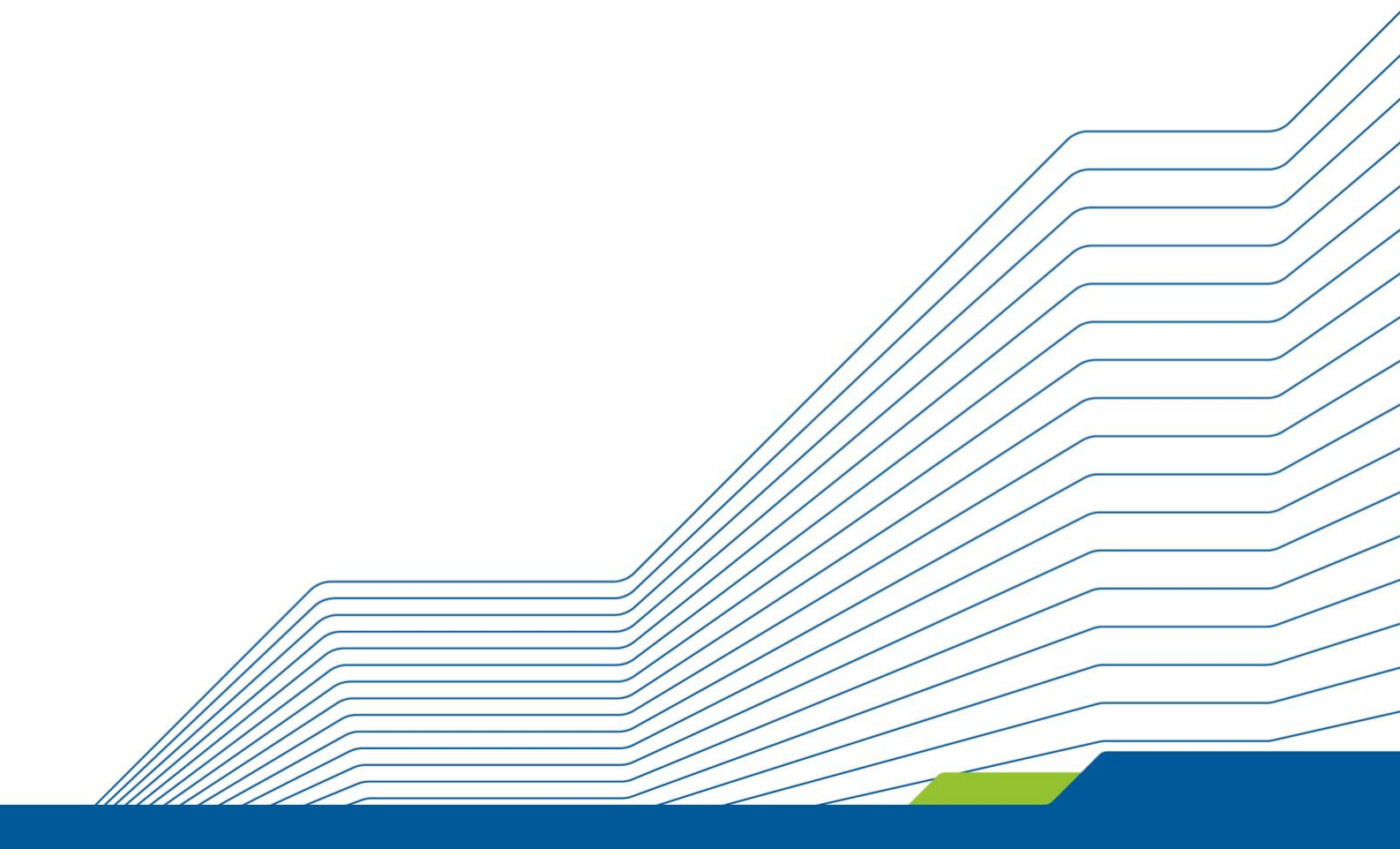
HELMHOLTZ
Open Science

# Dealing with research software: Recommendations for best practices



## Imprint
An online version of this publication can be found at:
DOI: https://doi.org/10.2312/os.helmholtz.003

## Publisher
Task Group "Zugang zu und Nachnutzung von wissenschaftlicher Software"
(Task Group Access to and Reuse of Research Software) of the Open Science Working Group of the Helmholtz Association

## Authors and Contributors
Kaja Scheliga (https://orcid.org/0000-0002-4863-5430)
Heinz Pampel (http://orcid.org/0000-0003-3334-2771)
Uwe Konrad (https://orcid.org/0000-0001-8167-9411)
Bernadette Fritzsch (https://orcid.org/0000-0002-0690-7151)
Tobias Schlauch (https://orcid.org/0000-0001-8760-8913)
Marco Nolden (https://orcid.org/0000-0001-9629-0564)
Wolfgang zu Castell, Ants Finke, Martin Hammitzsch, Oliver Bertuch, Michael Denker;
Task Group "Zugang zu und Nachnutzung von wissenschaftlicher Software":
https://os.helmholtz.de/open-science-in-der-helmholtz-gemeinschaft/akteure-und-ihre-rollen/arbeitskreis-open-science/task-group-zugang-zu-und-nachnutzung-von-wissenschaftlicher-software/#c17672

## Editing
Helmholtz Open Science Coordination Office (Kaja Scheliga)

## Contact
Helmholtz Open Science Coordination Office
c/o Helmholtz-Zentrum Potsdam
Deutsches GeoForschungsZentrum GFZ
Telegrafenberg, 14473 Potsdam
Email: open-science@helmholtz.de

## Version
February 2019

## Abstract

In this paper we present general recommendations for dealing with research software. We discuss incentives and metrics, software development and documentation, accessibility, publication and transfer strategies, infrastructures, quality assurance, licensing and other legal topics, education and training, as well as policies and guidelines. We consider research software, alongside text and data, as an essential element of open science. We argue that research software should be treated and acknowledged as a discrete product of the research process.

## Introduction

The increasing digitisation of education and research leads to a rising number of software-based solutions used or developed in labs and research institutions[1] [2]. In many research areas, source code is at the core of the research process and software-based solutions are indispensable tools in knowledge creation[3]. The documentation of and the access to the software used in the research process are also crucial for the replicability of the results[4]. In many areas the verifiability and reproducibility of research results can only be ensured when, alongside the research data, the source code is openly accessible and accompanied by an adequate documentation and clearly defined terms of use[5].

Even though research software increasingly gains importance as well as attention, there is still a lack of standards and guidelines, and also a lack of best practices for dealing with research software. Moreover, support mechanisms concerning the development, publication and maintenance of research software are needed.

In this paper[6], we take a general point of view and look at the whole lifecycle of software-based research projects, also taking into account the use of commercial software and existing open source software. We define research software as code (source code along with documentation, parameters, and workflows) that is developed or (re)used as part of research-based activities.

---

[1] S. Crouch, N. Chue Hong, S. Hettrik, M. Jackson, A. Pawlik, S. Sufi, L. Carr, D. De Roure, C. Goble, M. Parsons, The Software Sustainability Institute: changing research software attitudes and practices. Computing in Science & Engineering. 15, 74-80 (2013), doi: https://doi.org/10.1109/MCSE.2013.133

[2] J. N. Joppa, G. McInerny, R. Harper, L. Salido, K. Takeda, K. O'Hara, D, Gavaghan, S. Emmott, Troubling Trends in Scientific Software Use. Science. **340**:6134 (2013.), doi: https://10.1126/science.1231535

[3] Editorial: Software with impact. Nature Methods. **11**:211(2014), doi: https://doi.org/10.1038/nmeth.2880

[4] A. Morin, J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, P. Sliz, Shining light into black boxes. Science. **226**:6078 (2012), doi: https://doi.org/10.1126/science.1218263

[5] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. A. Ioannidis, M. Taufer, Enhancing Reproducibility for Computational Methods. Science. **354**:6317(2016), doi: https://10.1126/science.aah6168

[6] **Read me:** *This paper is based on the work of the Task Group "Zugang zu und Nachnutzung von wissenschaftlicher Software" ("Access to and Reuse of Research Software") of the Helmholtz Association. It draws on materials from the Helmholtz Open Science Workshop which took place in November 2016 at the Helmholtz Centre Dresden-Rossendorf (see Workshop Report) as well as a position paper developed by the Task Group and approved by the Open Science Working Group of the Helmholtz Association in March 2017. An extended version ("Empfehlungen zur Implementierung von Leit- und Richtlinien zum Umgang mit wissenschaftlicher Software an den Helmholtz-Zentren (in German)) of this text is available at https://os.helmholtz.de/index.php?id=3197*

We argue that research software is a discrete product of the research process. Analogue to other research methods, it must be documented, published and acknowledged. Thus, the access to and the reuse of research software is, together with open research data and open access to publications, an essential element of open science.

In the following sections of this paper we present general recommendations for dealing with research software. We discuss incentives and metrics, software development and documentation, accessibility, publication and transfer strategies, infrastructures, quality assurance, licensing and other legal topics, education and training, as well as policies and guidelines.

## Incentives and metrics

A sustainable and open approach to research software as well as the efficient use of resources in (re)using and developing software can best be encouraged with incentives and support for the respective practices. The intellectual effort involved in software development for research purposes should be acknowledged and valued as a distinct contribution to the research process. Research software should be considered a research product in its own right and publishing it should be supported and valued on a par with data and text. Repositories should treat source code and its documentation as a distinct publication type.

Software citations should be enforced by both research institutions and academic publishers. Furthermore, establishing transparent and open metrics are means to make the effort put into research software development measurable and for it to have an impact in scientometric analyses.

## Software development and documentation

Sustainable software development goes hand in hand with good software documentation practices. Both should be supported by adequate infrastructures and taught as an integral part of education, training and professional development in order to be established in research practice.

To make the steps that led to the research results replicable, changes in the source code should be documented (with version control) and runtime analyses should be clearly described. New employees should be made familiar with practices of good software development and documentation and also offered further professional development.

The basic requirements can be captured in policies and guidelines for good scientific practice[7] [8] [9]. These can be developed collaboratively, across research institutions.

Recommendations for software development and documentation should include specific references to methods, software tools and platforms that can be used for:

[7] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, et al., The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. **3**:160018 (2016), doi: https://doi.org/10.1038/sdata.2016.18
[8] Software Sustainability Institute, Guides (2016), https://www.software.ac.uk/resources/guides
[9] Software Sustainability Institute, Checklist for a Software Management Plan. v0.1 (2016), https://www.software.ac.uk/software-management-plans

- planning (use-cases, class diagrams, UML models, requirements, etc.)
- development (versioning, issue tracking, code review, style guidelines, etc.)
- testing (unit tests, integration tests, debugging, etc.)
- documentation (code documentation, API documentation, user documentation, etc.)
- distribution (release strategy, simple installation, etc.)

## Accessibility, publication and transfer strategies

The interplay of accessible software with text and data can ensure the replicability of research results. With regards to making research software accessible, it is important to differentiate between publishing and providing open access to research software in an academic context and the possible use of software developed for research purposes in a commercial context. Moreover, there is a distinction between simple applications for specific analyses, extensive software libraries and complex software systems.

In cases where there are no conflicting interests for a commercial use, research software should be made openly accessible (open source) in a trusted and reliable infrastructure. In order to grant scientists their competitive advantage, adequate embargo periods can also be considered.

Collaborative software development and interdisciplinary knowledge exchange should be supported. In order to grant secure access to and (re)use of research software for parties across research institutions and organisations, licenses and other legal aspects need to be defined. An important aspect in this context is making the software accessible on a long-term basis and adhering to the principles of version control in order to enable version-specific citations[10] [11]. We recommend using digital object identifiers (DOIs) and other persistent identifiers (PIDs) when publishing research software.

## Infrastructures

Adequate infrastructures can support researchers in developing sustainable research software. We advocate for research institutions to provide platforms for collaborative software development in order to guarantee long-term access to research software. This allows research institutions to maintain control over their intellectual property and to ensure independence from commercial infrastructure service providers. Such infrastructures can be managed and maintained by individual working groups, institutions or cooperatively across institutions, depending on the needs and the available resources.

[10] Y. Perez-Riverol, L. Gatto, R. Wang, T. Sachsenberg, J. Uszkoreit, FdV. Leprevost, C. Fufezan, T. Ternent, S. J. Eglen, D. S. Katz, T. J. Pollard, A. Konovalov, R. M. Flight, K. Blin, J. A. Vizcaíno, Ten simple rules for taking advantage of Git and GitHub. PloS Computational Biology. **12**(7): e1004947 (2016), doi: https://doi.org/10.1371/journal.pcbi.1004947
[11] A. Silver, Software simplified: containerization technology takes the hassle out of setting up software and can boost the reproducibility of data-driven research. Nature. **546**:173-174 (2017), doi: https://doi.org/10.1038/546173a

Those platforms can be linked and synchronized with open platforms (such as GitHub) to reach a broad open source community.

When setting up infrastructures for research software development, current tools and agile software development methods need to be taken into account.

Furthermore, complex research software can also be considered as an infrastructure that is developed and maintained over long periods of time. This entails requirements for reliability, quality, documentation, as well as training and community building.

We recommend considering the following measures and parameters for setting up infrastructures and related processes:

- Including research software products in publication guidelines and publication databases with referral to suitable infrastructures for archiving.
- Setting up subject-specific core facilities for supporting researchers in designing, implementing, and optimising software.
- Providing software development platforms which can be used for training, tests, and setting up new software projects.
- Provision of diverse and efficient test-platforms for continuous integration of the developed versions.
- Archiving of published releases in a repository and allocation of digital object identifiers (DOIs) or other persistent identifiers (PIDs).

When providing own infrastructures, a sustainable human resources policy should be adhered to in order to guarantee a seamless operation of the infrastructure and competent support for the researchers using the services.

## Quality assurance

Adhering to defined rules and applying tested methods in software development contributes to high quality standards, the reuse and further development of research software. Thus, it is important for research institutions to define, apply, and review appropriate quality standards for developing research software.

Checklists can be used to examine whether developed source code fulfils the quality requirements. This procedure can encompass various compliance levels, depending on the specifics of the software. Quality assurance can thus be applied to different quality levels.

Furthermore, research institutions are advised to set up processes for mutual support of researchers who are developing software as well as review procedures as part of collaborative working in interdisciplinary teams. Appropriate code review practices should be supported so that research software is reviewed from both a scientific and technological perspective. Moreover, code reviews

should accompany the development of software throughout the whole lifecycle[12] [13] and are of particular importance when research software is published.

When publishing research software, citation principles and the relevance of a particular software should count towards metrics, which in turn requires a reliable quality of the metadata[14]. Integrating methods of quality assurance into education and training is an important measure for establishing quality assurance as part of standard research practice.

It would be desirable for research institutions to agree upon standards and establish something like a "software seal of approval".

## Licensing and other legal topics

Since software is predominantly protected by copyright and can have commercial potential it is important to make a conscious decision about the type of use and the respective licensing. Here, it can be differentiated between licensing software for profit through collecting licensing fees associated with conventional licenses or expanded access to the software through using open-source-licenses. In the case of using open-source-licenses the software is freely accessible for third parties (although even under open-source-licenses revenues can be made by means of support services or access to extra features). We plead for using open and already established licenses, such as those accredited by the Open Source Initiative[15]. However, the pros and cons of copy-left licenses versus non-copy-left licenses need to be considered for each specific case.

## Education and training

The quality of research software depends on the skills of those developing it. Thus, education and training in software development and engineering should be shaped in dialogue with higher education bodies. Furthermore, software development skills should be taken into account with regards to career opportunities.

Next to embedding programming skills in the education of expert scientists, education paths for IT specialists in science should also be included. A dialogue between research institutions and universities as well as colleges should be established in order to enable knowledge exchange and to open up opportunities for collaboration. Qualification works (such as Bachelor, Master, and PhD thesis) where research software development plays a role should be designed in cooperation between scientific disciplines and computer science.

---

[12] N. Barnes, Publish your computer code: it is good enough. Nature. **467**:753,(2010), doi: https://doi.org/10.1038/467753a
[13] L. MacLeod, M. Greiler, M. A. Storey, C. Bird, J. Czerwonka, Code reviewing in the trenches: understanding challenges and best practices. IEEE Software. 99 (2017), doi: https://doi.org/10.1109/MS.2017.265100500
[14] A. M. Smith, D. S. Katz, K. E. Niemeyer, FORCE11 Software Citation Working Group. Software Citation Principles. PeerJ Computer Science. **2**:e86 (2016), doi: https://doi.org/10.7717/peerj-cs.86
[15] Open Source Initiative: https://opensource.org/

Internal as well as external education and training programmes should be complemented by sustainable human resource policies. Career paths for excellent graduates and other talented individuals should be provided. Long-term career prospects for roles involving research software development should be established. Training should be offered to both expert scientists to deepen their IT skills and IT specialists to deepen their discipline specific knowledge. Developing communications skills as well as building up an understanding for the mindset of collaboration partners also play an important role.

## Policies and guidelines

Policies and guidelines provide a framework for a coordinated and organised approach to dealing with research software. They support and unburden persons involved in developing or maintaining research software by serving as a point of orientation and providing general references to workflows, frameworks, and tools[16] [17]. Moreover, research institutions should also include contact partners who can offer individual support on a specific topic.

Corresponding policies and guidelines should encompass the whole software development lifecycle and in particular include statements referring to the following areas:

- incentives and metrics
- software development and documentation
- accessibility, publication and transfer strategies
- infrastructures
- quality assurance
- licensing and other legal topics
- education and training

Establishing policies and guidelines constitute a reliable basis for developing shared processes and infrastructures as well as standardised ways for dealing with research software in the digital era. The implementation of policies and guidelines can be endorsed by tangible templates covering core issues and designated persons of contact who can offer support concerning specific issues.

Furthermore, developing tangible templates for software management plans is also recommended. These can additionally support researchers in planning and implementing tasks in software development by illustrating options for actions and setting up a blueprint for dealing with research software responsibly.

---

[16] R. C. Jiménez, M. Kuzak, M. Alhamdoosh, et al., Four simple recommendations to encourage best practices in research software. F1000Research. **6**:ELIXIR-876 (2017), doi: https://doi.org/10.12688/f1000research.11407.1

[17] G. Wilson, D. A. Aruliah, C. T. Brown, N. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, P. Wilson. Best practices for scientific computing. PloS Biology. **12**(1):e1001745 (2014), doi: https://doi.org/10.1371/journal.pbio.1001745

## Summary

Research software development is an integral part of the research process. As an element of open science, the access to and the (re-)use of research software contribute substantially to making research results verifiable and reproducible. Consideration for the role of research software is needed throughout the whole research process and it is time for research institutions to play their part in supporting research software development.

HELMHOLTZ
Open Science