

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

Arfon M. Smith^{*1}, Kyle E. Niemeyer², Daniel S. Katz³, Lorena A. Barba⁴, George Githinji⁵, Melissa Gymrek⁶, Kathryn D. Huff⁷, Christopher R. Madan⁸, Abigail Cabunoc Mayes⁹, Kevin M. Moerman¹⁰, Pjotr Prins¹¹, Karthik Ram¹², Ariel Rokem¹³, Tracy K. Teal¹⁴, Roman Valls Guimera¹⁵, and Jacob T. Vanderplas¹³

¹Data Science Mission Office, Space Telescope Science Institute, Baltimore, MD, USA

²School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA

³National 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, USA

⁴Department of Mechanical & Aerospace Engineering, George Washington University, Washington, DC, USA

⁵KEMRI–Wellcome Trust Research Programme, Kilifi, Kenya

⁶Departments of Medicine & Computer Science and Engineering, University of California, San Diego, CA, USA

⁷Department of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana–Champaign, Urbana, IL, USA

⁸School of Psychology, University of Nottingham, Nottingham, United Kingdom

⁹Mozilla Foundation, Toronto, Ontario, Canada

¹⁰MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA & Trinity Centre for Bioengineering, Trinity College, The University of Dublin, Dublin, Ireland

¹¹University of Tennessee Health Science Center, Memphis, TN, USA & University Medical Centre Utrecht, Utrecht, The Netherlands

¹²Berkeley Institute for Data Science, University of California, Berkeley, Berkeley, CA, USA

¹³eScience Institute, University of Washington, Seattle, WA, USA

¹⁴Data Carpentry, Davis, CA, USA

¹⁵University of Melbourne Centre for Cancer Research, University of Melbourne, Melbourne, Australia

January 2018

Abstract

1

2

3

4

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

*Corresponding author, arfon@stsci.edu

5 and providing a mechanism for research software developers to receive credit. While designed
6 to work within the current merit system of science, *JOSS* addresses the dearth of rewards
7 for key contributions to science made in the form of software. *JOSS* publishes articles that
8 encapsulate scholarship contained in the software itself, and its rigorous peer review targets
9 the software components: functionality, documentation, tests, continuous integration, and the
10 license. A *JOSS* article contains an abstract describing the purpose and functionality of the
11 software, references, and a link to the software archive. The article is the entry point of a
12 *JOSS* submission, which encompasses the full set of software artifacts. Submission and review
13 proceed in the open, on GitHub. Editors, reviewers, and authors work collaboratively and
14 openly. Unlike other journals, *JOSS* does not reject articles requiring major revision; while
15 not yet accepted, articles remain visible and under review until the authors make adequate
16 changes (or withdraw, if unable to meet requirements). Once an article is accepted, *JOSS*
17 gives it a digital object identifier (DOI), deposits its metadata in Crossref, and the article
18 can begin collecting citations on indexers like Google Scholar and other services. Authors
19 retain copyright of their *JOSS* article, releasing it under a Creative Commons Attribution 4.0
20 International License. In its first year, starting in May 2016, *JOSS* published 111 articles, with
21 more than 40 additional articles **currently** under review. *JOSS* is a sponsored project of the
22 nonprofit organization NumFOCUS and is an affiliate of the Open Source Initiative (OSI). ■

23 1 Introduction

24 Modern scientific research produces many outputs beyond traditional articles and books. Among
25 these, research software is critically important for a broad spectrum of fields. Current practices
26 for publishing and citation do not, however, acknowledge software as a first-class research output.
27 This deficiency means that researchers who develop software face critical career barriers. The
28 *Journal of Open Source Software* (*JOSS*) was founded in May 2016 to offer a solution within the
29 existing publishing mechanisms of science. It is a developer-friendly, free and open-access, peer-
30 reviewed journal for research software packages. **By its first anniversary, *JOSS* had published more
31 than a hundred articles.** This article discusses the motivation for creating a new software journal,
32 delineates the editorial and review process, and summarizes the journal’s first year of operation via
33 submission statistics. We expect this article to be of interest to three core audiences: (1) researchers
34 who develop software and could submit their work to *JOSS*, (2) those in the community with an
35 interest in advancing scholarly communications who may appreciate the technical details of the
36 *JOSS* journal framework, and (3) those interested in possibilities for citing software in their own
37 research publications.

38 The sixteen authors of this article are the members of the *JOSS* Editorial Board at the end of
39 its first year (May 2017). Arfon Smith is the founding editor-in-chief, and the founding editors are
40 Lorena A. Barba, Kathryn Huff, Daniel Katz, Christopher Madan, Abigail Cabunoc Mayes, Kevin
41 Moerman, Kyle Niemeyer, Karthik Ram, Tracy Teal, and Jake Vanderplas. Five new editors joined
42 in the first year to handle areas not well covered by the original editors, and to help manage the
43 large and growing number of submissions. They are George Githinji, Melissa Gymrek, Pjotr Prins,
44 Ariel Rokem, and Roman Valls Guimera. (Since then, we **have** added three more editors: Jason
45 Clark, Lindsey Heagy, and Thomas Leeper.) ■

46 The *JOSS* editors are firm supporters of open-source software for research, with extensive knowl-
47 edge of the practices and ethics of open source. This knowledge is reflected in the *JOSS* submission
48 system, peer-review process, and infrastructure. The journal offers a familiar environment for de-
49 velopers and authors to interact with reviewers and editors, leading to a citable published work:

50 a software article. The article describes the software at a high level, and the software itself in-
51 cludes both source code and associated artifacts such as tests, documentation, and examples. With
52 a Crossref digital object identifier (DOI), the article is able to collect citations, empowering the
53 developers/authors to gain career credit for their work. *JOSS* thus fills a pressing need for compu-
54 tational researchers to advance professionally, while promoting higher quality software for science.
55 *JOSS* also supports the broader open-science movement by encouraging researchers to share their
56 software openly and follow best practices in its development.

57 2 Background and motivation

58 A 2014 study of UK Russell Group Universities [1] reports that ~90% of academics surveyed said
59 they use software in their research, while more than 70% said their research would be impractical
60 without it. About half of these UK academics said they develop their own software while in
61 the course of doing research. Similarly, a 2017 survey of members of the US National Postdoctoral
62 Association found that 95% used research software, and 63% said their research would be impractical
63 without it [2].

64 Despite being a critical part of modern research, software lacks support across the scholarly
65 ecosystem for its publication, acknowledgement, and citation [3]. Academic publishing has not
66 changed substantially since its inception. Science, engineering, and many other academic fields
67 still view research articles as the key indicator of research productivity, with research grants being
68 another important indicator. Yet, the research article is inadequate to fully describe modern, data-
69 intensive, computational research. *JOSS* focuses on research software and its place in the scholarly
70 publishing ecosystem.

71 2.1 Why publish software?

72 Most academic fields still rely on a one-dimensional credit model where academic articles and their
73 associated citations are the dominant factor in the success of a researcher’s career. Software creators,
74 in order to increase the likelihood of receiving career credit for their work, often choose to publish
75 “software articles” that act as placeholder publications pointing to their software. At the same time,
76 recent years have seen a push for sharing open research software [4–9].

77 Beyond career-credit arguments for software creators, publishing research software enriches the
78 scholarly record. Buckheit and Donoho paraphrased Jon Claerbout, a pioneer of reproducible
79 research, as saying: “An article about a computational result is advertising, not scholarship. The
80 actual scholarship is the full software environment, code and data, that produced the result.” [10].
81 The argument that articles about computational science are not satisfactory descriptions of the
82 work, needing to be supplemented by code and data, is more than twenty years old! Yet, despite
83 the significance of software in modern research, documenting its use and including it in the scholarly
84 ecosystem presents numerous challenges.

85 2.2 Challenges of publishing software

86 The conventional publishing mechanism of science is the research article, and a researcher’s career
87 progression hinges on collecting citations for published works. Unfortunately, software citation [11]
88 is in its infancy (as is data citation [12, 13]). Publishing the software itself and receiving citation
89 credit for it may be a better long-term solution, but this is still impractical. Even when software

90 (and data) are published so that they can be cited, we do not have a standard culture of peer review
91 for them. This leads many developers today to publish software articles.

92 The developer’s next dilemma is where to publish, given the research content, novelty, length
93 and other features of a software article. Since 2012, Neil Chue Hong has maintained a growing list of
94 journals that accept software articles [14]. He includes both generalist journals, accepting software
95 articles from a variety of fields, and domain-specific journals, accepting both research and software
96 articles in a given field. For many journals, particularly the domain-specific ones, a software article
97 must include novel results to justify publication.

98 From the developer’s point of view, writing a software article can involve a great deal of extra
99 work. Good software includes documentation for both users and developers that is sufficient to
100 make it understandable. A software article may contain much of the same content, merely in a
101 different format, and developers may not find value in rewriting their documentation in a manner
102 less useful to their users and collaborators. These issues may lead developers to shun the idea of
103 software articles and prefer to publish the software itself. Yet, software citation is not common
104 and the mostly one-dimensional credit model of academia (based on article citations) means that
105 publishing software often does not “count” for career progression [3, 11].

106 3 The Journal of Open Source Software

107 To tackle the challenges mentioned above, the *Journal of Open Source Software (JOSS)* launched
108 in May 2016 [15] with the goal of drastically reducing the overhead of publishing software articles.
109 *JOSS* offers developers a venue to publish their complete research software wrapped in relatively
110 short high-level articles, thus enabling citation credit for their work. In this section we describe
111 the goals and principles, infrastructure, and business model of *JOSS*, and compare it with other
112 software journals.

113 3.1 Goals and principles

114 *JOSS* articles are deliberately short and only include an abstract describing the high-level func-
115 tionality of the software, a list of the authors of the software (with their affiliations), a list of key
116 references, and a link to the software archive and software repository. Articles are not allowed to
117 include other content often found in software articles, such as descriptions of the API (application
118 programming interface) and novel research results obtained using the software. The software API
119 should already be described in the software documentation, and domain research results do not
120 belong in *JOSS*—these should be published in a domain journal. Unlike most journals, which ease
121 discoverability of new research and findings, *JOSS* serves primarily as a mechanism for software
122 developers/authors to improve and publish their research software. Thus, software discovery is a
123 secondary feature.

124 The *JOSS* design and implementation are based on the following principles:

- 125 • Other than their short length, *JOSS* articles are conventional articles in every other sense: the
126 journal has an ISSN, articles receive Crossref DOIs with high-quality submission metadata,
127 and articles are appropriately archived.
- 128 • Because software articles are “advertising” and simply pointers to the *actual* scholarship (the
129 software), short abstract-length submissions are sufficient for these “advertisements.”

- 130 • Software is a core product of research and therefore the software itself should be archived
131 appropriately when submitted to and reviewed in *JOSS*.
- 132 • Code review, documentation, and contributing guidelines are important for open-source soft-
133 ware and should be part of any review. In *JOSS*, they are the focus of peer review. (While
134 a range of other journals publish software, with various peer-review processes, the focus of
135 the review is usually the submitted article and reviewers might not even look at the code.)
136 The *JOSS* review process itself, described in §4, was based on the on-boarding checklist for
137 projects joining the rOpenSci collaboration [16].

138 Acceptable *JOSS* submissions also need to meet the following criteria:

- 139 • The software must be open source by the Open Source Initiative (OSI) definition ([open-
140 source.org](https://opensource.org)).
- 141 • The software must have a research application.
- 142 • The submitter should be a major contributor to the software they are submitting.
- 143 • The software should be a significant new contribution to the available open-source software
144 that either enables some new research challenge(s) to be addressed or makes addressing re-
145 search challenges significantly better (e.g., faster, easier, [simpler](#)).
- 146 • The software should be feature-complete, i.e., it cannot be a partial solution.

147 3.2 How *JOSS* works

148 *JOSS* is designed as a small collection of open-source tools that leverage existing infrastructure
149 such as GitHub, Zenodo, and Figshare. A goal when building the journal was to minimize the
150 development of new tools where possible.

151 The *JOSS* web application and submission tool

152 The *JOSS* web application and submission tool is hosted at <http://joss.theoj.org>. It is a simple
153 Ruby on Rails web application [17] that lists accepted articles, provides the article submission
154 form (see Figure 1), and hosts journal documentation such as author submission guidelines. This
155 application also automatically creates the review issue on GitHub once a submission has been
156 pre-reviewed by an editor and accepted to start peer review in *JOSS*.

157 Open peer review on GitHub

158 *JOSS* conducts reviews on the `joss-reviews` GitHub repository [18]. Review of a submission
159 begins **by**with the opening of a new GitHub issue, where the editor-in-chief assigns an editor, the
160 editor assigns a reviewer, and interactions between authors, reviewer(s), and editor proceed in the
161 open. Figure 2 shows an example of a recent review for the (accepted) `hdbscan` package [19]. The
162 actual review includes the code, software functionality/performance claims, test suite (if present),
163 documentation, and any other material associated with the software.

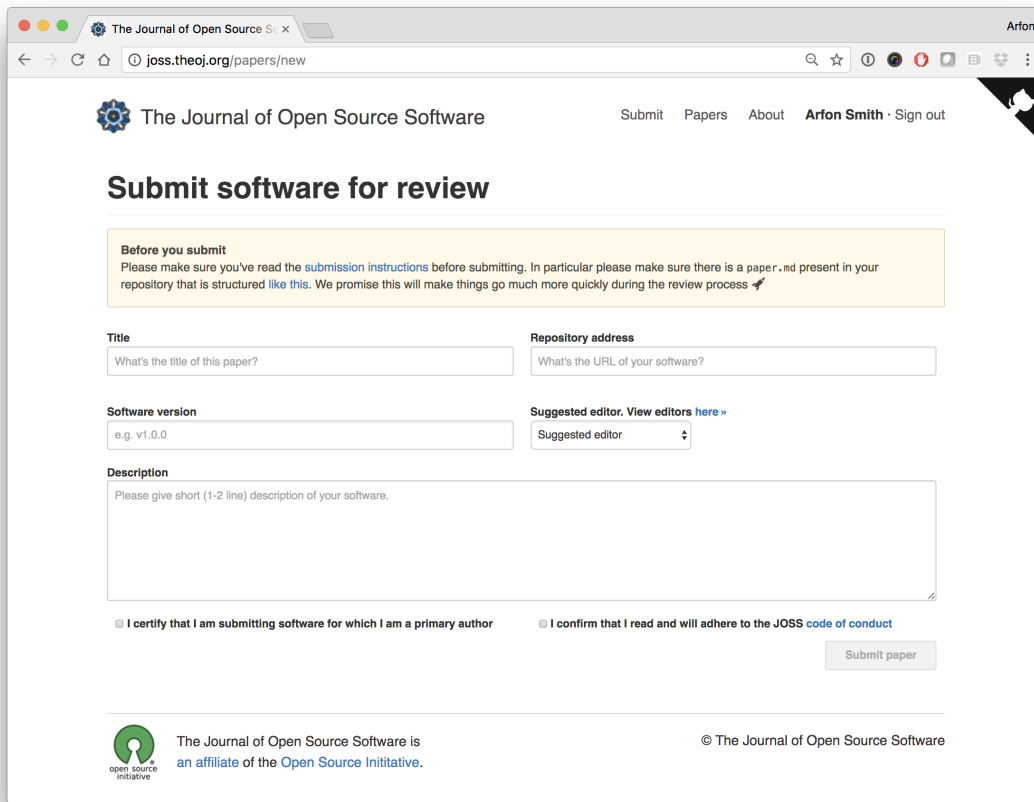


Figure 1: The *JOSS* submission page. A minimal amount of information is required for new submissions.

164 Whedon and the Whedon-API

165 Many of the tasks associated with *JOSS* reviews and editorial management are automated. A
166 core RubyGem library named *Whedon* [20] handles common tasks associated with managing the
167 submitted manuscript, such as compiling the article (from its Markdown source) and creating
168 Crossref metadata. An automated bot, *Whedon-API* [21], handles other parts of the review process
169 (such as assigning editors and reviewers based on editor input) and leverages the *Whedon* RubyGem
170 library. For example, to assign the editor for a submission, one may type the following command in
171 a comment box within the GitHub issue: `@whedon assign @danielskatz as editor`. Similarly,
172 to assign a reviewer, one enters: `@whedon assign @zhaozhang as reviewer` (where the reviewer
173 and editor GitHub handles identify them). The next section describes the review process in more
174 detail.

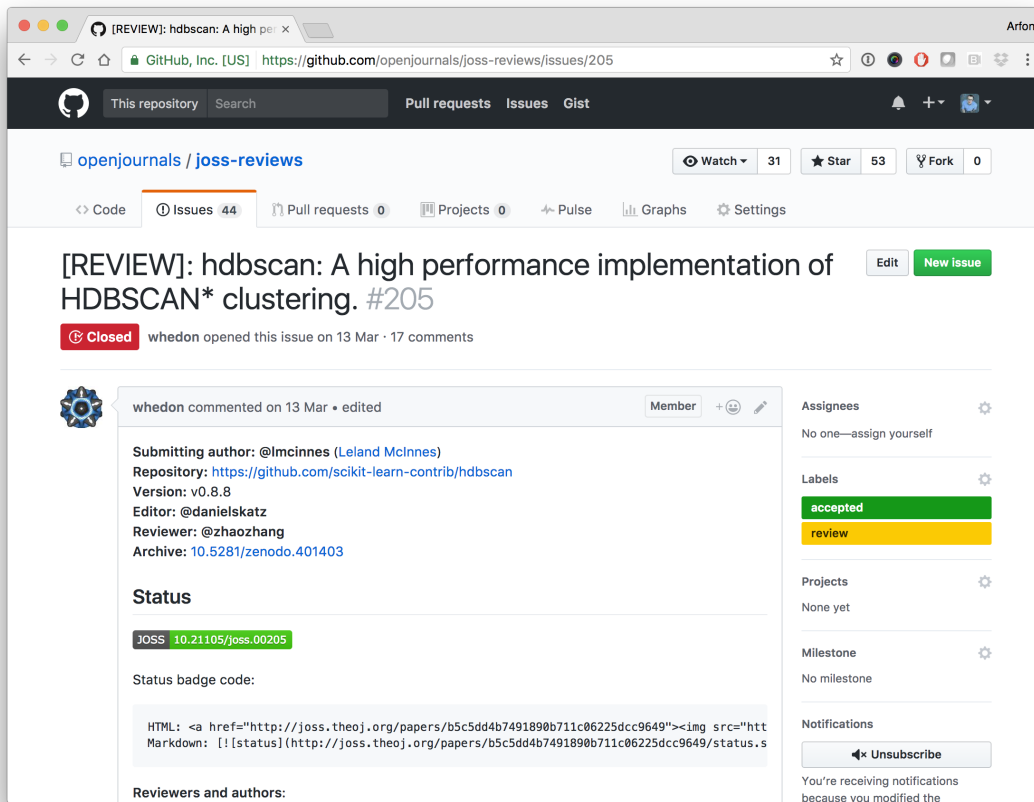


Figure 2: The hdbscan GitHub review issue.

175 3.3 Business model and content licensing

176 *JOSS* is designed to run at minimal cost with volunteer labor from editors and reviewers. The
 177 following fixed costs are currently incurred:

- 178 • Crossref membership: \$275. This is a yearly fixed cost for the *JOSS* parent entity—*Open*
 179 *Journals*—so that article DOIs can be registered with Crossref.
- 180 • Crossref article DOIs: \$1. This is a fixed cost per article.
- 181 • *JOSS* web application hosting (currently with Heroku): \$19 per month

Assuming a publication rate of 100 articles per year results in a core operating cost of ~\$6 per article. With 200 articles per year—which seems possible for the second year—the cost drops to

~\$3.50 per article:

$$(\$275 + (\$1 \times 100) + (\$19 \times 12))/100 = \$6.03 \quad (1)$$

$$(\$275 + (\$1 \times 200) + (\$19 \times 12))/200 = \$3.51 . \quad (2)$$

182 Submitting authors retain copyright of *JOSS* articles and accepted articles are published under
183 a Creative Commons Attribution 4.0 International License [22]. Any code snippets included
184 in *JOSS* articles are subject to the MIT license [23] regardless of the license of the submitted
185 software package under review, which itself must be licensed under an OSI-approved license (see
186 opensource.org/licenses/alphabetical for a complete list).

187 3.4 Comparison with other software journals

188 A good number of journals now accept, review, and publish software articles [14], which we
189 group into two categories. The first category of journals include those similar to *JOSS*, which
190 do not focus on a specific domain and only consider submissions of software/software articles: the
191 *Journal of Open Research Software* (*JORS*, openresearchsoftware.metajnl.com), *SoftwareX* (journals.elsevier.com/softwarex/), and now *JOSS*. Both *JORS* [24] and *SoftwareX* [25] now review
193 both the article text and the software. In *JOSS*, the review process focuses mainly on the software
194 and associated material (e.g., documentation) and less on the article text, which is intended to be
195 a brief description of the software. The role and form of peer review also varies across journals.
196 In *SoftwareX* and *JORS*, the goal of the review is both to decide if the article is acceptable for
197 publication and to improve it iteratively through a non-public, editor-mediated interaction between
198 the authors and the anonymous reviewers. In contrast, *JOSS* has the goal of accepting most arti-
199 cles after improving them as needed, with the reviewers and authors communicating directly and
200 publicly through GitHub issues.

201 The second category includes domain-specific journals that either accept software articles as a
202 special submission type or exclusively consider software articles targeted at the domain. For ex-
203 ample, *Collected Algorithms* (CALGO, acm.org/calgo) is a long-running venue for reviewing and
204 sharing mathematical algorithms associated with articles published in *Transactions on Mathematical
205 Software* and other ACM journals. However, CALGO authors must transfer copyright to
206 ACM and software is not available under an open-source license—this contrasts with *JOSS*, where
207 authors retain copyright and software must be shared under an open-source license. *Computer
208 Physics Communications* (journals.elsevier.com/computer-physics-communications) and *Geoscientific
209 Model Development* (geoscientific-model-development.net) publish full-length articles describ-
210 ing application software in computational physics and geoscience, respectively, where review pri-
211 marily focuses on the article. Chue Hong maintains a list of journals in both categories [14].

212 4 Peer review in *JOSS*

213 In this section, we illustrate the *JOSS* submission and review process using a representative example,
214 document the review criteria provided to authors and reviewers, and explain a fast-track option for
215 already-reviewed rOpenSci contributions.

216 4.1 The *JOSS* process

217 Figure 3 shows a typical *JOSS* submission and review process, described here in more detail using
218 the `hdbscan` package [19] as an example:

- 219 1. Leland McInnes submitted the `hdbscan` software and article to *JOSS* on 26 February 2017 using
220 the web application and submission tool. The article is a Markdown file named `paper.md`,
221 visibly located in the software repository (here, and in many cases, placed together with aux-
222 iliary files in a `paper` directory).
- 223 2. Following a routine check by a *JOSS* administrator, a “pre-review” issue was created in the
224 `joss-reviews` GitHub repository [26]. In this pre-review issue, an editor (Daniel S. Katz) was
225 assigned, who then identified and assigned a suitable reviewer (Zhao Zhang). Editors generally
226 identify one or more reviewers from a pool of volunteers based on provided programming
227 language and/or domain expertise.¹

228 The editor then asked the automated bot `Whedon` to create the main submission review issue
229 via the command `@whedon start review magic-word=bananas`. (“magic-word=bananas” is
230 a safeguard against accidentally creating a review issue prematurely.)

- 231 3. The reviewer then conducted the submission review [27] (see Figure 2) by working through
232 a checklist of review items, as described in §4.2. The author, reviewer, and editor discussed
233 any questions that arose during the review, and once the reviewer completed their checks,
234 they notified the submitting author and editor. Compared with traditional journals, *JOSS*
235 offers the unique feature of holding a discussion—in the open within a GitHub issue—between
236 the reviewer(s), author(s), and editor. Like a true conversation, discussion can go back and
237 forth in minutes or seconds, with all parties contributing at will. This contrasts with tradi-
238 tional journal reviews, where the process is merely an exchange between the reviewer(s) and
239 author(s), via the editor, which can take months for each communication, and in practice is
240 limited to one or two, perhaps three in some cases, exchanges due to that delay [28].

241 Note that *JOSS* reviews are subject to a code of conduct [29], adopted from the Contributor
242 Covenant Code of Conduct [30]. Both authors and reviewers must confirm that they have
243 read and will adhere to this Code of Conduct, during submission and with their review,
244 respectively.

- 245 4. After the review was complete, the editor asked the submitting author to make a perma-
246 nent archive of the software (including any changes made during review) with a service such
247 as Zenodo or Figshare, and to post a link to the archive in the review thread. This link,
248 in the form of a DOI, was associated with the submission via the command `@whedon set`
249 `10.5281/zenodo.401403 as archive`.

- 250 5. The editor-in-chief used the `Whedon` RubyGem library on his local machine to produce the
251 compiled PDF, update the *JOSS* website, deposit Crossref metadata, and issue a DOI for the
252 submission (`10.21105/joss.00205`).

- 253 6. Finally, the editor-in-chief updated the review issue with the *JOSS* article DOI and closed
254 the review. The submission was then accepted into the journal.

¹Potential reviewers can volunteer via <http://joss.theoj.org/reviewer-signup.html>

255 Authors can also first submit a pre-submission inquiry via an issue in the main *JOSS* repository
 256 [17] if they have questions regarding the suitability of their software for publication, or for any
 257 other questions.

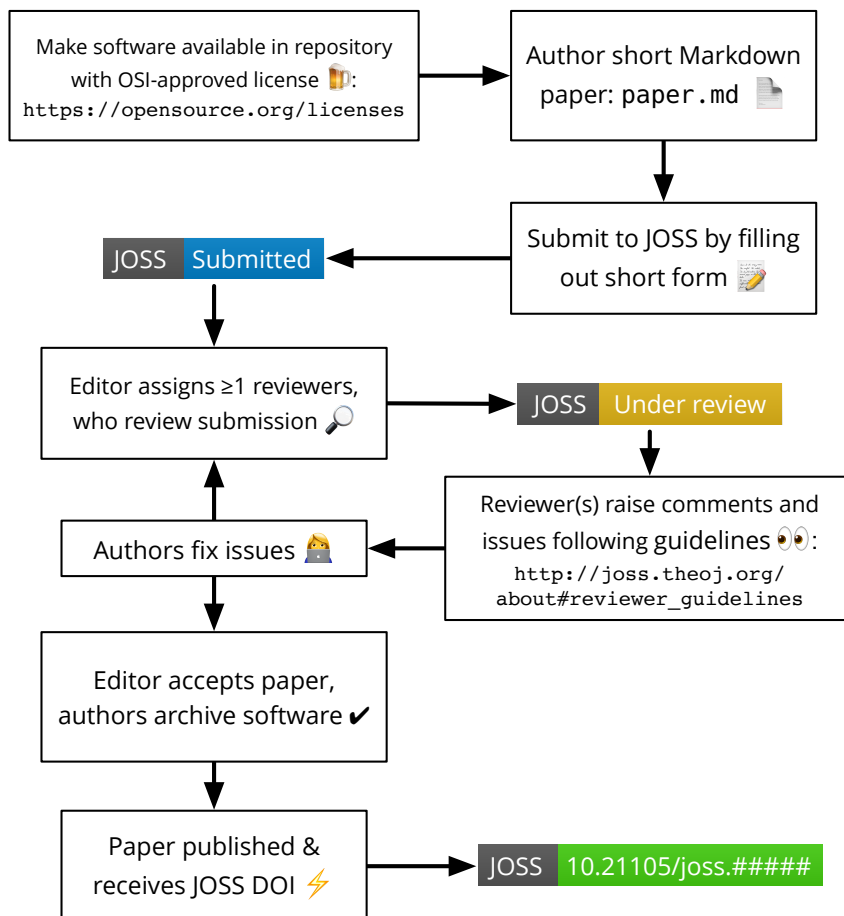


Figure 3: The *JOSS* submission and review flow including the various status badges that can be embedded on third-party settings such as GitHub README documentation [31].

258 4.2 *JOSS* review criteria

259 As previously mentioned, the *JOSS* review is primarily concerned with the material in the software
 260 repository, focusing on the software and documentation. We do not ask authors to use their software
 261 in a research study or include research results in their article beyond as examples; submissions
 262 focused on results rather than software should be submitted to research journals. The specific
 263 items in the reviewer checklist are:

- 264 • Conflict of interest

- 265 – As the reviewer I confirm that I have read the *JOSS* [conflict of interest policy](#) and that
266 there are no conflicts of interest for me to review this work.
- 267 • Code of Conduct
- 268 – I confirm that I read and will adhere to the *JOSS* [code of conduct](#).
- 269 • General checks
- 270 – **Repository:** Is the source code for this software available at the repository [URL](#)?
- 271 – **License:** Does the repository contain a plain-text LICENSE file with the contents of an
272 OSI-approved software license?
- 273 – **Version:** Does the release version given match the GitHub release?
- 274 – **Authorship:** Has the submitting author made major contributions to the software?
- 275 • Functionality
- 276 – **Installation:** Does installation proceed as outlined in the documentation?
- 277 – **Functionality:** Have the functional claims of the software been confirmed?
- 278 – **Performance:** Have any performance claims of the software been confirmed?
- 279 • Documentation
- 280 – **A statement of need:** Do the authors clearly state what problems the software is
281 designed to solve and who the target audience is?
- 282 – **Installation instructions:** Is there a clearly-stated list of dependencies? Ideally these
283 should be handled with an automated package management solution.
- 284 – **Example usage:** Do the authors include examples of how to use the software (ideally
285 to solve real-world analysis problems)?
- 286 – **Functionality documentation:** Is the core functionality of the software documented
287 to a satisfactory level (e.g., API method documentation)?
- 288 – **Automated tests:** Are there automated tests or manual steps described so that the
289 function of the software can be verified?
- 290 – **Community guidelines:** Are there clear guidelines for third parties wishing to 1)
291 contribute to the software, 2) report issues or problems with the software, and 3) seek
292 support?
- 293 • Software paper
- 294 – **Authors:** Does the `paper.md` file include a list of authors with their affiliations?
- 295 – **A statement of need:** Do the authors clearly state what problems the software is
296 designed to solve and who the target audience is?
- 297 – **References:** Do all archival references that should have a DOI list one (e.g., papers,
298 datasets, software)?

299 4.3 Fast track for reviewed rOpenSci contributions

300 For submissions of software that has already been reviewed under rOpenSci’s rigorous onboarding
301 guidelines [32, 33], *JOSS* does not perform further review. The editor-in-chief is alerted with a
302 note “This submission has been accepted to rOpenSci. The review thread can be found at [LINK
303 TO ONBOARDING ISSUE],” allowing such submissions to be fast-tracked to acceptance.

304 5 A review of the first year

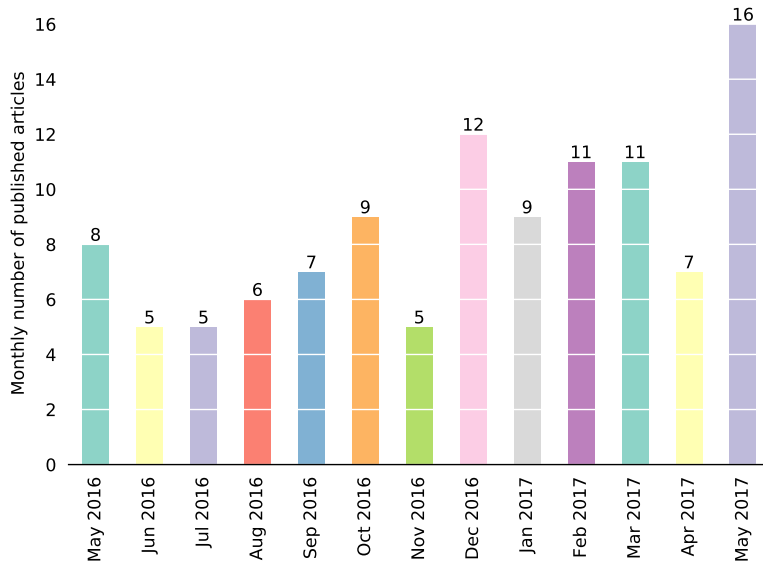
305 By the end of May 2017, *JOSS* published 111 articles since its inception in May 2016, and had an
306 additional 41 articles under consideration. Figure 4 shows the monthly and cumulative publication
307 rates; on average, we published 8.5 articles per month, with some (nonstatistical) growth over time.

308 Figure 5 shows the numbers of days taken for processing and review of the 111 published articles
309 (i.e., time between submission and publication), including finding a topic editor and reviewer(s).
310 Since the journal’s inception in May 2016, articles spent on average 45.5 days between submission
311 and publication (median 32 days, interquartile range 52.3 days) The shortest review took a single
312 day, for `Application Skeleton` [35], while the longest review took 190 days, for `walkr` [36]. In
313 the former case, the rapid turnaround can be attributed to the relatively minor revisions needed
314 (in addition to quick editor, reviewer, and author actions and responses). In contrast, the latter
315 case took much longer due to delays in selecting an editor and finding an appropriate reviewer, and
316 a multimonth delay between selecting a reviewer and receiving reviews. In other cases with long
317 review periods, some delays in responding to requests for updates may be attributed to reviewers
318 (or editors) missing GitHub notifications from the review issue comments. We have already taken
319 steps to improve the ability of authors, reviewers, and editors to keep track of their submissions,
320 including a prompt to new reviewers to unsubscribe from the main `joss-reviews` repository [18]
321 (to reduce unnecessary notifications) and a weekly digest email for *JOSS* editors to keep track of
322 their submissions. In the future we may collect the email addresses of reviewers so we can extend
323 this functionality to them.

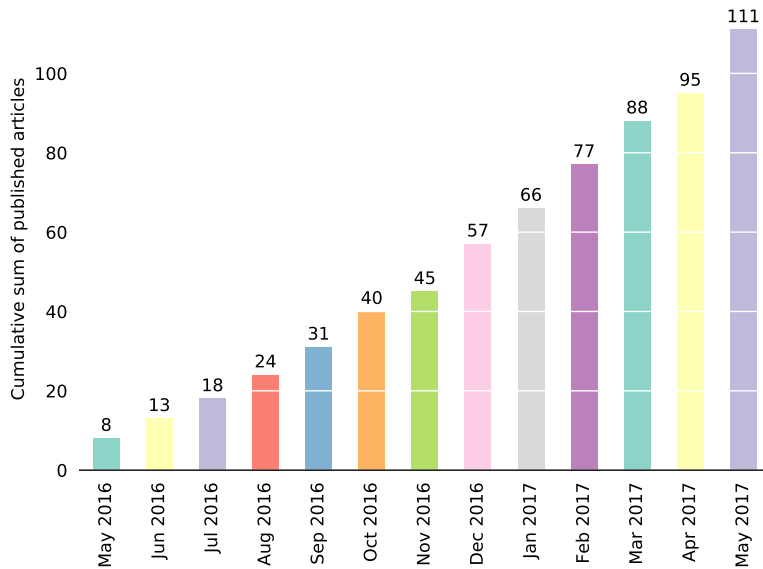
324 Figure 6 shows the frequency of programming languages appearing in *JOSS* articles. Python
325 appears the most with over half of published software articles (54), while R is used in nearly one-
326 third of articles (29). We believe the popularity of Python and R in *JOSS* submissions is the
327 result of (1) the adoption of these languages (and open-source practices) in scientific computing
328 communities and (2) our relationship with the rOpenSci project.

329 Each article considered by *JOSS* undergoes review by one or more reviewers. The set of 111
330 published articles have been reviewed by 93 unique reviewers. The majority of articles received
331 a review by one reviewer (average of 1.11 ± 0.34), with a maximum of three reviewers. Based on
332 available data in the review issues, on average, editors reached out to 1.85 ± 1.40 potential reviewers
333 (at most 8 in one case) via mentions in the GitHub review issue. This does not include external
334 communication, e.g., via email or Twitter. Overall, *JOSS* editors contacted 1.65 potential reviewers
335 for each actual review (based on means).

336 Interestingly, the current reviewer list contains only 52 entries, as of this writing [37]. Consid-
337 ering the unique reviewer count of 93, we clearly have reached beyond those that volunteered to
338 review a priori. Benefits of using GitHub’s issue infrastructure and our open reviews include: 1)
339 the ability to tag multiple people, via their GitHub handles, to invite them as potential reviewers;
340 2) the discoverability of the work so that people may volunteer to review without being formally
341 contacted; 3) the ability to get additional, unprompted feedback and comments; and 4) the ability



(a) Numbers of articles published per month.



(b) Cumulative sum of numbers of articles published per month.

Figure 4: Statistics of articles published in *JOSS* since its inception in May 2016 through May 2017. Data, plotting script, and figure files are available [34].

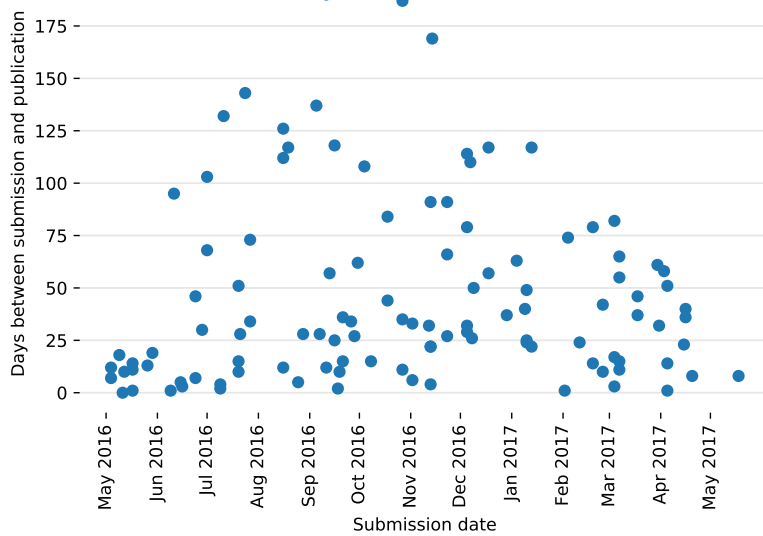


Figure 5: Days between submission and publication dates of the 111 articles *JOSS* has published, between May 2016–May 2017. Data, plotting script, and figure file are available [34].

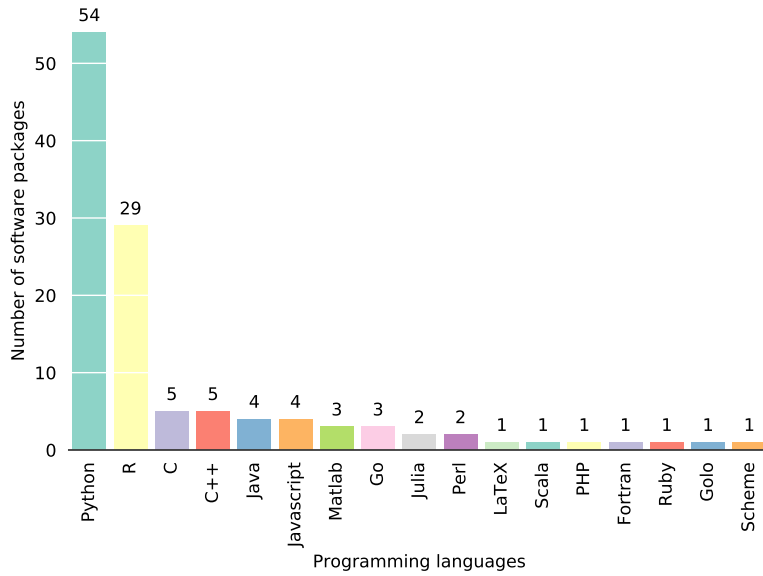


Figure 6: Frequency of programming languages from the software packages described by the 111 articles *JOSS* published in its first year. Total sums to greater than 111, because some packages are multi-language. Data, plotting script, and figure file are available [34].

342 to find reviewers by openly advertising, e.g., on social media. Furthermore, GitHub is a well-known,
343 commonly used platform where many (if not most) potential authors and reviewers already have
344 accounts.

345 Figure 7 shows the numbers of articles managed by each of the *JOSS* editors. Editor-in-chief
346 Arfon Smith stewarded the majority of articles published in the first year. This was somewhat
347 unavoidable in the first three months after launch, as Smith served as the de facto sole editor for all
348 submissions, with other members of the editorial board assisting. This strategy was not sustainable
349 and, over time, we adopted the pre-review/review procedure to hand off articles to editors. Also,
350 authors can now select during submission the appropriate editor based on article topic.

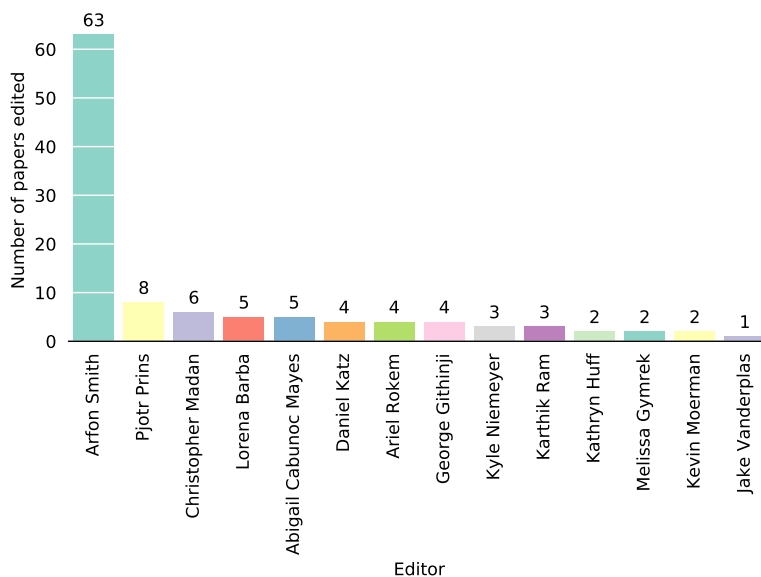


Figure 7: Numbers of articles handled by each of the *JOSS* editors. Data, plotting script, and figure file are available [34].

351 Lastly, we analyzed the affiliations of the 286 authors associated with articles published in the
352 first year. Figure 8 shows the number of authors per country; we represented authors with multiple
353 affiliations in different countries using their first affiliation. Authors with no affiliation, or where
354 we could not identify the country, are shown as “unknown.” From the articles published in the
355 first year, approximately 48% of authors live in the United States and approximately 40% live in
356 Europe (including Switzerland). The remaining 12% come from the rest of the world, most notably
357 Australia (6.6%) and Canada (2.1%). Moving forward, we hope to receive submissions from authors
358 in more countries that even better represent who develops research software around the world; one
359 strategy to achieve this involves continuing to expand our editorial board.

360 In its first year, *JOSS* also developed formal relationships with two US-based nonprofit or-
361 ganizations. In March 2017, *JOSS* became a community affiliate of the Open Source Initiative
362 (opensource.org), the steward of the open-source definition, which promotes open-source software
363 and educates about appropriate software licenses. And, in April 2017, *JOSS* became a fiscally
364 sponsored project of NumFOCUS (numfocus.org), a 501(c)(3) charity that supports and promotes

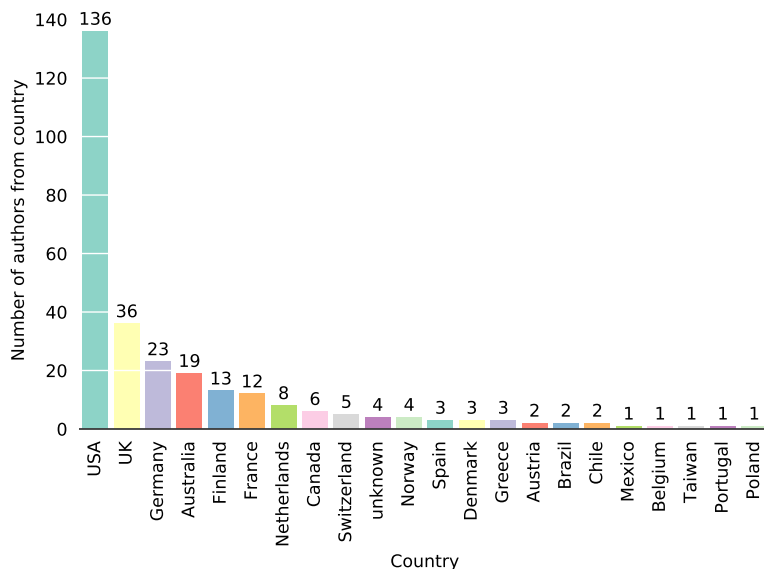


Figure 8: Numbers of authors from a particular country. Data, plotting script, and figure file are available [34].

365 “world-class, innovative, open source scientific computing.” Being associated with these two promi-
 366 nent community organizations increases the trust of the community in our efforts. Furthermore, as
 367 a NumFOCUS project, *JOSS* will be able to raise funding to sustain its activities and grow.

368 6 The second year for *JOSS*

369 Our focus for the [second](#) year will be on continuing to provide a high-quality experience for sub-
 370 mitting authors and reviewers, and making the best use of the editorial board. In our first year, we
 371 progressed from a model where the editor-in-chief handled most central functions to one with more
 372 distributed roles for the editors, particularly that of ensuring that reviews are useful and timely.
 373 Editors can now select and self-assign to submissions they want to manage, while the editor-in-chief
 374 only assigns the remaining submissions. As *JOSS* grows, the process of distributing functions across
 375 the editorial board will continue to evolve—and more editors may be needed.

376 In the [second](#) year, we plan to complete a number of high-priority improvements to the *JOSS*
 377 toolchain. Specifically, we plan on automating the final steps for accepting an article. For ex-
 378 ample, generating Crossref metadata and compiling the article are both currently handled by the
 379 editor-in-chief on his local machine using the `Whedon` RubyGem library. In the future, we would
 380 like authors and reviewers to be able to ask the `Whedon-API` bot to compile the paper for them,
 381 and other editors should be able to ask the bot to complete the submission of Crossref metadata on
 382 their behalf. Other improvements are constantly under discussion on the *JOSS* GitHub repository
 383 (github.com/openjournals/joss/issues). In fact, anyone is able to report bugs and suggest enhance-
 384 ments to the experience. And, since the *JOSS* tools are open source, we welcome contributions in

385 the form of bug-fixes or enhancements via the usual pull-request protocols.

386 Beyond roles and responsibilities for the editors, and improvements to the *JOSS* tools and
387 infrastructure, we will take on the more tricky questions about publishing software, such as how to
388 handle new software versions. Unlike traditional research articles that remain static once published,
389 software usually changes over time, at least for maintenance and to avoid software rot/collapse
390 (where software stops working because of changes in the environment, such as dependencies on
391 libraries or operating system). Furthermore, because all potential uses of the software are not
392 known at the start of a project, the need or opportunity arises to add features, improve performance,
393 improve accuracy, etc. After making one or more changes, software developers frequently update
394 the software with a new version number. Over time, the culmination of these changes may result in
395 a major update to the software, and with many new contributors a new version might correspond
396 to a new set of authors if the software is published. However, this process may not translate clearly
397 to *JOSS*. The editorial board will accept a new *JOSS* article published with each major version or
398 even a minor version if the changes seem significant enough to the editor and reviewer(s), but we
399 do not yet know if this will satisfy the needs of both developers and users (corresponding to *JOSS*
400 authors and readers, respectively).

401 The discussion about new software versions also generally applies to software forks, where soft-
402 ware is copied and, after some divergent development, a new software package emerges. Similar
403 to how we handle new software versions, the *JOSS* editorial board will consider publication of an
404 article describing a forked version of software if it includes substantial changes from a previously
405 published version. Authorship questions may be more challenging when dealing with forks com-
406 pared with new versions, since forks can retain varying amounts of code from the original projects.
407 However, while a version control history generally makes it easy to suggest people who should be
408 authors, deciding on authorship can be difficult and subjective, and is therefore ultimately project-
409 dependent. We prefer to leave authorship decisions to the projects, with discussion taking place as
410 needed with reviewers and editors.

411 7 Conclusions

412 Software today encapsulates—and generates—important research knowledge, yet it has not entered
413 the science publication ecosystem in a practical way. This situation is costly for science, through the
414 lack of career progression for valuable personnel: research software developers. We founded *JOSS*
415 in response to the acute need for an answer to this predicament. *JOSS* is a venue for authors who
416 wish to receive constructive peer feedback, publish, and collect citations for their research software.
417 By encouraging researchers to develop their software following best practices, and then share and
418 publish it openly, *JOSS* supports the broader open-science movement. The number of submissions
419 confirms the keen demand for this publishing mechanism: more than 100 accepted articles in the
420 first year and more than 40 others under review. By the end of 2017, *JOSS* has published nearly
421 200 articles. Community members have also responded positively when asked to review submissions
422 in an open and non-traditional format, contributing useful reviews of the submitted software.

423 However, we are still overcoming initial hurdles to achieve our goals. *JOSS* is currently not fully
424 indexed by Google Scholar, despite the fact that *JOSS* articles include adequate metadata and that
425 we made an explicit request for inclusion in March 2017 (see GitHub [issue #130](#)). Also, we may
426 need to invest more effort into raising awareness of good practices for citing *JOSS* articles. That
427 said, we have some preliminary citation statistics: according to Google Scholar, `corner.py` [38] and
428 `Armadillo` [39] have been cited the most at 116 and 79 times, respectively. Crossref’s Cited-by

429 service—which relies on publishers depositing reference information—reports 45 and 28 citations for
430 the same articles [40]. While most other articles have received no citations to-date, a few have been
431 cited between one and five times. We have had at least two “repeat” submissions, i.e., submissions of
432 a new version with major changes from a prior version. Clementi et al. [41] published PyGBe-LSPR,
433 a new version that added substantially new features over the original PyGBe of Cooper et al. [42].
434 Similarly, the software published by Sandersen and Curtin [43] extended on (and cited) their earlier
435 article [44].

436 The journal cemented its position in the first year of operation, building trust within the com-
437 munity of open-source research-software developers and growing in name recognition. It also earned
438 weighty affiliations with OSI and NumFOCUS, the latter bringing the opportunity to raise funding
439 for sustained operations. Although publishing costs are low at \$3–6 per article, *JOSS* does need
440 funding, with the editor-in-chief having borne the expenses personally to pull off the journal launch.
441 Incorporating a small article charge (waived upon request) may be a route to allow authors to con-
442 tribute to *JOSS* in the future, but we have not yet decided on this change. Under the NumFOCUS
443 nonprofit umbrella, *JOSS* is now eligible to seek grants for sustaining its future, engaging in new
444 efforts like outreach, and improving its infrastructure and tooling.

445 Outreach to other communities still unaware of *JOSS* is certainly part of our growth strategy.
446 Awareness of the journal so far has mostly spread through word-of-mouth and social networking [45,
447 46], plus a couple of news articles [47, 48]. As of August 2017, *JOSS* is also listed in the Directory
448 of Open Access Journals (DOAJ) (doaj.org/toc/2475-9066). We plan to present *JOSS* at relevant
449 domain conferences, like we did at the 2017 SIAM Conference on Computational Science & Engi-
450 neering [49] and the 16th Annual Scientific Computing with Python Conference (SciPy 2017). We
451 are also interested in partnering with other domain journals that focus on (traditional) research
452 articles. In such partnerships, traditional peer review of the research would be paired with peer
453 review of the software, with *JOSS* taking responsibility for the latter.

454 Finally, the infrastructure and tooling of *JOSS* have unexpectedly added values: while developed
455 to support and streamline the *JOSS* publication process, these open-source tools generalize to
456 a lightweight journal-management system. The *JOSS* web application and submission tool, the
457 Whedon RubyGem library, and the Whedon-API bot could be easily forked to create overlay journals
458 for other content types (data sets, posters, figures, etc.). The original artifacts could be archived on
459 other services such as Figshare, Zenodo, Dryad, arXiv, or engrXiv/AgriXiv/LawArXiv/PsyArXiv/
460 SocArXiv/bioRxiv. This presents manifold opportunities to expand the ways we assign career
461 credit to the digital artifacts of research. *JOSS* was born to answer the needs of research software
462 developers to thrive in the current merit traditions of science, but we may have come upon a
463 generalizable formula for digital science.

464 Acknowledgements

465 This work was supported in part by the Alfred P. Sloan Foundation. Work by K. E. Niemeyer
466 was supported in part by the National Science Foundation (No. ACI-1535065). Work by P. Prins
467 was supported by the National Institute of Health (R01 GM123489, 2017–2022). Work by K. Ram
468 was supported in part by The Leona M. and Harry B. Helmsley Charitable Trust (No. 2016PG-
469 BRI004). Work by A. Rokem was supported by the Gordon & Betty Moore Foundation and
470 the Alfred P. Sloan Foundation, and by grants from the Bill & Melinda Gates Foundation, the
471 National Science Foundation (No. 1550224), and the National Institute of Mental Health (No.
472 1R25MH112480).

473 References

- 474 [1] S. Hettrick, M. Antonioletti, L. Carr, N. Chue Hong, S. Crouch, D. De Roure, I. Emsley,
475 C. Goble, A. Hay, D. Inupakutika, M. Jackson, A. Nenadic, T. Parkinson, M. I. Parsons, A.
476 Pawlik, G. Peru, A. Proeme, J. Robinson, and S. Sufi. *UK Research Software Survey 2014*
477 *[dataset]*. University of Edinburgh on behalf of Software Sustainability Institute. 2015. DOI:
478 [10.7488/ds/253](https://doi.org/10.7488/ds/253).
- 479 [2] U. Nangia and D. S. Katz. “Track 1 Paper: Surveying the U.S. National Postdoctoral As-
480 sociation Regarding Software Use and Training in Research”. In: *Workshop on Sustainable*
481 *Software for Science: Practice and Experiences (WSSSPE 5.1)*. (Manchester, UK, Sept. 6,
482 2017). 2017. DOI: [10.6084/m9.figshare.5328442](https://doi.org/10.6084/m9.figshare.5328442).
- 483 [3] K. E. Niemeyer, A. M. Smith, and D. S. Katz. “The challenge and promise of software citation
484 for credit, identification, discovery, and reuse”. In: *Journal of Data and Information Quality*
485 7.4 (2016), p. 16. DOI: [10.1145/2968452](https://doi.org/10.1145/2968452).
- 486 [4] N. Barnes. “Publish your computer code: it is good enough”. In: *Nature* 467 (Oct. 2010),
487 p. 753. DOI: [10.1038/467753a](https://doi.org/10.1038/467753a).
- 488 [5] P. Vandewalle. “Code Sharing Is Associated with Research Impact in Image Processing”. In:
489 *Comput. Sci. Eng.* 14.4 (June 2012), pp. 42–47. DOI: [10.1109/MCSE.2012.63](https://doi.org/10.1109/MCSE.2012.63).
- 490 [6] A. Morin, J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, and P. Sliz. “Shining Light
491 into Black Boxes”. In: *Science* 336.6078 (Apr. 2012), pp. 159–160. DOI: [10.1126/science.](https://doi.org/10.1126/science.1218263)
492 [1218263](https://doi.org/10.1126/science.1218263).
- 493 [7] D. C. Ince, L. Hatton, and J. Graham-Cumming. “The case for open computer programs”. In:
494 *Nature* 482.7386 (Feb. 2012), pp. 485–488. DOI: [10.1038/nature10836](https://doi.org/10.1038/nature10836).
- 495 [8] Nature Methods Editorial Board. “Software with impact”. In: *Nature Methods* 11.3 (Mar.
496 2014), pp. 211–211. DOI: [10.1038/nmeth.2880](https://doi.org/10.1038/nmeth.2880).
- 497 [9] P. Prins, J. de Ligt, A. Tarasov, R. C. Jansen, E. Cuppen, and P. E. Bourne. “Toward
498 effective software solutions for big biology”. In: *Nat Biotech* 33.7 (2015), pp. 686–687. DOI:
499 [10.1038/nbt.3240](https://doi.org/10.1038/nbt.3240).
- 500 [10] J. B. Buckheit and D. L. Donoho. “WaveLab and Reproducible Research”. In: *Wavelets and*
501 *Statistics*. Ed. by A. Antoniadis and G. Oppenheim. New York, NY: Springer New York, 1995,
502 pp. 55–81. ISBN: 978-1-4612-2544-7. DOI: [10.1007/978-1-4612-2544-7_5](https://doi.org/10.1007/978-1-4612-2544-7_5).
- 503 [11] A. M. Smith, D. S. Katz, K. E. Niemeyer, and FORCE11 Software Citation Working Group.
504 “Software citation principles”. In: *PeerJ Computer Science* 2 (Sept. 2016), e86. DOI: [10.7717/](https://doi.org/10.7717/peerj-cs.86)
505 [peerj-cs.86](https://doi.org/10.7717/peerj-cs.86).
- 506 [12] FORCE11 Data Citation Synthesis Group. *Joint Declaration of Data Citation Principles*.
507 Ed. by M. Martone. 2014. URL: [https://www.force11.org/group/joint-declaration-](https://www.force11.org/group/joint-declaration-data-citation-principles-final)
508 [data-citation-principles-final](https://www.force11.org/group/joint-declaration-data-citation-principles-final) (visited on 06/21/2016).
- 509 [13] J. Starr, E. Castro, M. Crosas, M. Dumontier, R. R. Downs, R. Duerr, L. L. Haak, M.
510 Haendel, I. Herman, S. Hodson, J. Hourclé, J. E. Kratz, J. Lin, L. H. Nielsen, A. Nurnberger,
511 S. Proell, A. Rauber, S. Sacchi, A. Smith, M. Taylor, and T. Clark. “Achieving human and
512 machine accessibility of cited data in scholarly publications”. In: *PeerJ Computer Science* 1
513 (May 2015), e1. DOI: [10.7717/peerj-cs.1](https://doi.org/10.7717/peerj-cs.1).

- 514 [14] N. Chue Hong. *In which journals should I publish my software?* Software Sustainability In-
515 stitute. URL: [https://www.software.ac.uk/which-journals-should-i-publish-my-](https://www.software.ac.uk/which-journals-should-i-publish-my-software)
516 [software](https://www.software.ac.uk/which-journals-should-i-publish-my-software) (visited on 06/21/2016).
- 517 [15] A. M. Smith. *Announcing The Journal of Open Source Software*. 2016. URL: [http://web.](http://web.archive.org/web/20170415195544/http://www.arfon.org/announcing-the-journal-of-open-source-software)
518 [archive.org/web/20170415195544/http://www.arfon.org/announcing-the-journal-](http://web.archive.org/web/20170415195544/http://www.arfon.org/announcing-the-journal-of-open-source-software)
519 [of-open-source-software](http://web.archive.org/web/20170415195544/http://www.arfon.org/announcing-the-journal-of-open-source-software) (visited on 06/21/2016).
- 520 [16] C. Boettiger, S. Chamberlain, E. Hart, and K. Ram. “Building Software, Building Community:
521 Lessons from the rOpenSci Project”. In: *Journal of Open Research Software* 3.1 (2015), e8.
522 DOI: [10.5334/jors.bu](https://doi.org/10.5334/jors.bu).
- 523 [17] *JOSS web application*. 2016. URL: <https://github.com/openjournals/joss> (visited on
524 06/21/2016).
- 525 [18] *JOSS reviews*. 2016. URL: <https://github.com/openjournals/joss-reviews> (visited on
526 06/21/2016).
- 527 [19] L. McInnes, J. Healy, and S. Astels. “hdbscan: Hierarchical density based clustering”. In: *The*
528 *Journal of Open Source Software* 2.11 (Mar. 2017). DOI: [10.21105/joss.00205](https://doi.org/10.21105/joss.00205).
- 529 [20] A. M. Smith. *Whedon*. 2016. URL: <https://github.com/openjournals/whedon> (visited on
530 06/21/2016).
- 531 [21] A. M. Smith. *Whedon API*. 2016. URL: <https://github.com/openjournals/whedon-api>
532 (visited on 06/21/2016).
- 533 [22] Creative Commons. *Creative Commons Attribution 4.0 International*. Version 4.0. URL:
534 <https://creativecommons.org/licenses/by/4.0/legalcode> (visited on 06/21/2016).
- 535 [23] Open Source Initiative. *MIT License*. URL: <https://opensource.org/licenses/MIT> (visited
536 on 06/21/2016).
- 537 [24] Journal of Open Research Software. *Editorial Policies: Peer Review Process*. URL: [https://](https://openresearchsoftware.metajnl.com/about/editorialPolicies/)
538 openresearchsoftware.metajnl.com/about/editorialPolicies/ (visited on
539 07/02/2017).
- 540 [25] Elsevier. *Original Software Publications*. URL: [https://www.elsevier.com/authors/](https://www.elsevier.com/authors/author-services/research-elements/software-articles/original-software-publications)
541 [author-services/research-elements/software-articles/original-software-](https://www.elsevier.com/authors/author-services/research-elements/software-articles/original-software-publications)
542 [publications](https://www.elsevier.com/authors/author-services/research-elements/software-articles/original-software-publications) (visited on 07/02/2017).
- 543 [26] *hdbscan JOSS pre-review*. 2016. URL: [https://github.com/openjournals/joss-reviews/](https://github.com/openjournals/joss-reviews/issues/191)
544 [issues/191](https://github.com/openjournals/joss-reviews/issues/191) (visited on 06/21/2016).
- 545 [27] *hdbscan JOSS review*. 2016. URL: [https://github.com/openjournals/joss-reviews/](https://github.com/openjournals/joss-reviews/issues/205)
546 [issues/205](https://github.com/openjournals/joss-reviews/issues/205) (visited on 06/21/2016).
- 547 [28] J. P. Tennant, J. M. Dugan, D. Graziotin, D. C. Jacques, F. Waldner, D. Mietchen, Y.
548 Elkhatib, L. B. Collister, C. K. Pikas, T. Crick, P. Masuzzo, A. Caravaggi, D. R. Berg, K. E.
549 Niemeyer, T. Ross-Hellauer, S. Mannheimer, L. Rigling, D. S. Katz, B. Greshake, J. Pacheco-
550 Mendoza, N. Fatima, M. Poblet, M. Isaakidis, D. E. Irawan, S. Renaut, C. R. Madan, L.
551 Matthias, J. N. Kjær, D. P. O’Donnell, C. Neylon, S. Kearns, M. Selvaraju, and J. Colomb.
552 “A multi-disciplinary perspective on emergent and future innovations in peer review [version 3;
553 referees: 2 approved]”. In: *F1000Research* 6 (2017), p. 1151. DOI: [10.12688/f1000research.](https://doi.org/10.12688/f1000research.12037.3)
554 [12037.3](https://doi.org/10.12688/f1000research.12037.3).

- 555 [29] *JOSS Code of Conduct*. Git commit dc5f7ebe609e1fe900488efa1012cc27966ca129. 2017.
556 URL: https://github.com/openjournals/joss/blob/master/CODE_OF_CONDUCT.md
557 (visited on 06/21/2016).
- 558 [30] C. A. Ehmke. *Contributor Covenant Code of Conduct*. Version 1.4. 2016. URL: [http://](http://contributor-covenant.org/version/1/4/)
559 contributor-covenant.org/version/1/4/ (visited on 06/21/2016).
- 560 [31] K. E. Niemeyer. *JOSS publication flowchart*. Figshare. June 2017. DOI: [10.6084/m9.](https://doi.org/10.6084/m9.figshare.5147773.v1)
561 [figshare.5147773.v1](https://doi.org/10.6084/m9.figshare.5147773.v1).
- 562 [32] K. Ram, N. Ross, and S. Chamberlain. “A model for peer review and onboarding research
563 software”. In: *Proceedings of the Fourth Workshop on Sustainable Software for Science: Prac-*
564 *tice and Experiences (WSSSPE4)*. (Manchester, UK, Sept. 14, 2016). Ed. by G. Allen, J.
565 Carver, S.-C. T. Choi, T. Crick, M. R. Crusoe, S. Gesing, R. Haines, M. Heroux, L. J.
566 Hwang, D. S. Katz, K. E. Niemeyer, M. Parashar, and C. C. Venters. CEUR Workshop Pro-
567 ceedings 1686. urn:nbn:de:0074-1686-8. Aachen, 2016. URL: [http://ceur-ws.org/Vol-](http://ceur-ws.org/Vol-1686/WSSSPE4_paper_13.pdf)
568 [1686/WSSSPE4_paper_13.pdf](http://ceur-ws.org/Vol-1686/WSSSPE4_paper_13.pdf).
- 569 [33] K. Ram, S. Chamberlain, N. Ross, and M. Salmon. *rOpenSci Onboarding*. [https://github.](https://github.com/ropensci/onboarding)
570 [com/ropensci/onboarding](https://github.com/ropensci/onboarding). Commit 328434b5e37898e1e321e40f1f13e4bbe78f3d09. 2017.
- 571 [34] K. E. Niemeyer. *JOSS first-year publication data and figures*. Figshare. June 2017. DOI: [10.](https://doi.org/10.6084/m9.figshare.5147722.v1)
572 [6084/m9.figshare.5147722.v1](https://doi.org/10.6084/m9.figshare.5147722.v1).
- 573 [35] Z. Zhang, D. S. Katz, A. Merzky, M. Turilli, S. Jha, and Y. Nand. “Application Skeleton:
574 Generating Synthetic Applications for Infrastructure Research”. In: *The Journal of Open*
575 *Source Software* 1.1 (May 2016). DOI: [10.21105/joss.00017](https://doi.org/10.21105/joss.00017).
- 576 [36] A. Y. Z. Yao and D. Kane. “walkr: MCMC Sampling from Non-Negative Convex Polytopes”.
577 In: *The Journal of Open Source Software* 2.11 (Mar. 2017). DOI: [10.21105/joss.00061](https://doi.org/10.21105/joss.00061).
- 578 [37] *reviewers.csv*. Git commit 2006d1e05ad83c07d6f1bf14787e211167883a6d. 2017. URL:
579 <https://github.com/openjournals/joss/blob/master/docs/reviewers.csv> (visited on
580 06/01/2017).
- 581 [38] D. Foreman-Mackey. “corner.py: Scatterplot matrices in Python”. In: *The Journal of Open*
582 *Source Software* 1.2 (June 2016), p. 24. DOI: [10.21105/joss.00024](https://doi.org/10.21105/joss.00024).
- 583 [39] C. Sanderson and R. Curtin. “Armadillo: a template-based C++ library for linear algebra”.
584 In: *The Journal of Open Source Software* 1.2 (June 2016), p. 26. DOI: [10.21105/joss.00026](https://doi.org/10.21105/joss.00026).
- 585 [40] *JOSS Cited-by report*. 2017. URL: [https://data.crossref.org/depositorreport?pubid=](https://data.crossref.org/depositorreport?pubid=J299881)
586 [J299881](https://data.crossref.org/depositorreport?pubid=J299881) (visited on 12/15/2017).
- 587 [41] N. C. Clementi, G. Forsyth, C. D. Cooper, and L. A. Barba. “PyGBe-LSPR: Python and
588 GPU Boundary-integral solver for electrostatics”. In: *The Journal of Open Source Software*
589 2.19 (Nov. 2017), p. 306. DOI: [10.21105/joss.00306](https://doi.org/10.21105/joss.00306). URL: [https://doi.org/10.21105/](https://doi.org/10.21105/joss.00306)
590 [joss.00306](https://doi.org/10.21105/joss.00306).
- 591 [42] C. D. Cooper, N. C. Clementi, G. Forsyth, and L. A. Barba. “PyGBe: Python, GPUs and
592 Boundary elements for biomolecular electrostatics”. In: *The Journal of Open Source Software*
593 1.4 (Aug. 2016), p. 43. DOI: [10.21105/joss.00043](https://doi.org/10.21105/joss.00043). URL: [https://doi.org/10.21105/](https://doi.org/10.21105/joss.00043)
594 [joss.00043](https://doi.org/10.21105/joss.00043).

- 595 [43] C. Sanderson and R. Curtin. “gmm_diag and gmm_full: C++ classes for multi-threaded
596 Gaussian mixture models and Expectation-Maximisation”. In: *The Journal of Open Source
597 Software* 2.18 (Oct. 2017). DOI: [10.21105/joss.00365](https://doi.org/10.21105/joss.00365). URL: [https://doi.org/10.21105/
598 joss.00365](https://doi.org/10.21105/joss.00365).
- 599 [44] C. Sanderson and R. Curtin. “Armadillo: a template-based C++ library for linear algebra”.
600 In: *The Journal of Open Source Software* 1.2 (June 2016), p. 26. DOI: [10.21105/joss.00026](https://doi.org/10.21105/joss.00026).
601 URL: <https://doi.org/10.21105/joss.00026>.
- 602 [45] J. K. Tauber. *pyuca* Published in *The Journal of Open Source Software*. URL: [https://
603 jktauber.com/2016/05/19/pyuca-published-journal-open-source-software/](https://jktauber.com/2016/05/19/pyuca-published-journal-open-source-software/) (visited
604 on 12/15/2017).
- 605 [46] C. T. Brown. *Publishing Open Source Research Software in JOSS - an experience report*.
606 Sept. 2016. URL: [http://ivory.idyll.org/blog/2016-publishing-oss-research-
607 software.html](http://ivory.idyll.org/blog/2016-publishing-oss-research-software.html) (visited on 12/15/2017).
- 608 [47] J. Perkel. “TechBlog: *JOSS* gives computational scientists their academic due”. In: *Naturejobs
609 Blog* (Apr. 2017). URL: [http://blogs.nature.com/naturejobs/2017/04/04/joss-gives-
610 computational-scientists-their-academic-due/](http://blogs.nature.com/naturejobs/2017/04/04/joss-gives-computational-scientists-their-academic-due/).
- 611 [48] M. Moore. “Journal of Open Source Software helps researchers write and publish papers on
612 software”. In: *SD Times* (May 2017). URL: [http://sdtimes.com/journal-open-source-
613 software-helps-researchers-write-publish-papers-software/](http://sdtimes.com/journal-open-source-software-helps-researchers-write-publish-papers-software/).
- 614 [49] A. M. Smith, L. A. Barba, G. Githinji, M. Gymrek, K. Huff, D. S. Katz, C. Madan, A. C.
615 Mayes, K. M. Moerman, K. E. Niemeyer, P. Prins, K. Ram, A. Rokem, T. Teal, and J.
616 Vanderplas. *The Journal of Open Source Software*. Poster presented at SIAM Computational
617 Science & Engineering 2017 (CSE17), Atlanta, GA, USA. 27 February–3 March 2017. Feb.
618 2017. DOI: [10.6084/m9.figshare.4688911](https://doi.org/10.6084/m9.figshare.4688911).