
    exp = jupiter(today)  
    obs = kepler_loc(p1, p2, 1, 1)  
    assert exp == obs
```

For approximate exactness:

```
import numpy as np  
def test_kepler_loc():  
    p1 = jupiter(two_days_ago)  
    p2 = jupiter(yesterday)  
    exp = jupiter(today)  
    obs = kepler_loc(p1, p2, 1, 1)  
    assert np.allclose(exp, obs)
```

# Test using `pytest`

```
# content of test_sample.py
def inc(x):
    return x + 1

def test_answer():
    assert inc(3) == 5
```

```
$ pytest
```

`pytest` finds all testing modules and functions, and runs them.

# Kinds of tests

**interior test:** precise points/values do not matter

**edge test:** test examines beginning or end of a range

**Best practice:** test all edges and at least one interior point.

**Also corner cases: two or more edge cases combined.**

```
import numpy as np

def sinc2d(x, y):
    '''(Describe the function here)'''
    if x == 0.0 and y == 0.0:
        return 1.0
    elif x == 0.0:
        return np.sin(y) / y
    elif y == 0.0:
        return np.sin(x) / x
    else:
        return (np.sin(x) / x) * (np.sin(y) / y)
```

```
import numpy as np
from mod import sinc2d

def test_internal():
    exp = (2.0 / np.pi) * (-2.0 / (3.0 * np.pi))
    obs = sinc2d(np.pi / 2.0, 3.0 * np.pi / 2.0)
    assert np.allclose(exp, obs)

def test_edge_x():
    exp = (-2.0 / (3.0 * np.pi))
    obs = sinc2d(0.0, 3.0 * np.pi / 2.0)
    assert np.allclose(exp, obs)
```

```
def test_edge_y():
    exp = (2.0 / np.pi)
    obs = sinc2d(np.pi / 2.0, 0.0)
    assert np.allclose(exp, obs)

def test_corner():
    exp = 1.0
    obs = sinc2d(0.0, 0.0)
    assert np.allclose(exp, obs)
```

# Types of tests

- **unit test:** interrogate individual functions and methods
- **integration test:** verify that multiple pieces of the code work together
- **regression test:** confirm that results match prior code results (which are assumed correct)

# Test generators

```
import numpy as np
import pytest

# contents of add.py
def add2(x, y):
    return x + y

class Test(object):
    @pytest.mark.parametrize('exp, x, y', [
        (4, 2, 2),
        (5, -5, 10),
        (42, 40, 2),
        (16, 3, 13),
        (-128, 0, -128),
    ])
    def test_add2(self, x, y, exp):
```

```
$ pytest
```

---

# Test-Driven Development (TDD)

Write the tests first.

Before you write any lines of a function, first write the test for that function.

---

```
from numpy import allclose
from mod import std

def test_std1():
    obs = std([0.0, 2.0])
    exp = 1.0
    assert np.allclose(exp, obs)
```

```
def std(vals):
    # this must be cheating.
    return 1.0
```

```
def test_std1():
    obs = std([0.0, 2.0])
    exp = 1.0
    assert np.allclose(exp, obs)

def test_std2():
    obs = std()
    exp = 0.0
    assert np.allclose(exp, obs)

def test_std3():
    obs = std([0.0, 4.0])
    exp = 2.0
    assert np.allclose(exp, obs)
```

```
def std(vals):
    # a bit better, but still not quite generic
    if len(vals) == 0:
        return 0.0
    return vals[-1] / 2.0
```

```
def test_std1():
    obs = std([0.0, 2.0])
    exp = 1.0
    assert np.allclose(exp, obs)

def test_std2():
    obs = std()
    exp = 0.0
    assert np.allclose(exp, obs)

def test_std3():
    obs = std([0.0, 4.0])
    exp = 2.0
    assert np.allclose(exp, obs)

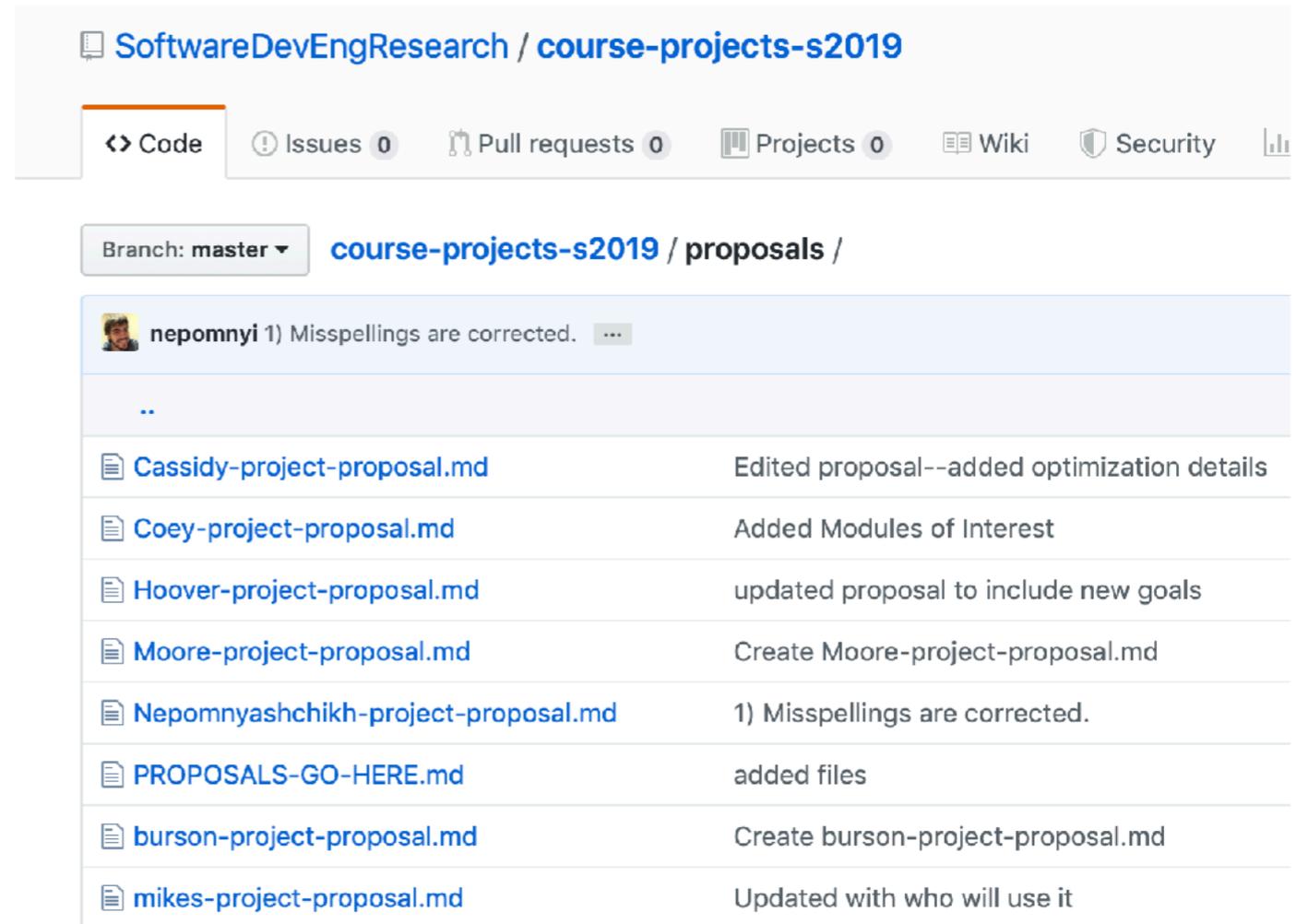
def test_std4():
    obs = std([1.0, 3.0])
    exp = 1.0
    assert np.allclose(exp, obs)

def test_std4():
    obs = std([1.0, 1.0, 1.0])
    exp = 0.0
    assert np.allclose(exp, obs)
```

```
def std(vals):
    # finally some math
    n = len(vals)
    if n == 0:
        return 0.0
    mu = sum(vals) / n
    var = 0.0
    for val in vals:
        var = var + (val - mu)**2
    return (var / n)**0.5
```

# Project

- At beginning of class, students propose a project for the term connected to their thesis research.
- This is submitted as a PR to the `course_projects` repo



The screenshot shows a GitHub pull request interface for the repository `SoftwareDevEngResearch / course-projects-s2019`. The current branch is `master` and the pull request is for the `proposals` directory. The pull request is titled "1) Misspellings are corrected." and is authored by `nepomnyi`. The pull request description shows a list of files that have been modified or added:

| File   | Change                                      |
|--|---|
| <code>Cassidy-project-proposal.md</code>         | Edited proposal--added optimization details |
| <code>Coey-project-proposal.md</code>            | Added Modules of Interest                   |
| <code>Hoover-project-proposal.md</code>          | updated proposal to include new goals       |
| <code>Moore-project-proposal.md</code>           | Create Moore-project-proposal.md            |
| <code>Nepomnyashchikh-project-proposal.md</code> | 1) Misspellings are corrected.              |
| <code>PROPOSALS-GO-HERE.md</code>                | added files                                 |
| <code>burson-project-proposal.md</code>          | Create burson-project-proposal.md           |
| <code>mikes-project-proposal.md</code>           | Updated with who will use it                |

# Project Stages

# Project Stages

Project proposal

# Project Stages

Project proposal

Create project repo & fork to account

# Project Stages

Project proposal

Create project repo & fork to account

Create first module with tests, submit as PR for review

# Project Stages

Project proposal

Create project repo & fork to account

Create first module with tests, submit as PR for review

Configure continuous integration

# Project Stages

Project proposal

Create project repo & fork to account

Create first module with tests, submit as PR for review

Configure continuous integration

Create command-line interface

# Project Stages

Project proposal

Create project repo & fork to account

Create first module with tests, submit as PR for review

Configure continuous integration

Create command-line interface

Package software for distribution

# Project Stages

Project proposal

Create project repo & fork to account

Create first module with tests, submit as PR for review

Configure continuous integration

Create command-line interface

Package software for distribution

Configure Zenodo integration

# Project Stages

Project proposal

Create project repo & fork to account

Create first module with tests, submit as PR for review

Configure continuous integration

Create command-line interface

Package software for distribution

Configure Zenodo integration

Write report and cite software

# Results from 2018–2019

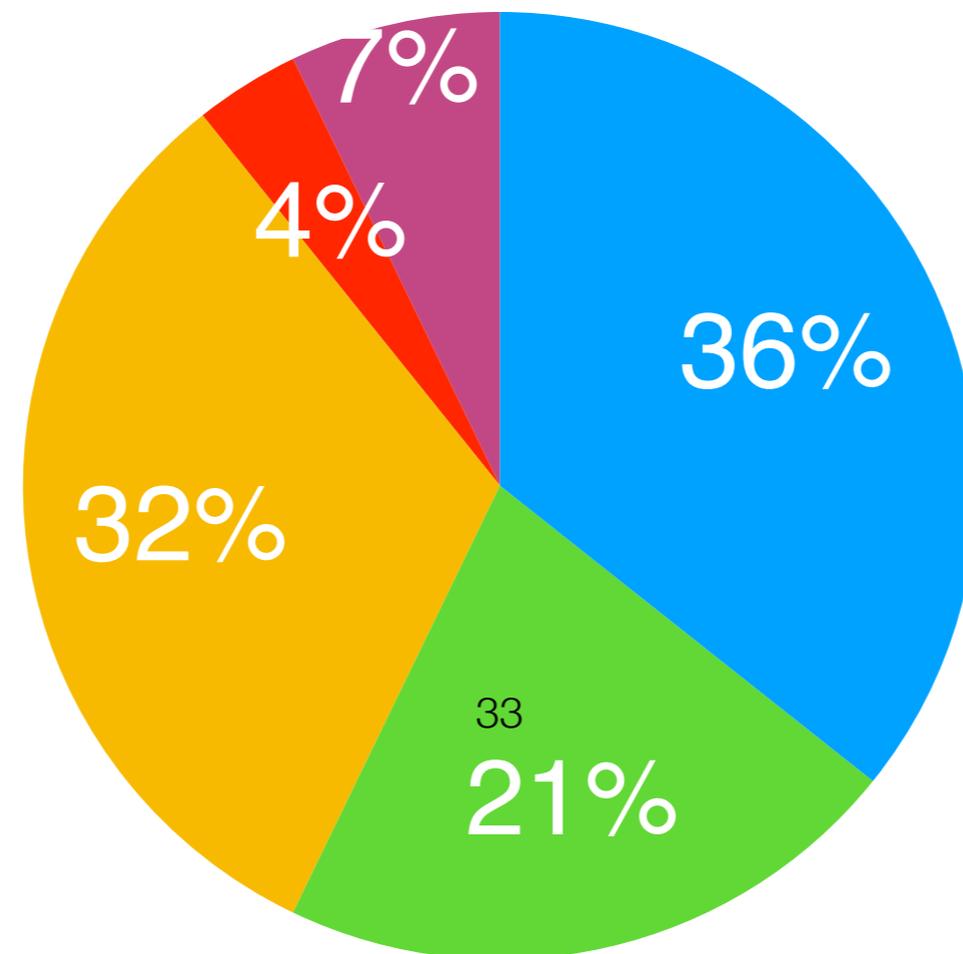
- Offered in Spring 2018 and Spring 2019 as “ME 599: Software Development for Engineering Research”.
- (However, nothing specifically Mechanical Engineering —or even Engineering—about it.)
- Enrollment: 17 students in 2018, 9 students in 2019.

# Project Topics

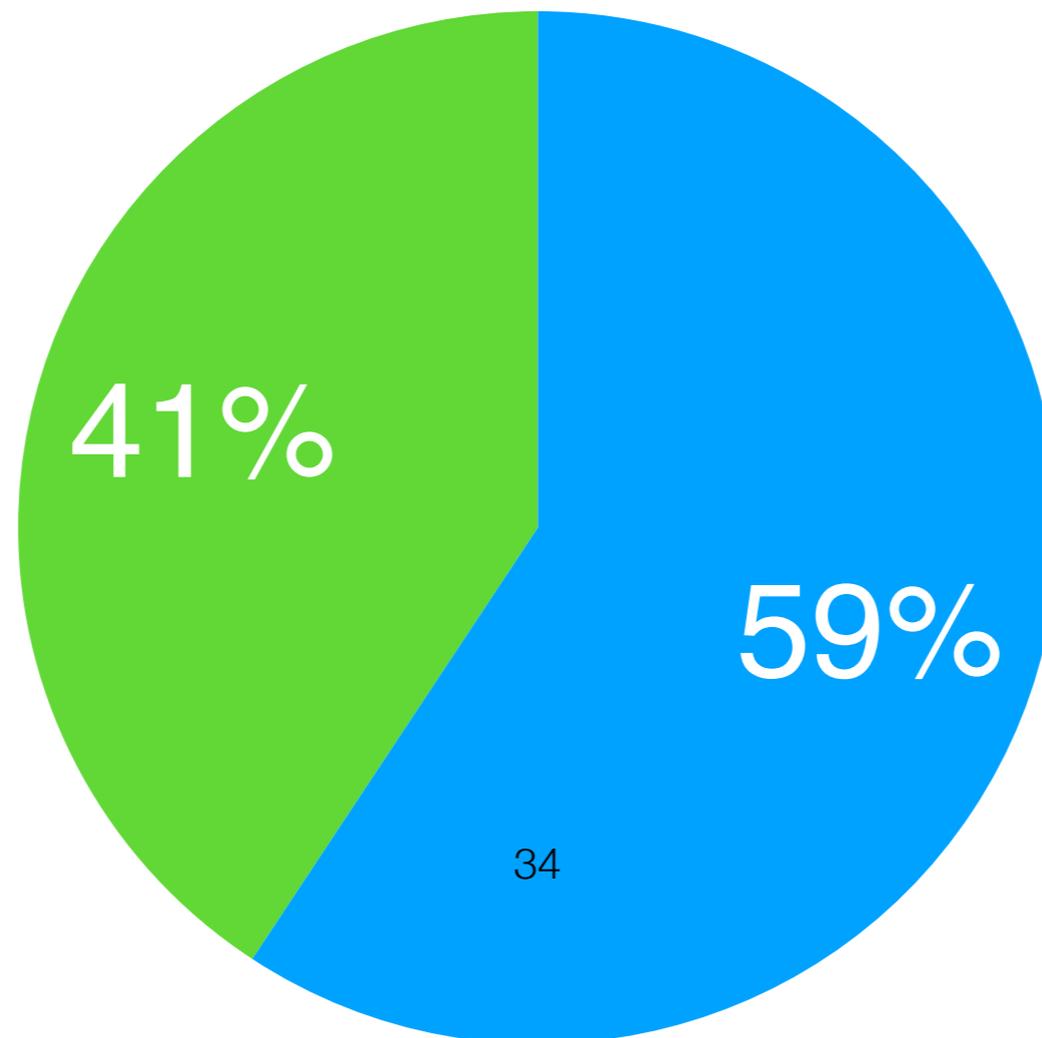
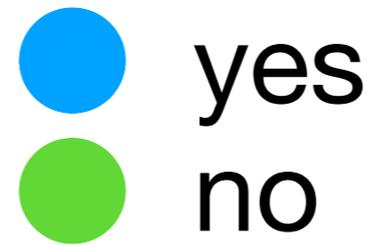
- designing detonation tubes
- extracting features with machine learning from nuclear physics simulations
- interfacing with an 8-channel digital pulse processor board
- simulating and analyzing a formula SAE vehicle engine
- optimizing and analyzing wind-farm layouts
- analyzing spin stabilization of solid rocket motors
- nodal quasi-diffusion solver for nuclear fission
- agent-learning for autonomous path finding
- generating input for Monte-Carlo radiation transport
- calculating solar-energy terms based on location
- analyzing radioxenon spectra
- calculating deep-learning layers for multi-agent reinforcement learners
- analyzing solvent extraction kinetics
- simulating rapid compression machine experiments
- calibrating blackbody infrared cameras
- simulating transient heat transfer in a microchannel
- biomass cookstove optimization tool
- projectile testing prediction
- transit system anomaly detection
- Automatic functional design representation
- Heat exchanger analysis
- Influx modeling for multiphase flows
- Control package for liquid rocket engine test stand

# Student composition

- 1st-year MS/PhD student
- 2nd-year MS/PhD student
- $\geq 3$ rd-year PhD student
- Undergraduate
- MEng

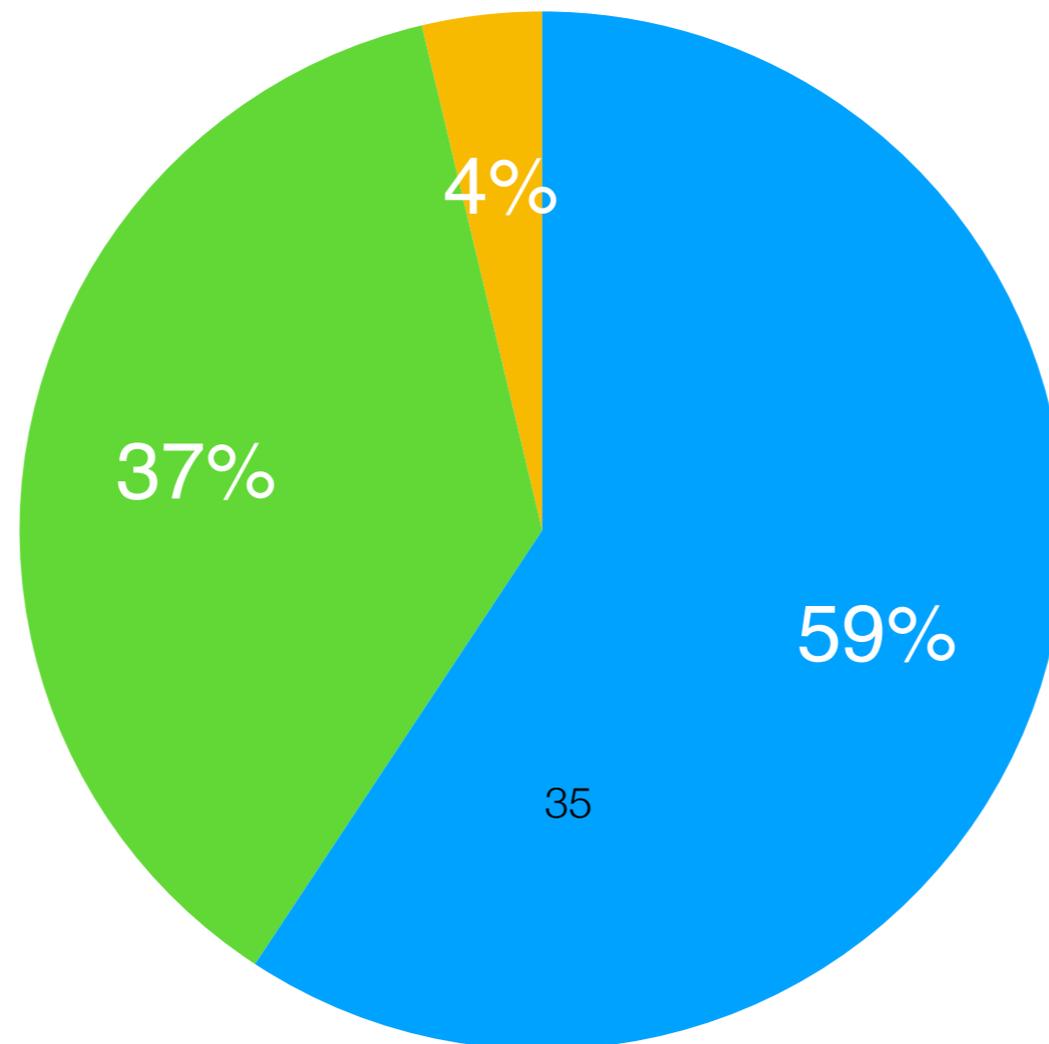


# Whether students took Python class



# Comfort with command line

- I've used it, though I'm not exactly comfortable with it.
- I'm comfortable working on the command line, but not an expert
- What is the command line?
- I'm a command-line ninja 🤘



# Student Feedback

# Student Feedback

- In both cases, students rate the course as a whole highly

# Student Feedback

- In both cases, students rate the course as a whole highly
- Comments suggest that all graduate students working with software/programming need this material

# Student Feedback

- In both cases, students rate the course as a whole highly
- Comments suggest that all graduate students working with software/programming need this material
- Some feelings of being too advanced for their experience level

# Post-Course Work

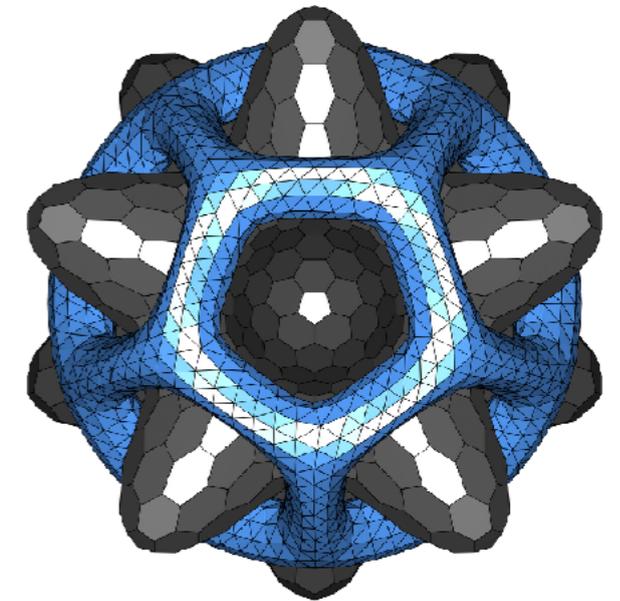
# Post-Course Work

- In both classes so far, roughly half the students indicated that they plan to use and/or develop their packages further for their research

# Post-Course Work

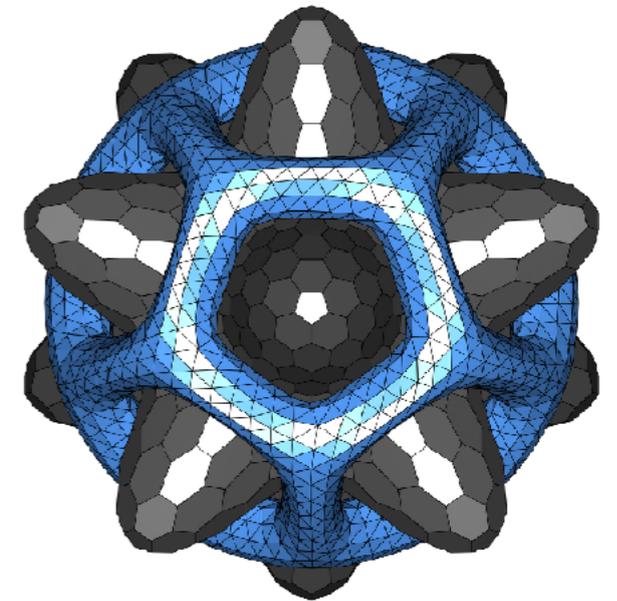
- In both classes so far, roughly half the students indicated that they plan to use and/or develop their packages further for their research
- One ultimate goal: encourage submission to JOSS— one under review, and another to be submitted!

# JOSS: Journal of Open Source Software



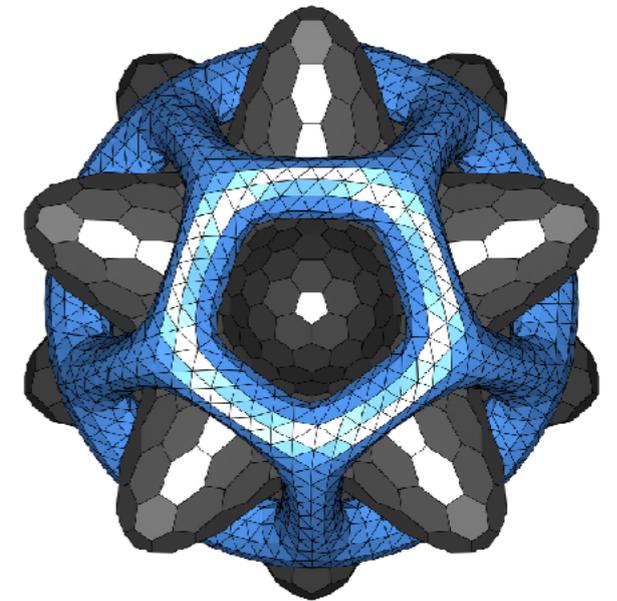
# JOSS: Journal of Open Source Software

- <http://joss.theoj.org/>



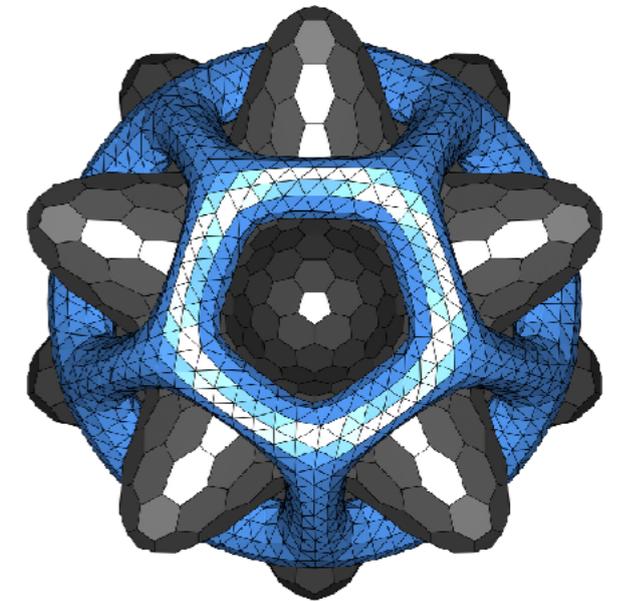
# JOSS: Journal of Open Source Software

- <http://joss.theoj.org/>
- Developer-friendly journal for research software packages



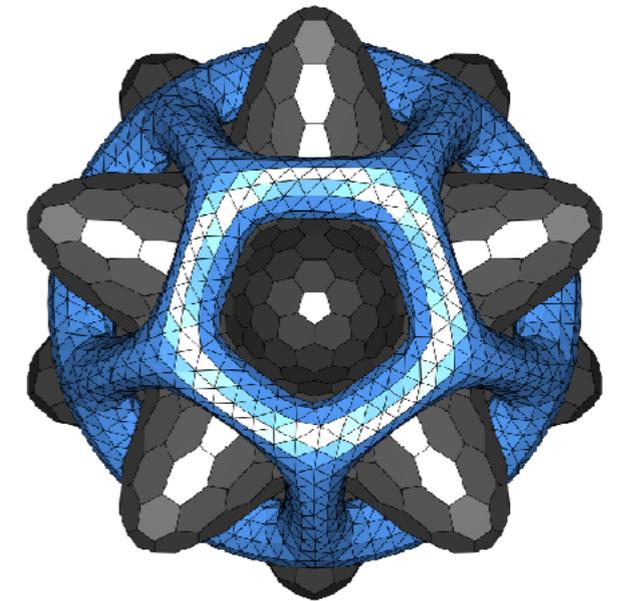
# JOSS: Journal of Open Source Software

- <http://joss.theoj.org/>
- Developer-friendly journal for research software packages
- Affiliate of Open Source Initiative



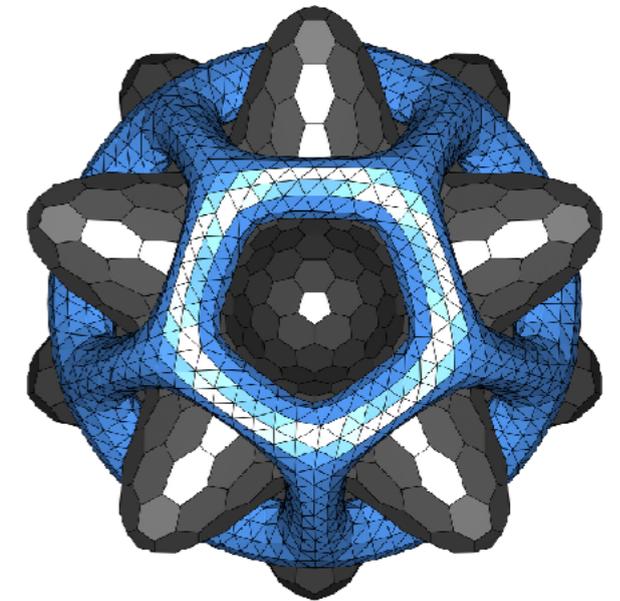
# JOSS: Journal of Open Source Software

- <http://joss.theoj.org/>
- Developer-friendly journal for research software packages
- Affiliate of Open Source Initiative
- Open access, no fees



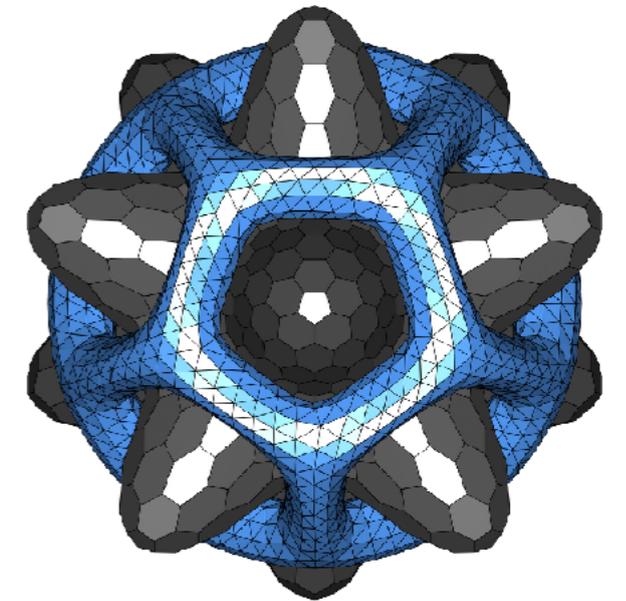
# JOSS: Journal of Open Source Software

- <http://joss.theoj.org/>
- Developer-friendly journal for research software packages
- Affiliate of Open Source Initiative
- Open access, no fees
- As of this morning, 597 submissions published



# JOSS: Journal of Open Source Software

- <http://joss.theoj.org/>
- Developer-friendly journal for research software packages
- Affiliate of Open Source Initiative
- Open access, no fees
- As of this morning, 597 submissions published



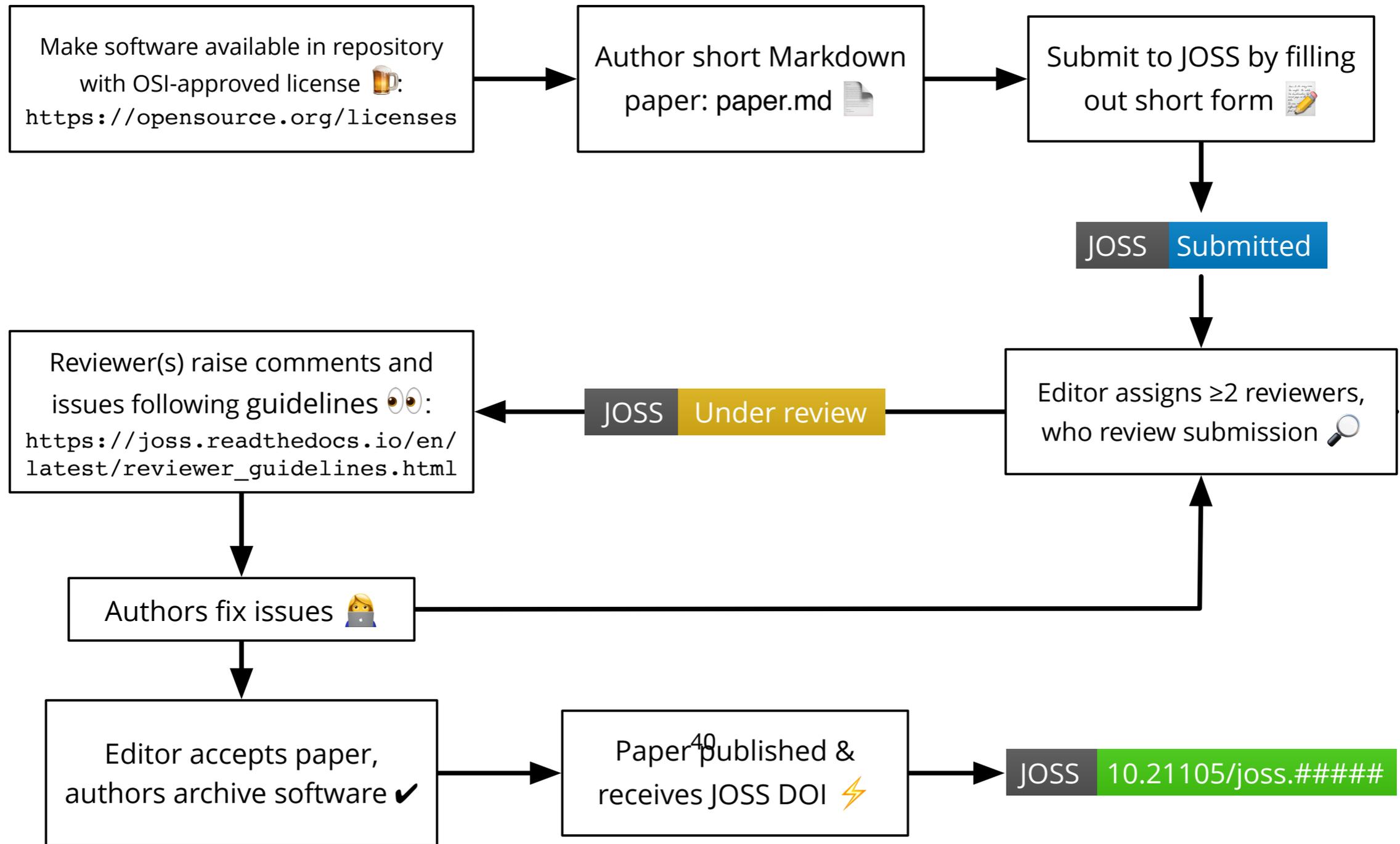
“If you've already licensed your code and have good documentation then we expect that it should take less than an hour to <sup>38</sup>prepare and submit your paper to JOSS.”



# JOSS Editorial Board



# JOSS Workflow



The Journal of Open Source Software

Submit Papers About Arfon Smith · Sign out

## 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.md` 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. View editors [here](#) »**  
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.

© The Journal of Open Source Software

Issues · openjournals/joss-reviews

GitHub, Inc. [US] <https://github.com/openjournals/joss-reviews/issues>

This repository Search Pull requests Issues Gist

openjournals / joss-reviews Watch 31 Star 53 Fork 0

Code Issues 44 Pull requests 0 Projects 0 Pulse Graphs Settings

Filters is:issue is:open Labels Milestones New issue

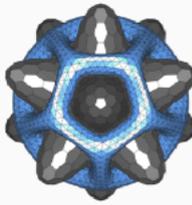
| <input type="checkbox"/> | 44 Open ✓ 176 Closed  | Author                            | Labels   | Projects | Milestones | Assignee | Sort |
|--------------------------|---|-----------------------------------|----------|----------|------------|----------|------|
| <input type="checkbox"/> | <b>[REVIEW]: Brightway: An open source framework for life cycle assessment</b> review                     | #236 opened a day ago by whedon   | 0 of 17  |          |            |          | 1    |
| <input type="checkbox"/> | <b>[REVIEW]: Text detection in screen images with a Convolutional Neural Network</b> review               | #235 opened 4 days ago by whedon  | 0 of 17  |          |            |          | 1    |
| <input type="checkbox"/> | <b>[REVIEW]: biogo/ncbi: interfaces to NCBI services for the Go language</b> review                       | #234 opened 6 days ago by whedon  | 0 of 17  |          |            |          | 3    |
| <input type="checkbox"/> | <b>[REVIEW]: nails: Network Analysis Interface for Literature Studies</b> review                          | #233 opened 7 days ago by whedon  | 14 of 17 |          |            |          | 7    |
| <input type="checkbox"/> | <b>[REVIEW]: libqcpp: A C++14 sequence quality control library</b> review                                 | #232 opened 7 days ago by whedon  | 0 of 17  |          |            |          | 1    |
| <input type="checkbox"/> | <b>[REVIEW]: flusight</b> review  | #231 opened 8 days ago by whedon  | 0 of 17  |          |            |          | 3    |
| <input type="checkbox"/> | <b>[PRE REVIEW]: qGaussian: An R package that deal with the Tsallis statistics</b> pre-review             | #229 opened 9 days ago by whedon  |          |          |            |          | 6    |
| <input type="checkbox"/> | <b>[REVIEW]: cbcbeat: an adjoint-enabled framework for computational cardiac electrophysiology</b> review | #224 opened 10 days ago by whedon | 0 of 17  |          |            |          | 1    |

# JOSS paper reviews

The screenshot shows a GitHub issue page for the repository 'openjournals / joss-reviews'. The issue title is '[REVIEW]: hdbscan: A high performance implementation of HDBSCAN\* clustering. #205'. The issue is marked as 'Closed' and was opened by 'whedon' on 13 Mar with 17 comments. The issue content includes submission details: author '@lmcinnes (Leland McInnes)', repository 'https://github.com/scikit-learn-contrib/hdbscan', version 'v0.8.8', editor '@danielskatz', reviewer '@zhaozhang', and archive '10.5281/zenodo.401403'. A 'Status' section shows a badge for 'JOSS 10.21105/joss.00205' and provides HTML and Markdown code for the status badge. The right sidebar shows 'Assignees' (None), 'Labels' (accepted, review), 'Projects' (None yet), 'Milestone' (None), and 'Notifications' (Unsubscribe).

# JOSS paper review

# More about JOSS



**Journal of Open Source Software Blog**

Home

Blog for the Journal of Open Source Software <http://joss.theoj.org>



4 JUN, 2019 · BLOG

## Cost models for running an online open journal

The Journal of Open Source Software (JOSS) is a free, open-access online journal, with no article processing charge (APC). We are committed to operating as a free service to our community, and we do so thanks to the volunteer labor of editors and reviewers, and by taking advantage of existing infrastructure. In this post, we examine the true costs of running a journal such as JOSS, and make the case that even when considering all services we don't currently pay for, the true cost per paper would not exceed \$100. Current APCs at many "gold" open-access journals exceed that by one or more orders of magnitude, (see, for example, [PNAS](#), [Nature](#), [IEEE](#), etc.)

### Real costs we *\*have\** to pay

- Annual Crossref membership: \$275/year
- JOSS paper DOIs: \$1/accepted paper
- JOSS website hosting: \$19/month
- JOSS domain name registration: \$10/year

At 300 papers/year, this is \$813, or \$2.71/paper. This is what we actually pay today, covered by a grant from the Alfred P. Sloan Foundation.



## Journal of Open Source Software (JOSS): design and first-year review

Arfon M. Smith<sup>1</sup>, Kyle E. Niemeyer<sup>2</sup>, Daniel S. Katz<sup>3</sup>, Lorena A. Barba<sup>4</sup>, George Githinji<sup>5</sup>, Melissa Gymrek<sup>6</sup>, Kathryn D. Huff<sup>7</sup>, Christopher R. Madan<sup>8</sup>, Abigail Cabunoc Mayes<sup>9</sup>, Kevin M. Moerman<sup>10,11</sup>, Pjotr Prins<sup>12,13</sup>, Karthik Ram<sup>14</sup>, Ariel Rokem<sup>15</sup>, Tracy K. Teal<sup>16</sup>, Roman Valls Guimera<sup>17</sup> and Jacob T. Vanderplas<sup>15</sup>

- <sup>1</sup>Data Science Mission Office, Space Telescope Science Institute, Baltimore, MD, United States of America  
<sup>2</sup>School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, United States of America  
<sup>3</sup>National Center for Supercomputing Applications & Department of Computer Science & Department of Electrical and Computer Engineering & School of Information Sciences, University of Illinois at Urbana-Champaign, Urbana, IL, United States of America  
<sup>4</sup>Department of Mechanical & Aerospace Engineering, The George Washington University, Washington, D.C., United States of America  
<sup>5</sup>KEMRI—Wellcome Trust Research Programme, Kilifi, Kenya  
<sup>6</sup>Departments of Medicine & Computer Science and Engineering, University of California, San Diego, La Jolla, CA, United States of America  
<sup>7</sup>Department of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, United States of America  
<sup>8</sup>School of Psychology, University of Nottingham, Nottingham, United Kingdom  
<sup>9</sup>Mozilla Foundation, Toronto, Ontario, Canada  
<sup>10</sup>MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, United States of America  
<sup>11</sup>Trinity Centre for Bioengineering, Trinity College, The University of Dublin, Dublin, Ireland  
<sup>12</sup>University of Tennessee Health Science Center, Memphis, TN, United States of America  
<sup>13</sup>University Medical Centre Utrecht, Utrecht, The Netherlands  
<sup>14</sup>Berkeley Institute for Data Science, University of California, Berkeley, CA, United States of America  
<sup>15</sup>eScience Institute, University of Washington, Seattle, WA, United States of America  
<sup>16</sup>Data Carpentry, Davis, CA, United States of America  
<sup>17</sup>University of Melbourne Centre for Cancer Research, University of Melbourne, Melbourne, Australia

Submitted 6 October 2017  
Accepted 24 January 2018  
Published 12 February 2018

Corresponding authors  
Arfon M. Smith, [arfon@stsci.edu](mailto:arfon@stsci.edu)  
Kyle E. Niemeyer,  
[Kyle.Niemeyer@oregonstate.edu](mailto:Kyle.Niemeyer@oregonstate.edu),  
[arfon.smith@gmail.com](mailto:arfon.smith@gmail.com)

Academic editor  
Edward Fox

Additional Information and  
Declarations can be found on  
page 19

DOI 10.7717/peerj-cs.147

© Copyright  
2018 Smith et al.

Distributed under  
Creative Commons CC-BY 4.0

OPEN ACCESS

### ABSTRACT

This article describes the motivation, design, and progress of the Journal of Open Source Software (JOSS). JOSS is a free and open-access journal that publishes articles describing research software. It has the dual goals of improving the quality of the software submitted and providing a mechanism for research software developers to receive credit. While designed to work within the current merit system of science, JOSS addresses the dearth of rewards for key contributions to science made in the form of software. JOSS publishes articles that encapsulate scholarship contained in the software itself, and its rigorous peer review targets the software components: functionality, documentation, tests, continuous integration, and the license. A JOSS article contains an abstract describing the purpose and functionality of the software, references, and a link to the software archive. The article is the entry point of a JOSS submission, which encompasses the full set of software artifacts. Submission and review proceed in the open, on GitHub. Editors, reviewers, and authors work collaboratively and openly. Unlike other journals, JOSS does not reject articles requiring major revision; while not

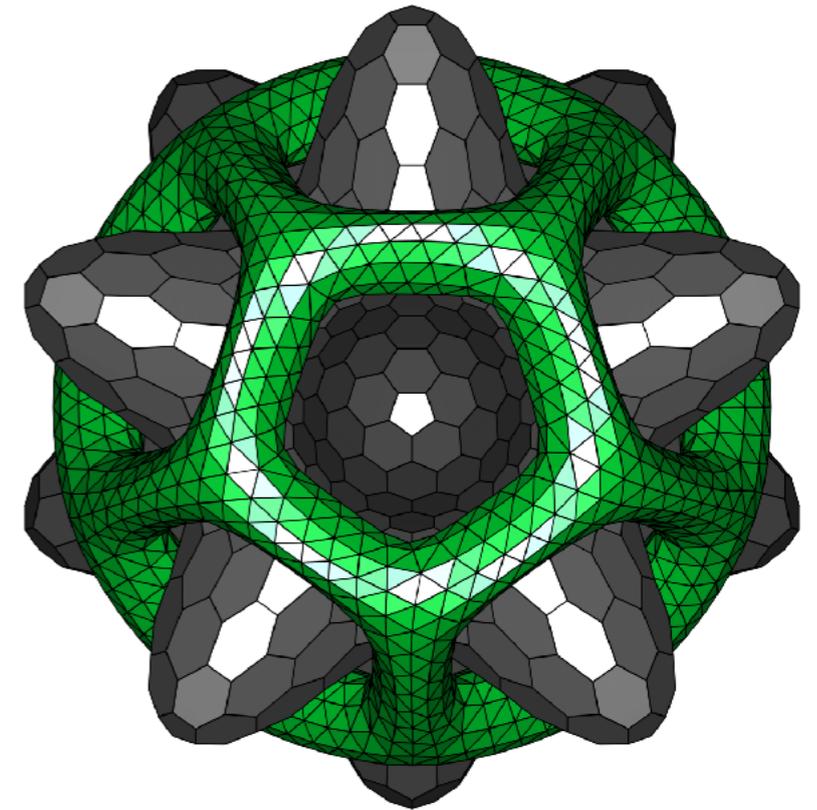
44  
JOSS blog: <http://blog.joss.theoj.org>

Smith et al. (2018), Journal of Open Source Software (JOSS): design and first-year review. *PeerJ Comput. Sci.* 4:e147; DOI 10.7717/peerj-cs.147

# Sister Journal: JOSE

## Journal of Open Source Education

- Part of the Open Journals, sibling journal to JOSS (literally forked from it!). Also NumFOCUS affiliate.
- JOSE publishes two types of articles that describe:
  - \*open educational software tools
  - \*open-source learning modules
- Motivation: credit efforts to develop software for assisting teaching/learning and open-source educational content
- Submissions are peer-reviewed, with the intent of *improving the quality of the software or content submitted*



# JOSE Editorial Board



**Lorena Barba**  
George Washington Univ.



**Katy Huff**  
UIUC



**Juan Flopped**  
University of Cape Town



**Jason Moore**  
UC Davis



**Kyle Niemeyer**  
Oregon State



**Charles Severance**  
Univ. Michigan



**Robert Talbert**  
Grand Valley State U.



**Tracy Teal**  
Univ. Michigan



**Carol Willing**  
Cal Poly San Luis Obispo

# Summary and Future Work

# Summary and Future Work

- Course offered twice with ME 599 temporary class identifier; plan to offer again in Spring 2021 — will need permanent “location”.

# Summary and Future Work

- Course offered twice with ME 599 temporary class identifier; plan to offer again in Spring 2021 — will need permanent “location”.
- Most students come in with little/no Unix command line experience, and only some with Python programming — plan to offer regular Software Carpentry workshops for first-year grad students in engineering.

# Summary and Future Work

- Course offered twice with ME 599 temporary class identifier; plan to offer again in Spring 2021 —will need permanent “location”.
- Most students come in with little/no Unix command line experience, and only some with Python programming—plan to offer regular Software Carpentry workshops for first-year grad students in engineering.
- Developing standalone modules for topics, to be shared openly

# Summary and Future Work

- Course offered twice with ME 599 temporary class identifier; plan to offer again in Spring 2021 — will need permanent “location”.
- Most students come in with little/no Unix command line experience, and only some with Python programming — plan to offer regular Software Carpentry workshops for first-year grad students in engineering.
- Developing standalone modules for topics, to be shared openly
- Submit to JOSS and/or JOSE!

# Thank you!

## Feedback and questions are welcome.

`softwaredevengresearch.github.io/syllabus-s2019`

`niemeyer-research-group.github.io`

**Funding:** Better Scientific Software Fellowship, part of the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.