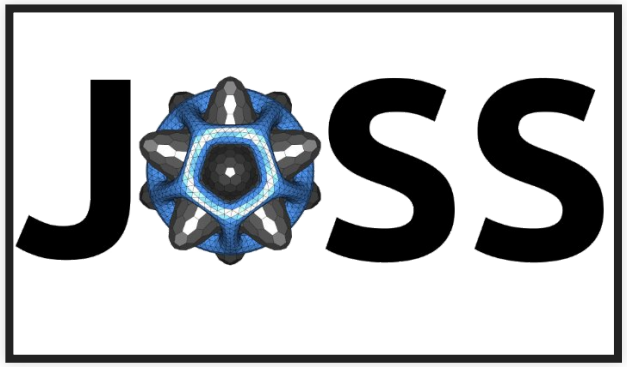
PUBLISHING YOUR SOFTWARE PROJECT

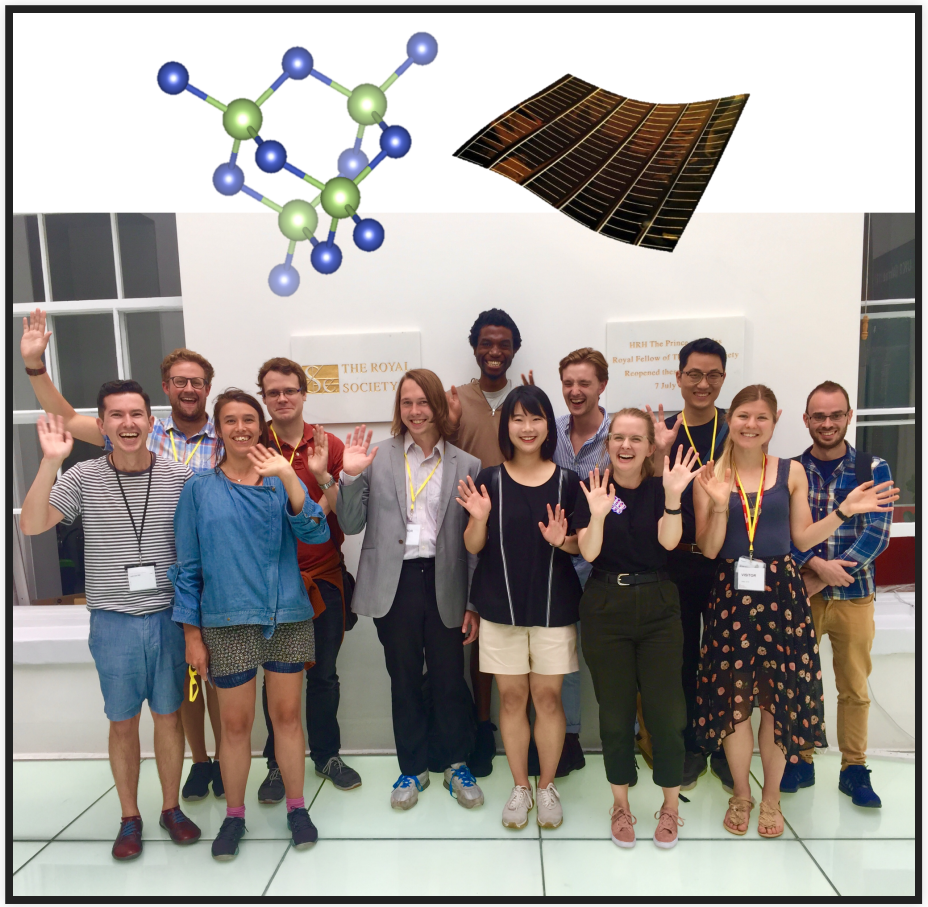
WITH THE JOURNAL OF OPEN SOURCE SOFTWARE



Lucy Whalley

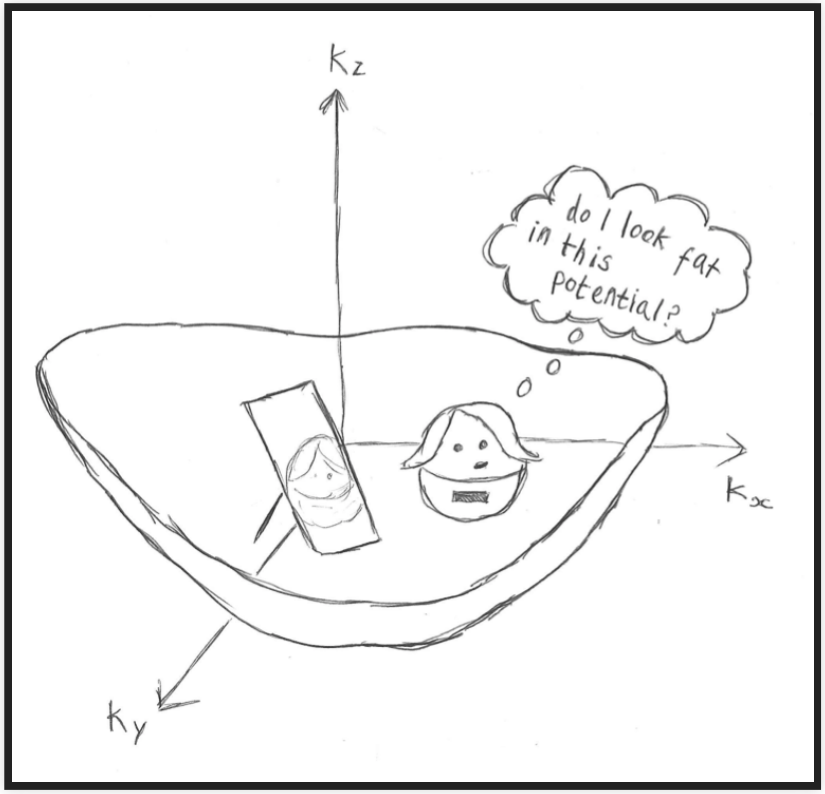
lucydot.github.io/slides

MATERIALS DESIGN GROUP @ ICL



github.com/WMD-group

`effmass` PYTHON PACKAGE



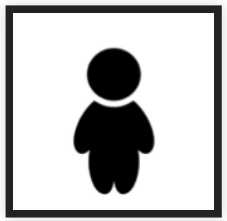
code: github.com/lucydot/effmass

research paper: arxiv.org/abs/1811.02281

FROM THE JOSS GUIDING PRINCIPLES

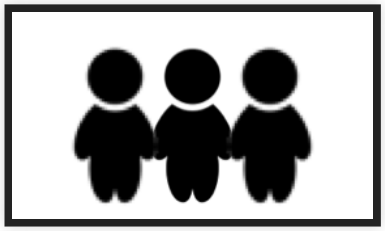
"We like to think of JOSS as a 'developer friendly' journal. That is, if the submitting authors have followed best practices (have documentation, tests, continuous integration, and a license) then their review should be rapid."

WHY SHOULD I SUBMIT TO JOSS?



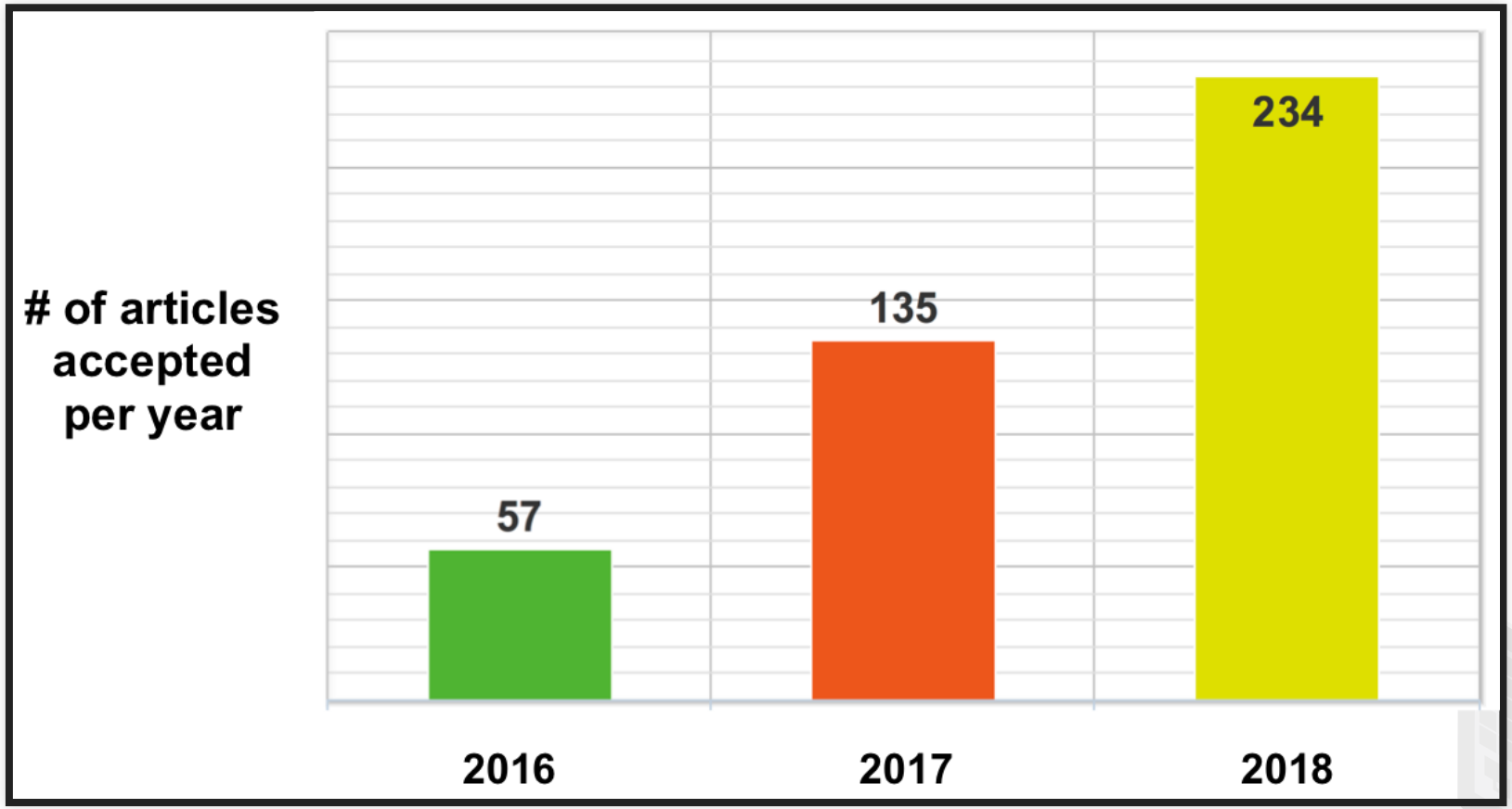
- published paper and citations
- an incentive to learn new tools
- peer review process brings increased confidence
- good way to promote your code to the community

WHY SHOULD *WE* SUBMIT TO JOSS?

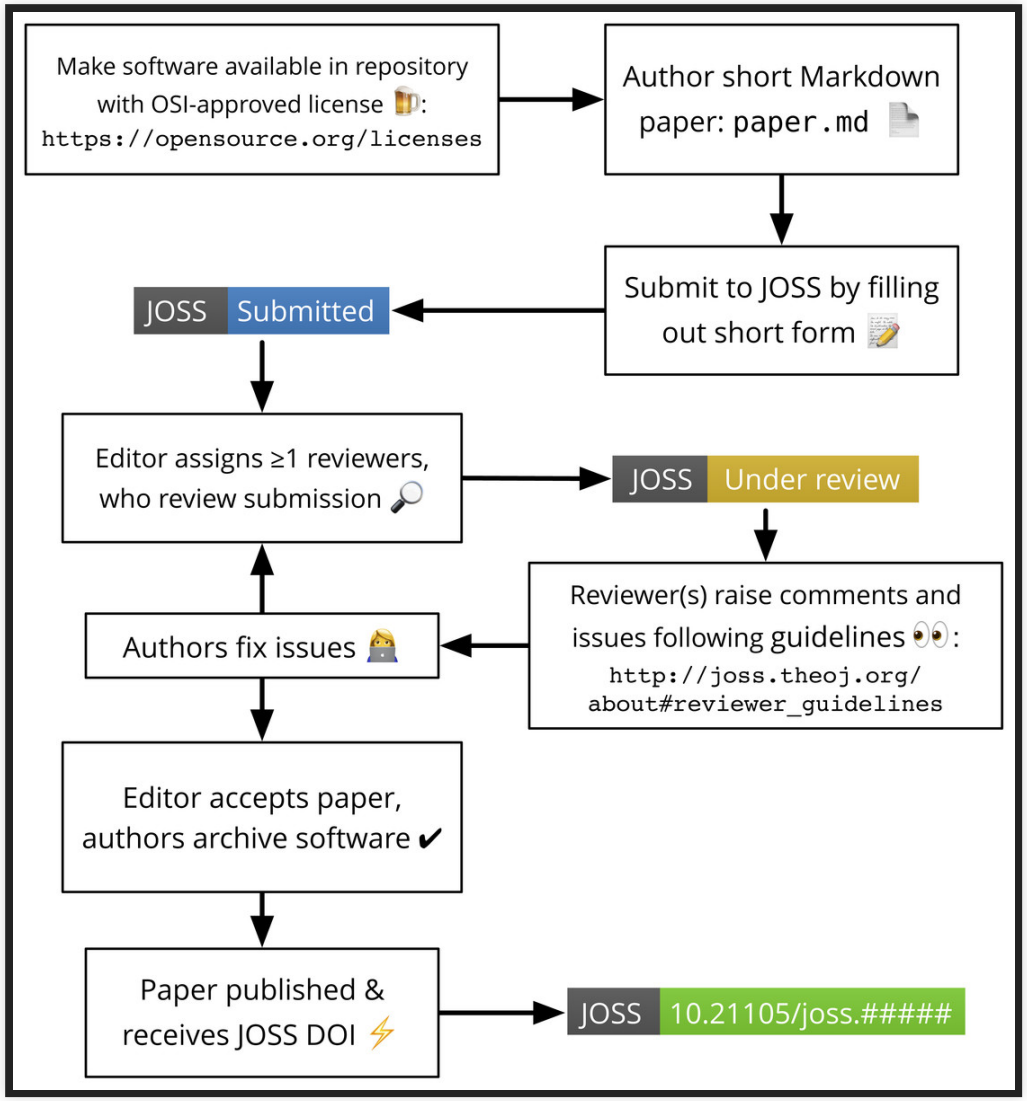


- well-documented and well-tested software freely available to the research community
- reproducibility, sustainability, maintainability
- article: "The Scientific Paper Is Obsolete"

JOSS IS GROWING



THE JOSS SUBMISSION AND REVIEW FLOW



A JOSS PAPER CONTAINS...

- A list of the software authors and their affiliations
- A summary describing the high-level functionality
- A statement of need
- A list of key references
- A summary of research projects using the software

THE JOSS REVIEW CRITERIA

- Software license
- Functionality
- Installation instructions
- Community guidelines
- **Tests**
- **Documentation**

TESTS

A simple unit test

```
from mean import *

def test_ints():
    num_list = [1,2,3,4,5]
    obs = mean(num_list)
    exp = 3
    assert obs == exp
```

Example from Katy Huff's Python testing workshop

TESTS

Another simple unit test

```
def test_zero():
    num_list=[0,2,4,6]
    obs = mean(num_list)
    exp = 3
    assert obs == exp
```

Example from Katy Huff's Python testing workshop

TESTS

- **unit tests**: test individual functions
- **integration tests**: test that the functions work together correctly
- **end-to-end tests**: test from start to finish

Tools: pytest, Travis CI, Jupyter Notebook

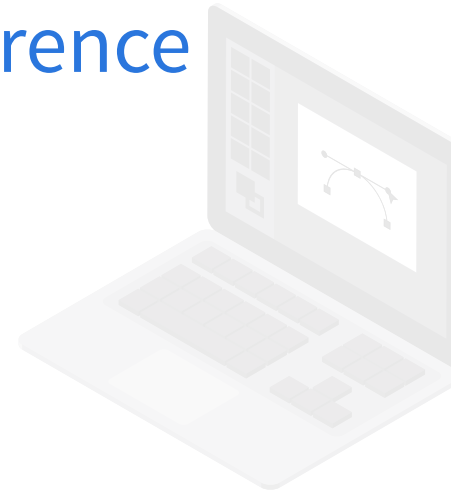
`effmass`: unit tests, CI, (manual) E2E

DOCUMENTATION

- **tutorials**: how to complete a particular task
- **explanation**: background theory
- **reference**: API-documentation

Tools: Jupyter Notebook, ReadTheDocs, Sphinx

`effmass`: tutorial, background, reference



FINAL THOUGHTS

- Other relevant journals: www.codeisscience.com
- JOSS are always looking for new reviewers
- Possible ways to support each other? --> workshops, code review, mentoring.

slides and image credits at lucydot.github.io/slides

DISCUSSION QUESTIONS

- How can the research software community support work in the materials department?
- Which languages, software and tools do you use and why? Are there others you would like to learn/use?
- What are the main software-related challenges to your work?