OPINION ARTICLE

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


1ELIXIR Hub, Wellcome Genome Campus, Hinxton, CB10 1SD, UK
2Netherlands eScience Center, Science Park 140, Amsterdam, 1098 XG, The Netherlands
3CSL Limited, Bio21 Institute, 30 Flemington Road, Parkville, Victoria, 3010, Australia
4National eResearch Collaboration Tools and Resources, Victoria, 3010, Australia
5ELIXIR-DE and de.NBI, Bioinformatics Group, Department of Computer Science, University of Freiburg, Freiburg, Germany
6ELIXIR-ES, National Bioinformatics Infrastructure Sweden (NBIS), Scilifelab, Department of Biochemistry and Biophysics (DBB), Stockholm University, Stockholm, Sweden
7ELIXIR-ES, Spanish National Bioinformatics Institute (INB), Spanish National cancer Research Centre (CNIO), Calle de Melchor Fernández Almagro 3, Madrid, 28029, Spain
8Software Sustainability Institute, JCMB, University of Edinburgh, Edinburgh, EH9 3FD, UK
9Repositive Ltd, Future Business Centre, Cambridge, UK
10EMBL Australia Bioinformatics Resource, Lab-14, The University of Melbourne, 700 Swanston St, Parkville, Victoria, 3053, Australia
11EMBL-EBI, Wellcome Genome Campus, Hinxton, CB10 1SD, UK
12Barcelona Supercomputing Center, Barcelona, 08034, Spain
13Department of Biochemistry and Molecular Biomedicine, Universitat de Barcelona, Barcelona, 08028, Spain
14ELIXIR-UK, Software Sustainability Institute, School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, UK
15Oxford e-Research Centre, University of Oxford, Oxford, UK
16BBMRI-ERIC, Neue Stiftungtalstraße 2/8/6, Graz, 8010, Austria
17Dutch TechCenter for Life Sciences and ELIXIR-NL, Utrecht, The Netherlands
18ELIXIR-DK, Technical University of Denmark, Denmark, Denmark
19National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, IL, USA
20School of Information Sciences, University of Illinois Urbana Champaign, Urbana, IL, USA
21Department of Electrical and Computer Engineering, University of Illinois Urbana Champaign, Urbana, IL, USA
22Department of Computer Science, University of Illinois Urbana Champaign, Urbana, IL, USA
23ELIXIR-SI, Faculty of Medicine, University of Ljubljana, Ljubljana, Slovenia
Abstract
Scientific research relies on computer software, yet software is not always developed following practices that ensure its quality and sustainability. This manuscript does not aim to propose new software development best practices, but rather to provide simple recommendations that encourage the adoption of existing best practices. Software development best practices promote better quality software, and better quality software improves the reproducibility and reusability of research. These recommendations are designed around Open Source values, and provide practical suggestions that contribute to making research software and its source code more discoverable, reusable and transparent. This manuscript is aimed at developers, but also at organisations, projects, journals and funders that can increase the quality and sustainability of research software by encouraging the adoption of these recommendations.

Keywords
Open Source, code, software, guidelines, best practices, recommendations, Open Science, quality, sustainability, FAIR

This article is included in the ELIXIR gateway.
Corresponding authors: Rafael C. Jiménez (rafael.jimenez@elixir-europe.org), Mateusz Kuzak (m.kuzak@esciencecenter.nl), Steve Crouch (s.crouch@software.ac.uk)

Competing interests: At the time of writing MC is an employee of Reposite Ltd.

Grant information: This work was partially supported by ELIXIR-EXCELERATE and CORBEL. ELIXIR-EXCELERATE and CORBEL are funded by the European Commission within the Research Infrastructures programme of Horizon 2020, grant agreement numbers 676559 and 654248. The European workshops were supported by ELIXIR and organised in collaboration with the Software Sustainability Institute and Netherlands eScience Center. The workshop in Australia was supported by EMBL-ABR via its main funders: The University of Melbourne and Bioplatforms Australia.

Copyright: © 2017 Jiménez RC et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Jiménez RC, Kuzak M, Alhamdoosh M et al. Four simple recommendations to encourage best practices in research software [version 1; peer review: 3 approved] F1000Research 2017, 6:876 https://doi.org/10.12688/f1000research.11407.1

First published: 13 Jun 2017, 6:876 https://doi.org/10.12688/f1000research.11407.1
Introduction

New discoveries in modern science are underpinned by automated data generation, processing and analysis: in other words, they rely on software. Software, particularly in the context of research, is not only a means to an end, but is also a collective intellectual product and a fundamental asset for building scientific knowledge. More than 90% of scientists acknowledge software is important for their own research and around 70% say their research would not be feasible without it (Hannay et al., 2009; Hettrick et al., 2016).

Scientists are not just users of software; they are also prime producers (Goble, 2014). 90% of scientists developing software are primarily self-taught and lack exposure and incentives to adopt software development practices that are widespread in the broader field of software engineering (Wilson et al., 2014). As a result, software produced for research does not always meet the standards that would ensure its quality and sustainability, affecting the reproducibility and reusability of research (Crouch et al., 2013).

Open Source Software (OSS) is software with source code that anyone can inspect, modify and enhance. OSS development is used by organisations and projects to improve accessibility, reproduction, transparency and innovation in scientific research (Mulgan et al., 2005; Nosek et al., 2015). OSS not only increases discoverability and visibility, but it also engages developer and user communities, provides recognition for contributors, and builds trust among users (McKiernan et al., 2016). OSS development significantly contributes to the reproducibility of results generated by the software and facilitates software reusability and improvement (Ince et al., 2012; Perez-Riverol et al., 2014).

Opening code to the public is also an opportunity for developers to showcase their work, so it becomes an incentive for adoption of software development best practices (Leprevost et al., 2014). Thus, OSS can be used as a vehicle to promote the quality and sustainability of software, leading to the delivery of better research.

This manuscript describes a core set of OSS recommendations to improve the quality and sustainability of research software. It does not propose new software development best practices, but rather provides easy-to-implement recommendations that encourage adoption of existing best practices. These recommendations do not aim to describe in detail how to develop software, but rather lay out practical suggestions on top of Open Source values that go towards making research software and its source code more discoverable, reusable and transparent.

The OSS recommendations should be applied following existing and complementary guidelines like best practices, manifestos and principles that describe more specific procedures on how to develop and manage software. Some of these complementary guidelines are related to version control, code review, automated testing, code formatting, documentation, citation and usability. (Artaza et al., 2016; DagstuhlEAS, 2017; Gilb, 1988; Leprevost et al., 2014; List et al., 2017; Perez-Riverol et al., 2016; Pric & Procter, 2012; Smith et al., 2016; Wilson et al., 2014; Wilson et al., 2016).

This manuscript also aims to encourage projects, journals, funders and organisations to both endorse the recommendations and to drive compliance through their software policies. The recommendations are accompanied by a list of arguments addressing common questions and fears raised by the research community when considering open sourcing software.

In this manuscript, software is broadly defined to include command line software, graphical user interfaces, desktop and mobile applications, web-based services, application program interfaces (APIs) and infrastructure scripts that help to run services.

Target audience

Our target audience includes leaders and managers of organisations and projects, journal editorial bodies, and funding agencies concerned with the provision of products and services relying on the development of open research software. We want to provide these stakeholders with a simple approach to drive the development of better software. Though these OSS recommendations have mostly been developed within, and received feedback from, the life science community, the document and its recommendations apply to all research fields.

Strategies to increase software quality usually target software developers, focusing on training and adoption of best practices (Wilson et al., 2014). This approach can yield good results, but requires a significant effort as well as personal commitment from developers (Wilson, 2014). For an organisation employing scientists and developers with different sets of programming skills and responsibilities, it is not easy to endorse specific best practices or define a broad range of training needs. It is easier to endorse a set of basic recommendations that are simple to monitor, simple to comply with, and which drive the adoption of best practices and reveal training needs. The OSS recommendations aim to create awareness, encourage developers to be more conscious of best practices, and make them more willing to collaborate and request support. The recommendations define broad guidelines, giving developers freedom to choose how to implement specific best practices.

In terms of the adoption of these recommendations, we see endorsement as the first step: that is, agreeing to support the OSS recommendations without a formal process for implementation. Promotion is a second step: that is, actively publicising and incentivising the OSS recommendations within the organisation as well as globally. Compliance is the third step: to formally implement them within the organisation, with ongoing monitoring and public reporting if possible. To facilitate progress, we propose that organisations, projects, journals, as well as funding agencies include these OSS recommendations as part of their policies relating to the development and publication of software.
Open Source Software is not just adopted by non-profit organisations, but also by commercial companies as a business model (Popp, 2015). Therefore, we encourage not only publicly funded projects but also for-profit entities to adopt OSS and support these recommendations.

**Recommendations**

1. **Make source code publicly accessible from day one**
   Develop source code in a publicly accessible, version controlled repository (e.g., GitHub and Bitbucket) from the beginning of the project. The longer a project is run in a closed manner, the harder it is to open it later (Fogel, 2005). Opening code and exposing the software development life cycle publicly from day one:
   - Promotes trust in the software and broader project
   - Facilitates the discovery of existing software development projects
   - Provides a historical public record of contributions from the start of the project and helps to track recognition
   - Encourages contributions from the community
   - Increases opportunities for collaboration and reuse
   - Exposes work for community evaluation, suggestions and validation
   - Increases transparency through community scrutiny
   - Encourages developers to think about and showcase good coding practices
   - Facilitates reproducibility of scientific results generated by all prior versions of the software
   - Encourages developers to provide documentation, including a detailed user manual and clear in-code comments

Some common doubts and questions about making software Open Source are discussed in the Supplementary File S1, “Fears of open sourcing and some ways to handle them”.

2. **Make software easy to discover by providing software metadata via a popular community registry**
   Facilitate discoverability of the software project and its source code by registering metadata related to the software in a popular community registry. Metadata might include information like the source code location, contributors, licence, version, identifier, references and how to cite the software. Metadata registration:
   - Increases the visibility of the project, the software, its use, its successes, its references, and its contributors
   - Provides easy access for software packagers to deploy your software, thus increasing visibility
   - Encourages software providers to think about the metadata that describes software as well as how to expose such metadata
   - Helps to expose the software metadata in a machine readable format via the community registry
   - Increases the chances of collaboration, reuse, and improvement

Examples of community registries of software metadata are bio. tools (Ison et al., 2016), (Ison et al., 2016) biojs.io (Corpus et al., 2014; Gómez et al., 2013) and Omic Tools (Henry et al., 2014) in the life sciences and DataCite (Brase, n.d.) as a generic metadata registry for software as well as data.

3. **Adopt a licence and comply with the licence of third-party dependencies**
   Adopt a suitable Open Source licence to clarify how to use, modify and redistribute the source code under defined terms and conditions. Define the licence in a publicly accessible source code repository, and ensure the software complies with the licences of all third party dependencies. Providing a licence:
   - Clarifies the responsibilities and rights placed on third parties wishing to use, copy, redistribute, modify and/or reuse your source code
   - Enables using the code in jurisdictions where “code with no licence” means it cannot be used at all
   - Protects the software’s intellectual property
   - Provides a model for long-term sustainability by enabling legally well-founded contributions and reuse

We advise choosing a OSI-approved Open Source Licence unless your institution or project requires a different licence. Websites like “Choose an open source license” provide guidelines to help users to select an OSI-approved Open Source Licence. Organisations like the OSS Watch also provide advice on how to keep track of the licences of software dependencies. For reusability reasons, we also advise authors to disclose any patents and pending patent applications known to them affecting the software.

4. **Define clear and transparent contribution, governance and communication processes**
   Open sourcing your software does not mean the software has to be developed in a publicly collaborative manner. Although it is desirable, the OSS recommendations do not mandate a strategy for collaborating with the developer community. However, projects should be clear about how contributions can be made and incorporated by having transparent governance model and communication channels. Clarity on the project structure, as well as its communication channels and ways to contribute:
   - Increases transparency on how the project and the software is being managed
   - Helps to define responsibilities and how decision are made in the software project
• Helps the community know how to collaborate, communicate and contribute to the project

For instance the Galaxy project’s website describes the team’s structure, how to be part of the community, and their communication channels.

Alignment with FAIR data principles
The FAIR Guiding Principles for scientific data management and stewardship provide recommendations on how to make research data findable, accessible, interoperable and reusable (FAIR) (Wilkinson et al., 2016). While the FAIR principles were originally designed for data, they are sufficiently general that their high level concepts can be applied to any digital object including software. Though not all the recommendations from the FAIR data principles directly apply to software, there is good alignment between the OSS recommendations and the FAIR data principles (see Table 1).

<table>
<thead>
<tr>
<th>The FAIR Guiding Principles</th>
<th>OSS recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>To be Findable: F1. (meta)data are assigned a globally unique and persistent identifier; F2. data are described with rich metadata (defined by R1 below); F3. metadata clearly and explicitly include the identifier of the data it describes; F4. (meta)data are registered or indexed in a searchable resource</td>
<td>“R2. Make software easy to discover by providing software metadata via a popular community registry” aligns with the Findability principle, helping to increase visibility and helping software providers to think about how to describe software metadata (versions, identifiers, contributors, citations, etc.)</td>
</tr>
<tr>
<td>To be Accessible: A1. (meta)data are retrievable by their identifier using a standardized communications protocol; A1.1 the protocol is open, free, and universally implementable; A1.2 the protocol allows for an authentication and authorization procedure, where necessary; A2. metadata are accessible, even when the data are no longer available</td>
<td>“R1. Make source code publicly accessible from day one” focuses on openness including accessibility. The FAIR accessible principle instead opens the door to data that is restricted access e.g. for privacy reasons. Since such reasons do not apply for software, the OSS recommendations prefer to direct towards openness instead, supporting open science to the maximum extent.</td>
</tr>
<tr>
<td>To be Interoperable: I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; I2. (meta)data use vocabularies that follow FAIR principles; I3. (meta)data include qualified references to other (meta)data</td>
<td>This OSS recommendations do not aim to address software interoperability directly but contribute to a more homogenous description of software by encouraging software providers to register software metadata into registries providing specific metadata guidelines.</td>
</tr>
<tr>
<td>To be Reusable: R1. (meta)data are richly described with a plurality of accurate and relevant attributes; R1.1. (meta)data are released with a clear and accessible data usage license; R1.2. (meta)data are associated with detailed provenance; R1.3. (meta)data meet domain-relevant community standards</td>
<td>“R3. Adopt a license and comply with the licence of third-party dependencies” aligns with the Reusability principle, helping to define to what extent the source code can be used and reused by the community, as a standalone software or as part of other software. Open availability of tools and libraries working with data formats can be a great help in making data interoperable: e.g. reuse of the same tools to read and write data can prevent subtle interoperability problems. Reproducibility of experiments and reuse of data is facilitated by the open availability of the associated software which is part of the provenance. All of the OSS recommendations thereby facilitate data Reusability.</td>
</tr>
</tbody>
</table>

Table 1. Comparison between the OSS recommendations and the FAIR data principles (Wilkinson et al., 2016).
guidelines on how to provide software metadata following the FAIR Guiding Principles (Wilkinson et al., 2016).

Conclusion
The OSS recommendations aim to encourage the adoption of best practices and thus help to develop better software for better research. These recommendations are designed as practical ways to make research software and its source code more discoverable, reusable and transparent, with the desired objective to improve its quality and sustainability. Unlike many software development best practices tailored for software developers, the OSS recommendations aim to target a wider audience, particularly research funders, research institutions, journals, group leaders, and managers of projects producing research software. The adoption of these recommendations offer a simple mechanism for these stakeholders to promote the development of better software and an opportunity for developers to improve and showcase their software development skills.

Author contributions
Steve Crouch, Neil Chue Hong, Mateusz Kuzak, Manuel Corpus, Jason Williams, Maria Victoria Schneider and Rafael C Jimenez also contributed organising and facilitating workshops. Federico Lopez developed the reference website to provide information and a point of contact for these recommendations. All the authors contributed providing feedback to shape this manuscript and recommendations.

Supplementary material
Supplementary File 1: ‘Fears of open sourcing and some ways to handle them’. In this appendix we aim to expose some of the common fear scenarios related to open sourcing, and some ways to handle them.

Click here to access the data.

Competing interests
At the time of writing MC is an employee of Repositive Ltd.

Grant information
This work was partially supported by ELIXIR-EXCELERATE and CORBEL. ELIXIR-EXCELERATE and CORBEL are funded by the European Commission within the Research Infrastructures programme of Horizon 2020, grant agreement numbers 676559 and 654248. The European workshops were supported by ELIXIR and organised in collaboration with the Software Sustainability Institute and Netherlands eScience Center. The workshop in Australia was supported by EMBL-ABR via its main funders: The University of Melbourne and Bioplatforms Australia.

Acknowledgments
The authors wish to thank all the supporters of the OSS recommendations.

The OSS recommendations presented in this manuscript have been been open for discussion for more than a year. This allowed them to be developed by a wide range of stakeholders, including developers, managers, researchers, funders and project coordinators and anybody else concerned with the production of quality software for research. We also organised several workshops and presented this work in several meetings to engage more stakeholders, collect feedback and refine the recommendations. For further information, about the OSS recommendations please visit the following site: https://SoftDev4Research.github.io/recommendations/

References

Open Peer Review

Current Peer Review Status: ☑️ ☑️ ☑️

Stefanie Betz
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

The article advocates research software openness presenting four recommendations to improve research software visibility, re-usability, and transparency. I really like the article and I think it is important to open research software. Please find below my feedback. (Overall, I agree with the comments of Milad Miladi):

- In my opinion, the title does not reflect the content of the article. The focus is on adopting OSS and supporting the provided four recommendations to help to develop better software for better research. Currently, only the second part (supporting the provided four recommendations to help to develop better software for better research) is reflected in the title not the part about OSS.
- I am not sure everybody is familiar with the open registry platforms. Thus, some more information regarding them or a link to background information would be nice.
- I think recommendation four should include documentation.

Is the topic of the opinion article discussed accurately in the context of the current literature?
Yes

Are all factual statements correct and adequately supported by citations?
Yes

Are arguments sufficiently supported by evidence from the published literature?
Yes

Are the conclusions drawn balanced and justified on the basis of the presented arguments?
Yes

Competing Interests: No competing interests were disclosed.
**Reviewer Expertise:** Software Engineering

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

---

**Gregory V. Wilson**  
Rangle.io, Toronto, ON, Canada

The article presents focused, well-argued advocacy for improving software development practices in the sciences. None of the recommendations will be surprising to those already involved in open science, but as only a small minority of researchers actually do them, it is worth presenting them forcefully and succinctly. I would recommend shortening the introductory material (sections "Introduction" and "Target Audience"), but that is a minor point.

**Is the topic of the opinion article discussed accurately in the context of the current literature?**  
Yes

**Are all factual statements correct and adequately supported by citations?**  
Yes

**Are arguments sufficiently supported by evidence from the published literature?**  
Yes

**Are the conclusions drawn balanced and justified on the basis of the presented arguments?**  
Yes

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** software engineering education

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.
This article presents four simple recommendations that may improve the overall quality and visibility of research software. This reviewer agrees with the basic principles set forth by the authors, and hopes they will be widely shared and adopted at least for software that is expected to last longer than the time it takes for the corresponding research paper to be accepted and/or presented.

Is the topic of the opinion article discussed accurately in the context of the current literature?
Yes

Are all factual statements correct and adequately supported by citations?
Yes

Are arguments sufficiently supported by evidence from the published literature?
Yes

Are the conclusions drawn balanced and justified on the basis of the presented arguments?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: formal methods, software engineering, programming languages, logics, theoretical computer science, semantics, parallel programming, open source software, package management, archival

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.
Very interesting article! Research software importance and recognition is increasing rapidly in the scientific landscape and this work provides relevant recommendations to be followed by research software producers in order to improve the disseminations practices.

The article proposals have some common points with the dissemination procedure that I described in the preprint:
"Free software, Open source software, licenses. A short presentation including a procedure for research software and data dissemination" (September 2014), https://zenodo.org/record/11709/
It was presented at the EGI Conference 2015, in the session:
Workshop on open licenses: Data licencing and policies
https://indico.egi.eu/indico/event/2452/session/75

**Competing Interests:** As competing interest, I have collaborated with one of the authors in the last four years (grant application).

---

Reader Comment 20 Jul 2017

**Paul Kersey**, European Bioinformatics Institute, UK

I think this is an important paper, and I broadly agree with its recommendations, but it could do (IMO) with a little copy-editing before it is distributed to funding agencies and other people identified as among its target audience. I'm writing these comments here at the encouragement of the lead author.

**A.TITLE**
I don't think the title is quite right. The paper recommends practices for the development of best software, i.e. (i) open sourcing of code is a practice, not a means to encourage a practice (ii) the adoption of practices (whatever they are) the means not the end
I would suggest "Good Practices in Research Software" as a better title: 5 words not 10, and the dropped words just obfuscate.

**B.RECOMMENDATION 2**
“Make software easy to discover by providing software metadata via a popular community registry” – what matters is that the registry is popular. What is meant by a community registry? Why it is important that a registry is run by a community? Wouldn't you prefer a Google-run registry?

**C.CONCLUSIONS**
"The OSS recommendations aim to encourage the adoption of best practices (1) and thus help to develop better software for better research (2). These recommendations are designed as practical ways to make research software and its source code more discoverable, reusable and transparent, with the desired objective to improve its quality and sustainability (3). Unlike many software development best practices tailored for software developers, the OSS recommendations aim to target a wider audience, particularly research funders, research institutions, journals, group leaders, and managers of projects producing research software (4). The adoption of these recommendations offer a simple mechanism for these stakeholders to promote the development
of better software and an opportunity for developers to improve and showcase their software development skills (5)."

1. This amounts to saying “We recommend that people follow these recommendations”. The recommendations are recommendations of good practice. These practices are recommended with the aim of doing something - that aim is not to encourage their own adoption!

2. Better software or better research? How is the research better? You address this in the introduction but the treatment here is a bit glib.

3. The structure of this sentence - "we do A to do B to do C" - makes it unclear what the ultimate goal is, and the individual links in the logic chain are unsupported. I think the way I would put it is: “Research software is often tapped in a vicious cycle: small user communities can sustain only a limited effort in maintenance, development and documentation. We believe that an open source approach, as recommended here, can help to overcome these problems”.

4. “Unlike many software development best practices tailored for software developers, the OSS recommendations aim to target a wider audience, particularly research funders, research institutions, journals, group leaders, and managers of projects producing research software.” “Aiming to target” is a neoplasm. Also, the “unlike” is a bit tenuous – all development practices are tailored for developers (or else they’re a bit silly), but obviously managers are also generally interested and involved. I think the point is simply “We urge not only software developers but also research funders, research institutions, journals, group leaders, and managers of projects producing research software, to consider these recommendations”.

5. The last sentence is a bit weak as a conclusion. There’s a powerful point (which is well-made in the introduction) that open source software can be validated, improved upon, and used to enable research; that the culture of open source promotes the development of both the coder and the code; and that the spoils of publicly-funded tool development properly belong in the public domain. But instated, the final conclusion is a piece of career advice, and moreover, amounts to saying “people should follow these recommendations because they are good”, rather than demonstrating why they are good.

**Competing Interests:** No competing interests were disclosed.
"openness" and "encouragement" is not understood.

* Although the importance of documentation has been stated in numerous previous works, it worth to be mentioned here again. Specially in my opinion this meta-data should be available independent and in parallel to of the availability via community registry.

* From the text the reader may not be fully convinced about the advantages of using open registry platforms over plain text format beside the source code. A brief overview of the mentioned platforms (bio.tools,..) like number of current entries, the differences, ... would be nice.

* Reusability and reproducibility of the research tools are among major concerns of nowadays research ecosystem. This has been stated in the abstract and introduction sections, however it would be very valuable if the authors could include specific recommendations on how these metrics can be improved for research softwares.

*Competing Interests*: No competing interests were disclosed.

Reader Comment 13 Jun 2017

Mehran Karimzadeh,

Very interesting work. You may be interested in our previous perspective, "Top considerations for creating bioinformatics software documentation" (https://doi.org/10.1093/bib/bbw134), which might be a useful reference in the part of your article that mentions existing guidelines in documentation and elsewhere.

*Competing Interests*: No competing interests were disclosed.
The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com