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

Funding agencies increasingly ask applicants to include data and software management plans into proposals. In addition, the author guidelines of scientific journals and conferences more often include a statement on data availability, and some reviewers reject unreproducible submissions. This trend towards open science increases the pressure on authors to provide access to the source code and data underlying the computational results in their scientific papers. Still, publishing reproducible articles is a demanding task and not achieved simply by providing access to code scripts and data files. Consequently, several projects develop solutions to support the publication of executable analyses alongside articles considering the needs of the aforementioned stakeholders. The key contribution of this paper is a review of applications addressing the issue of publishing

executable computational research results. The review can support publishers to decide which system to integrate into their submission process, editors to recommend tools for researchers, and authors of scientific papers to adhere to reproducibility principles.

Materials and methods

Our analysis considered 10 applications that **(i)** were actively maintained at the time the data for this paper was collected, **(ii)** support the publication of executable code and data, **(iii)** are connected to the publication process. By investigating the software documentation and published articles, we compared the applications considering the needs of the stakeholders involved in the scholarly publication process (publishers, editors, authors, reviewers, readers, and librarians) across a set of criteria (e.g., hosting options, submission types, and functionalities).

Results

Most applications allow publishers to self-host the system, but online versions and commercial providers exist as well. Executable analyses are predominantly submitted using literate programming approaches, such as Jupyter Notebooks. All approaches provide further features, e.g., one-click reproducible results, but most applications also allow modifying/deleting materials after publication which is critical for archiving.

Conclusions

The applications all provide a rich set of functionalities and address many reproducibility issues. However, issues related to ethics, privacy, big data, costs, and long computation times are not fully solved, yet.

Background

Many scientific articles report on results based on computations, e.g., a statistical analysis implemented in R. Publishing the used source code and data to adhere to open reproducible research (ORR) principles (i.e., public access to code and data underlying the reported results [1]) seems simple. However, several studies concluded that papers rarely link to these materials [2, 3]. Moreover, due to technical challenges, e.g., capturing the original computational environment of the analyst, even accessible materials do not guarantee reproducibility [4, 5]. These issues have several implications [6]: It is difficult (often even impossible) to find errors within the analysis, but publishing erroneous papers can damage an author's reputation [7] as well as trust in science [8]. Also, reviewers cannot verify the results, because they need to understand the analysis just by reading the text [9]. Furthermore, other researchers cannot build upon existing work but have to collect data and implement the analysis from scratch [10]. Finally, libraries cannot preserve the materials for future use or education. These issues are also to society's disadvantage as it cannot benefit fully from publicly funded research [11]. Fortunately, funding bodies, e.g., Horizon 2020 increasingly consider data and software management plans as part of grant proposals. Accordingly, more editors add a section on code and data availability into their author guidelines [12, 13], and reviewers consider reproducibility in their decision process [14]. Nevertheless, these cultural and systematic developments [15] alone do not solve the plethora of reproducibility issues. Authors often do not know how to fulfill the requirements of funding bodies and journals, such as the TOP guidelines [16]. 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 projects work on solutions to support the publication of executable supplements. The key contribution of this paper is a review of applications that support the publication of executable computational research for transparent and reproducible research. This review can be used as decision support by publishers who want to comply with reproducibility principles, editors and

programme committees planning to adopt reproducibility requirements in the author guidelines and integrate code evaluation in their review process [17], applicants in the process of creating data and software management plans for their funding proposals, and authors searching for tools to disseminate their work in a convincing, sustainable, and effective manner. We also consider aspects related to preservation relevant for librarians dealing with long-term accessibility of research materials. Based on the survey, we critically discuss trends and limitations in the area of reproducible research infrastructures.

Scope: This work focuses on applications that support the publication of research results based on executable source code scripts (e.g., R or Python) and the underlying data. Hence, we did not consider workflow systems (e.g., Taverna [18]) or online repositories (e.g., Open Science Framework, https://osf.io/, last access for this and the following URLs: 20th Dec 19). Also, this paper does not discuss how to work reproducibly since this is covered already in literature, e.g., by [19, 20, 21, 22]. The review is a snapshot of the highly dynamic area of publishing infrastructures. Hence, some of the collected information might become outdated, 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 is helpful for publishers, editors, reviewers, authors, and librarians. All collected data is available in the supplements (see Availability of data and materials). The paper is structured as follows: First, we survey fundamental concepts and tools underlying the applications. We then introduce each application and the comparison criteria followed by the actual comparison. The paper concludes by a discussion about the observations we made, trends, and limitations.

Fundamental concepts and tools

Before reviewing applications for publishing reproducible research, we briefly survey fundamental concepts and tools that underpin the applications. This overview is needed to understand how the applications work and what the limitations are.

Packaging computational research reproducibly

The traditional research article alone is not sufficient to communicate a complex computational analysis [23]. To address this issue, computational reproducibility concerns the publication of code and data underlying a research paper. This form of publishing research allows reviewers to verify the reported results and readers to reuse the materials [24]. To achieve that, all materials are needed, including not only the data and code but also the computational environment. A basic concept for such a collection is the research compendium, a “mechanism that combines text, data, and auxiliary software into a distributable and executable unit” [25]. The concept was extended by a description and snapshot of the software environment using containerization resulting in the executable research compendium [26]. Containerization and virtualization are mechanisms to capture the full software stack of a computational environment, including all software dependencies in a portable snapshot [27]. In contrast to containerization, virtualization also includes the operating system kernel. Despite this difference, both approaches have proven to improve transparency and reproducibility [28, 29]. One containerization technology is Docker, which is based on so-called Dockerfiles, human and machine readable recipes to create the image of a virtual environment [28]. These recipes add an additional layer of documentation making Docker a popular tool in the area of computational reproducibility [30]. A research compendium should contain an entry point, i.e., a main file that needs to be executed to run the entire analysis. One option to realize these entry points is the concept of literate programming, an approach for interweaving source code and text in one notebook [31]. Two popular realizations of such notebooks are Jupyter Notebooks [32] and R Markdown [33]. Combining source code and data in one document is advantageous over other approaches, such as having code scripts and the article separated, which might result in inconsistencies between the two. A further advantage is the possibility to execute the analysis with a single click, so called one-click-reproduce [34]. This form of making computational results

available lowers the barrier for others to reproduce the results and thus increases trust and transparency of computer-based research.

Licensing and Citation

Appropriate licensing of research components is crucial yet complex, as copyright laws differ between component types, e.g., data, software, and text [35]. This is particularly important when it comes to reusing research components, which is one of the main goals of research compendia. A further level of complexity emerges if research compendia include, for example, parts of the data and the code of several already published papers. A typical use case is reusing code of a specific version published in a repository, while the same code is developed and stored on a public repository (e.g., GitLab). Besides conscious handling of licenses and copyrights, building on top of the work of others requires adequate citations. This can be supported by connecting the research components with the help of metadata including permanent and global identifiers, e.g., digital object identifier (DOIs) [1], which can be also used for data [36] and software [37].

Ethical and technical issues

Frequently mentioned issues related to computational reproducibility concern sensitive data and large data files. To tackle the issue of sensitive data, a first step would be to anonymize the data. Another option is to involve a trustworthy authority which ensures that the results in the article can be achieved based on the used data [38]. In this case, public access is not required. To ensure that these solutions are not exploited, authors should argue why hiding or providing synthetic data is required and reviewers can then decide whether the reasons are valid. A further solution is the concept of cloud-based data enclaves, which provide data access only to authorized persons [39]. Such approaches for access control could be connected with the applications discussed in this paper. Large data files, e.g., global remote sensing datasets quickly reach several petabytes. However, a large number of papers are based on datasets that can be stored on public and free data repositories,

such as Open Science Framework (file size limit only for individual files, https://help.osf.io/hc/en-us/articles/360019737894-FAQs#what-is-the-individual-file-size-limit) or Zenodo (max. 50GB by default, extension possible, https://help.zenodo.org/whatsnew/). Further limiting factors are long computation times and the need for specialized hardware, such as high-performance computing clusters [40].

Materials and Methods

To obtain an overview of what the applications supporting the publication of reproducible analyses provide as well as trends and limitations, we compared them across a set of criteria.

Materials

To ensure that the stakeholders receive current recommendations, we considered an application as part of our analysis if **(i)** it was actively maintained at the time the data for this paper was collected (5th-13th Dec 2019), **(ii)** it supported publishing executable code and data which can be inspected and reused, and **(iii)** the application was explicitly connected to the publication process. Hence, we did not consider technologies that alone cannot support the publication process of code and data as further infrastructure is needed (e.g., Docker) or applications that only provide access to data or code (e.g., Zenodo). We found the applications during literature research and discussions at conferences or workshops.

Applications

Based on the sample criteria, ten applications were selected for the review. In the following, we briefly introduce them in alphabetical order.

Researchers having a repository (e.g., on GitHub/Lab, Zenodo) including, e.g., a Jupyter Notebook can use **Binder** (https://mybinder.org/) to make it available in an executable environment [41].

Readers can launch the analysis from a Binder-ready repository and inspect the workflow in a browser. Binder creates a containerized environment from a repository based on configuration files. In **Code Ocean** [42], authors can create so-called "capsules" which contain code, data, and the computational environment including the version of the operating system and dependencies. Readers can, while studying the article, execute and inspect the analysis in a separate window below the online version of the article or on Code Ocean's website. The **eLife Reproducible Document Stack** (RDS, https://elifesciences.org/labs/b521cf4d/reproducible-document-stack-towards-a-scalable-solution-for-reproducible-articles) enables authors to 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** [43] is a web-based application for developing computational analyses without programming expertise. Scientists can upload and analyze data by using Jupyter Notebooks [44]. **Gigantum** (https://gigantum.com/) builds on top of Git and packages code, data, the computational environment, and the work history into a Git repository. Gigantum is composed of a client application for creating as well as executing analyses locally, and a cloud-based infrastructure for sharing computations and collaborating with peers. **Manuscripts** (https://www.manuscripts.io/about/) 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** [26] addresses publishers who want to extend their existing infrastructure by a reproducibility service during the process of paper submission [45]. Authors can also create interactive figures, allowing reviewers and readers to check the robustness of the results, e.g., by changing model parameters using a slider [46]. **REANA** [47, 4] provides a formal specification to guide authors through the process of capturing input datasets, code, and the computational environment. Based on this structure and after creating some configuration files manually, REANA provides a set of

command line interface (CLI) commands to run large analyses on a remote REANA cloud. **ReproZip** [48, 49] provides a set of CLI commands for encapsulating data, code, and the computational environment automatically. Users can execute the resulting bundle on a server provided by ReproZip [50] or locally on different computer systems. With **Whole Tale** [51], 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.

Rationale for the comparison criteria

We identified the comparison criteria considering the needs of stakeholders of the scholarly publication process described by [26], i.e., those of publishers, editors, authors, reviewers, readers, and librarians. There is some overlap regarding stakeholder needs, for example, publishers as well as authors aim at attracting readers and providing a convenient reading experience for reviewers.

Publishers need to know whether they can integrate the application into their existing infrastructure. The applications can be either made available as open source tools for own hosting or as a service hosted by the provider. If the tool is available for free under an open license, publishers only have to consider costs for maintaining the infrastructure. Moreover, publishers gain full control and can customize the interface or processes according to their own specifications. In case of a paid service, publishers can take advantage of not being responsible for the maintenance. A further criterion relevant for publishers is the development stage of the application, i.e., if it was already used in published articles.

Editors of journals need to ensure that a service for publishing reproducible research is consistent with the tools the authors typically use and common practices in their scientific field. For example, journals regularly receiving submissions containing Jupyter Notebooks should not choose a service that supports only R Markdown. This aspect might also affect the author and reviewer guidelines, for which the editors are responsible. A further relevant aspect is the addressed research area. Some

applications might address specific fields and thus provide features tailored to domain-specific requirements.

Authors need to submit research materials efficiently. Hence, we checked how authors can upload their files and which submission formats and programming languages are supported. We also considered which license submitted materials receive, since this is a frequently mentioned aspect of papers discussing reproducibility guidelines [1]. Although licensing is relevant for all stakeholders, authors are particularly responsible for taking care of it. We also checked whether the applications can deal with sensitive data.

For **readers** and **reviewers**, open reproducible research comes with several benefits, such as advanced search capabilities, re-running workflows, inspecting results in detail (i.e. looking at code or data files), modifying parameter settings, and reusing the data or the analysis for the own work [52]. We thus checked whether the tools provide any specific support for such investigations of the research materials.

Librarians are tasked with preserving research materials. We checked how the materials are stored and shared, and if modifying or deleting them after publication is possible.

Based on these comparison criteria, we investigated the project websites, the actual applications, GitHub/Lab repositories, scientific articles (if available), and blog posts. Since most of the sources were not scientific articles, the supplements contain screenshots and URLs to show where we found the corresponding information.

Results

In the following, we compare the applications considering the needs of the stakeholders.

Table 1 summarizes aspects relevant for publishers, i.e., if self-hosting is possible, which license is assigned to the application, whether it is already in use or in a beta stage, and the funding source. From the ten applications, eight allow self-hosting. Code Ocean and Gigantum provide the service

themselves. eLife RDS, o2r, and REANA (in Table 1 marked by *) require own installations since no free online deployments exist. Three applications are released under the BSD-3-Clause License, three under MIT of which Gigantum assigned this license to the local tool and not to the cloud service, one under Apache 2.0, one under the CPAL 1.0, and one under Academic Free License 3.0. These licenses allow operators to host their own service as well as to modify the software according to their individual needs and styles. This means, however, they also have to maintain the infrastructure and provide the required technological resources as well as personnel. In contrast, Code Ocean's infrastructure and Gigantum's cloud service are provided in exchange for payment. From the reviewed applications, four are rather experimental and six are already in use as shown by the example papers with workflows based on the corresponding application. Seven applications receive funding from public or private science foundations. Code Ocean and Gigantum offer a commercial service.

Table 2 summarizes aspects relevant for editors and authors, i.e., the scientific domains, supported submission formats, upload mechanisms, and license terms. Although none of the investigated applications are strictly tied to a specific domain, we observed that some of them focus on particular areas. For example, Galaxy provides a rich set of features tailored to use cases in the life sciences. Other applications originate from a particular domain, e.g., eLife's RDS comes from the life sciences whereas REANA focuses on particle physics. From the ten applications, nine support literate programming approaches by default, e.g., Jupyter Notebook or R Markdown. Manuscripts supports Markdown, but also code execution via embedded Jupyter Notebooks. Seven applications are extensible and provide the possibility to configure the application to support further submission formats or programming languages. Except for Code Ocean which also supports MATLAB and Stata, all applications only support non-proprietary programming languages. For making code and data available on the platform, five applications provide file upload. Five applications provide the possibility to upload materials via an external cloud or repository, e.g., Zenodo.

Table 1: Overview of properties relevant for publishers, i.e., if self-hosting is possible (* denotes **only** self-hosting is possible), which license the applications have, the stage of the project (in use or beta), and the funding source.

	Self-hosting	**Open license**	**Stage**	**Funding**
Binder	yes	BSD 3-Clause "New" or "Revised"	in use by [2]	Moore Foundation Google Cloud Platform
Code Ocean	no	Commercial application	in use by [53]	commercial
eLife RDS	yes*	MIT	in use by [54]	Howard Hughes Medic. Inst, Max Planck Society, Wellcome Trust, Knut and Alice Wallenberg Foundation
Galaxy	yes	Academic Free 3.0	in use by [55]	National Institutes of Health, National Science Foundation, Penn State, Johns Hopkins, and the Pennsylvania Department of Public Health
Gigantum	no	MIT	beta	commercial
Manuscripts	yes	CPAL-1.0	beta	no information available
o2r	yes*	Apache 2.0	beta	DFG (German Funding Agency)
REANA	yes*	MIT	in use by [56]	CERN, National Science Foundation
ReproZip	yes	BSD 3-Clause "New" or "Revised"	in use by [49]	Moore and Sloan Foundation
Whole Tale	yes	BSD 3-Clause "New" or "Revised"	beta	National Science Foundation

However, uploading materials might be disadvantageous for papers based on large data files. For these cases, eLife's RDS (based on Stencila), REANA, and ReproZip allow local usage. Researchers can also work locally with Gigantum, but then need to synchronize with the online service to access all features. Despite the importance of licensing, we could not find information on copyright for research materials in four applications. Whole Tale and Gigantum only allow open

licenses whereas Code Ocean, Galaxy, and o2r encourage it. eLife assigns an open license to the article text only.

Table 2: Overview of aspects relevant for editors and authors, i.e., the addressed research area, supported submission formats, how authors can upload materials, and copyright.

	Research area	**Submission formats/ Program. languages**	**Upload**	**Copyright**
Binder	all	R Markdown, Jupyter Notebooks, extensible	via URL/DOI from Git(Hub/Lab), Gist, Zenodo, Figshare, Dataverse	no information found
Code Ocean	all	R Markdown, Jupyter Notebooks, C/C++, Fortran, Java, Lua, MATLAB, Stata, extensible	File upload, via URL from Git repository	self-determined, MIT for code/ CC0 for data encouraged
eLife RDS	all, focus on life sciences	R Markdown, Jupyter Notebooks, Markdown, Excel, Word, LaTeX, JATS, extensible	created locally using Stencila	CC-BY for text, for code/data not discussed
Galaxy	all, focus on life sciences	Jupyter Notebooks, extensible	File upload, FTP, SRA	encourage open license for software
Gigantum	all	R Markdown, Jupyter Notebooks	Synchronization	self-determined but has to be open
Manuscripts	all	Markdown, Word, Latex, JATS, R, Julia, Python	File upload	no information found
o2r	all, focus on geosciences	R Markdown	File upload, ownCloud	self-determined but open is encouraged
REANA	all, focus on particle physics	Jupyter Notebooks, extensible	created locally	no information found
ReproZip	all	Jupyter Notebooks, extensible	created locally	no information found
Whole Tale	all	R Markdown, Jupyter Notebooks, extensible	File upload, URL/DOI from DataOne/ Dataverse, Materials Data Facility	self-determined but has to be open

Table 3 summarizes aspects relevant for reviewers and readers. From the ten applications, five provide a keyword-based search for papers whereas five do not provide any search feature. o2r provides a spatiotemporal search combined with thematic properties, such as libraries used in the code. Nine applications provide tools for inspecting code and data, six of them by providing an own user interface (UI) and three by embedding a programming environment (e.g., JupyterLab, RStudio). Though REANA does not provide supportive tools for inspection, the materials can be viewed when stored on public repositories, e.g., GitLab. Nine applications provide tools for downloading materials. Projects created with REANA can be downloaded if stored on public repositories which already provide a download functionality. Eight applications allow readers to execute the analysis in the browser on a remote server. Gigantum provides a UI running locally, REANA projects are executed via the CLI in a remote REANA cloud. Each application allows manipulating the code and rerunning it based on a new parameter. Most commonly, users can directly manipulate the code in the browser (6 applications provide this option) or locally (Gigantum). In REANA, users can pass new parameter values via the CLI, in ReproZip via the CLI or input fields using ReproServer. The o2r platform allows authors to configure UI widgets giving reviewers/readers the chance to interactively manipulate parameter values, e.g., by using a slider to change a model parameter within a certain range.

Table 4 addresses libraries and other institutions with a mandate to preserve and provide access to research outputs. It includes information on how the research materials are stored and shared, and whether modifying or deleting content once published is possible. Five applications provide storage, though it remains unclear whether they run the servers by themselves or by third-party services, and what kind of backup and archiving is implemented. Seven applications give hosts the option to store research materials independently, e.g., on the publisher's infrastructure. The free available instance of Binder, MyBinder.org (https://mybinder.org/), stores Docker images temporarily but beyond that,

Table 3: Overview of features relevant for reviewers and readers, i.e., searching for papers and materials, inspecting code and data, downloading materials, executing the analysis, and manipulating the code.

	Searching	**Inspection**	**Download**	**Execution**	**Manipulation**
Binder	no support	within UI of JupyterLab in browser	via UI	within UI in browser	manually within code in browser
Code Ocean	keyword-based	below article, or in UI of Code Ocean	via UI	within UI in browser	manually within code in browser
eLife RDS	keyword-based	within article in browser	via UI	within UI in browser	manually within code in browser
Galaxy	keyword-based	within UI of JupyterLab in browser	via UI	within UI in browser	manually within code in browser
Gigantum	no support	within UI of local installation	via UI	within UI of local installation	within UI of local installation
Manuscripts	no support	within UI of Manuscripts	via UI	within UI in browser	manually within code in browser
o2r	spatiotemporal and keyword-based search	within UI of o2r	via UI	within UI in browser	using UI widgets
REANA	no support	no support	no support	via CLI in remote Reana cloud	manually via CLI
Repro.zip	no support	within UI of ReproServer	via UI	locally via CLI, within UI in browser	manually via CLI/ input fields in browser
Whole Tale	keyword-based	within UI of JupyterLab/RStudio in browser	via UI	within UI in browser or locally	manually within code in browser

no storage is provided. Whole Tale and o2r use existing long-term preservation services, e.g., Zenodo and DataOne. Regarding the possibility to modify or delete materials once published, we assigned “possible” if there is any way to do so. In Binder, REANA, and ReproZip,

modifying/deleting content is possible if the research materials are stored on GitHub/Lab, but not when stored on Zenodo. The same is true for Galaxy, Gigantum, and Manuscripts, which allow users to edit/delete contents stored in the cloud. Code Ocean and Whole Tale assign DOIs to published contents making it impossible to edit these after publication. The same applies to o2r but only if the materials are archived. Finally, in eLife's RDS, the article is composed of text and code. Deleting it is thus equivalent to withdrawing a paper. All in all, seven applications allow modifying published materials. However, this issue is mitigated when researchers "go the extra mile" and also publish their materials in long-term repositories, such as Zenodo. One exception is Code Ocean which allows modifications (no deletions) but assigns a new DOI to the modified content. Finally, authors need a way to share reproducible results in their paper. This is possible via a URL/DOI to the application (eight applications provide this possibility) or a URL to an online repository (2).

Table 4: Overview of properties relevant for long-term preservation, i.e., how the research materials are stored, if they can be modified/deleted after publication, and how they can be shared in articles.

	Storing	**Modify/Delete after publication**	**Sharing**
Binder	by host	possible	URL to Binder instance
Code Ocean	provided	not possible	URL/DOI to Code Ocean
eLife RDS	by host	not possible	URL/DOI to eLife
Galaxy	provided, by host	possible	URL to Galaxy
Gigantum	provided	possible	URL to Gigantum
Manuscripts	provided, by host	possible	URL to Manuscripts
o2r	by host, Zenodo	possible	URL to o2r
REANA	by host	possible	URL to online repository
Repro.zip	provided, by host	possible	URL to ReproServer
Whole Tale	DataOne	not possible	DOI to DataOne

Discussion

Several projects develop applications for publishing computational research. One might think the applications, since they all strive for the same overall goal, resemble each other. However, the overview in this paper (see Tables 1-4) shows that the applications address different issues and needs. This increases the chances for stakeholders to find a suitable application for their individual requirements.

Needs of stakeholders

Publishers: A critical decision is whether publishers want to host an infrastructure by themselves or engage a provider. Applications exist for both approaches though the majority of them allow self-hosting. Accordingly, all self-hosting solutions have an open license enabling operators to create customized versions of the platforms and the peer review process. A further advantage is the mitigation of risks regarding vendor lock-in or grant-based projects which expire at some point.

Nevertheless, it remains unclear which costs publishers have to expect when hosting an infrastructure. The final costs strongly depend on the number of views and execution attempts, workflow sizes, and manipulation options. These parameters differ between use cases and could be the basis for future research, e.g., on stress tests. Therefore, the metrics of existing publications might provide first insights to calculate the required resources. The Binder instance MyBinder.org provides an initial estimate regarding costs (https://mybinder.org/v2/gh/jupyterhub/binder-billing/master?urlpath=lab/tree/analyze_data.ipynb). However, further data on infrastructure costs from the other services would help to calculate costs in a more accurate way, albeit this transparency is only realistic for non-profit projects.

Working with applications that are already in use and those in a beta stage can both have advantages and disadvantages. While applications already in use provide initial evidence that they work, it might take more effort to make adjustments in order to fit the publisher’s infrastructure. In contrast,

applications in a beta stage can adjust their feature set and consider new contributions without worrying about running instances and backwards compatibility, but the deployment of the applications might reveal new issues.

Editors and authors: The research area does not narrow down the number of options. Although some applications come from specific domains, e.g., the life sciences, none of them is restricted to a specific field. Regarding submission formats, there is a trend towards literate programming approaches. Most applications either support Jupyter Notebook or R Markdown which both have proven to support reproducibility [57]. However, some journals and publishers have particular requirements, e.g., they rely on LaTeX. Since transformations to other document types are often cumbersome and shifting 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 shows that a scientific article combining executable code with narrative is possible today and comes with advantages with respect to communicating scientific results, e.g., studying text and analysis in parallel while also being able to manipulate the analysis.

A critical issue is that not all applications explicitly handle the copyright of the shared materials. Those who do, fortunately, either require or encourage open licenses. Licensing is important to enable reusability and thus a recommendation mentioned frequently in papers discussing reproducibility guidelines [16, 35]. Therefore, the platforms should inform users about licenses, e.g., by referring to existing advising resources (e.g., https://choosealicense.com/) and ideally require open licenses.

A further limitation of the applications is that the anonymity of the authors is not guaranteed during the review process. All applications require an account for creating reproducible results and the name of the creator is usually visible making double-blind review impossible. However, access to code and data is particularly important for reviewers, since they decide on accepting or rejecting a

submission. One solution might be to create an anonymous version of the materials, as it is possible with Open Science Framework (https://help.osf.io/hc/en-us/articles/360019930333-Create-a-View-only-Link-for-a-Project) or to adopt an open peer-review process.

Reviewers and readers: Being able to reproduce computational results in a paper is a clear benefit. However, open reproducible research comes with a number of further incentives with respect to finding and inspecting papers [15]. Most search tools provided by the applications do not take full advantage of the information contained in code and data files, e.g., spatiotemporal properties. Instead, they either 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, RStudio. In both cases, users can directly access, manipulate, and reuse the code. However, readers might still need to understand 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 [58] or bindings [46] 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., the input datasets. This might be due to the plethora of complex interoperability issues with respect to data formats or column names in tabular data. Only ReproZip [49] and o2r [52] provide basic means to substitute input datasets, yet they require users to ensure compatibility.

Librarians: 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 not given anymore. 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. A further dependence is the fundamental technology. Without Docker, a Dockerfile cannot be run anymore but it is still readable and provides important machine- and human-readable information on how to run the analysis. This is also true for source code scripts which are plain text files and thus can be opened using any editor. These examples demonstrate the importance of using open and text-based instead of proprietary and binary file formats in science. Due to these issues, researchers should consider archiving the research materials on platforms, such as Zenodo in addition to an executable version (which should be the same version) using one of the applications.

Limitations

This work is subject to a number of limitations. The scope of the paper is narrow and does not cover all applications that support the publication of computational research (e.g., workflow system, such as Taverna). In addition, the information in this review paper might become outdated quickly but having a structured overview can still be helpful for the involved stakeholders to decide on a technology. Finally, the properties we investigated in this survey are certainly not complete. Still, stakeholders requiring more information can use the overview as a starting point for further research.

Conclusions

Open science is on the rise and obtains more attention and expressions of support from all stakeholders including publishers, editors, reviewers, authors, readers, and institutions responsible for archiving, e.g., libraries. Despite these developments, publishing computational results reproducibly is still a challenge for all parties involved in scholarly communication. Fortunately, several projects aim at tackling these issues by designing applications to support the publication of executable research results. The key contribution of this paper is an overview of these applications, their commonalities, and differences. This overview can be used as a decision support for publishers

facing the question of whether to host an application by themselves, for editors who want to ensure that the application is conform with the author and reviewer guidelines, for authors who want to create reproducible analyses efficiently, and for reviewers who want to verify the results and to suggest potential solutions during the review process. Moreover, the overview considers the needs of readers who want to better understand and reuse research materials, such as code scripts and data. Finally, the survey contains aspects relevant for archiving. The applications all provide a rich set of functionalities and address many reproducibility issues. However, issues related to ethics, privacy, big data, and long computation times are not fully solved, yet. Beyond these challenges, considering executable submissions already during the review process comes with a number of novel research questions: How many reviewers try to reproduce submissions using one of the applications? How does a reproducibility attempt (whether it failed or succeeded) affect a reviewer's decision? Is the effort required for reviewing reproducible papers the same as for a traditional article (i.e., only reading the text), or will reviewers refuse reviews since they fear additional work? And finally, how much time does it take to review a paper supplemented by reproducible documents and how much additional understanding do reviewers obtain? These quantitative (time, user interactions) and qualitative (interviews, questionnaires) measures can help to improve the applications and eventually foster the success of open reproducible research.

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 and code used to create the tables are openly available on Zenodo: https://doi.org/10.5281/zenodo.3562270. The table allows substituting the input data by an updated record. The repository includes a list of all projects we looked at and the reason for why we excluded some of them.

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.

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. Morin, A., Urban, J., Adams, P. D., Foster, I., Sali, A., Baker, D., & Sliz, P. (2012). Shining Light into Black Boxes. *Science*, 336(6078), 159–160. doi:10.1126/science.1218263
7. 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

8. National Academies of Sciences, Engineering, Medicine & others (2019). Reproducibility and Replicability in Science. National Academies Press.
9. 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
10. Powers, S. M. & Hampton, S. E. (2018). Open science, reproducibility, and transparency in ecology. *Ecological Applications*, 29(1). doi:10.1002/eap.1822
11. Piwowar, H. (2007). Sharing Detailed Research Data Is Associated with Increased Citation Rate. *Nature Precedings*. doi:10.1038/npre.2007.361.1
12. 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
13. Hrynaszkiewicz, I. (2019). Publishers' Responsibilities in Promoting Data Quality and Reproducibility. *Handbook of Experimental Pharmacology*. Doi:10.1007/164_2019_290
14. Stark, P. B. (2018). Before reproducibility must come preproducibility. *Nature*, 557(7707), 613–613. doi:10.1038/d41586-018-05256-0
15. 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
16. 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.
17. 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
18. 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
19. Rule, A., Birmingham, A., Zuniga, C., Altintas, I., Huang, S.-C., Knight, R., … Rose, P. W. (2019). Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLOS Computational Biology*, 15(7), e1007007. doi:10.1371/journal.pcbi.1007007

20. Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10), e1003285. doi:10.1371/journal.pcbi.1003285
21. Greenbaum, D., Rozowsky, J., Stodden, V., & Gerstein, M. (2017). Structuring supplemental materials in support of reproducibility. *Genome Biology*, 18(1). doi:10.1186/s13059-017-1205-3
22. Markowetz, F. (2015). Five selfish reasons to work reproducibly. *Genome Biology*, 16(1). doi:10.1186/s13059-015-0850-7
23. Donoho, D. L. (2010). An invitation to reproducible computational research. *Biostatistics*, 11(3), 385–388. doi:10.1093/biostatistics/kxq028
24. Barba, L. A. (2018). Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311*
25. 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
26. 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
27. Perkel, J. M. (2019). Make code accessible with these cloud services. *Nature*, 575(7781), 247–248. doi:10.1038/d41586-019-03366-x
28. Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1), 71–79. doi:10.1145/2723872.2723882
29. Howe, B. (2012). Virtual Appliances, Cloud Computing, and Reproducible Research. *Computing in Science & Engineering*, 14(4), 36–41. doi:10.1109/mcse.2012.62
30. Nüst, D. & Hinz, M. (2019). containerit: Generating Dockerfiles for reproducible research with R. *Journal of Open Source Software*, 4(40), 1603. doi:10.21105/joss.01603
31. Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, 27(2), 97–111. doi:10.1093/comjnl/27.2.97
32. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, F., Kelley, K. et al. (2016). Jupyter Notebooks-a Publishing Format for Reproducible Computational Workflows. *Proceedings of the 20th International Conference on Electronic Publishing*, 87–90.

33. Baumer, B., Cetinkaya-Rundel, M., Bray, A., Loi, L., & Horton, N. J. (2014). R Markdown: Integrating a reproducible analysis tool into introductory statistics. *arXiv* preprint arXiv:1402.1894
34. Pebesma, E. (2013). Earth and Planetary Innovation Challenge. Earth and Planetary Innovation Challenge. *Elsevier*. http://pebesma.staff.ifgi.de/epic.pdf
35. 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
36. Fenner, M., Crosas, M., Grethe, J. S., Kennedy, D., Hermjakob, H., Rocca-Serra, P., … Clark, T. (2019). A data citation roadmap for scholarly data repositories. *Scientific Data*, 6(1). doi:10.1038/s41597-019-0031-8
37. Park, H. & Wolfram, D. (2019). Research software citation in the Data Citation Index: Current practices and implications for research software sharing and reuse. *Journal of Informetrics*, 13(2), 574–582. doi:10.1016/j.joi.2019.03.005
38. Pérignon, C., Gadouche, K., Hurlin, C., Silberman, R., & Debonnel, E. (2019). Certify Reproducibility with Confidential Data. *Science* 365 (6449). doi:10.1126/science.aaw2825
39. 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
40. Ahn, D. H., Lee, G. L., Gopalakrishnan, G., Rakamarić, Z., Schulz, M., & Laguna, I. (2013). Overcoming extreme-scale reproducibility challenges through a unified, targeted, and multilevel toolset. *Proceedings of the 1st International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering*. Doi:10.1145/2532352.2532357
41. 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
42. 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
43. 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
44. 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

45. Nüst, D. (2018). Reproducibility Service for Executable Research Compendia: Technical Specifications and Reference Implementation (version 1.0.0). *Zenodo*. doi:10.5281/zenodo.2203844
46. 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
47. 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
48. Steeves, V., Rampin, R., & Chirigati, F. (2017). Using ReproZip for Reproducibility and Library Services. *IASSIST Quarterly*, 42(1), 14. doi:10.29173/iq18
49. 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
50. Rampin, R., Chirigati, F., Steeves, V., & Freire, J. (2018). ReproServer: Making Reproducibility Easier and Less Intensive. arXiv Preprint arXiv:1808.01406.
51. 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
52. 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
53. 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
54. 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
55. 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

56. Prelipcean, D. (2019). Physics Examples for Reproducible Analysis. *CERN*. https://cds.cern.ch/record/2690231
57. 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
58. 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