Challenges of Curating for Reproducible and FAIR Research Output

DOI: 10.15497/RDA00063

Authors: Limor Peer, Florio Arguillas, Tom Honeyman, Nadica Miljković, Karsten Peters-von Gehlen and CURE-FAIR subgroup 3

Published: 1 July 2021

Version: 2.1

Abstract: The goal of the CURE-FAIR WG is to establish standards-based guidelines for curating for reproducible and FAIR data and code. The final deliverable of the Working Group (WG) is a document outlining CURE-FAIR standards-based guidelines for best practices in publishing and archiving computationally reproducible studies. To support the deliverable, the WG created four subgroups, each tasked with studying and summarizing a particular aspect of this work: CURE-FAIR Definitions, Practices, Challenges, and Alliances.

This document reports on the work of subgroup 3, on CURE-FAIR Challenges. The aim of Subgroup 3 is to describe the challenges of preparing and reusing materials required for computational reproducibility. The group has reviewed the literature and collected use cases, stories, and interviews with various stakeholders (researchers, publishers, funders, data professionals, IT, repositories) who are trying to reproduce computation-based scientific results and processes. This report represents the WG's first milestone toward the final deliverable: CURE-FAIR survey of current state.

Language: English

RDA webpage: https://www.rd-alliance.org/group/cure-fair-wg/outcomes/challenges-curating-reproducible-and-fair-research-output

Citation and Download: Peer, L., Arguillas, F., Honeyman, T., Miljković, N., Peters-von-Gehlen, K., & CURE-FAIR WG Subgroup 3. (2021). Challenges of Curating for Reproducible and FAIR Research Output. Research Data Alliance. DOI: 10.15497/RDA00063

Challenges of Curating for Reproducible and FAIR Research Output

A Report by the RDA CURE-FAIR Working Group, Subgroup 3 on CURE-FAIR Challenges

Limor Peer, Florio Arguillas, Tom Honeyman, Nadica Miljković, Karsten Peters-von Gehlen and CURE-FAIR subgroup 3

July 1, 2021

V2.1
V2.0 [posted April 12, 2021 for community review]
V1.1 [posted March 17, 2021 for WG review]
V1.0 [posted February 15, 2021 for preliminary WG review]

Table of Contents

Executive Summary

The goal of the CURE-FAIR WG is to establish standards-based guidelines for curating for reproducible and FAIR data and code. The final deliverable of the Working Group (WG) is a document outlining CURE-FAIR standards-based guidelines for best practices in publishing and archiving computationally reproducible studies. To support the deliverable, the WG created four subgroups, each tasked with studying and summarizing a particular aspect of this work: CURE-FAIR Definitions, Practices, Challenges, and Alliances.

This document reports on the work of subgroup 3, on CURE-FAIR Challenges.[1] **The aim of Subgroup 3 is to describe the challenges of preparing and reusing materials required for computational reproducibility**. The group has reviewed the literature and collected use cases, stories, and interviews with various stakeholders (researchers, publishers, funders, data professionals, IT, repositories) who are trying to reproduce computation-based scientific results and processes. This report represents the WG's first milestone toward the final deliverable: CURE-FAIR survey of current state.

The main challenges experienced by those who are preparing or attempting computational reproducibility are:

1. **Lack of clarity on standards for computational reproducibility**. We find variation in the meaning and the practice of computational reproducibility. There is a need to define criteria for what should be reproduced, what should be archived (and for how long), and how to determine whether computational reproducibility is possible or practical.

2. **Sociocultural factors.** Several challenges have to do with "human factors" at the micro level (e.g., insufficient training and skills) and macro level (e.g., lack of awareness in some domains, professional incentive structures). These factors in combination contribute to insufficient progress on creating conditions for facile and ubiquitous computational reproducibility.

3. **The R in FAIR (Reusable).** The ability to reuse digital materials when preparing or attempting computational reproducibility is the most common challenge reported. In the context of computational reproducibility, the "digital object" is a *process* that comprises various *components* (i.e., code, data, workflow, and execution or computational environment).[2] The components and the process as a whole need to be FAIR.

4. **Machine readable is not enough.** FAIR must be human-readable. Computational reproducibility by humans requires that they understand what scripts are intended to do and why. Moreover, since in many cases not all steps in a workflow are script-based, the

[1] The final WG deliverable will include the work of Subgroup 1 on CURE-FAIR definitions and Subgroup 2 on CURE-FAIR practices, and be informed by the work of Subgroup 4 on CURE-FAIR and RDA.
[2] See definition of "computational environment," The Turing Way guide.

next best thing is proper documentation and metadata so that humans can interpret the scripts and understand their function in order to attempt computational reproducibility.

5. **Solutions tend to be ad-hoc and not FAIR**. Some practical difficulties are met with improvised solutions by those attempting computational reproducibility. These solutions often do not advance transparency and reproducibility on the whole.

6. **Differences across stakeholders and domains**. Special attention must be given to the requirements and practices of different scholarly domains, and indeed to the raw materials that comprise an academic field. In addition, different stakeholders emphasize different computational reproducibility challenges. Solutions addressing the feasibility of computational reproducibility need to consider these contexts.

1. Background

The CURE-FAIR WG is focused on the curation practices that support computational reproducibility and FAIR principles. As stated in the CURE-FAIR WG charter, the ultimate objective of the Working Group (WG) is to improve FAIR-ness and long-term usability of "reproducible file bundles" across domains.

Following the National Academies of Sciences, Engineering, and Medicine **we define computational reproducibility as the ability to repeat the analysis and arrive at the same result** [1]. In this framework, the object of the curation is a "reproducible file bundle," and curation refers to the activities designed for "maintaining, preserving and adding value to digital research data throughout its lifecycle" (Digital Curation Center).

Among the reasons to engage in computational reproducibility in the context of scholarly research are the preservation of a complete scientific record, verification of scientific claims, doing science and building upon the findings, and teaching.

A reproducible file bundle includes the digital research objects used to arrive at a scientific claim. At the core of this bundle are data (raw, analysis, input, output) and code (cleaning, analysis). Additional objects are often included in the bundle: ReadMe files, codebooks, data dictionaries, workflows, pipelines, repositories, virtual environments and containers, and other digital artifacts used in the course of the research, as well as various metadata and documentation. This is also referred to as a "research compendium" [6].

The CURE-FAIR WG final deliverable is a document outlining CURE-FAIR standards-based guidelines for best practices in publishing and archiving computationally reproducible studies. To inform these guidelines, Subgroup 3 has been charged with describing the challenges of preparing and reusing materials required for computational reproducibility (that is, research outputs such as data, software, codebooks, etc.). To that end, the group has been working to collect use cases, stories, and interviews with various stakeholders (researchers, publishers, funders, data professionals, IT, repositories) who are trying to reproduce computation-based scientific results, workflows, and processes to learn about any pain points, especially across domains.

From September 2020 through January 2021, the CURE-FAIR WG Subgroup 3 held a series of meetings. The meetings were used to plan for ways to collect information about challenges, analyze that information and discuss a framework for the challenges, and begin drafting a report based on our findings. We describe below the information we gathered and the insights we gained. We also highlight the highly relevant work of adjacent RDA working groups, specifically, the FAIR4RS WG [2]. This report represents the WG's first milestone: CURE-FAIR survey of current state.

We note that the information presented here reflects our gleaning from the sources we employed here: the literature, crowdsourced use cases, stories, and interviews with community members. We present the information as is and summarize it without much editorializing. To the extent that there are gaps in our telling -- for example, solutions that are not mentioned, information that is outdated or inaccurate, recommendations that are not made -- it is a reflection of the current state of knowledge, practice, and thought about this topic.

2. Challenges

We collected information about the challenges of preparing and reusing materials required for computational reproducibility through various methods: Literature review, crowd-sourced use cases, in-depth interviews, and RDA VP16 session participants. We identified the challenges and mapped them to the FAIR principles, categorizing each challenge as falling under Findable, Accessible, Interoperable, or Reusable. As intended by the original articulation of the FAIR Principles, we apply the principles to "all scholarly digital research objects—from data to analytical pipelines" [3]. We also added an "Other" category for challenges that we considered beyond the FAIR principles (see [4] on implementation considerations).

We summarize the challenges by the method we sourced them. The Insight section of this report provides a synthesis of the CURE-FAIR challenges, and the Appendix provides more details.

2a. Challenges reported in the literature

Survey results presented in [5] revealed challenges researchers are facing when sharing their code and identified the following categories that we mapped onto challenges that are FAIR and beyond-FAIR:

- Software dependencies and execution environment (technical challenge);
- Documentation (FAIR challenge);
- Accessibility and licensing (FAIR challenge);

- Time and skill (institutional challenge);
- Quality control (institutional challenge);
- Software sustainability and management plan (technical challenge);
- Metadata, i.e., lack of metadata (FAIR challenge).

We identified challenges in computational reproducibility from several perspectives, including, Editors, Publishers, Policy Makers, Reviewers, Researchers, Developers, Software Curators, Data Stewards, and Funders (see References), and split them into two categories based on similar challenges and concerns.

The first category is concerned with reproducibility in the context of scholarly communication (including Editors, Publishers, Policy Makers, and Reviewers). For this group, the main challenges we identified in the literature are related to institutional and technical challenges (that is, beyond-FAIR):

- Complexity of reviewing and publishing research compendium (text, data, and software code) [6] (technical challenge);
- Ways to define a "minimum standard" and procedures for publishing and citing reproducible research (e.g., capturing execution commands, creation of Docker files, platform integration such as for code versioning and for data archiving, cloud-based computational environment, application of dynamic documents such as Jupyter Notebook) [7-13] (institutional challenge);

To overcome these challenges, the following recommendations were proposed:

- Strong incentives should be provided for sharing data and code in order to motivate authors to publish their research and not turn away from journals imposing new requirements for data and code sharing [7-9, 11-14];
- In order to ensure computational reproducibility, openness (e.g., in relation to open data, open code, open formats) is crucial if there are no legal obstacles, and it should be applied even for subscription journals [7-8, 10-12, 14-15].

The second category reflects the perspectives of those conducting or practicing computational reproducibility, including Researchers, Developers, Curators, Data Stewards, and Funders. We identified the following challenges, a majority of which are also beyond FAIR:

- It takes more time to prepare and read research compendium consisting of text, data, and software code [6] (institutional challenge);
- We need more established relations between FAIR policies and free and open source software for reproducible research as openness is crucial for computational reproducibility [5, 7, 10-11, 14] (FAIR (re)definition for software challenge);
- There is no definition of good practices in computational reproducibility and data science in general (e.g., incorporate computational reproducibility early in the

data management plan, DMP, provide instructions for software citations) [10-11, 16] (institutional challenge);

- Ways to share proprietary research software should be established so as to not infringe upon licenses or at least metadata should be shared (for example, via software registries) [7, 10, 14-15] (technical challenge);
- How to define important principles of computational reproducibility in research (versioning, documenting and sharing code, and computational environment) [11] (institutional challenge).

The following recommendations were proposed:

- Computational reproducibility might be introduced as a subject taught at Universities in order to ensure adequate knowledge and skills [16];
- Availability and sharing of data, software, and other artifacts are crucial for data science and other disciplines, and a discipline-oriented approach might be needed [8, 16];

More generally in the literature, we found a discussion of key issues relating to challenges of conducting, or preparing for, computational reproducibility:

<u>Open science</u>: It has been stated that the journal publication process is the main place to apply practices of computational reproducibility and journal policy is a vehicle for promoting reproducible computational research [7, 12, 17]. In this context, computational reproducibility is linked to open science practices. A key consideration is that these policies are domain-specific with a coherent scope [16]. For example, editorial perspective on open science practices and computational reproducibility in the area of MRI (Magnetic Resonance Imaging) revealed that special precautions should be taken as MRI is driven by copyright law and patents [8]. Open science practices should also include establishment of a development community (e.g., organizing Hackathons) and using notebooks for code [8]. However, sharing software code and data for post-publication can be a double-edged sword as the complexity can be increased for journals and researchers, and also for the reader as she/he might not be able to intuitively get the access to a published research compendium [6].

<u>Research software</u>: There is current interest[3] in developing a set of FAIR principles for research software. The lack of guidance or principles for a FAIR-compatible approach to the code or software components of a reproducible bundle is in part dependent on this activity being completed. It has been suggested that, "software developed to be used by others ... can be expected to reach a higher degree of FAIRness than software that has not been implemented with reuse as a primary goal" [10]. In producing guidelines for constructing a "reproducible bundle" this working group is suggesting that even software implemented originally without reuse as a primary goal will still need to be sufficiently FAIR so as to enable reuse because it is part of a reproducible bundle.

[3] Definitions of Research Software and FAIR Principles for Research Software are being developed by the RDA FAIR4RS WG [2].

Related but apart from this is the need to bridge research and software engineering practices (e.g., version control, testing, maintenance, etc) [5, 10, 18-19]. Current practices for computational research are related to the extensive use of development platforms such as GitHub and Gitlab, but these are not traditional archival systems that provide rich metadata or persistent identifiers, although they are popular solutions for software specialists [9-11]. Recent advances address some of these issues (see the GitHub Archive Program; Automatic deposit from GitHub to Zenodo to obtain a DOI).

A general awareness of issues in computational reproducibility has given rise to a plethora of applications and solutions that address some of the challenges discussed in this report. There are several solutions that utilize containers or virtual machines to include all components in a readily available form (e.g., ReproZip, Code Ocean, Whole Tale, Binder, Renku with Dataverse Repositories). Some of these are cloud based, to make them more readily available for use. Some solutions are more oriented to shift in practice, for instance using computational notebooks to incorporate more description and discussion into the code. A relatively recent innovation is when the computational reproduction is integrated into a publication or an archive [9] (which builds on the tools and techniques of the other solutions).

Common problems for all these solutions are that infrastructure development (data and software repositories, human infrastructure) is at different levels across the various inputs to a reproducible bundle. Consequently, bundling them together into a monolithic bundle may solve infrastructure gaps but create new complexities: It can lead to complex licensing requirements, increased preservation needs and complexity, and the solutions may mask granular identification and hinder reuse or the ability to interrogate the overall bundle.

Software Heritage is a promising platform for archiving and identifying free and open source software [5, 10]. Software registries are a common mechanism for discovery in some domains or disciplines, e.g., swMath, SciCrunch, ASCL. This approach fulfills a software identification need in the absence of support for identifiers and quality metadata in the software repository space (where development platforms are used as repositories without the same level of support for metadata and preservation as is seen for research data). But it can be problematic because many software registries allow public submissions, leading to challenges of metadata governance, curation, and integrity. Another approach is emulation-as-a-service; for example, EaaSI is a resource that preserves and provides access to the original computational environment, thus enabling reproducibility.

Open data and open software: Although practices from open data are usually translated to open software [7], the main difference between data and software is that software is executable, can and does change, and of composite nature (usually with composite dependencies). Further, unstable foundations can lead to a so-called software collapse -- where software stops working if not actively maintained [2, 10, 20-21]. Also, software licencing and conditions for reuse can be complicated, especially in cases when the final software package incorporates dependencies between different software licenses passed on from earlier versions. Although the software licences are beyond the control of the software authors, authors can often choose which dependencies to include. Research software is usually defined as software used to generate, process, and analyse results presented in a publication [10, 22], but the proposed definition of research software might be more aligned with the core open science and reproducibility principles by including four essential freedoms in free software (e.g, freedom to change/improve the program) [15]. Research software should be considered in all its aspects: as a tool, as a research outcome and as the object of research (c.f. [2, 5] for more information).

<u>FAIR software and reproducible software</u>: FAIR software contributes to, but does not guarantee, computational reproducibility. For reproducible research, a common refrain in the literature is to use free and open source software as it provides permissive licences and therefore promotes software reusability and accessibility [10-11, 15]. An identified issue in designing reproducible software in relation to FAIR principles is in the application of “mouse-operated point-and-click interface" especially when performed with commercial software as it is inherently difficult to record and replay the analysis procedure accurately. Additionally, it prevents research workflows from being tracked and reused efficiently [11, 23]. Therefore, a Data Life Cycle approach with thematic abstraction applied to software as well as to data [16] might be a solution for defining computational reproducibility rather than just focusing on FAIR. The terms partial reproducibility and full reproducibility have been proposed in [14] and defined by the following three aspects that are not considered by the FAIR principles:

- Depth of sharing (e.g., sharing just a code or software system as white or black box), also addressed in [24];
- Code portability (the same and/or similar Operating System);
- Coverage (how much of the published results can be reproduced).

Having all that in mind, the application of the FAIR principles should be done carefully [10].[4]

<u>FAIR Workflows</u>: When we talk about computational reproducibility, the implementation (or planning thereof) of workflows using specialized applications is rapidly emerging. The workflow is considered one of the artifacts that enables reproducibility, alongside code, data, and computational environment information [16]. This introduces a set of challenges, see e.g., Goble et al. [25]. This also relates closely to the difficulties of applying the FAIR principles to software alluded to above, i.e., the FAIR principles are currently not applicable to workflows and would need a redefinition or update. This is further complicated since workflows are not always script-based but rather a series of manual tasks. Whereas scripted workflows, which “explicitly and unambiguously carry out instructions [embody] the principles of reproducibility and transparency” [11] manual tasks rely heavily on documentation for reproducibility. Portable workflows have been recently proposed since they can provide, to some extent, automatic adaptation for code execution on different platforms and environments [26].

2b. Challenges reported to the CURE-FAIR WG

Between September 2020 and January 2021, subgroup 3 gathered information about CURE-FAIR challenges using a variety of methods. The goal is to have a holistic view of the issue from multiple stakeholder points of view, including researchers, data professionals, and technical infrastructure experts across research domains and geographical boundaries.

[4] The RDA FAIR4RS WG [2] distinction between “software in research” and “Research Software” is relevant here.

In this section we present a summary of our findings. Please see the Appendix for more details.

Crowd-sourced use cases

We created a google form to gather information about the challenges faced by various stakeholders when making computational reproducibility possible or attempting computational reproducibility. Each entry on the form represents a "user story" and respondents can enter multiple stories. The form structures each story as follows: AS A <role> I WANT TO <activity/goal> SO THAT <benefit> BUT <problem>. For example, "AS A researcher I WANT TO reuse analysis code SO THAT I can test consistency across computing environments BUT a dependency is no longer available." The form also allowed respondents to add any further comments.

Among challenges classified as FAIR, **Reusability** is the most dominant challenge to computational reproducibility. Perhaps this is not surprising since reproducibility is a particular type of reuse and since it is predicated on Reusability not only of each element, but also the interaction between them (i.e., data, software, workflow, or other digital objects). The main challenges that emerge from the crowd-sourced use cases have to do with changes to software over time and with inadequate documentation of software, dependencies, and the computation environment (both the original environment used to compute and the requirements for reproducing the computation).

A frequently reported challenge to reproducibility has to do with the **dynamic nature of software**. Problems encountered due to different versions of software and code, or due to having to run software on systems different from the one on which it was developed, can prevent users from using the materials at all. This includes updated software that does not support older versions or when the code itself does not use FAIR vocabularies or programming language. Respondents report problems with, for example, an R package that is no longer available (ID15, Research Assistant) or that has been phased out (decommissioned) (ID6, Researcher), or issues with "software updates and compatibility issues [which] often mean that I have to go back and fix bugs months or even years later" (ID11, Data Scientist).

The challenge of using software in computational reproducibility is closely linked to the challenge of **inadequate documentation**. Lack of specificity about software version means that a future end user attempting computational reproducibility may Find and Access software, but it is outdated and therefore not "optimized for reuse."[5] Respondents report that data, code, software and other materials (e.g., dependencies, operating systems) are insufficiently documented which prevents from using the materials in a meaningful way. This theme is expressed by various stakeholders: "The dependencies are not documented, or the documentation is too difficult to find" (ID46, Researcher); "The submission has insufficient documentation for a third party to execute the whole workflow, even though some parts of

[5] "The ultimate goal of FAIR is to optimise the reuse of data," FAIR Principles

data and code are published” (ID35, Reproducibility Reviewer); “There is not enough documentation for entering required metadata when publishing or archiving the dataset; there is no data management plan that can be used to guide the work” (ID53, Data Steward); “When we hear about computational reproducibility the focus is often on the compute environment itself, but the documentation is just as important and offers critical information for ensuring the submission is reproducible and understandable” (ID23, Data Curator).

Accessibility challenges to computational reproducibility were also reported by respondents. Lack of access to input data may be due to the fact that they are not shared by the author, as noted by one respondent: “Datasets are often not publicly available; similarly, access to datasets can often depend on the goodwill of their creators, even if they are publicly funded” (ID19, Researcher). In some disciplines, inability to access data has to do with the high cost of archiving big data or the fact that “the resources required are too large to ensure reproducibility (at least within a reasonable time frame)” (ID56, service provider). Along these lines, lack of access to a specialized computing environment used in the original research can impede reproducibility. Respondents report lack of access to licensed products or a “compute environment [that] is no longer available” (ID57, service provider) as challenges to reproducibility.

Beyond-FAIR challenges are most revealing about the difficulties for practicing reproducibility and cultivating a reproducibility culture. Prominent among these are challenges relating to the **scholarly publication system**, misaligned **incentive structure**, and **skills and knowledge** of researchers. These impediments to reproducibility are separate from the attributes of the digital objects themselves and can have a profound impact on reproducibility work. These challenges tell the story of the cultural and institutional factors that often impede reproducibility. Moreover, in some cases, these challenges are overcome with ad-hoc solutions that themselves may not be FAIR. For example, when a journal refused to publish a correction as a result of a post-publication reproducibility attempt that failed, the original authors, as a workaround, “published a correction on their lab web page, but this is obviously much less Findable and might be removed if the PI leaves the university” (ID9, Researcher).

The community also offered recommendations and solutions to address some of the challenges via the google form. We report these here.

1. **More training**;
    - “Further training or specialised assistance through designated University staff would be desirable for this to become common practice and benefit the field” (ID27, Researcher and Data Steward);
    - “We don't just have to teach reproducibility basics but also some basics of how PCs work” (ID16, Teacher);
    - “RSEs are not always aware of the verification landscape” (ID31, Research Software Engineer);

2. **More carrots / incentives to encourage reproducibility**;
    - "We need much bigger carrots to avoid using any sticks at all" (ID34, Reproducibility Chair);
    - "Reproducibility is hard! It can rarely be achieved by one person alone but requires independent attempts and feedback, so that habits are changed" (ID35, Reproducibility Reviewer);

3. **Establish journal standards for reporting and documentation**;
    - "We required the author to specify in their README that secondary users wishing to verify the results themselves had to use the specific version of that package, otherwise the code would fail" (ID25, Data Curator);
    - "We usually resolve this issue by running on the same operating system and asking the authors to provide text in the README highlighting this in case secondary users attempt to verify on the wrong OS" (ID26, Data Curator);
    - "Currently at Odum we are building guidance for authors submitting to journals with data verification policies to help address the gaps we frequently experience in their verification submission packages. The main areas we are focusing on right now are with the documentation -- the README and Codebook -- which most authors seem to have issues completing properly. The README provides context for the submission package such as the compute environment requirements, the files and their functions, as well as instructions for obtaining original source data (if appropriate). The Codebook provides context for the analysis data and must include all variables, variable definitions, values and value labels. We also require full data citations for any original source data used in the analysis data. Hopefully our guides will help to reduce the number of re-submissions due to issues in the documentation, but training and/or more resources might help. When we hear about computational reproducibility the focus is often on the compute environment itself, but the documentation is just as important and offers critical information for ensuring the submission is reproducible and understandable" (ID23, data curator);
    - "While I am able to upload source code and annotated scripts to platforms like the Open Science Framework, there still seems to be a major hurdle when it comes to reproducibility by others. Most recently, a reviewer failed to look through the annotated RNotebooks and wrongly assumed that I had not provided sufficient information / output tables. It would be great if the community could settle on standardised templates for transparent reporting, so researchers can provide source data and code in a structured format which the reviewers are familiar with. Also, as someone who is not computationally trained originally, I find it hard to create self-contained 'packages' that make it even easier for others to reproduce my work (e.g., using Docker). Further training or specialised assistance through designated University staff would be desirable for this to become common practice and benefit the field" (ID27, Researcher and Data Steward).

4. **Inclusion of libraries and dependencies in the reproducibility package**;

- “Had to figure out how to modify the code; ideally, the libraries and dependencies used in the original analysis are saved with the data and the code” (ID6, Researcher);
- “Use tools, for example, the Guix-Past channel (ID18, Research \Engineer).

5. **Use of open software or making source codes freely available**;
   - “I believe that source code or at least detailed functionality of signal processing methods should be freely available for scientists to be able to improve it and to understand its effect and to test it across platforms. It harms science when functionality is not the same across programming languages” (ID12, Researcher).
   - “No code was published, so I create my own implementation based on the Methods description in the paper, but this gives different results from those in the paper, and the authors do not reply to my emails pointing this out” (ID8, Researcher);
   - “Rewrite old solutions based on published paper, compare rewritten solution with results of executable and continue” (ID42, Researcher).

6. **Scripted process / Interactive research article**;
   - “My dream is to have a simple user-friendly environment where I can read an interactive research article. Data and methodology are available, so that if I want to change data set or modify the code I can simply edit the changes and run the code to see the result” (ID44, Journal Editor).
   - “I have to introduce manual corrections in order to ensure that only the data with the good quality i.e., satisfactory signal to noise ratio is preserved and this is not possible with purely automatic procedure… Currently, the only way to solve this is to introduce an exhaustive description of manual corrections. I would like to see other ways how to overcome this in the future” (ID14, Researcher);
   - “Rewriting old solutions using the same programming language, and unifying trace file format” (ID41, Researcher)

7. **Use of a third-party that verifies reproducibility of the study / Institutional support for curating for reproducibility**;
   - “I've experienced this even with published collections of code and data. Unit tests, code reviews, even just having someone try it out other than yourself, all these things can help” (ID29, Research Software Engineer);
   - “We've come up with a minimal approach to reproducibility: one person executed the workflow once and writes a little report - that's it. Finding people who would be excited to execute workflows seems possible. Having an extra role of "codechecker" in peer review has great potential to involve RSEs and ECRs in scholarly communication” (ID36, Reproducibility Advocate).
     - “Reproducibility is hard! It can rarely be achieved by one person alone but requires independent attempts and feedback, so that habits are changed” (ID35, Reproducibility Reviewer);

Please see the Appendix for more details.

In-depth interviews

Subgroup 3 interviewed several community members about the challenges they face relating to computational reproducibility.

The themes in these interviews generally echo those we identify in the use cases described in the google form: Researchers report FAIR challenges having to do primarily with **Reusability** (e.g., differences in computational environments, software versions, lack of documentation) and also **Interoperability** (e.g., lack of integration of curation systems or code execution platforms and data reportisoties). Beyond-FAIR challenges discussed related to the additional **burden** on researchers who are either preparing or reviewing reproducible materials, the lack of **standards** and a clear **incentive** structure to reward reproducible practices, and the general lack of **awareness** of reproducible practices. Another challenge noted here is that the criteria of machine readability, while necessary, is insufficient for humans attempting to **interpret**, understand, and make sense of the materials.

Please see the Appendix for more details.

RDA Plenary participants

During RDA VP16 in November 2020, the CURE-FAIR WG held a session titled, Current state of curation practices that support computational reproducibility. Among the topics discussed at the session were, defining computational reproducibility, curation practices supporting computational reproducibility, effective curation workflows, FAIR and computational reproducibility, and code vs software.

From the perspective of researchers, challenges to reproducibility include **documentation**: Domain scientists make decisions and conduct methodologies outside of the computational workflow that are rarely documented, shared, and reproducible and are generally resistant to using prescribed methods or even using consistent standards for organizing files. This is closely linked to the challenge of process or **workflow standardization**: The diversity in workflows, procedures, and tools makes it difficult to interpret and to reproduce researchers' work.

Repository professionals point out that challenges may be technology- or human-based. **Reusability** is a main concern as these professionals attempt to use the materials that they have been handed. Problems include code that does not run because of differences in computational environments, software versions, or lack of documentation. Poor code quality (including poor structure, poor formatting, or poor spelling, see [27] may affect machines (and therefore FAIR Reusability) but it also affects interpretability of code by humans. Repository professionals, potentially the first reusers of reproducible materials [28], often find that code does not execute. Repository professionals point to their own lack of **skills** and **institutional capacity** as culprits, a point that was reiterated by data professionals. At the same time, researchers often deposit materials that are **poorly documented** and may not be agreeable to comments and feedback from repository staff or to skilling up. A challenge is that **standards** and **guidance** about how to prepare materials for reproducibility are often nonexistent or poorly clearly communicated to researchers. Data professionals attempting to verify computational reproducibility reiterate many of these challenges and add specific challenges related to **Accessibility**, including lack of access to sensitive data or to software dependencies (e.g., the same software version cannot be found, is too expensive, is ”impossible” to install using existing documentation).

Research software professionals see software and software environment **sustainability** (affecting FAIR Reusability) as a major challenge. The group points to the dynamic nature of software, which means that reproducibility is sensitive to software versions and environments, requiring that a specific software artifact is identified using, for example, SHWIDs. A related but distinct challenge is that software itself may not be deterministic and produce different results every time by design. Practical considerations also come into play. For example, in some cases using high performance computing (HPC) resources may make computational reproducibility impractical but not impossible, while in other cases using a real-time data stream may be practical but make reproducibility impossible. More generally, a lack of **standards** for computational reproducibility -- including the lack of clarity about how to handle the reproducibility of results that rely on services, online dependencies, or GUI-driven software and even a clear definition of software and code -- is a challenge discussed by this group.

Publishers advocating for computational reproducibility would like standardized interactivity and universal deployment capability, but this may not be practical for some computational problems, including those that depend on specific external resources, such as data sources and computational platforms.[6] Problems include dependencies that are no longer available, repositories that are shut down, file formats that change or are no longer readable, and data that are inaccessible or no longer available. These are primarily FAIR **Reusability** and **Accessibility** challenges.

Please see the Appendix for more details.

3. Insights

These are the themes that emerge from this report about the challenges of attempting computational reproducibility and curating for reproducible and FAIR research output:

1. **Lack of clarity on standards for computational reproducibility**. A larger question emerges about the goal of computational reproducibility and how it may vary across domains. There is a need to address key questions, such as, what should be reproduced

[6] See Daniel S. Katz on the difference between “reproducible in theory” (e.g., computation can be reproduced but would take a year to run on the leading HPC resource) and “in practice.”

(for example, in some cases the exact output is the object of interest while in others it is the ability to run a script with *any* input; in some cases the focus is on the reproducibility of a specific result, in others it is on the reproducibility of the computation itself)? Under what criteria should it be determined that computational reproducibility has been achieved? What is the object, the referent, of a particular execution of computational reproducibility and how should it relate to the reproducibility bundle? Under what criteria should it be determined that computational reproducibility is possible? Or practical? What materials should be archived, and for how long?

2. **Sociocultural factors.** Many challenges are not mapped onto FAIR and have to do with "human factors." These impediments to reproducibility are separate from the attributes of the digital objects themselves and can have a profound impact on reproducibility work. We categorize human factors into micro and macro challenges. At the micro level, factors such as insufficient training and skills pose challenges to those who are preparing or attempting computational reproducibility.[7] This applies across the board, as gaps in skills are identified by all stakeholders, including researchers, reviewers, curators, and publishers. At the macro level, cultural factors as well as global and regional contexts -- including lack of awareness in some domains (albeit growing in recent years), publishers whose policies and practices lag behind some of the technological affordances, and professional incentive structures that still do not privilege reproducibility work -- bear responsibility for insufficient progress on creating conditions for facile and ubiquitous computational reproducibility.

3. **The R in FAIR (Reusable).** Most challenges reported have to do with reusability. This is intuitive since computational reproducibility is a type of reuse, and "the ultimate goal of FAIR is to optimise the reuse of data" [or software, or their interaction]. As stated in the FAIR principles, the digital object and the metadata "should be well-described so that they can be replicated and/or combined in different settings." When it comes to computational reproducibility, the "digital object" is a process that comprises various components (i.e., code, data, workflow, and computational or execution environment), and the components and process as a whole need to be FAIR.

4. **Machine readable is not enough.** FAIR has to be also human-readable, especially to understand what scripts are intended to do and why. In addition, since in many cases not all steps in a workflow are script-based, the next best thing is proper documentation to allow humans to make meaning and attempt computational reproducibility. Metadata and documentation applies to all the *objects* (e.g., data, software, dependencies, the computation environment, both the original environment used to compute and the requirements for reproducing the computation) and the *processes* the objects were put through in the course of the research. Given that a lack of standardization is a common challenge, documentation has an important bridging role to play *before* standards are developed (but, of course, is still relevant afterwards).

[7] Several initiatives provide training in reproducibility, including Collaborative Replications and Education Project (CREP), Project TIER, and the Berkeley Initiative for Transparency in the Social Sciences (BITSS).

5. **Solutions tend to be ad-hoc and not FAIR**. Some practical difficulties are met with improvised solutions by those attempting computational reproducibility. For example, creating an implementation based on the Methods description in the paper when the original code is not provided. This solution may or may not reproduce the results but, more importantly, may never be linked to the original artifacts or made FAIR in its own right. In that sense, this is a solution that does not advance transparency and reproducibility on the whole, though it may provide individual benefits.

6. **Differences across stakeholders and domains**. Special attention must be given to the requirements and practices of different scholarly domains, and indeed to the raw materials that comprise an academic field. For example, reproducibility might be a challenge when access to data or software or platforms is restricted (e.g., sensitive data, commercial or otherwise restricted software, a particular platform with unusual hardware or software). Facilitating computational reproducibility in that case may require specific guidelines and shifts in current practices such as, for example, allowing authors to request data access for themselves (to perform data analysis) and for a trusted third party (to verify computational reproducibility). In other domains, challenges to reproducibility center around large data and computational requirements, i.e. workflows built around running research code on HPC systems requiring hundreds of nodes and producing petabytes' worth of data per experiment. Solutions addressing the feasibility of computational reproducibility in these situations need to be considered.

   Stakeholders largely identify a common list of computational reproducibility challenges, but we do find some differences in emphases. Researchers are most often frustrated with the lack of documentation and the diversity of workflows and research practices that require time investment just for understanding and interpreting the materials. Repository professionals and curators point to insufficient resources and need for upskilling in order to support computational reproducibility.[8] Research software professionals point to complicated issues around the sustainability of software as a major challenge. Publishers lament that the research process is not more uniformly scripted such that interactive research articles can more easily be implemented.[9]

CURE-FAIR subgroup 3 members

Florio Arguillas
Sonia Barbosa
Andrew Davison
Kenan Direk
Luiz Gadelha

[8] This includes software curation [30, 31].

[9] We note the variability among publishers in policy and practice, including the implementation of badges, which while not widely adopted has been shown to effectively promote transparency [29].

Tom Honeyman
Anthony Juehne
Katie Mika
Nadica Miljković
Wolmar Nyberg Åkerström
Karsten Peters-von Gehlen
Limor Peer
Ana Trisovic

Acknowledgements

Members of the CURE-FAIR Subgroup 3 are grateful to community members who have commented on and contributed to this document. We give special thanks to Daniel S. Katz and to Lars Vilhuber for expert input and valuable comments. We thank community members for helping us crowdsource stories about challenges with computational reproducibility, our interviewees for devoting time and engaging in open conversations with us, and participants in RDA plenary sessions for their active involvement in fruitful discussions.

References

1. National Academies of Sciences, Engineering, and Medicine. (2019). Reproducibility and Replicability in Science. Washington, DC: *The National Academies Press*. doi: 10.17226/25303.
2. Katz, D. S., Gruenpeter, M., Honeyman, T., Hwang, L., Wilkinson, M. D., Sochat, V., ... & Goble, C. (2021). A Fresh Look at FAIR for Research Software. *arXiv preprint arXiv:*2101.10883.
3. Wilkinson, M., Dumontier, M., Aalbersberg, I. et al. (2016). The FAIR Guiding Principles for Scientific Data Management and Stewardship. Sci Data 3, 160018. doi: *10.1038/sdata.2016.18*.
4. Jacobsen, A., de Miranda Azevedo, R., Juty, N., Batista, D., Coles, S., Cornet, R., ... & Schultes, E. (2020). FAIR Principles: Interpretations and Implementation Considerations, *2*:(1-2), 10-29. doi: *10.1162/dint_r_00024*.
5. Webinar, FAIR + Software: decoding the principles, November 23, 2020. Gruenpeter, Morane, Di Cosmo, Roberto, Koers, Hylke, Herterich, Patricia, Hooft, Rob, Parland-von Essen, Jessica, … Jones, Sarah. (2020). M2.15 Assessment Report on 'FAIRness of Software' (Version 1.1). Zenodo. doi: *10.5281/zenodo.4095091*.
6. Nüst, D., Boettiger, C., & Marwick, B. (2018). How to Read a Research Compendium. *arXiv preprint arXiv:*1806.09525.
7. Stodden, V., Guo, P., & Ma, Z. (2013). Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals. *PloS one*, *8*(6), e67111. doi: *10.1371/journal.pone.0067111.*
8. Stikov, N., Trzasko, J. D., & Bernstein, M. A. (2019). Reproducibility and the Future of MRI Research. *Magnetic resonance in medicine*, *82*(6), 1981-1983. doi: *10.1002/mrm.27939.*

9. Trisovic, A., Durbin, P., Schlatter, T., Durand, G., Barbosa, S., Brooke, D., & Crosas, M. (2020). Advancing Computational Reproducibility in the Dataverse Data Repository Platform. *arXiv preprint arXiv:*2005.02985.
10. Lamprecht, A. L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., ... & McQuilton, P. (2019). Towards FAIR Principles for Research Software. *Data Science*, (Preprint), 1-23. doi: *10.3233/DS-190026*.
11. Marwick, B. (2017). Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation. *Journal of Archaeological Method and Theory*, *24*(2), 424-450. doi: *10.1007/s10816-015-9272-9*.
12. Peng, R. D. (2009). Reproducible Research and Biostatistics. *Biostatistics*, *10*(3), 405-408. doi: *10.1093/biostatistics/kxp014*.
13. Katz, D. S., Bouquin, D., Hong, N. P. C., Hausman, J., Jones, C., Chivvis, D., ... & Zhang, Q. (2019). Software Citation Implementation Challenges. *arXiv preprint arXiv:*1905.08674.
14. Freire, J., Bonnet, P., & Shasha, D. (2012, May). Computational Reproducibility: State-of-the-art, Challenges, and Database Research Opportunities. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data* (pp. 593-596). doi: *10.1145/2213836.2213908*.
15. Free Software for Open Science - 36c3 (2020) held by Purine Bitter, *Free Software Foundation Europe*.
16. Stodden, V. (2020). The Data Science Life Cycle: A Disciplined Approach to Advancing Data Science as a Science. *Communications of the ACM*, *63*(7), 58-66. doi: *10.1145/3360646*.
17. Katz, D. S., Hong, N. P. C., Clark, T., Muench, A., Stall, S., Bouquin, D., ... & Yeston, J. (2020). Recognizing the Value of Software: A Software Citation Guide. *F1000Research*, *9*. doi: 10.12688/f1000research.26932.2.
18. Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good Enough Practices in Scientific Computing. *PLoS computational biology*, *13*(6), e1005510. doi: *10.1371/journal.pcbi.1005510*.
19. Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., ... & Wilson, P. (2014). Best Practices for Scientific Computing. *PLoS Biol*, *12*(1), e1001745. doi: *10.1371/journal.pbio.1001745*.
20. Katz, D. S., Niemeyer, K. E., Smith, A. M., Anderson, W. L., Boettiger, C., Hinsen, K., ... & Rios, F. (2016). Software vs. Data in the Context of Citation. *PeerJ Preprints*, *4*, e2630v1. doi: *10.7287/peerj.preprints.2630v1*.
21. Hinsen, K. (2019). Dealing with Software Collapse. *Computing in Science & Engineering*, *21*(3), 104-108. doi: *10.1109/MCSE.2019.2900945*.
22. Hettrick, S., Antonioletti, M., Carr, L., Chue Hong, N., Crouch, S., De Roure, D., ... & Sufi, S. (2014). UK Research Software Survey 2014. doi: 10.5281/zenodo.14809.
23. Barba, L.A. (2016). The Hard Road to Reproducibility. *Science* 354(6308), 142–142. doi: 10.1126/science.354.6308.142.
24. Soergel, D. (2014). Confirmation Depth as a Measure of Reproducible Scientific Research. *Post*. http://davidsoergel.com/posts/confirmation-depth-as-a-measure-of-reproducible-scientific-research
25. Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M.R., Peters, K., & Schober, D. (2020). FAIR Computational Workflows. *Data Intelligence*, *2*, 108–121. doi: *10.1162/dint_a_00033*.
26. Fursin, G. (2020). Collective Knowledge: Organizing Research Projects as a Database of Reusable Components and Portable Workflows with Common APIs. *arXiv preprint arXiv:*2011.01149v2.

27. Neville-Neil, G. V. (2010). Literate Coding. *Communications of the ACM*, *53*(12), 37-38. doi: *10.1145/1859204.1859219*.
28. Peer, L. (2013). The repository as data (re) user: Hand curating for replication.
29. Kidwell, M.C., Lazarević, L.B., Baranski, E., Hardwicke, T.E., Piechowski, S., Falkenberg, L. , … & Nosek, B.A. (2016). Badges to Acknowledge Open Practices: A Simple, Low-Cost, Effective Method for Increasing Transparency. PLoS Biol 14(5): e1002456. doi: *10.1371/journal.pbio.1002456*.
30. Chassanoff, A., AlNoamany, Y., Thornton, K., & Borghi, J. (2018). Software Curation in Research Libraries: Practice and Promise. Journal of Librarianship and Scholarly Communication, 6(General Issue), eP2239. doi: *10.7710/2162-3309.2239*.
31. Rios, F., Almas, B., Chassanoff, A., Contaxis, N., Jabloner, P., (2017). Exploring Curation-ready Software: Improving Curation-readiness. doi:*10.17605/OSF.IO/T9G3Q*.

Appendix: Challenges reported to the CURE-FAIR WG

Literature

We present an overview of 22 references rather than an exhaustive literature review. We first identified the corpora of more than 50 items of journal and conference papers, webinars, web pages, and other sources dealing with computational reproducibility and open science we included in an internal Zotero group library. We started with a small group of core references on tools for computational reproducibility and built upon it in a manner similar to a snowball sample. Scanning their context, we selected the 22 references summarized in this report. Although we were not focused on a specific discipline, the papers represent case studies in Archeology, Magnetic Resonance Imaging, Biostatistics, and the Social Sciences. Our overview incorporates a small portion of the body of knowledge and represents an insight into selected challenges and experiences related to computational reproducibility in science. Additional references were added after incorporating suggestions during the review process.

Crowd-sourced use cases

We invited the RDA community to contribute use cases. Between October 8, 2020 and January 21, 2021, we have recorded a total of 60 responses. Respondents represent a wide variety of roles. The majority are researchers, mostly from STEM domains, and about one-fourth are from the social sciences. Other respondents included research software engineers, data professionals, reproducibility advocates, data scientists, journal editors, students, and teachers. We used Atlas.ti to manage and process the information from the google form and present the “codes” we used below.

We analyzed the challenges involved in preparing and reusing materials required for computational reproducibility by using the FAIR foundational principles as a framework. The FAIR principles “define characteristics that contemporary data resources, tools, vocabularies and infrastructures should exhibit to assist discovery and reuse by third-parties.” Applying the FAIR framework can illuminate areas of weakness when it comes to reproducibility, as we elaborate below.

Mapping CURE-FAIR challenges onto FAIR

Findable: This governing principle is concerned with the ability to unambiguously identify the data, software, workflow, or other digital objects required for reproducibility. The challenges include,

- Difficulty finding data;
- Difficulty finding software;
- No software / data citation.

Accessible: The main issue with accessibility is the ability to retrieve a working version of data, software, workflow, or other digital objects required for reproducibility. The challenges include,

- Authors not making the data, software, workflow, or other digital objects available;
- Inability to access software because it is proprietary;
- Inability to access data because of the high cost of archiving big data;
- Inability to access data due to the absence of a persistent identifier;
- Inability to access data because the repository no longer exists;
- Inability to access dependencies;
- Inability to access original computing environment;
- Inability to access data when conditions of use are unclear.

Interoperable: The concern here is that data, software, workflow, or other digital objects are working with other objects or are portable so as to enable reproducibility. Challenges include,

- Files not working in another computing environment.

Reusability: This principle concerns the ability to optimize the use of data, software, workflow, or other digital objects for an intended purpose. Focusing on computational reproducibility, the main challenges include,

- Little or no documentation;
- Code not working / not executable;
- Code did not run as intended;
- Code is obsolete, or written in a different format;
- Differences between software versions;
- Differences between operating systems;
- Uncertainty about whether the reproducibility package will still work in the future;
- Curating for reproducibility is very hard to do.
- Unclear usage license

Beyond-FAIR challenges

Some of the challenges of preparing and reusing materials required for computational reproducibility go beyond the FAIR principles. These challenges include factors having to do with the current workings of scholarly publication, with a misaligned incentive structure, and with the skills and knowledge of researchers.

In terms of **scholarly publication**, community members see journals as slow to adapt to cultural and technological advances relating to computational reproducibility. Increasing requirements on authors and reviewers is seen as a challenge. Authors are asked to provide materials (e.g., README, Codebook, data citation) that they may not have prepared in addition to the data and the code. Reviewers for those journals that implement a reproducibility review process are often asked to handle code that “has not gone through QA of any sort and may not run" (ID28, Research Software Engineer). In some domains, the cost of sharing raw data to fulfill a journal's data and code sharing requirements is high. In domains that rely heavily on modelling, for example, the value of raw data may decrease over time. For example, a service provider working with climate scientists says that, "raw data (model output) storage is no longer necessary" when it exceeds the computer capacities (ID58, Service Provider). This point highlights that different domains may have different standards about what needs to be preserved and what exactly needs to be reproduced.

Additionally, respondents point out that many journals are not set up for executing workflows in order to verify computational reproducibility or integrating such activities in the traditional review process: "The tasks of journal editors gets harder (needs to match expertise), the review process can become more complex, review works best if communication between author and code checker is effective (non-anonymous) yet publishing platforms are not build for that, the publishing business is very slow to change and hesitant to take on additional tasks even if obviously beneficial, and increasing requirements for authors may lead authors to submit elsewhere which is undesirable" (ID36, reproducibility advocate). Another respondent, a Genomics researcher, has had the experience of failing to computationally reproduce results published in an article, contacting the authors who acknowledged an error, but the journal was not open to publishing a correction (ID9, Researcher).

Inadequate **incentives and rewards systems** to do reproducible science, and to meaningfully share it, is another challenge mentioned by respondents. Respondents point out that cultural change is slow, that even when required to share their materials, there is currently no benefit to sharing good quality material.

The need for better **skills and knowledge** about both doing and publishing reproducible science is another issue that respondents have brought up. One area that has been identified is the lack of training about the documentation necessary for future users to interpret and use the materials or even basic file management. A data curator points out that, "training and resources for creating robust documentation about verification submissions is limited" (ID23, Data Curator). In some domains, code or software necessary to reproduce results is not widely available and, as one respondent, a researcher in bioinformatics, says that, "in order to analyse the dataset, I have to learn how to use some specific software, which I don't think is well-spent time." (ID50, Researcher).

In-depth interviews

With consent from the interviewees, we include a summary of these conversations here.

Predrag Pejović, Department of Electrical Engineering, University of Belgrade (via Zoom). Dr. Pejović identifies the following challenges, **cross-platform** (e.g., to use Derive code in Maxima) and **cross-machine** (e.g., to use Windows for code written under Linux OS, transition from 34 to 64 bit OS and vice versa) reproducibility are much difficult to deal with than cross-version (e.g., run Python 2 code in Python 3 environment; these can be classified as Interoperable and Reusable FAIR challenges); Code verification is very important and at the same time **cumbersome procedure for journal publication** for all participants (this can be classified as an institutional challenge); Formats are crucial especially for the computational reproducibility in time i.e., **document** reproducibility (this can be classified as a Reusable FAIR challenge). Dr. Pejović also offers the following recommendations: Students of Electrical Engineering and Computer Science should be made aware of computational reproducibility; and scientific software should be free software.[10]

Daniel Heydebreck, German Climate Computing Center in Hamburg, Germany (via Zoom). Dr. Heydebreck has a scientific background in Earth System Science (ESS) and until recently worked as data management support scientist developing methods to assist researcher make their data reusable – he therefore reported on his experience regarding the challenges associated with computational reproducibility from both perspectives, i.e. that of a researcher and that of a data management support scientist. From the perspective of a scientist in ESS, a major challenge hindering the reproducibility of computational analyses/workflows is **missing documentation** as well as a **lack of technical understanding** of applied software and/or analysis methods. Furthermore, **updates to software** and the associated libraries as well as computing hardware, e.g. compute clusters, often lead to old codes not executing anymore. For his own purposes, Daniel has used git to track his own versioning history – which has shown to be of great advantage. From the perspective of the data management support scientist, the main challenge to ensure computational reproducibility was the amount of time and **effort** it takes for the scientists to prepare their analysis workflows and make them reusable. Also, depending on the sub-discipline, the perception, that the produced data is **not going to be used** by anyone anyway, is widespread. Simply put, scientists want to actually do science rather than documenting their work. Further challenges are the general lack of support staff and funding, the lack of **education** in research data management topics and group or model specific data formats requiring a high-level of expertise to use.

Christopher Kadow, German Climate Computing Center in Hamburg, Germany (via Zoom). Dr. Kadow has a scientific background in meteorology and has been developing and deploying custom tailored collaborative research environments in the past years. From his point of view, the main challenges of achieving computational reproducibility are the constantly **changing computational environments** which the scientists are exposed to. It is practically impossible to anticipate future computational environments, both in terms of hard- and software, and how current analysis workflows would execute under those conditions, e.g. in 10 years from now. Determining the amount and content of **documentation** needed is therefore not straightforward. Furthermore, the data basis for scientific analyses is constantly updated, which gives rise to the question about what one would be willing to reproduce in the future. For Christopher, it is therefore clear that reproducing the (close to) raw data is not necessarily useful (as it would be too cumbersome), as it is rather the possibility to check whether scientific results from earlier

[10] A full transcript of the interview is available via Zenodo, http://doi.org/10.5281/zenodo.4469568

work are still valid given new approaches. Ideally, having a piece of software determining the criteria allowing for the reproducibility (to a certain degree) of a given dataset would be ideal. Another aspect currently hindering the creation of computationally reproducible workflows/analyses are still not established **reward** systems.

Markus Demleitner, German Astrophysical Virtual Observatory (via email). Dr. Demleitner points out that the main challenge is the FAIR-imposed requirement of **machine-readability** of any digital data object. This may be important for datasets, but a scientific analysis also needs to be highly human readable in order to comprehend the methodology and sense behind it. Focusing on machine-readability may reduce interpretability by humans. Furthermore, the lack of an existing **reward** system is not conducive to scientists producing useful provenance information. Instead, imposing the requirement to provide provenance information without showing an actual benefit to the respective researcher may result in bogus provenance. Finally, the point is made that it **must be easier** for scientists to provide documentation and provenance information to their workflows, e.g. through standardized formats such as W3C provenance.

[name] Code Ocean (via email). Harvard University has a grant to explore metadata for containers. Stencila on github and containerMD on bibnum. The journal, Political Analysis on Harvard Dataverse repository started using Code Ocean for computational reproducibility in late 2019. Their primary challenge is that **authors are not familiar** with what a cloud container is and how they operate. In the process of implementing this, it took a long time for the journal to document for their authors how computational reproducibility works; they encountered limitations with both authors and the Code Ocean environment that had to be overcome; and **integration** with general repositories is an issue because general repositories have size ingest limitations and no directed support for computational reproducibility clients.

Ana Trisovic, Harvard Dataverse (via email). Challenges of computational reproducibility include the lack of **standards** for sharing research code in the repositories (for example, Python has standard configuration files to capture environments, which can be encouraged in the repositories, but other programming languages do not provide such files). Another issue is the lack of code metadata which would support advanced components such as containers and workflows. Furthermore, we observe that even when there is support for rich metadata (standard exists), many of the fields are not used and are later missing. How can we make completing metadata easier? See paper [9] and talk on enabling FAIRness with containers and the slides.

Lars Vilhuber, American Economics Association (AEA) Data Editor (via blog). Working with journals to verify computational reproducibility of materials deposited by authors in support of their manuscripts, Dr. Vilhuber identifies the back and forth **communication** with authors as one of the challenges. Reasons for contacting authors often have to do with **permission** to use and disseminate the data. For example, dealing with a data provider's "take down request" which means the author needs to update the deposit before it can be republished, or working with the legal office to include the Data Editor in the Data Use Agreement.

Another reason to communicate with authors is the need for more **documentation** including, for example, a copy of the Jupyter notebooks or a README. Debugging **code errors** is a major component of the Data Editor's work, sometimes also requiring communication with the authors. Other challenges include assessing and securing **computing resources** to run the computation, and the inability to find and cross-list replication packages or data from a trusted repository that is not the journal's repository.[11] In separate correspondence, Dr. Vilhuber mentions that several economics journals have endorsed and systematically require the use of a "Template README for the Social Sciences," reporting excellent impact.

RDA Plenary Participants

RDA VP16 Participants who attended the session, Current state of curation practices that support computational reproducibility, self-selected into virtual breakout rooms to discuss the question, "**What are the primary challenges you have encountered (or that you anticipate encountering) in curating for reproducibility?**"

Many issues were discussed, and we list them here:

- FAIR
- FAIR is not enough
- Workflow issues
- Acceptance of blackbox environment
- Sensitive Data issues
- Large Data/Large computations
- PIDs: more is better, early and often
- Capacity
- Costs
- Software dependencies
- Incentives
- Poor code/documentation
- Changing the cultural to one that welcomes feedback/verification
- Updated stats packages=different results
- Training for curators and researchers
- Start early (researchers)
- Communication!
- Paradata
- System compatibility
- Software environment sustainability
- Software preservation/networks
- Standardizing definitions of code/software/etc…
- Services and online dependencies
- GUI-driven software results
- Practical vs possible reproducibility

[11] See "A Day in the Life of a Data Editor," https://aeadataeditor.github.io/posts/2021-01-31-day-in-the-life

- Versioning of software (PIDs)