Publishing computational research - A review of infrastructures for reproducible and transparent scholarly communication

Authors: Markus Konkol* (m.konkol@uni-muenster.de), Daniel Nüst, Laura Goulier (Institute for Geoinformatics, University of Münster, Münster, Germany)

* Corresponding author

Keywords: Open reproducible research, open science, computational statistics, scholarly communication

Abstract

Background

The trend toward open science increases the pressure on authors to provide access to the source code and data they used to compute the results reported in their scientific papers. Since sharing materials reproducibly is challenging, several projects have developed solutions to support the release of executable analyses alongside articles.

Methods

We reviewed eleven applications that can assist researchers in adhering to reproducibility principles. The applications were found through a literature search and interactions with the reproducible research community. An application was included in our analysis if it **(i)** was actively maintained at the time the data for this paper was collected, **(ii)** supports the publication of executable code and data, **(iii)** is connected to the scholarly publication

process. By investigating the software documentation and published articles, we compared the applications across 19 criteria, such as deployment options and features that support authors in creating and readers in studying executable papers.

Results

From the eleven applications, eight allow publishers to self-host the system for free, whereas three provide paid services. Executable analyses are predominantly submitted using literate programming approaches: Ten applications support Jupyter Notebooks or R Markdown documents. All approaches provide features for readers, e.g., one-click reproducible results or tools for manipulating the analysis parameters. Six applications allow for modifying materials after publication.

Conclusions

This paper surveys the features provided by certain applications to assist in reproducible research and describes their limitations while considering stakeholders involved in scholarly communication. For example, publishers can opt for a paid service or host an infrastructure by themselves, but there is not much data on the costs of running one's own service. Further, executable documents might require editors to change the journal guidelines and adjust the review process. From the author's view, there are already open platforms authors can use today, e.g., as supplements. The ability to modify published content is beneficial for authors' ongoing projects, but it might become unclear which version was used in the paper. Finally, considering readers, readers can obtain tools to execute and manipulate the computational analysis, but they might need to cope with complex code.

Background

In many scientific fields, the results of scientific articles can be based on computations, e.g., a statistical analysis implemented in R. For this type of research, publishing the used source code and data to adhere to **open reproducible research (ORR)** principles (i.e., public access to the code and data underlying the reported results [1]) seems simple. Nevertheless, several studies have concluded that papers rarely include or link to these materials [2, 3]. Reasons for that are manifold:

First, due to *technical challenges*, e.g., capturing the analyst's original computational environment, even having accessible materials does not guarantee that results can be reproduced [4, 5]. Second, many authors hesitate to share their work because publishing erroneous papers can *damage an author's reputation* [6] as well as *trust* in science [7]. These perspectives, however, overlook the fact that engaging in open practices offers some career advantages [8, 9] and can help in identifying and correcting mistakes [10, 11].

As a result of authors not including their source code and underlying data, several further problems arise. For example, reviewers *cannot verify the results*, because without the code, they are required to understand the analysis just by reading the text [12]. Hence, finding errors in results is difficult and often impossible [6], raising the question of whether the traditional research article is suitable for conveying a complex computational analysis [13]. Additionally, other researchers working in similar areas *cannot continue building on existing work* but have to collect data and implement the analysis from scratch [14]. All these issues

are also to society's disadvantage, as the public cannot benefit fully from publicly funded research [15].

Fortunately, funding agencies, e.g., Horizon 2020, are increasingly requiring data and software management plans as part of grant proposals. Accordingly, more journal editors are starting to make sure that author guidelines include a section on code and data availability [16, 17], and reviewers are now considering reproducibility in their decision processes [10]. Moreover, concepts and tools to package code, the computing environment, data, and the text of a research workflow (a so-called "Research Compendium" [18]) are becoming more advanced and applied. This form of publishing research allows reviewers to verify the reported results and readers to reuse the materials [19].

Nevertheless, neither the cultural and systematic developments [20] for ORR nor the existence of technologies for packaging research reproducibly can alone solve the plethora of reproducibility issues. Authors often do not know how to fulfill the requirements of funding bodies and journals, such as the Transparency and Openness Promotion (TOP) guidelines [21], or they do not have the programming skills. It is important to consider that the range of researchers' programming expertise varies from trained research software engineers to self-taught beginners. For these reasons, more and more applications have been created to help support the publication of executable computational research for transparent and reproducible research. This paper aims at reviewing these applications in order to help researchers find the application that best suits their individual needs.

Methods

Study design

In this review study, we surveyed and compared eleven applications that assist authors in publishing reproducible research. The goal of the review was to obtain an overview of the benefits and limitations of these applications considering the challenges outlined in the previous section. We contrasted the solutions to create a set of criteria that addresses the needs of the stakeholders involved in the scholarly publication process, i.e. publishers, editors, authors, readers/reviewers, and librarians [22].

Sample

We identified the applications during a literature search as well as through discussions at conferences[1] and workshops[2]. We included an application in our analysis if it **(i)** was actively maintained at the time the data for this paper was collected (5th-13th Dec 2019[3]), **(ii)** supports publishing executable code and data that can be inspected and reused, and **(iii)** is explicitly connected to the publication process. Hence, we did not consider technologies (e.g., containerization) that alone cannot support the publication process of code because further infrastructure is needed, systems that only provide access to materials (e.g., Zenodo), or workflow systems (e.g., Taverna [23]). Based on the sample criteria, we selected the following eleven applications for the review, presented in alphabetical order (see Table 1).

---

[1] One conference we attended: EGU General Assembly 2019 (last access of this and the following URLs: 22nd May 2020); it also had a session on open science.

[2] Workshop: eLife innovation sprint 2019, which brought together people interested in open science.

[3] A reviewer directed us to the application Authorea, which we missed in our first analysis, and the lack of pricing information. We thus collected data to address these aspects on 22nd May 2020

Application	Description
Authorea	In Authorea, authors can create executable papers collaboratively. They can attach code and data to figures to make them reproducible. Authors can also directly submit to a journal and, at the same time, publish a preprint.
Binder	Binder creates a containerized executable environment based on a repository (e.g., on GitHub/Lab, Zenodo) including a Jupyter Notebook [24]. Readers can launch the analysis and inspect the workflow in a browser.
Code Ocean	Code Ocean creates "capsules" containing code, data, and the computational environment. While reading, users can execute and inspect the analysis in a separate window below the article or on Code Ocean's website [25].
eLife Reproducible Document Stack (RDS)	RDS originates from the life sciences. Authors can publish executable documents based on Stencila (https://stenci.la/), an open-source editor for articles. The executable document, which contains the whole narrative and executable code snippets, is not only a supplement but the actual scientific article.
Galaxy	Galaxy [26] provides features tailored to use cases in the life sciences. It is a web app for developing comput. analyses without programming expertise. Scientists can upload and analyze data using Jupyter Notebooks [27].
Gigantum	Gigantum packages code, data, the computational environment, and the work history into a Git repository. Gigantum is composed of a client app for creating as well as executing analyses locally and a cloud-based infrastructure for sharing computations and collaborating with peers.
Manuscripts	Manuscripts is an online tool for writing executable documents collaboratively based on the concept of literate programming, but featuring a "What you see is what you get" user interface. The runtime environment of the author is, however, not considered.
o2r	o2r [22] originates from the geosciences and addresses publishers who want to extend their infrastructure via a reproducibility service during the process of paper submission [28]. Authors can create interactive figures, allowing readers to change model parameters using a slider [29].
REANA	REANA [30, 4] originates from particle physics and provides a specification for capturing data, code, and the comput. environment. Based on this structure and manually created configuration files, REANA provides command line interface (CLI) commands to run large analyses on a remote REANA cloud.
ReproZip	ReproZip [31, 32] provides a set of CLI commands for encapsulating data, code, and the computational environment. Users can execute the resulting bundle on a server provided by ReproZip [33] or locally on different systems.
Whole Tale	With Whole Tale [34], authors can create so-called "Tales" that combine narrative, data, code, and the computational environment. Readers can inspect the materials and execute the analysis in the original environment.

Table 1: Overview of applications we included in the analysis.

Variables

In a next step, we reviewed literature to identify a set of comparison criteria (highlighted in bold in the following) relevant for the stakeholders mentioned above. According to Hrynaszkiewicz [17], publishers refrain from hosting data, raising the question of whether the applications need to be **self-hosted** by the publishers. Since self-hosting might require changes to the software, we also checked whether the applications are released under an **open license**. Next, a proxy for assessing the stage and the reliability of an application is to check whether it is already **in use**. To provide an initial estimate of the application's longevity, we looked up whether the applications are **grant-based**, since such funds are usually temporary. Also, because using literate programming tools is a frequently mentioned recommendation for creating executable documents [35], we checked whether the applications support **R Markdown** and **Jupyter Notebooks**. However, since researchers might have individual requirements [22], we also investigated whether the applications are **extensible**, meaning, for example, whether users can add a new submission format. A further relevant piece of information for authors is whether they need to **upload** their materials to the application and whether **copyright** is addressed explicitly in the documentation. Copyright is not only a main concern of authors in the context of ORR [36], but also essential when it comes to reusing research materials [37]. We also checked whether **sensitive data** can be shared. Based on the benefits of ORR summarized by Konkol et al. [36], we checked whether the applications provide tools to enable reusability, such as tools to **discover** articles, **inspect** the materials, execute the analyses (**one-click reproducible**), **manipulate** parameter values, **substitute**

datasets, and **download** the materials. Finally, papers describing open science guidelines [21] or assessing reproducibility of published papers [3] often refer to the importance of making materials permanently available, for example for future use and education [38]. Hence, we investigated whether it is possible to **modify or delete materials after publication** and whether these materials can be **shared via a DOI or URL**.

Data collection

Based on the comparison criteria, two authors collected information iteratively by investigating the project websites, applications, GitHub/Lab repositories, scientific articles, and blog posts. In the first iteration, one author of this paper gathered information on the applications one by one and took screenshots. In the second iteration, another author independently checked the information collected in the first iteration. Conflicts concerning the data were resolved through discussion amongst all authors. In order to give the scientific community, particularly the developers of the considered applications, the opportunity to comment on the analysis, we published a preprint [39] of the paper three months before submission. All collected data is available in the supplement (see Availability of data and materials). Thus, it is possible to continue the work in this paper as a "rolling review". Since some sources (e.g., documentation) were not scientific articles, we also attached links to and screenshots of the original information.

Results

Table 2 summarizes aspects relevant for publishers, editors, authors, readers, and librarians.

	Auth-orea	Bin-der	Code Ocean	eLife RDS	Gala-xy	Gigan-tum	Manu-scripts	o2r	RE-ANA	Repro-Zip	Whole Tale
Free self-hosting	-	+	-	+*	+	-	+	+	+*	+	+
Open license	-	+	-	+	+	+/-	+	+	+	+	+
In use	in use[40]	in use[2]	in use[41]	in use[42]	in use[43]	-	-	-	in use[44]	in use[31]	-
Grant-based	-	+	-	+	+	-	N/D	+	+	+	+
R Markdown	-	+	+	+	-	+	-	+	-	-	+
Jupyter Noteb.	+	+	+	+	+	+	-	-	+	+	+
Extensible	-	+	+	+	+	-	-	-	+	+	+
Upload	+	+	+	-	+	-	+	+	-	-	+
Copyright	+	N/D	+	N/D	+	+	N/D	+	N/D	N/D	+
Sensitive data	-	-	-	-	-	-	-	-	-	-	-
Discovery	+	-	+	+	+	-	-	+	-	-	+
Inspection	+	+	+	+	+	+	+	+	-	-	+
Execution	+	+	+	+	+	+	+	+	+	+	+
Manipulation	+	+	+	+	+	+	+	+	+	+	+
Substitution	-	-	-	-	-	-	-	+	-	+	-
Download	+	+	+	+	+	+	+	+	-	+	+
Modify/Delete after publishing	-	+	-	-	+	+	+	-	+	+	-
Shared via DOI	+	-	+	+	-	-	-	-	-	-	+
Shared via URL	+	+	+	+	+	+	+	+	-	+	-

Table 2: Overview of which application supports the corresponding criteria. (N/D=no data).

From the eleven applications, eight allow self-hosting for free. eLife RDS and REANA (in Table 2 marked by *) require deployment, since no free online version exists. Eight applications are released under an open license, and Gigantum has only published the client tool under an open license. The open source applications that have an online version running can be used by researchers for free. The three commercial providers, namely Authorea, Code

Ocean, and Gigantum, provide the service in exchange for payment but they also offer free subscriptions with limited features and resources (i.e. storage and computation time).

In total, seven applications are already in use, as shown by the example papers with reproducible workflows. Seven applications currently receive funding from public or private organizations[4].

Ten applications support literate programming, e.g., R Markdown or Jupyter Notebooks; the Manuscripts application supports Markdown but also code execution via embedded Jupyter Notebooks. Seven applications are extensible and can be configured to support further programming languages. Except for Code Ocean, which also supports MATLAB and Stata, all applications only support non-proprietary programming languages.

Seven applications require authors to create their projects online, whereas eLife's RDS (based on Stencila), REANA, and ReproZip allow local usage. Researchers can also work locally with Gigantum, but they then need to synchronize with the online service to access all features.

Regarding copyright, we could not find explicit information on assigning copyright for research materials in five applications. Whole Tale and Gigantum only allow open licenses, whereas Code Ocean, Galaxy, and o2r encourage them. We could not find information on sensitive data in any of the applications.

From the eleven applications, six provide a keyword-based search for papers, of which o2r provides a spatiotemporal search combined with thematic properties, such as libraries used in

[4] Further details on funding are available in the supplement.

the code. Five applications embed a programming environment (e.g., JupyterLab, RStudio) for inspecting code and data, whereas four provide their own user interface (UI).

All applications provide support for executing the analysis. REANA projects are executed via the CLI in a remote REANA cloud, and this also applies to ReproZip, which in addition to a remote cloud also provides a ReproServer for executing code online. Gigantum's local client allows users to run code in the browser. The remaining applications allow users to execute the analysis in a browser on a remote server.

Each application allows users to manipulate the code and rerun it based on a new parameter. Most commonly, users can directly manipulate the code in the browser (8 applications provide this option). In REANA and ReproZip, users can pass new parameter values via the CLI, whereas the o2r platform enables authors to configure UI widgets that allow reviewers/readers to manipulate parameter values interactively, e.g., by using a slider to change a model parameter within a defined range. Features for substituting the input datasets used in an analysis are provided by o2r and ReproZip.

While ten applications provide a feature for downloading materials, REANA projects need to be stored on a third-party service to be downloadable.

Overall, six applications allow users to modify/delete materials after publication. In Binder, REANA, and ReproZip, modifying/deleting content is possible if the research materials are stored on GitHub/Lab, but not when they are stored on Zenodo. Authorea, Code Ocean, eLife RDS, and Whole Tale assign DOIs to published content, ensuring long-term availability and making it impossible to edit these after publication. In eLife's RDS, the article is composed of text and code; thus, deleting it is equivalent to withdrawing a paper.

Discussion

Several developers have created applications for publishing computational research. One might think the applications, since they all strive for the same overall goal, resemble each other. However, we showed in this paper that the applications address different issues and needs, which increases the chances that stakeholders will find an application that best suits their individual requirements. The review can be used by the various stakeholders in different ways: Publishers who want to comply with reproducibility principles may use it to decide for a certain application, editors and program committees may use it when planning to include code review in their review process [45], applicants designing data and software management plans may use it when writing their funding proposals, and authors who are searching for tools to disseminate their work in a convincing, sustainable, and transparent manner may also find it valuable. In addition to these stakeholders, we also considered librarians, who are tasked with aspects related to the preservation and long-term accessibility of research materials. Given the variety of the stakeholders and their considerations, it is difficult to determine the best application or objectively provide a ranking. Identifying the ideal application strongly depends on the needs and goals of the stakeholders.

Hosting of the applications

Publishers need to decide whether they want to host an infrastructure by themselves or engage a provider. Applications exist for both approaches, though the majority of them can be self-hosted. Some of the self-hosting solutions are also available as online versions, but it

should be considered that these have limited resources regarding storage and processing power. Moreover, it is difficult to estimate when the projects will expire. If the applications are grant-based, they might receive follow-up funding, but this depends on the projects' success and whether they aim at carrying out research (which might come with many changes to the software) or developing a scalable and sustainable platform. As public information on all funding levels and grant durations are too uncertain and incomplete to be included in the analysis, we refrained from drawing concrete conclusions in terms of longevity and how likely they will exist in the next years.

All self-hosting solutions have an open license, allowing operators to host their own service as well as modify the software according to their individual needs and styles. A further advantage of self-hosting is the mitigation of risks regarding vendor lock-in. However, hosting one's own service means that publishers also have to provide the required technological resources and personnel. It remains unclear what kinds of costs publishers will have to expect when hosting a platform and incorporating it into their publishing infrastructure. The final costs strongly depend on the number of views, execution attempts, workflow sizes, and ease of integration into technical systems. These parameters differ between use cases and could be used as measures for future research, e.g., on stress tests. Therefore, the metrics of existing publications might provide the first ways to calculate the required resources. While the Binder instance MyBinder.org published an initial estimate regarding costs[5], further data from other services would help to calculate costs more

[5] MyBinder costs:
https://mybinder.org/v2/gh/jupyterhub/binder-billing/master?urlpath=lab/tree/analyze_data.ipynb

specifically. Moreover, it would be interesting to see usage statistics showing how often the services are used, for example, by authors, readers, and reviewers, albeit this transparency is only realistic for non-profit projects. Nevertheless, since reproducibility studies are rarely successful [5], using these services seems to be uncommon.

A further criterion we investigated was whether the applications are in use. While applications in use offer initial evidence that they work, it might take more effort to adjust them to fit a publisher's infrastructure. In contrast, beta applications can adjust their features without worrying about running instances and backwards compatibility, but the deployment of such applications might reveal new issues.

Creating executable analyses

Regarding submission formats, there is a trend toward literate programming approaches. Most applications either support Jupyter Notebooks or R Markdown, which both have proven to support reproducibility [46]. However, some journals and publishers rely on different formats, e.g., LaTeX. Since transformations to other document types are often cumbersome and adapting author requirements can be a lengthy process, it might be easier to have reproducible documents as a supplement, potentially for a transition period, until the executable documents are widely accepted. Nevertheless, eLife's RDS has already shown that combining executable code with narrative in a scientific article is possible today and comes with advantages related to communicating scientific results. For example, readers can, while studying the text, also manipulate the analysis. A limitation in this context is related to the

peer-review process. All applications require an account for creating reproducible results, and since the name of the creator is usually visible, a double-blind review is not guaranteed. However, access to code and data is particularly important for reviewers who need to recommend acceptance or rejection of a submission. One solution might be to create anonymous versions of the materials, as is possible with Open Science Framework[6], or to adopt an open peer-review process.

A further critical issue is that not all applications address copyright in their documentation. Those that do, fortunately, either require or encourage open licenses, which is a recommendation mentioned frequently in papers discussing reproducibility guidelines [21, 37]. Hence, the platforms should inform users about licenses, e.g., by referring to existing advising resources (e.g., https://choosealicense.com/). Licensing is important to enable reusability and, thus, is ideally assigned to code, data, and text separately, as is done, for example, by Authorea. Computational reproducibility is challenging also because of sensitive data. None of the applications address this issue, but it can be combined with existing solutions, such as involving trustworthy authorities [47] and cloud-based data enclaves [48].

Studying reproducible research

Being able to reproduce the computational results in a paper is a clear benefit, but open reproducible research comes with a number of further incentives [20]. Concerning the discovery of papers, most search tools provided by the applications do not take full advantage

[6] Anonymized links:
https://help.osf.io/hc/en-us/articles/360019930333-Create-a-View-only-Link-for-a-Project

of the information contained in code and data files, e.g., spatiotemporal properties. Instead, they either only provide a keyword-based search or no search at all. For inspecting materials, most solutions either provide their own UI or integrate a development environment, e.g., JupyterLab. In both cases, users can directly access, manipulate, and reuse the code. However, readers (including experienced programmers) might still struggle with understanding complex code scripts. Moreover, identifying specific parameters buried in the code and finding out how to change these can be a daunting task. The concept of nano-publications [49] or bindings [29] might help to solve these issues. A further need in this context is a UI for comparing original and manipulated figures, since differences in the figure after changing parameters might be difficult to spot. Most applications do not provide any support for substituting research components, e.g., by other input datasets, which might be due to the plethora of complex interoperability issues with respect to data formats or column names in tabular data. Only ReproZip [32] and o2r [36] provide basic means to substitute input datasets, yet they require users to ensure compatibility.

Researchers who are writing or studying computational research articles might be programming beginners or experts. Hence, while the learning curve may be either shallow or steep, it is nonetheless present. Although the applications are well documented, programming novices in particular might need to invest effort at the beginning of use. For example, they would need to learn how to write R Markdown documents and create configuration files manually. Some of the creation steps might be automated, but this usually comes at the cost of flexibility. The learning curve not only exists for authors but also for consumers,

particularly reviewers who need to verify the results and those who want to build upon the materials. Nevertheless, such an effort only needs to be invested once and will eventually pay off in the form of more convincing and transparent research.

Sharing computational research

The state of the research materials is an essential issue when it comes to publication. While some applications fix the state of the research materials by assigning a DOI and archiving a snapshot, others allow changing and deleting them. This is a disadvantage with respect to reproducibility since verifiability and accessibility are lost. In addition, if self-hosting is not possible, the computational analysis of an article will be executable only as long as the project and its infrastructure exist; this dependence is a crucial aspect with respect to archiving. However, this issue can be mitigated if researchers “go the extra mile” and also publish their materials in long-term repositories in addition to an executable version using one of the applications.

A further dependence is the technology underlying the infrastructure. For example, without the Docker container runtime, the captured computing environment might not work even though it remains human readable [50]. This is also true for source code scripts, which are plain text files and, thus, can be opened using any editor, even if they cannot be compiled and executed. These examples demonstrate the importance of using open and text-based file formats instead of proprietary and binary file formats in science.

Limitations

This work is subject to a number of limitations. The scope of this review is narrow and does not cover all applications that are connected with computational research (e.g., workflow systems, such as Taverna [23]). Also, we have no access to publishers' actual systems, preventing us from being able to evaluate the usability of APIs and documentation and how easy they can be incorporated into existing infrastructures. In addition, this review is a snapshot of the highly dynamic area of publishing infrastructures. Hence, the information might become outdated quickly, e.g., an application might extend the set of functionalities or be discontinued. Still, reviewing the current state of the landscape to reflect on available options might be helpful for researchers. Furthermore, the properties we investigated in this survey do not cover all possible aspects and discipline-specific needs, but, nevertheless, stakeholders requiring more information can use the overview as a starting point for further research. Also, we collected and interpreted the data ourselves and did not contact the application developers, which might have increased the accuracy of the data. Finally, our evaluation only considered documented features. However, programmers with sufficient expertise can build upon the open source applications and implement missing features.

Conclusions

Several projects aim at tackling issues related to computational reproducibility by designing applications that assist researchers in publishing and studying executable research results. The key contribution of this paper was a survey of such applications' benefits and limitations, which can be used by stakeholders involved in scholarly publication to help them make

decisions. Publishers have the choice between using provided services or self-hosting solutions, but more data is needed to estimate the costs for publishers to maintain their own infrastructure. Although most applications support submissions in the form of Jupyter Notebooks or R Markdown documents, this trend toward literate programming requires authors to adjust their workflows and requires journals to adapt their guidelines. These changes require time, which could be bridged by sharing notebooks in the supplements. The applications we reviewed indicate that further incentives can be realized if the research findings are open and reproducible. For example, reviewers and readers can inspect an analysis in detail and gain a deeper understanding by manipulating the assumptions underlying the analysis. The survey also addressed aspects relevant for archiving. We found that several applications allow users to change the materials after publication. This might result in conflicts between the version referred to in an article and the available version, which might have been changed since the article was first published.

The applications can be used to investigate several research questions: How does a reproducibility attempt (failed or successful) affect a reviewer's decision? How much effort is needed to review reproducible papers compared to traditional articles? Will reviewers refuse reviews if they fear additional work? Answering these questions will help to improve the applications and eventually foster the success of open reproducible research. To find these answers more quickly, it might be helpful to establish a collaboration of commercial and independent publishers on open infrastructures for reproducible and transparent scholarly communication[7].

[7] https://investinopen.org/

List of abbreviations

CLI = Command line interface

DOI = Digital Object Identifier

o2r = Opening reproducible research (project)

ORR = Open reproducible researchers

REANA = Reproducible Analysis (project)

RDS = Reproducibility Document Stack

UI = User Interface

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability of data and materials

The data is openly available on Zenodo: https://zenodo.org/record/3839753. The repository includes a list of all applications we looked at and, for excluded applications, the reasons for exclusion.

Competing interests

The authors of this paper are members of the o2r project that was also discussed in this paper (http://o2r.info/).

Funding

This work is supported by the project Opening Reproducible Research 2 (https://www.uni-muenster.de/forschungaz/project/12343) funded by the German Research Foundation (DFG) under project numbers KR 3930/8-1; TR 864/12-1; PE 1632/17-1. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Authors' contributions

Markus Konkol wrote the paper, collected the data, and conceptualized the analysis. Daniel Nüst wrote the paper. Laura Goullier collected data and wrote the paper. All authors discussed the results and approved the final manuscript.

Acknowledgements

Not applicable. All contributors are listed as authors. We thank Timothy Errington and Mario Malicki for their reviews, Vicky Steeves for a helpful discussion on the article preprint [63], and Celeste Brennecka for proofreading.

References

1. Stodden, V., McNutt, M., Bailey, D. H., Deelman, E., Gil, Y., Hanson, B., … Taufer, M. (2016). Enhancing reproducibility for computational methods. *Science*, 354(6317), 1240–1241. doi:10.1126/science.aah6168

2. Stagge, J. H., Rosenberg, D. E., Abdallah, A. M., Akbar, H., Attallah, N. A., & James, R. (2019). Assessing data availability and research reproducibility in hydrology and water resources. *Scientific Data*, 6(1). doi:10.1038/sdata.2019.30
3. Nüst, D., Granell, C., Hofer, B., Konkol, M., Ostermann, F. O., Sileryte, R., & Cerutti, V. (2018). Reproducible research and GIScience: an evaluation using AGILE conference papers. *PeerJ*, *6*, e5072. doi:10.7287/peerj.preprints.26561
4. Chen, X., Dallmeier-Tiessen, S., Dasler, R., Feger, S., Fokianos, P., Benito Gonzalez, J., Hirvonsalo, H. et al. (2018). Open Is Not Enough. *Nature Physics* 15 (2). 113–19. doi:10.1038/s41567-018-0342-2
5. Konkol, M., Kray, C., & Pfeiffer, M. (2018). Computational reproducibility in geoscientific papers: Insights from a series of studies with geoscientists and a reproduction study. *International Journal of Geographical Information Science*, 33(2), 408–429. doi:10.1080/13658816.2018.1508687
6. Herndon, T., Ash, M., & Pollin, R. (2013). Does high public debt consistently stifle economic growth? A critique of Reinhart and Rogoff. *Cambridge Journal of Economics*, 38(2), 257–279. doi:10.1093/cje/bet075
7. National Academies of Sciences, Engineering, Medicine & others (2019). Reproducibility and Replicability in Science. National Academies Press.
8. Markowetz, F. (2015). Five selfish reasons to work reproducibly. *Genome Biology*, 16(1). doi:10.1186/s13059-015-0850-7
9. McKiernan, E. C., Bourne, P. E., Brown, C. T., Buck, S., Kenall, A., Lin, J., … Yarkoni, T. (2016). Author response: How open science helps researchers succeed. doi:10.7554/elife.16800.008
10. Stark, P. B. (2018). Before reproducibility must come preproducibility. *Nature*, 557(7707), 613–613. doi:10.1038/d41586-018-05256-0
11. Vazire, S. (2020). A toast to the error detectors. Nature.
12. Bailey, D. H., Borwein, J. M., & Stodden, V. (2016). Facilitating Reproducibility in Scientific Computing: Principles and Practice. *Reproducibility: Principles, Problems, Practices, and Prospects*, 205–231. doi:10.1002/9781118865064.ch9
13. Donoho, D. L. (2010). An invitation to reproducible computational research. *Biostatistics*, 11(3), 385–388. doi:10.1093/biostatistics/kxq028
14. Powers, S. M. & Hampton, S. E. (2018). Open science, reproducibility, and transparency in ecology. *Ecological Applications*, 29(1). doi:10.1002/eap.1822
15. Piwowar, H. (2007). Sharing Detailed Research Data Is Associated with Increased Citation Rate. *Nature Precedings*. doi:10.1038/npre.2007.361.1
16. Nüst, D., Ostermann, F. O. Sileryte, R., Hofer, B., Granell, C., Teperek, M., Graser, A., Broman, K.W., & Hettne, K. M. (2019). AGILE Reproducible Paper Guidelines. doi:10.17605/OSF.IO/CB7Z8
17. Hrynaszkiewicz, I. (2019). Publishers' Responsibilities in Promoting Data Quality and Reproducibility. *Handbook of Experimental Pharmacology*. Doi:10.1007/164_2019_290
18. Gentleman, R. & Temple Lang, D. (2007). Statistical Analyses and Reproducible Research. *Journal of Computational and Graphical Statistics*, 16(1), 1–23. doi:10.1198/106186007x178663
19. Barba, L. A. (2018). Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311*

20. Munafò, M. R., Nosek, B. A., Bishop, D., Button, K. S., Chambers, C. D., Sert, N. P., Simonsohn, U., Wagenmakers, E-J., Ware, J. J., & Ioannidis, J. PA. (2017). A Manifesto for Reproducible Science. *Nature Human Behaviour 1* (1). doi:10.1038/s41562-016-0021
21. Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., ... & Contestabile, M. (2015). Promoting an open research culture. *Science*, *348*(6242), 1422-1425.
22. Nüst, D., Konkol, M., Pebesma, E., Kray, C., Schutzeichel, M., Przibytzin, H., & Lorenz, J. (2017). Opening the Publication Process with Executable Research Compendia. *D-Lib Magazine*, 23(1/2). doi:10.1045/january2017-nuest
23. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., … Goble, C. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1), W557–W561. doi:10.1093/nar/gkt328
24. Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., … Willing, C. (2018). Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. *Proceedings of the 17th Python in Science Conference*. doi:10.25080/majora-4af1f417-011
25. Clyburne-Sherin, A., Fei, X., & Green, S. A. (2019). Computational Reproducibility via Containers in Social Psychology. *Meta-Psychology 3*. doi:10.15626/MP.2018.892
26. Goecks, J., Nekrutenko, A., Taylor, J., & Galaxy Team, T. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8), R86. doi:10.1186/gb-2010-11-8-r86
27. Grüning, B. A., Rasche, E., Rebolledo-Jaramillo, B., Eberhard, C., Houwaart, T., Chilton, J., … Nekrutenko, A. (2017). Jupyter and Galaxy: Easing entry barriers into complex data analyses for biomedical researchers. *PLOS Computational Biology*, 13(5), e1005425. doi:10.1371/journal.pcbi.1005425
28. Nüst, D. (2018). Reproducibility Service for Executable Research Compendia: Technical Specifications and Reference Implementation (version 1.0.0). *Zenodo*. doi:10.5281/zenodo.2203844
29. Konkol, M., Kray, C., & Suleiman, J. (2019). Creating Interactive Scientific Publications Using Bindings. *Proceedings of the ACM on Human-Computer Interaction,* 1–18. doi:10.1145/3331158
30. Šimko, T., Heinrich, L., Hirvonsalo, H., Kousidis, D., & Rodríguez, D. (2019). REANA: A System for Reusable Research Data Analyses. *EPJ Web of Conferences*, 214, 06034. doi:10.1051/epjconf/201921406034
31. Steeves, V., Rampin, R., & Chirigati, F. (2017). Using ReproZip for Reproducibility and Library Services. *IASSIST Quarterly*, 42(1), 14. doi:10.29173/iq18
32. Chirigati, F., Doraiswamy, H., Damoulas, T., & Freire, J. (2016). Data Polygamy. *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*. doi:10.1145/2882903.2915245
33. Rampin, R., Chirigati, F., Steeves, V., & Freire, J. (2018). ReproServer: Making Reproducibility Easier and Less Intensive. arXiv Preprint arXiv:1808.01406.
34. Brinckman, A., Chard, K., Gaffney, N., Hategan, M., Jones, M. B., Kowalik, K., Stodden, V., Turner, K. et al. (2019). Computing environments for reproducibility: Capturing the "Whole Tale." *Future Generation Computer Systems*, 94, 854–867. doi:10.1016/j.future.2017.12.029
35. Peng, R. D., Dominici, F., & Zeger, S. L. (2006). Reproducible epidemiologic research. *American journal of epidemiology*, *163*(9), 783-789.

36. Konkol, M., & Kray, C. (2018). In-depth examination of spatiotemporal figures in open reproducible research. *Cartography and Geographic Information Science*, 46(5), 412–427. doi:10.1080/15230406.2018.1512421
37. Stodden, V. (2009). The Legal Framework for Reproducible Scientific Research: Licensing and Copyright. *Computing in Science & Engineering*, 11(1), 35–40. doi:10.1109/mcse.2009.19
38. Sayre, F., & Riegelman, A. (2019). Replicable Services for Reproducible Research: A Model for Academic Libraries. *College & Research Libraries, 80*(2), 260. doi:10.5860/crl.80.2.260
39. Konkol, M., Nüst, D., and Goulier, L., 2020. Publishing computational research - A review of infrastructures for reproducible and transparent scholarly communication. arXiv preprint arXivarXiv:2001.00484.
40. Hanwell, M. D., Harris, C., Genova, A., et al. (2020). Open Chemistry, JupyterLab, REST, and Quantum Chemistry. *Authorea*. doi:10.22541/au.158687268.81852407
41. Chitre, M. (2018). Editorial On Writing Reproducible and Interactive Papers. *IEEE Journal of Oceanic Engineering*, 43(3), 560–562. doi:10.1109/joe.2018.2848058
42. Lewis, L. M., Edwards, M. C., Meyers, Z. R., Talbot Jr, C. C., Hao, H., Blum, D., & others. (2018). Replication Study: Transcriptional Amplification in Tumor Cells with Elevated c-Myc. *Cancer Biology 7*. doi:10.7554/eLife.30274
43. Ide, N., Suderman, K., Verhagen, M., & Pustejovsky, J. (2016). The Language Application Grid Web Service Exchange Vocabulary. *Lecture Notes in Computer Science*, 18–32. doi:10.1007/978-3-319-31468-6_2
44. Prelipcean, D. (2019). Physics Examples for Reproducible Analysis. *CERN*. https://cds.cern.ch/record/2690231
45. Eglen, S. & Nüst, D. (2019). CODECHECK: An open-science initiative to facilitate sharing of computer programs and results presented in scientific publications. *Septentrio Conference Series*, (1). doi:10.7557/5.4910
46. Grüning, B., Chilton, J., Köster, J., Dale, R., Soranzo, N., van den Beek, M., … Taylor, J. (2018). Practical Computational Reproducibility in the Life Sciences. *Cell Systems*, 6(6), 631–635. doi:10.1016/j.cels.2018.03.014
47. Pérignon, C., Gadouche, K., Hurlin, C., Silberman, R., & Debonnel, E. (2019). Certify Reproducibility with Confidential Data. *Science* 365 (6449). doi:10.1126/science.aaw2825
48. Foster, I. (2017). Research Infrastructure for the Safe Analysis of Sensitive Data. *The ANNALS of the American Academy of Political and Social Science*, 675(1), 102–120. doi:10.1177/0002716217742610
49. Kuhn, T., Chichester, C., Krauthammer, M., Queralt-Rosinach, N., Verborgh, R., Giannakopoulos, G., … Dumontier, M. (2016). Decentralized provenance-aware publishing with nanopublications. *PeerJ Computer Science*, 2, e78. doi:10.7717/peerj-cs.78
50. Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1), 71–79. doi:10.1145/2723872.2723882