Characterising Reproducibility Debt in Scientific Software: A Systematic Literature Review

Zara Hassan[a,*], Christoph Treude[b], Michael Norrish[a], Graham Williams[a] and Alex Potanin[a]

[a]The Australian National University
[b]Singapore Management University

ARTICLE INFO

Keywords:
Reproducibility Debt
Technical Debt
Computational Reproducibility
Scientific Software
Systematic Literature Review

ABSTRACT

Context: In scientific software, the inability to reproduce results is often due to technical issues and challenges in recreating the full computational workflow from the original analysis. We conceptualise this problem as *Reproducibility Debt* (RpD). Much research has been performed to propose solutions to tackle these issues across various computational science disciplines. It is essential to identify and accumulate existing knowledge on reproducibility issues and state-of-the-art solutions so as to provide researchers and practitioners with information that enables further research activities and RpD management in practice. **Objective:** In the context of scientific software, we aim to characterise RpD by providing a taxonomy of issues contributing towards its emergence and identification (causes, effects) and the common solutions discussed in the existing literature. **Method:** We conducted a systematic literature review, considering 2198 studies until January 2024, including 214 primary studies. **Results:** We propose the first taxonomy of RpD items consisting of 37 causes attributed towards its emergence, 63 corresponding effects under seven main categories, and 29 prevention strategies. Moreover, we also identify 39 specialised tools/frameworks supporting reproducibility. **Conclusion:** The main contributions of this work are (1) a formal definition of RpD; (2) a taxonomy of issues contributing towards RpD; (3) a list of causes and effects having implications for software professionals to identify and measure RpD in their projects; (4) a list of strategies and tools to prevent or remove RpD; (5) the identification of gaps in existing research to guide future studies.

1. Introduction

Technical Debt (TD) is defined as a "collection of design or implementation constructs that are beneficial in the short-term but set up a technical context that can make future changes more costly or impossible" (Ernst et al., 2021), meaning that developers adopt practices that are expedient in the short term but compromise the overall quality of software (Rocha et al., 2017; Tom et al., 2013). Multiple TD types have been identified, along with their occurrences (known as 'smells') and corresponding management strategies (Alves et al., 2016; Rios et al., 2018). TD is commonly introduced to achieve short-term goals, whilst its impact on a project can be mitigated and controlled through appropriate management processes (Freire et al., 2020; Fernández-Sánchez et al., 2017; Lenarduzzi et al., 2021). Identifying the existing TD types is essential to enable strategies for their timely management (Rios et al., 2018); therefore, we must continue investigating emerging TD types.

Scientists implementing scientific software[1] are also prone to introducing TD; even though they are domain experts, they often lack knowledge of common Software Engineering (SE) practices to develop high-quality software (Pinto et al., 2018; Heaton and Carver, 2015), or they may give priority to publishing results, thereby avoiding detailed documentation and rigorous testing (Johanson and Hasselbring, 2018; Kanewala and Bieman, 2014). Consequently, TD is not exclusive to traditional and commercial software but also manifests in scientific software, where its adverse effects can become even more concerning. This added complication is caused by scientific software's purpose, i.e., to produce research outcomes in a myriad of disciplines, which form the basis for future scientific research (Vidoni, 2021; Hannay et al., 2009).

The pervasiveness of scientific software across many computational science disciplines has resulted in the increased importance of open science practices, "a movement to make all research artefacts available to the public, thus, ensuring the transparency and reproducibility of scientific processes" (Méndez Fernández et al., 2019). Addressing this, several organisations are now promoting open science and reproducibility by providing tools to share relevant scientific software and corresponding datasets. Examples are Research Software Alliance (ReSA), which now has chapters across different regions, including Australia and New Zealand; the Australian Research Data Commons (ARDC), which provides open access to categorised datasets; and the Australian Research Council (ARC), whose commitment to open science is a requisite in their discovery grants. The National Science Foundation (NSF) also supports reproducibility and recommends publishing source code and data.

Despite gaining much importance, the reproducibility of scientific software and associated results remains a challenge

*Corresponding author
zara.hassan@anu.edu.au (Z. Hassan); ctreude@smu.edu.sg (C. Treude); michael.norrish@anu.edu.au (M. Norrish); graham.williams@anu.edu.au (G. Williams); alex.potanin@anu.edu.au (A. Potanin)
ORCID(s): 0000-0003-3416-2991 (Z. Hassan)

[1]"End-user application software that is written to achieve scientific objectives (e.g., Climate models), or tools that support writing code that expresses a scientific model and the execution of scientific code, or research software written to publish papers, production software written for real users" Kanewala and Bieman (2014).

across the myriad of computational sciences, resulting in growing challenges (Sculley et al., 2015; Ivie and Thain, 2019). In scientific software, the inability to reproduce results is often due to technical issues and challenges in recreating the full computational workflow from the original analysis (Canon, 2020; Peng, 2011; Ivie and Thain, 2019), which we conceptualise as *Reproducibility Debt* (RpD). However, only a few studies acknowledge it as TD (Abubakar et al., 2020; Geiger et al., 2018; Sculley et al., 2015). Moreover, there is a lack of a standardised RpD definition, smells are not categorised, and descriptions of issues and challenges are scattered throughout the literature; complicating the establishment of a shared vocabulary for the area.

Our Systematic Literature Review (SLR) aims to 'Characterise RpD in Scientific Software'. This includes providing a consolidated definition and taxonomy of RpD items derived from prior primary studies. We uncover and classify the causes and common problems attributed to its emergence alongside the organisation of solutions for its proactive management. Gaps are identified, and opportunities for the development of new research are presented, supporting further research in this area.

In the following, Section 2 presents related work and Section 3 details the methodology we used to perform the SLR. Our results are presented and discussed in Sections 4 and 5, while Section 6 identifies possible validation issues for this research. In Section 7, we conclude our review and identify future studies. Our reproducibility package containing raw data, qualitative and quantitative analysis files can be viewed here [2]

2. Related Work

This section reviews the significant contributions from relevant research in TD and identifies the importance of reproducibility and openness in scientific research software from different disciplinary perspectives.

2.1. Technical Debt (TD)

2.1.1. General Technical Debt

Since Cunningham (1992) first proposed the metaphor of TD, numerous studies have identified different types of TD and have devised processes and strategies for their Management (Avgeriou et al., 2016; McConnell, 2008; Li et al., 2014). Alves et al. (2016) conducted a systematic mapping study, considering 100 papers published from 2010 to 2014, to propose an initial taxonomy of TD types together with a list of existing strategies for their identification and management. Rios et al. (2018) then conducted a systematic tertiary study between 2012 and 2018, considering 13 secondary studies. They evolved the taxonomy of TD types and produced a list of indicators of their occurrence, organising a map of activities, strategies and tools to manage TD. Although the results of both studies encompassed several TD types, they worked with primary studies centred on traditional software development and did not consider RpD or scientific software. Likewise, Lacerda et al. (2020) only focused on identifying Code Smells and Code Debt to provide an assessment of which detection and refactoring tools/processes could be applied to counteract those smells.

From a human-centred perspective, Freire et al. (2020) used a system of validated surveys (InsighTD) to study the relationships between preventive actions and TD types, and although they did assess a large array of debts, they did not consider RpD either. Codabux et al. (2021) explored open peer-reviews written as GitHub issues to identify the TD in R packages submitted to *rOpensci*. They manually analysed 5000 comments from 157 packages and evolved a taxonomy of TD specific to R packages, including the perspective of editors, developers and reviewers. Once again, RpD was not discussed.

Several other studies identified TD through source code inspection (Brown et al., 2010; Zazworka et al., 2014), examining traditional software written in Java and C. Tsoukalas et al. (2021) applied machine learning (ML) to identify and assess TD in Java projects. They used 18 metrics related to each Java class for statistical and ML models to classify them as 'High-TD or not' and proposed a prototype tool for auto-assessment of TD in Java projects. Subsequently, Tsoukalas et al. (2022) introduced a tool named 'TD Classifier' based on earlier work. TD Classifier incorporates the knowledge extracted by accumulating the results of three widely used TD assessment tools Squore (Baldassari, 2013), SonarQube (Saarimaki et al., 2019) and CAST (Curtis et al., 2012), and relied on other open-source tools to spot high-TD classes in Java projects from their Git repository. While TD Classifier assists the developer in TD management activities, it is limited to the Java programming language and would need to be expanded to support other languages used in the development of scientific software.

Tang et al. (2021) formulated a taxonomy of refactoring in ML and deep learning (DL) systems. They introduced 14 ML-specific and 7 TD-specific categories by analysing 26 ML projects. As a result, they suggested best practices and anti-patterns to assist practitioners and developers in evolving long-lasting ML systems and to assist educators in teaching methods for tackling TD in ML and DL systems.

In summary, while TD has been extensively explored within software development, RpD has not yet received significant attention. The software development community has recognised the importance of addressing TD to ensure the long-term viability and maintainability of software systems; the scientific community is increasingly grappling with issues related to the reproducibility of scientific software and associated results.

2.1.2. Self-Admitted Technical Debt (SATD)

Potdar and Shihab (2014) used source code comments from four open-source Java projects to investigate TD and named it SATD in recognition of the developers admitting to having TD in their project declared through source-code comments. Later, Liu et al. (2020) identified SATD in DL

[2] https://doi.org/10.6084/m9.figshare.25540978.v1

frameworks, widely used in scientific applications of computational sciences. Inline comments admitting to incurring (knowingly or not) different TD types were analysed in seven popular open-source DL frameworks, including TensorFlow [3] and Keras [4], uncovering the presence of significant TD. The situation was concerning since TD affects the quality and reliability of applications developed using such frameworks, especially due to the pressure for a reduced delivery time (Lim et al., 2012). Their work identifies seven different TD types (design, algorithm, requirements, defect, documentation, test and compatibility) but does not mention RpD.

Vidoni (2021) conducted mixed-methods studies to investigate SATD in R Programming, mining 500 R packages from GitHub, representing over 164,000 source code comments, and conducting two surveys. This manual study identified 12 TD types (including Algorithm Debt) but did not address RpD. Xavier et al. (2022) investigated situations that led developers to document TD using either code comments or issues. They used a large dataset comprising 74,000 comments and 20,000 issue instances from 190 GitHub projects and suggested guidelines for developers to decide between issues or comments for reporting TD. Tan et al. (2022) conducted an empirical study to investigate self-fixed TD using static analysis of some 44,000 commits from Java and Python projects. They assessed the extent to which developers fixed their own introduced TD, the type of debt (Documentation, Code, Design, Test, and Defect) more likely to be self-fixed, and their remediation time compared to non-self-fixed TD.

Seminal works in SATD (Potdar and Shihab, 2014; da Silva Maldonado et al., 2017) assessed automated approaches for SATD detection in many data sources and with multiple classifiers (Bavota and Russo, 2016; Flisar and Podgorelec, 2019; Sierra et al., 2019; Xavier et al., 2020). Manual investigations were also conducted to uncover specific nuances (Maldonado and Shihab, 2015; Fucci et al., 2021). These studies focused almost exclusively on a common dataset of ten corporate, large-scale Java-based repositories from Potdar and Shihab (2014) and do not mention RpD.

2.2. Reproducibility and Openness in Scientific Research

2.2.1. Conferences' Interests in Reproducibility

Openness and transparency of the scientific research process are fundamental aspects for the progression of science (Raghupathi et al., 2022). The increasing reliance on computing models in scientific research has prompted the scientific community to address concerns regarding the reproducibility of scientific computations (Bajpai et al., 2017; Freire et al., 2012). Consequently, prominent engineering and scientific forums prioritise raising awareness and implementing policies to facilitate the reproducibility of scientific computations.

The Association for Computing Machinery (ACM) has implemented rigorous policies to ensure the reproducibility of research outputs. This includes the establishment of the Reproducibility Task Force[5], which collaborates with Special Interest Groups (SIGs) conferences[6], and Engineering Interactive Computing Systems (EICS)[7]. Their efforts focus on promoting best practices in reviewing software and data artefacts and enhancing re-usability through improved documentation and review processes. According to the ACM, research is reproducible "when its results can be generated by an independent team using the same experimental setup as (the) original researchers"[8]. Furthermore, the Institute of Electrical and Electronics Engineers (IEEE) established a task force in 2020 dedicated to advancing open science and reproducibility. This task force aims to analyse models, practices, and experiences to support open science and reproducibility within the IEEE Computer Society and among peer societies and publishers.[9]

MethodX[10], *SoftwareX*[11] and Journal of Systems and Software (JSS) [12] are renowned journals published by Elsevier, dedicated to promoting openness and reproducibility in research methodologies and software. *MethodX* specifically adheres to the FAIR data principles (Findable, Accessible, Interoperable, and Reusable), aiming to enhance the discoverability of methods, protocols, reviews, and associated research. By fostering collaboration and facilitating discussions for improvement throughout the research cycle, *MethodX* contributes significantly to open science and the enhancement of reproducibility. Similarly, *JSS* has recently introduced the JSS Open Science initiative to support reproducibility in scientific research. A paper published in this special section is reviewed by the JSS Open Science board to ensure transparent and reproducible research.

In contrast, *SoftwareX* promotes domain-independent software by supporting their publication, establishing their scientific relevance, and making them accessible for inspection, validation, and re-use. Beyond merely facilitating the reproducibility of software and associated data, *SoftwareX* also provides credit to the authors and contributes to academic advancement by enabling software citation.

[3] https://github.com/tensorflow/tensorflow

[4] https://keras.io/

[5] https://www.acm.org/publications/artifacts

[6] https://www.acm.org/special-interest-groups/volunteer-resources/officers-manual/conferences

[7] https://eics.acm.org/

[8] https://www.acm.org/publications/policies/artifact-review-and-badging-current

[9] https://ieeecs-media.computer.org/media/tech-news/ieee-reproducibility-practices-survey-summary-of-findings-1.pdf

[10] https://www.sciencedirect.com/journal/methodsx

[11] https://www.sciencedirect.com/journal/softwarex

[12] https://www.sciencedirect.com/journal/journal-of-systems-and-software/special-issue/10XMGH48FFT

2.2.2. *Reproducibility in Empirical Software Engineering (ESE)*

Among the ESE research community, problems related to reproducibility have gained interest during the last two decades (Miller, 2005; Vegas et al., 2006; Shull et al., 2008). This has led to the development of the International Workshop on Replication in ESE Research (RESER).[13] The workshop aims to establish a platform for ESE researchers where they can debate theoretical foundations, methods and results of replication studies.

González-Barahona and Robles (2012) highlighted the importance of reproducibility as the desirable property of research studies. They provided a systematic methodology for assessing the reproducibility of ESE studies based on data retrieved by mining software repositories. Similarly, Anchundia and Fonseca C. (2020) identified tools to maximise reproducibility in SE experiments and analysed replication from new perspectives, namely, communication, knowledge management and motivation. Apart from working on reproducibility from a new point of view, their research is limited to the reproducibility of ESE research and does not mention the reproducibility issues in scientific software across disciplines. Rodríguez-Pérez et al. (2018) studied reproducibility and credibility in ESE using the case study of the popular SZZ (Śliwerski, Zimmermann and Zeller) algorithm (Śliwerski et al., 2005), but their findings are limited to ESE research and are solely based on an isolated study of primary research using the SZZ algorithm.

Piccolo and Frampton (2016) assessed the strengths and limitations of various tools, techniques and strategies to achieve better computational reproducibility. Trisovic et al. (2020) approached the integration of data repositories and reproducibility tools to ensure computational reproducibility. Samuel and König-Ries (2022) assessed the understandability of scientific experiments as a critical reproducibility component. Still, none identified specific types of smells, causes, or techniques to manage them in the context of RpD.

3. Methodology

We conduct a Systematic Literature Review (SLR) according to the guidelines proposed by Kitchenham and Charters (2007). SLR is a systematic and auditable method to identify and interpret the evidence available in primary studies. The process of SLR is summarised in Figure 1 and explained in the following subsections.

3.1. Research Questions (RQs)

This study aims to characterise RpD in scientific software, thus identifying and consolidating existing knowledge on scientific software reproducibility across all computational science disciplines. The study addresses reported issues, developers' challenges, and proposed solutions by investigating related primary studies. We formulate the following RQs:

[13] https://dl.acm.org/doi/10.1145/1810295.1810429

RQ1: *Which primary studies have approached reproducibility and its aspects in scientific software, and how can they be categorised?*

We believe ours to be the first SLR in the area of scientific software reproducibility; therefore, categorising and organising existing literature is the first step to guide our further work. We first identify the number of primary studies published on scientific software reproducibility per year and per journal/conference, as performed by Rios et al. (2018) in their systematic mapping studies. We also quantify the number of studies published per scientific discipline. This information represents the extent to which each scientific discipline contributes towards solving the reproducibility issues. Moreover, it helps us identify the variations in the manifestation of RpD across various scientific domains by combining the results from RQ2 and RQ3.

RQ2: *What are the main categories of issues that contribute to RpD in scientific software?*

Many initiatives have been undertaken to report issues and developer's challenges that contribute towards RpD in scientific software projects (Ivie and Thain, 2019; Brinckman et al., 2019). We organise these to provide a structured overview and to enable the research community to share a common vocabulary. This research question identifies and defines the categories of issues that have gained the research community's attention, thus providing a high-level taxonomy of RpD items and a formal definition of RpD.

RQ3: *What are the underlying causes and effects of RpD in scientific software?*

Identification of TD is not only about understanding how and where it occurs but also analysing its causes and their corresponding effects (Melo et al., 2022; Rios et al., 2020). Rios et al. (2018) report that 'causes for TD insertion in software projects' are little explored in academic research. Therefore, this question aims to identify the causes attributed towards the emergence and identification of RpD in scientific software projects. To answer this RQ, we perform an aspect-wise analysis of extracted data, thus presenting a comprehensive list of RpD causes and their effects, aiding in its identification, measurement and prevention.

RQ4: *What solutions are presented in the existing literature to tackle reproducibility problems in scientific software?*

To resolve reproducibility issues, scientists have opted for several solutions to mitigate the reproducibility problem. Several guidelines have been proposed (Botvinik-Nezer and Wager, 2023; Maghami et al., 2023). Specialised tools and frameworks have been developed, such as *WholeTale* (Brinckman et al., 2019), *Galaxy Framework* (Goecks et al., 2010), as well as utilising existing software development tools and technologies (Ziemann et al., 2023; Canon, 2020). This RQ aims to identify and organise these solutions to support researchers and practitioners in preventing RpD in scientific software projects.

In wrapping up, according to Avgeriou et al. (2016), TD has three main components: debt items (cause), interest (effect), and prevention (principal). Hence, by answering RQ 1, 2, 3, and 4, we can characterise reproducibility as a type

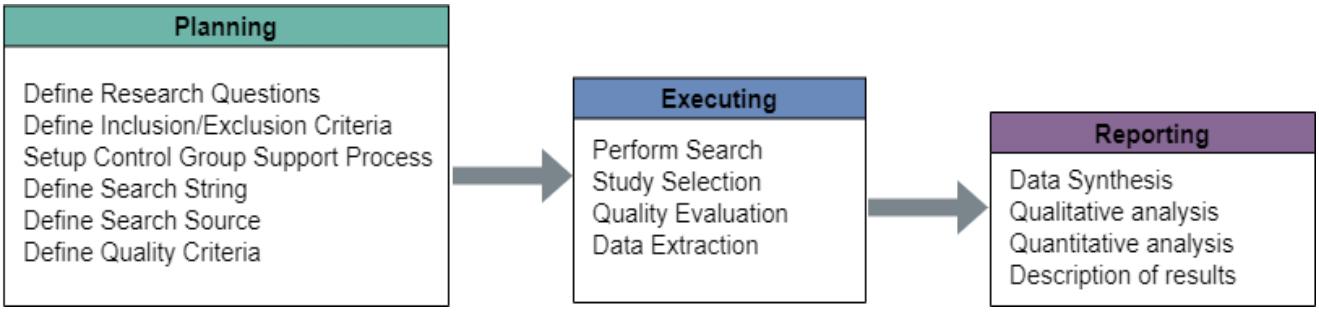

Figure 1: A systematic literature review process as we have adopted for characterising Reproducibility Debt, based on the guidelines proposed by Kitchenham and Charters (2007).

Table 1
Inclusion and Exclusion Criteria (IEC) for selecting primary studies.

Code	Criteria Description
I.1	Paper discusses open science, reproducibility and/or replicability (under this name or another) of scientific software (or related aspects) in any discipline.
I.2	The paper was published in a peer-reviewed academic venue.
I.3	Extended versions of conference papers; albeit, if the conference version is at least 30% different from its extended version, we keep both.
I.4	The articles published in any year.
E.1	The paper discusses open science but does not approach reproducibility or replicability.
E.2	Studies published in the form of abstracts/extended abstracts, keynotes, tutorials, Posters.
E.3	Venue is not peer-reviewed.

of TD in scientific software using evidence from existing literature.

3.2. Search Strategies

A broad search strategy in selecting the primary literature is instrumental to our review. We employ a Control Group Support Process adopted by Anchundia and Fonseca C. (2020) for evolving the search string. It is then necessary to define inclusion/exclusion criteria (IEC) at the beginning of the study to ensure the quality and relevance of the work being considered.

3.2.1. Inclusion/Exclusion Criteria

IECs are defined based on our research objectives to ensure the inclusion of peer-reviewed literature that discusses open science, reproducibility and/or replicability (under this name or another) of scientific software (or related aspects) in any discipline and correspondingly set aside all the venues domain experts have not reviewed. Moreover, no exclusion was done based on publication year to mitigate the threat of missing relevant papers. IECs are summarised in Table 1.

3.2.2. Control Group Support Process

After defining the IEC, we set up a Control Group (seminal papers discussing scientific software reproducibility) based on our research objectives following the IECs. Control papers (CPs) were studied in detail: (1) to extract the relevant terms for synthesising a search string; (2) to identify research gaps, thereby establishing the foundation for our study; (3) to verify that the search process easily finds the CPs. Each control paper has been assigned a `T.code` following this scheme: `CP-01`, where `CP` means control paper and `01` is used for indexing (See Table 2). A brief overview of each paper is given below:

In `CP-01` González-Barahona and Robles (2012) argued that Empirical Software Engineering (ESE) studies, those based on data retrieved from development repositories, are suitable for reproduction, as data and tools employed in these studies can be easily shared or described in detail. However, they identified that many studies in the ESE area are not reproducible, leading them to study factors (elements) that impact the reproducibility of ESE research. Based on the identified elements *(data source, retrieval methodology, raw dataset, extraction methodology, study parameters, processed dataset, analysis methodology, result dataset)*, they proposed a methodology for evaluating the reproducibility of a study in the ESE domain.

In `CP-02` Anchundia and Fonseca C. (2020) identified tools that maximise results' reproducibility in SE experiments based on analysis of 40 primary studies. They analysed replication from a new perspective, i.e., communication, knowledge management and motivation. They argued that tools and practices depend on the experiment domain (human- and technology-oriented experiments) and still lack acceptability and usability among ESE researchers, adversely affecting ESE results' reproducibility.

In `CP-03` Rodríguez-Pérez et al. (2018) studied reproducibility and credibility in ESE using the case study of SZZ (Śliwerski, Zimmermann and Zeller) algorithm (Śliwerski et al., 2005), widely used to detect the origin of the bug. The authors conducted an SLR to evaluate studies that use the SZZ algorithm. They reviewed a total of 187 papers to analyse whether the aspects of reproducibility were addressed, reporting limitations and use of improved versions of the algorithm. They observed that ESE research lacks reproducibility of results, concluding that results are not credible and dependable. Their research establishes a need

Table 2
The set of control papers used to obtain the relevant terms for the formulation of Search String.

T.Code	Author	Title	Publication Year
CP-01	(González-Barahona and Robles, 2012)	On the reproducibility of empirical software engineering studies based on data retrieved from development repositories	2011
CP-02	(Anchundia and Fonseca C., 2020)	Resources for Reproducibility of Experiments in Empirical Software Engineering: Topics Derived From a Secondary Study.	2020
CP-03	(Rodríguez-Pérez et al., 2018)	Reproducibility and credibility in empirical software engineering: A case study based on a systematic literature review of the use of the SZZ algorithm.	2018
CP-04	(Krafczyk et al., 2019)	Scientific Tests and Continuous Integration Strategies to Enhance Reproducibility in the Scientific Software Context.	2019
CP-05	(Ivie and Thain, 2019)	Reproducibility in Scientific Computing.	2018
CP-06	(Brinckman et al., 2019)	Computing environments for reproducibility: Capturing the "Whole Tale".	2018
CP-07	(Raghupathi et al., 2022)	Reproducibility in Computing Research: An Empirical Study.	2022

for improvement in ESE research and provides reproducibility guidelines for researchers and reviewers to address this problem.

In CP-04 Krafczyk et al. (2019) proposes scientific tests and continuous integration strategies to enhance the reproducibility of scientific software. The concept of scientific black box testing was introduced, comparing computational scientific results with those published to ensure their reliability and accuracy. They also proposed production-minimised computational experiments to run in a continuously integrated environment and publish the results with the main results. Moreover, the authors introduced the concept that a "scientific test produces computational results from a published article".

In CP-05 Ivie and Thain (2019) stated that publishing the source code and relevant data with sufficient detail permits other researchers to understand one's methodology and reproduce results. However, in practice, there are various technical and social barriers to achieving reproducibility in scientific computing, e.g., computing environment, source code, naming issues, and developers' motivations. They performed a detailed survey on scientific reproducibility, highlighting technical barriers in reproducing workflows and analysing extant approaches to achieve reproducibility.

In CP-06 Brinckman et al. (2019) presented "Whole Tale" software, a reproducible research environment which facilitates the linkage of code and data with publication. The Whole Tale aims to redefine the model through which computational and data-driven science is conducted, published, verified and reproduced. It supports the entire research process, from pre-publication to post-publication, by exposing the salient details via access to the persistent versions of the code and data used in the research. The Whole Tale strengthens the three layers of scholarly publication, i.e., data, process and computational analysis. Its architecture uses a range of flexible APIs to enable users to ingest and manage data and front ends and capture, replay, and extend publications.

In CP-07 Raghupathi et al. (2022) assessed the present state of reproducibility of research in computing based on previous research across all domains. They identified 25 variables relevant to reproducibility and categorised them into three factors, i.e., method, data and experiment, to measure three different degrees of reproducibility. The authors analysed almost 100 research articles and found that none of them documented all variables. However, only a few variables for each factor were documented, and the reproducibility score decreased with an increased requirement for documentation. Therefore, they argued that reproducibility in computing can only be increased as researchers prioritise reproducibility and utilise methods that ensure reproducibility; also, the publishers should increase their focus on the reproducibility aspects of research articles.

Although the work presented by these CPs has made a significant contribution in terms of factor identification, tools, methodology, and guidelines for reproducibility, none of them discern the issues of reproducibility as a TD. Also, most research is specific to a particular domain, resulting in isolated solutions. Therefore, a systematic approach to classify reproducibility as a type of TD (identifying all underlying causes, effects and prevention strategies) across all scientific disciplines remains a research gap.

3.2.3. Search String

While reading and analysing CPs, we identified and extracted the terms used to describe reproducibility in scientific software and other closely related concepts. To mitigate the threat of missing relevant papers, the *evaluation criteria* for the search string was defined at the start inspired by the work from Anchundia and Fonseca C. (2020), i.e., (1) the number of papers found is neither too large (More than 5k) nor too small (less than 50); (2) papers included in the control group are shown within 1-3 pages from top search results; (3) the titles of the searched studies are relevant to the research objectives.

Subsequently, we formulated and tested various search strings using combinations of all identified terms, i.e., `reproducibility, reproduce, reproducible, replicability, replicable, repeatability, repeatable, transparency, open science, open source, open code, scientific software,`

Table 3
Quality Assessment Criteria

ID	Quality Criteria (QC)	Score
QC1	The Paper discusses reproducibility issues and their solutions/practices, in the context of scientific software.	Yes=1, No=0
QC2	The objective of the research is clearly stated.	Yes=1, No=0
QC3	Is the research design appropriate to meet the objectives?	Yes=1, No=0
QC4	The research results and discussion support the objectives.	Yes=1, No=0
QC5	The related work section of the paper should be presented.	Yes=1, No=0
QC6	Paper cites at least 5 papers related to reproducibility.	Yes=1, No=0

`scientific workflows, computational software, scientific computing`. Since the default search options for different digital libraries work differently, we experimented with all combinations of search strings in all digital libraries and observed the outputs. We observe in our control papers that the term `reproducible` and `reproduce` always co-occur with `reproducibility`, so we use only `reproducibility` in the final search string.

When applied to *titles* and *full text*, the following combination of keywords using `AND` and `OR` operators give the optimum results according to our evaluation criteria, i.e., a reasonable number of papers, control papers appeared in top search, and titles seems relevant to our research objectives.

```
("reproducibility" OR "replicability" OR "repeatability") AND ("open source" OR "open science" OR "open code") AND ("scientific software" OR "scientific computing" OR "computational software")
```

3.2.4. Study Search

We conducted our search in September 2022 and re-ran it in January 2024, using a range of digital libraries, including ACM Digital Library, IEEE Xplore, Elsevier ScienceDirect, Wiley, Taylor & Francis, and SpringerLink. These selections were based on recommendations from prior studies in SE (Kitchenham et al., 2010; Petersen et al., 2015). Additionally, to prevent publisher bias and ensure comprehensive coverage, we included Google Scholar, following guidelines by Wohlin (2014). In our Google Scholar search, we considered papers from the top 50 pages. Our search process yielded a total of 2198 papers, which were collected using Zotero, a reference management system designed to facilitate the organisation and handling of academic papers.

3.3. Selection Process

The selection of primary studies involves the following steps:

1. *Duplicate Removal:* We first removed the duplicates, resulting in 2004 papers ready for the next filtering round.
2. *First Filtering:* We analysed the metadata (title, keywords, and venue) of each paper in comparison to the inclusion criteria I2, I3 and E2, E3 (See Table 1). At this point, we did not use I1 or E1 since it was not possible to know whether the article under evaluation discusses open science, reproducibility and/or replicability of scientific software based on the metadata alone.
3. *Second Filtering:* The abstracts of the papers surviving the first filtering were subjected to criteria I1 and E1 in the second filtering. I2, I3, E2, and E3 were not applied at this stage, as they had already been applied in the first round. Any article with a reasonable degree of doubt regarding its relevance to the research topic was kept for the next round of filtering.
4. *Quality Assessment:* We evaluated the studies' quality to ensure that the final selection list included the most relevant papers to our research objectives. The papers selected after the second filtering were assessed based on the quality criteria (QC) checklist presented in Table 3. We developed 6 QCs following the quality checklist template provided by (Kitchenham and Charters, 2007). Quality score was calculated for each primary study in the pool by a complete reading of the manuscript. Following the recommendations of Lima et al. (2019), the paper must satisfy a minimum of four QCs to qualify for inclusion in the final selection of this SLR, i.e., it had to achieve more than 50% of the criteria.
   We allocated a maximum score of 1 to every QC. QC1 strongly supports our primary research question as it allows us to include studies discussing issues contributing towards RpD and their proposed solutions. QC2, QC3 and QC4 further supplement QC1 by evaluating the quality of the paper based on its research objectives, design and results. We kept the flexibility by adding QC5 and QC6, as we worked with papers from multiple disciplines with different structures and published in venues with different requirements. Therefore, any primary study not meeting certain criteria but still conveying meaningful information about reproducibility issues and reaching a quality score of 4 was included as a part of the study. The reason for taking QC5 as an optional criterion is that many short peer-reviewed papers provide meaningful information but are limited in related work sections. QC6 is also an optional criterion because many non-SE venues may discuss reproducibility without citing anything related to it. Moreover, it mitigates the threat of losing any paper discussing reproducibility based on previous

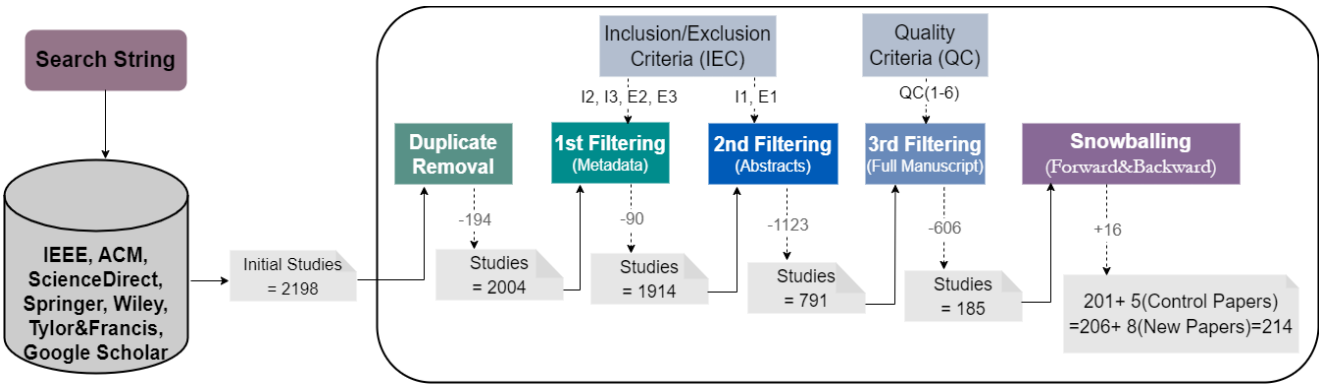

Figure 2: Number of included articles during the study selection process.

literature but not structured well in terms of research objectives and results.

5. *Snowballing:* We applied a snowballing *(forward and backward)*, to find relevant papers that were potentially missed out, and effectively mitigate this threat (Wohlin, 2014). In backward snowballing, the reference list of already selected papers was used to identify new papers. In forward snowballing, we used Google Scholar to search for new papers based on the list of already included papers. All obtained papers were selected by applying the same three-stage filtering process and selection criteria. This process yielded 15 new papers after 1st iteration, 1 paper in 2nd iteration and no new paper in 3rd iteration.

The study search, selection and filtering process is summarised in Figure 2

Filtering Protocol: During all three stages of filtering, the first author independently analysed metadata, abstracts and manuscripts. All results were reviewed and discussed between all authors of this paper in weekly meetings to mitigate researcher bias. ***Note:*** Due to the open nature of the discussion, inter-rater agreement was not calculated. To keep the filtering process transparent, the Zotero library can be viewed here [14].

3.4. Data Extraction

Data extraction is a critical element of the systematic review process, where required text and data from selected primary studies are collected consistently and explicitly (Cruzes and Dyba, 2011). To answer RQs mentioned in Section 3.1, we extracted data (text, phrases) from the selected primary studies according to the template presented in Table 4. To strengthen the validity of our data extraction process and to minimise the research bias, we used a systematic approach for data extraction, discussed in the next paragraph.

After selecting primary studies, we designed the extraction form in a spreadsheet format. During this stage, control papers were re-analysed to derive fields (categories) for data extraction. All fields were discussed to clarify which information should be included in a particular field. The original data extraction began once consensus was achieved on the extraction strategy. However, we remain open to adding new categories (columns in the spreadsheet), as new data/categories that might be substantial to answer RQs will be identified while reading manuscripts. Moreover, these refined categories help us directly answer RQ2, i.e., high-level taxonomy of RpD items, the presence of which hinders the reproducibility of scientific outcomes.

Extraction Protocol: During extraction, the first author independently extracts data from selected primary studies, which the second author reviewed. All authors met weekly and discussed discrepancies and feedback on the data extraction and classification (Kitchenham and Charters, 2007). During manuscript reading, we also considered where the information extracted came from; for example, some came from the methodology section, some from the results section, and some from a conclusion and discussion part. We also refined a guideline for reading the paper over time. Data extraction was done by copy-pasting fragments of data from paper into a spreadsheet. The data extraction file is shared in the replication package to maintain transparency.

3.5. Data Synthesis and Analysis

We conducted both qualitative and quantitative analyses on the extracted data to address the proposed RQs. The quantitative analysis of the characteristics of primary studies (publication year and venue) enabled us to answer RQ1. To answer RQ2, RQ3 and RQ4, we first performed the qualitative analysis of gathered reproducibility issues and solutions. Later, we performed a quantitative analysis of the emerged themes to describe their manifestation more precisely and accurately. This includes calculating frequencies or percentages of their occurrence in selected primary studies to quantify the prevalence and severity.

3.5.1. Qualitative Analysis Process

We used an open coding approach for qualitative analysis, which is a part of grounded theory Corbin and Strauss (2008). Using this approach, we aim to produce a set of

[14] https://www.zotero.org/groups/4762953/test_repro/library

Table 4
Data Extracted from Primary Studies

RQ	Data Item	Meaning
RQ1	Study ID	Unique identifier assigned to each primary study.
RQ1	Title	Title of the article under study.
RQ1	Year	Year in which research article is published.
RQ1	Venue	Venue at which article is published i.e., Journal or Conference.
RQ1	Research Domain	Refers to the specific field or area of study that the paper is focused on.
RQ1	DOI	Unique Digital Object Identifier assigned to each manuscript.
RQ2, 3	Tools-centric issues	Issues associated with the infrastructure, tools, and technologies used to develop and store scientific software.
RQ2, 3	Human-centric issues	Issues associated with individuals involved in scientific research.
RQ2, 3	Process-centric issues	Issues related to inadequate application of Software Engineering practices in scientific software development.
RQ2, 3	Documentation issues	Issues related to incomplete or unclear documentation, i.e., missing details about data sources, software dependencies, or specific configurations.
RQ2, 3	Legal issues	Issues related to the use of proprietary software or data. Also, licensing and accessibility issues of open-source.
RQ2, 3	Version-centric issues	Issues related to the version of software, code, or data used in the research.
RQ2, 3	Code-centric issues	Issues related to the development, organisation, documentation, and dissemination of scientific software code.
RQ2, 3	Data-centric issues	Issues related to the processing, storage, and dissemination of scientific data.
RQ4	Tools-centric solutions	Tools and technologies to solve reproducibility-related problems.
RQ4	Human-centric solutions	Human-centric guidelines or skills required to enhance reproducibility in scientific research.
RQ4	Process-centric solutions	Software Engineering practices to achieve reproducibility of results.
RQ4	Documentation-centric solutions	Documentation practices or guidelines to ensure rigour and reproducibility in scientific research.
RQ4	Solutions for legal issues	Guidelines or list of licences to overcome legal issues related to scientific software and data.
RQ4	Version-centric solutions	Guidelines or tools to resolve versioning issues.
RQ4	Code-centric solutions	Guidelines and best practices related to the development, organisation, and documentation of scientific software code.
RQ4	Data-centric solutions	Guidelines and best practices related to the processing, storage, and dissemination of scientific research data

concepts that fit our extracted data following a three-stage coding process.

1. *Line-by-line codes:* In this phase, we generated line-by-line codes from the extracted data. Since the data had been previously categorised, each category was coded individually.
2. *Axial codes within categories:* In the second stage, we performed an aspect-wise analysis of all emerged open codes within the categories and separated them into higher-level categories of *causes and effects* of RpD. This stage was iterative until we merged all codes obtained in the first stage into higher-level axial codes.
3. *Merging and moving of axial codes across categories:* During this stage, we revisited axial codes across all categories to check their relation with other codes. We performed two parallel operations during this phase. *First*, closely related axial codes across categories were merged based on common themes and citations supporting those themes to avoid duplication of codes and citations. *Second*, the final obtained axial code was moved to the most relevant category. For example, *Unclear descriptions of parameter settings used in experiments, simulations, or software tools* in Documentation-centric issues was merged with *Inadequately documented dependencies, versions and workflows* in Tools-centric issues as *Inadequately documented parameter settings, dependencies, versions and workflows* and was moved to Documentation-related issues. All line-by-line codes were revisited to check their relevance with merged axial codes to mitigate the threat of merging irrelevant codes at this stage.

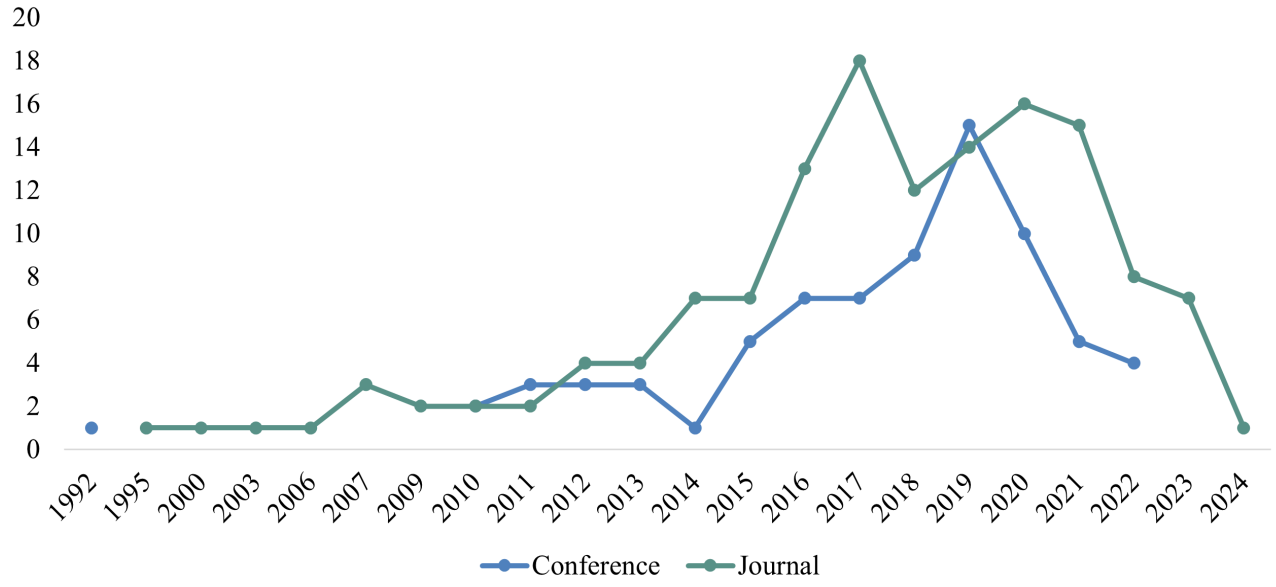

Figure 3: Study Distribution by Year and Publication Venue

Open Coding Protocol: Qualitative coding was performed by the first author and reviewed by the second author. Final codes in each category (both line-by-line and axial) and all merged codes across categories are reviewed and discussed among all authors. All axial codes were assigned unique IDs before merging, for example, DC1: Unclear descriptions of parameter settings used in experiments, simulations, or software tools and T3: Inadequately documented dependencies, versions and workflows. New code emerged after merging as DC1/T3, which was kept unique to maintain traceability in the replication package. All analysis and coding files are shared in the replication package.

New Search 2024: The initial study search was conducted in September 2022, and results are reported in 2024. To mitigate the threat of missing any relevant primary studies published during 2023, we ran another search in January 2024 after completing qualitative coding of already extracted data. We used the same search string and three-stage filtering process to select the eight new primary studies (Botvinik-Nezer and Wager, 2023; Wagner et al., 2024; Zhu et al., 2023; Maghami et al., 2023; Choi et al., 2023; Chan and Schoch, 2023; Ivimey-Cook et al., 2023; Ziemann et al., 2023). However, we did not extract data from these studies into a spreadsheet; instead, codes were directly applied to the text in the manuscript.

4. Results

In this section, we present the evidence found in the SLR to answer the RQs presented in Section 3.1.

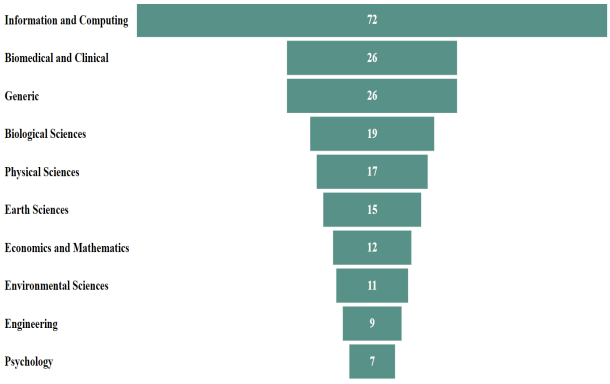

Figure 4: Number of primary studies per scientific discipline

4.1. RQ1: *Which primary studies have approached reproducibility and its aspects in scientific software, and how can they be categorised?*

The identification and filtering of studies discussing reproducibility in scientific software is summarised in Figure 2. We applied a three-stage filtering process to select 214 primary studies from seven digital libraries based on predefined IECs and QC. Figure 3 presents the distribution of selected primary studies, both over *year* and *publication venue* till January 2024. The graph shows that the first selected study was published in 1992, and only 19 studies were recorded till 2011. Almost 91% of selected studies were published after 2011. With this, we can say that reproducibility was recognised as a plausible issue among the research community, primarily after 2011. The maximum number of studies on reproducibility were published in 2019 (around 29), and the numbers slightly decreased afterwards.

When considering the distribution of studies based on publication venue, a relatively high percentage was identified in Journals (65% or 140 studies). Conferences started taking an interest in reproducibility after 2011. The year 2019 stood out with a maximum of 14 conference papers; the number started decreasing afterwards, with only 19 publications until 2022. From the trend, we can say that journals are more interested in publishing work relevant to computational reproducibility.

One of the goals of this work is to identify researchers' interest in computational reproducibility across various scientific disciplines and the types of reproducibility issues reported by them. In this regard, a text describing the scientific discipline was extracted from the paper and recorded in the spreadsheet. We used Australian and New Zealand Standard Research Classification (ANZSRC) Field of Research (FoR) codes to classify and present the collected literature meaningfully to guide our analysis. The categories in the classification include major fields and related subfields of research. The extracted data determined the research field and papers with related sub-fields were merged into higher-level categories of the major field according to ANZSRC classification guidelines. *For example*, research articles from software engineering, machine learning, and high-performance computing were merged into *Information and Computing Sciences*. *Biomedical and Clinical* articles include neuroimaging, bio-informatics, anatomy, and physiology; *Physical Sciences* include astronomy, high energy physics, plasma physics, remote sensing, and molecular dynamics; and *Earth Sciences* include hydrology and ocean science. All other manuscripts discussing computational reproducibility, in general, were grouped into one generic category.

We identified that 214 primary studies were distributed among nine main scientific disciplines. We also identified that papers were written by almost 190 different authors (we record the first author's name only), showing a broad interest in this subject across various disciplines. The graph in Figure 6 shows that Information and Computing Science has the highest number of papers highlighting reproducibility issues, with 72 papers (approximately 33%). This indicates a significant focus on reproducibility within this field. Following closely behind Information and Computing Science, Biomedical and Clinical Science have 26 papers addressing reproducibility issues, highlighting the importance of reproducibility in medical research. We identified 26 primary studies discussing computational reproducibility in a generic manner covering all computational science disciplines which involve complex simulations and modelling, where reproducibility is essential for validating results and ensuring their reliability. Other scientific fields also have their contribution in highlighting and addressing reproducibility issues, i.e., Biological Sciences has 19 papers; Physical and Earth Science follows with 17 and 15 papers; Economics and Mathematics has 12 papers collectively; Environmental Science follows with 11 papers; Engineering has 9 papers. Psychology has the lowest number of papers, approximately 3% of the total.

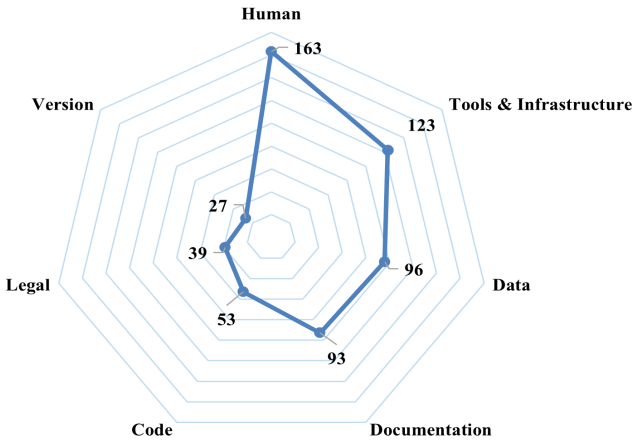

Figure 5: Type of reproducibility issues mentioned in primary studies

4.2. RQ2:*What are the main categories of issues that contribute to RpD in scientific software?*

Prior work by Izurieta et al. (2016) demonstrated that identifying TD items, i.e., a list of the bad practices or issues that create debt, is usually the first step in identifying any TD in software systems. We, therefore, categorised the issues or bad practices contributing towards RpD into seven main categories. This classification gives us the high-level taxonomy of RpD items as follows:

1. Data-Centric Issues: These are related to scientific data processing, storage, and dissemination.
2. Code-Centric Issues: These are related to developing, organising, and disseminating scientific software code.
3. Documentation-Centric: Incomplete or unclear documentation can hinder understanding the processes required to reproduce research findings. This includes missing details about data sources, software dependencies, or specific configurations used in analysis.
4. Infrastructure and Tools-Centric Issues: These are associated with the infrastructure, tools, and technologies used to develop and store scientific software.
5. Versioning Issues: Arise when the version of software, code, or data used in the original research is unavailable or incompatible with other software used in replication.
6. Human-Centric Issues: These are associated with individuals involved in scientific research, including software developers, domain researchers, reviewers, and funding organisations.
7. Legal Issues: Refers to the intellectual property and ownership issues in open research software and data. Also, proprietary software or data can create reproducibility barriers, especially when others lack access to the same tools or licenses.

The quantitative analysis of all emerged codes under these categories gives us the representation of their occurrence in selected primary studies. The graph in Figure 5 shows the types of issues contributing towards RpD and the total number of primary studies mentioning them, either directly (self-identified) or indirectly (cited others as a baseline of their work). It is observed that issues and challenges related to the people involved in the scientific software development process are mentioned in 163 primary studies. Following human-centric issues are tools and infrastructure-related issues highlighted by 123 primary studies and subsequently data-centric (96 studies), documentation-centric (93 studies), code-centric (53 studies), legal issues (39 studies), and finally, versioning issues (27 primary studies). We observed that, on average, at least three reproducibility issues are mentioned in each primary study.

This taxonomy of RpD items helped us to define RpD as:

> Reproducibility Debt (RpD) is a type of technical debt primarily impacting scientific software. It refers to the accumulative issues and challenges that hinder the ability to reproduce scientific outcomes, stemming from challenges inherent in scientific software code and data, including development, organisation, dissemination, and documentation. RpD may accumulate when researchers and scientific software developers engage in sub-optimal activities, often for short-term benefits, which ultimately compromise the reproducibility of research outcomes.

4.3. RQ3: *What are the underlying causes and effects of RpD?*

In accordance with the research conducted by Li et al. (2015); Avgeriou et al. (2016); Rios et al. (2018), a pivotal factor in the identification of TD is the provoking elements that caused its emergence. In this regard, this question aims to identify the main factors *(causes)* of the emergence of RpD and their corresponding effects to facilitate its identification and to present the main indicators of its occurrence. After the analysis, we identified 100 codes which are classified as *causes and effects* of RpD, based on the evidence obtained in the recorded data. In total, 37 codes were classified as causes of RpD, and 63 codes appeared as their effects. The complete catalogue of causes and effects of RpD are shown in Table 5 along with their evidence of occurrence in selected primary studies. We classified them under seven main categories discussed in RQ2.

Data-Centric: In total, five data-related causes associated with the emergence of RpD appeared with their eight corresponding effects, presented in the first block of Table 5. We identified that the ripple effect of *non-systematic data processing and analysis* manifests in the form of unorganised data and analysis files. Moreover, the consequence of non-systematic approaches extends to limited traceability of data and metadata. Another listed RpD item *incomplete or selective reporting of research data sets* is driven by constraints in storage space, a desire for taking full advantage of data, or inadvertent oversight, thus introducing RpD into the research landscape. This results in low-quality data sets with missing values, thus frustrating other researchers while reproducing and building upon that data. The absence of standardised practices, i.e., *non-standardised data and metadata formats for storage, sharing and reuse*, creates a fragmented landscape, making it challenging for researchers to seamlessly share, understand, and reuse each others' data. Moreover, the consequential effect of employing varying data and metadata formats manifests as limited data interoperability.

Code-Centric: The second block of the table 5 presents five code-centric causes attributed towards the emergence of RpD and their eight effects. RpD is compounded in scientific software when researchers employ *intricate algorithms and complex code*, often necessitated by the complexity of their research questions. As a result, a layer of complexity is added, hindering subsequent efforts to reuse the code. Other code-related causes of RpD are *lack of good programming practices, lack of comprehensive testing and formal code review processes* resulting in low-quality and ill-maintained code. Moreover, RpD manifests when researchers rely on *software or libraries that are either unstable or lack proper maintenance*. Scientific software developers often lack awareness about the importance and process of code-sharing and do not prioritise systematic code-sharing practices.

Documentation-Centric: causes and effects attributed towards the emergence and identification of RpD are presented in block three of Table 5. The first RpD item in this block is related to the *insufficient or outdated documentation of data* involved in scientific research, which manifests in the form of missing details about data pre-processing, analysis and loss of data provenance information. Moreover, *inadequately documented workflows and dependencies* also incorporate RpD, resulting in a limited understanding of developed software for future use. Moreover, RpD is incurred when *researchers employ manual and ad-hoc methods for preparing and sharing documentation*. The resulting inconsistent and unstructured documentation affects the verifiability, reusability and maintainability of developed software. The restricted length and format of research articles in various conferences and journals is another cause of RpD, manifesting in the form of incomplete descriptions of computational steps in scholarly publications.

Tool-Centric: Several causes attributed towards the emergence and identification of RpD relevant to the software/hardware tools and technologies used to develop and execute scientific software are presented in block 4 of Table 5. RpD emerges if *necessary dependencies required for the build process are missing or get updated over time*. This restricts others' ability to re-execute a program, or results are inconsistent across every new build. Similarly, *inconsistent software/hardware settings in different computing environments* by original researchers and those trying to reproduce may cause RpD to manifest as unexpected behaviour or

Table 5
Causes attributed towards emergence of RpD and their corresponding effects

ID	RpD Causes	Effects (Interest)
1.0	**Data-Centric**	
1.1	Non-systematic data processing and analysis	Unorganised data and analysis files, Limited traceability of data and metadata
	Huppmann et al. (2019); Gil et al. (2016); Orozco et al. (2020); Marwick (2017); Ram (2013); Millman and Pérez (2018); Davis-Turak et al. (2017); Harrell et al. (2022); Lifschitz et al. (2011); Rokem et al. (2017); Peng (2011); Gentleman and Lang (2007); Lefebvre and Spruit (2023); Taubert and Bücker (2017); Peng et al. (2006); Bell et al. (2017); Jimenez et al. (2017b); Akhlaghi et al. (2021); Stevens (2017)	
1.2	Incomplete or selective reporting of datasets	Frustrating process of replication, Missing values, Low-quality datasets
	Krafczyk et al. (2019); Goble et al. (2013); Wittek et al. (2021); Bell et al. (2017); Bugbee et al. (2020); Waltemath and Wolkenhauer (2016); Vilhuber (2020); Choi et al. (2023)	
1.3	Non-standardised data and metadata formats for storage, sharing and reuse	Inconsistent results
	Brinckman et al. (2019); Lefebvre and Spruit (2023); Feger et al. (2020); Zhao et al. (2018); Waltemath and Wolkenhauer (2016); Wilson et al. (2017); Mecum et al. (2018); Alencar et al. (2018); Peer et al. (2021); Rokem et al. (2017); Gorgolewski et al. (2017); Tierney and Ram (2021); Marwick (2017); Kim et al. (2018); Yu et al. (2016); Hinsen (2011); Goecks et al. (2010)	
1.4	Lack of an integrated and trusted infrastructure for storing, processing and distributing growing volume of data	Uncertain long-term data availability
	González-Barahona and Robles (2012); Ivie and Thain (2019); Brinckman et al. (2019); Huber et al. (2021); Garrett-Ruffin et al. (2021); Spencer Smith et al. (2016); Brunsdon and Comber (2020); Piccolo and Frampton (2016); Davis-Turak et al. (2017); Feger et al. (2020); Bell et al. (2017); Bugbee et al. (2020); Kanwal et al. (2017); Bánáti et al. (2015); Jenkins et al. (2016); Cook et al. (2012); Jansen et al. (2019); Mcdougal et al. (2016); Alencar et al. (2018); Fiore et al. (2018); Gorgolewski et al. (2017); Tierney and Ram (2021); White et al. (2019); Leek and Jager (2017); Chen et al. (2019); Yu et al. (2016); Ibanez et al. (2018); Bjorn et al. (2019); Gomes et al. (2022); Peng (2011); Gentleman and Lang (2007); Howe (2012); Maghami et al. (2023)	
1.5	Heterogeneous and complex datasets	Challenging data understanding and re-usability
	Raghupathi et al. (2022); Garrett-Ruffin et al. (2021); Denaxas et al. (2017); Eckersley et al. (2003); Mendez et al. (2020); Brunsdon and Comber (2020); Bell et al. (2017); Bugbee et al. (2020); Wilson et al. (2017); McPhillips et al. (2019); Stodden and Miguez (2014); Tierney and Ram (2021); Marwick (2017); Yu et al. (2016); Blinov et al. (2021); Gomes et al. (2022); Peng et al. (2006); Goecks et al. (2010)	
2.0	**Code-Centric**	
2.1	Complex algorithms and code	Limited re-usability of code, Inability to track code versions
	Orchard and Rice (2014); Morrison (2018); Garcia-Silva et al. (2019); Lupelli et al. (2015); Bilke et al. (2019); Poldrack et al. (2019); Fernandez-Prades et al. (2018); Cook et al. (2012); LeVeque (2009); Hutton and Henderson (2018); Boettiger (2015); Dylan Chapp (2020); Jimenez et al. (2017b); Peng (2011); Krafczyk et al. (2019); Rollins et al. (2014)	
2.2	Lack of good programming practices by developers and lack of formal code review process	Low-quality code and non-reproducible results
	Niso et al. (2022); Pernet and Poline (2015); Denaxas et al. (2017); Chue Hong (2018); Bast (2019); Alarid-Escudero et al. (2019); Widder et al. (2019); Bahaidarah et al. (2022); Gomes et al. (2022); Botvinik-Nezer and Wager (2023); Chan and Schoch (2023); Ivimey-Cook et al. (2023)	
2.3	Unstable or ill-maintained software or libraries	Critical issues or bugs may remain unaddressed, Lack of compatibility with the latest updates
	Chue Hong (2018); Fernandez-Prades et al. (2018); Di Meglio et al. (2012); Dalle (2012); Pörtner et al. (2018); Cruz et al. (2018); Kim et al. (2018); Gille et al.	
2.4	Lack of awareness about the importance and process of code-sharing	Low priority accorded to share-ability of code by researchers, Use of non-systematic approach and methods for code sharing on demand
	Rollins et al. (2014); Hinsen (2011); Isdahl and Gundersen (2019); Mcdougal et al. (2016); Hutton and Henderson (2018); Chen et al. (2019); Brito et al. (2020); Morin et al. (2012); Bahaidarah et al. (2022); Gomes et al. (2022); Peng (2011); Gentleman and Lang (2007); Botvinik-Nezer and Wager (2023); Wagner et al. (2024)	

Table 5 Continued

ID	RpD Causes	Effects (Interest)
2.5	Lack of comprehensive testing (unit, integration, and regression tests) Pimentel et al. (2019); Ram et al. (2019); Lee et al. (2021)	Low maintainability
3.0	**Documentation-Centric**	
3.1	Inadequate or outdated data documentation González-Barahona and Robles (2012); Garcia-Silva et al. (2019); Denaxas et al. (2017); Piccolo and Frampton (2016); Davis-Turak et al. (2017); Zhao et al. (2018); Waltemath and Wolkenhauer (2016); Dylan Chapp (2020); Marwick (2017); Vilhuber (2020); Leipzig et al. (2021); Bánáti et al. (2016); Mecum et al. (2018); Ghoshal et al. (2021); Raghupathi et al. (2022); Cito and Gall (2016); Irving (2016); Lee et al. (2021); Santana-Perez and Pérez-Hernández (2015); Popp and Biskup (2022); Goecks et al. (2010); Ziemann et al. (2023)	Missing data preprocessing and analysis details, Missing details about parameters used for generation of simulation data, Loss of data provenance information
3.2	Inadequately documented dependencies, versions and workflows González-Barahona and Robles (2012); Essawy et al. (2020); Kanwal et al. (2017); Waltemath and Wolkenhauer (2016); Balz and Rocca (2020); Ghoshal et al. (2021); Pörtner et al. (2018); Cruz et al. (2018); Hutton and Henderson (2018); Cito and Gall (2016); Föll et al. (2019); Meng et al. (2015); Gorgolewski et al. (2017); Rougier et al. (2017); Jimenez et al. (2017b); Gomes et al. (2022); Peng (2011); Robles (2010); Chan and Schoch (2023); Ziemann et al. (2023)	Lack of understanding about used software and versions
3.3	Missing/outdated/unstructured documentation and code comments for large growing projects Essawy et al. (2017); Orozco et al. (2020); Kedron et al. (2021); Feinberg et al. (2020); Perkel (2020); Ram et al. (2019); Piccolo and Frampton (2016); Crick et al. (2017); Knoll and Heedt (2020); Fehr et al. (2016); Wagner et al. (2024); Zhu et al. (2023)	Limited utility of information in the document, Reduced code understand-ability
3.4	Manual ad-hoc methods of preparing and sharing documentation Denaxas et al. (2017); Orzechowski et al. (2020); Baiocchi (2007); Castleberry et al. (2012); Mauerer and Scherzinger (2021); Balz and Rocca (2020); Kalenkovich and Levchenko (2021); Smith et al. (2016); Widder et al. (2019)	Affects verify-ability, re-usability and maintain-ability of software
3.5	Restricted length and format of research articles Rollins et al. (2014); Niso et al. (2022); Skaggs et al. (2015); Goble et al. (2013); Edmunds et al. (2017); Baiocchi (2007); Buckheit and Donoho (1995); Mcdougal et al. (2016); Marwick (2017); Krafczyk et al. (2021); Fidler et al. (2017); Schwab et al. (2000)	Incomplete and selective description of methods and computational steps, black box science
3.6	Disconnect between scholarly publications and their underlying data, models, code and methodology used to produce the findings Brinckman et al. (2019); Hinsen (2011); Wattanakriengkrai et al. (2022); Howison and Bullard (2016); Skaggs et al. (2015); Baiocchi (2007); Erdemir et al. (2016); Vitek and Kalibera (2011); Cito and Gall (2016); Irving (2016); Boettiger (2015); Nüst and Eglen (2021); Fidler et al. (2017); Stodden et al. (2016); Crook et al. (2013); Gentleman and Lang (2007); Raff and Farris (2023)	Impairs a researchers' ability to build upon results

build failure. Moreover, the *lack of standardised and trusted infrastructure* for permanent storage and sharing scientific workflows is another contributing factor towards RpD. It results in the lack of access to the scientific software package over time with a consequential effect of uncertainty in the long-term re-execution of shared workflows. Researchers in various scientific disciplines prefer to *use trivial or old software tools* for computational purposes, thus incorporating RpD, which manifests as compatibility issues with the latest software or tools.

Version-Centric: The causes mentioned under this category are particular to the issues arising from version changes impacting reproducibility. For example, the *version upgradation in programming languages, supporting libraries, underlying operating systems, and other dependencies* may cause RpD to manifest as limited usability, portability and backward compatibility of the developed software.

Human-Centric: TD in scientific software, much like traditional software, is accrued by the individuals participating in the development process. In this context, the development process refers to the series of activities, decisions, and tasks undertaken by a team of developers, scientists, and researchers involved in creating and maintaining scientific software. Hence, the last block of Table 5 presents the causes

Table 5 Continued

ID	Causes (RpD)	Effects (Interest)
4.0	**Tools and Infrastructure**	
4.1	Missing dependencies for complex software systems	Inability to re-execute the program, Inconsistent results
	Rollins et al. (2014); Essawy et al. (2020, 2017); Garcia-Silva et al. (2019); Choi et al. (2021); Chue Hong (2018); Pimentel et al. (2019); Knoll and Heedt (2020); Bánáti et al. (2016); Apostal et al. (2018); Kanwal et al. (2017); Bánáti et al. (2015); Dalle (2012); Balz and Rocca (2020); Wang et al. (2020b); Pörtner et al. (2018); Nguyen et al. (2019); Canon (2020); Hosny et al. (2016); Gorgolewski et al. (2017); Marwick (2017); Hale et al. (2017); Parashar (2020); Crick et al. (2017); Vaillancourt et al. (2020); Benthall and Seth (2020); Rougier et al. (2017); Blomer et al. (2015); Jimenez et al. (2017b); Akhlaghi et al. (2021); Clyburne-Sherin et al. (2019); Howe (2012); Piccolo and Frampton (2016); Davis-Turak et al. (2017); Chan and Schoch (2023); Ziemann et al. (2023)	
4.2	Inconsistent software/hardware settings in different computing environments	Unexpected behaviour or even failure to build
	Ivie and Thain (2019); Choi et al. (2021); Apostal et al. (2018); Cook et al. (2012); Mcdougal et al. (2016); Wang et al. (2020b); Pörtner et al. (2018); Canon (2020); Tatman et al. (2018); Hosny et al. (2016); Meng et al. (2015); Gorgolewski et al. (2017); Marwick (2017); Hale et al. (2017); Brito et al. (2020); Vaillancourt et al. (2020); Jimenez et al. (2017b); Kellogg et al. (2019); Mauerer and Scherzinger (2022); Canon and Younge (2019); Maghami et al. (2023)	
4.3	Lack of integrated and trusted infrastructure for permanent storage, sharing and execution of scientific workflows	Lack of access to a software package, uncertain long-term re-execution of workflows
	Rollins et al. (2014); Gil et al. (2016); Edmunds et al. (2017); Waltemath and Wolkenhauer (2016); Hidayetoğlu et al. (2022); Jimenez et al. (2017a); Ghoshal et al. (2021); Hutton and Henderson (2018); Ram et al. (2019); Bast (2019); Widder et al. (2019); von Hahn and Mechefske (2022); Leek and Jager (2017); Benthall and Seth (2020); Akhlaghi et al. (2021); Santana-Perez and Pérez-Hernández (2015); Peng (2011); González-Barahona and Robles (2012); Goble et al. (2013); Denaxas et al. (2017); Taylor et al. (2016); Di Meglio et al. (2012); Robles (2010); Mcdougal et al. (2016); Cushing et al. (2018); da Silva and Guareis de Farias (2019); Blomer et al. (2015); Nüst et al. (2017)	
4.4	Outdated or non-trivial software tools/infrastructure usage	Incompatibility with modern tools
	Essawy et al. (2020); Mcdougal et al. (2016); Peer et al. (2021); Santana-Perez and Pérez-Hernández (2015); Ivie and Thain (2019)	
4.5	Custom Scripts and Tools	Tightly coupled systems
	Kanwal et al. (2017); Canon (2020); Jimenez et al. (2017b)	
4.6	High resource requirement for replication (HPCs or GPUs)	Lengthy download and runtime
	Krafczyk et al. (2019); Erdemir et al. (2016); Taylor et al. (2016); Cook et al. (2012); Jansen et al. (2019); Balz and Rocca (2020); LeVeque (2009); Marrone et al. (2019); Ghoshal et al. (2021); Widder et al. (2019); Hey and Payne (2015); Maghami et al. (2023)	
4.7	Hidden states and out-of-order executions in notebooks	Inconsistencies between the reproduced and the originally recorded results
	Wang et al. (2020a,b); Akhlaghi et al. (2021); Casseau et al. (2021)	
4.8	Non-deterministic order of execution in parallel systems	Accumulation of rounding errors
	Waltemath and Wolkenhauer (2016); Taufer et al. (2010); Jalal Apostal et al. (2020); Jean-Paul et al. (2019); Revol and Théveny (2014); Langlois et al. (2015); Marrone et al. (2019); Canon (2020); Dylan Chapp (2020); Parashar (2020); Rougier et al. (2017); Jézéquel et al. (2015); Boettiger (2015); Ziemann et al. (2023)	

relevant to the people involved in the scientific software development process. Scientific software often requires a large volume of data from diverse contexts to produce results; however, the *lack of sufficient knowledge and training in data management skills* leads to RpD, manifested in low-quality research artefacts. Moreover, a *lack of training towards using state-of-the-art SE tools and technologies* also restricts them from tracking changes in code and software configurations over time. It may also result in reduced collaboration among large interdisciplinary teams and insufficient planning, design and documentation.

Table 5 Continued

ID	Causes (RpD)	Effects (Interest)
5.0	**Version-Centric**	
5.1	Version upgradation in programming languages, libraries, operating systems, and other dependencies	Limited usability, portability and backward compatibility
	Krafczyk et al. (2019); Denaxas et al. (2017); Chue Hong (2018); Piccolo and Frampton (2016); Mukherjee et al. (2021); Bentley et al. (2019); Pimentel et al. (2019); Wang et al. (2020a); Fernandez-Prades et al. (2018); Jean-Paul et al. (2019); Hidayetoğlu et al. (2022); Cook et al. (2012); Dalle (2012); Mcdougal et al. (2016); Bast (2019); Hosny et al. (2016); Lee et al. (2021); Gorgolewski et al. (2017); Perkel (2020); Parashar (2020); Bjorn et al. (2019); Crook et al. (2013); Clyburne-Sherin et al. (2019); Ivie and Thain (2019); Baiocchi (2007); Peer et al. (2021)	
5.2	Issues with version relaxation	Non-deterministic build process
	Goswami et al. (2020)	
6.0	**Human-Centric**	
6.1	Lack of knowledge and training in data management	Low quality research artefacts
	Baiocchi (2007); Davis-Turak et al. (2017); Feger et al. (2020); Mauerer and Scherzinger (2022); Pröell et al. (2015); Wilson et al. (2017); Fiore et al. (2018); Tierney and Ram (2021); Leek and Jager (2017); Lowndes et al. (2017); Gomes et al. (2022)	
6.2	Lack of training or inadequate application of software engineering practices and tools (version control and containerisation technologies)	Inability to track changes in code and software configurations over time, Reduced collaboration among large interdisciplinary teams, insufficient planning, design and documentation
	Brinckman et al. (2019); Wattanakriengkrai et al. (2022); Spencer Smith et al. (2016); Orozco et al. (2020); Wilson et al. (2014); von Hahn and Mechefske (2022); Hey and Payne (2015); Lowndes et al. (2017); Crick et al. (2017); Beaulieu-Jones and Greene (2017); Benthall and Seth (2020); Jimenez et al. (2017b); Kellogg et al. (2019); Crouch et al. (2013); Raghupathi et al. (2022); Bell et al. (2017); Widder et al. (2019); Kanwal et al. (2017); Ram et al. (2019); Smith et al. (2016); Chirigati et al. (2013); Clyburne-Sherin et al. (2019); Goecks et al. (2010); Zhu et al. (2023); Chan and Schoch (2023)	
6.3	Lack of recognition, reward and incentives for reproducibility-oriented practices (data and code sharing, exhaustive testing and documentation)	Reduced priority of reproducibility practices, Prolonged research cycles, Culture of non-sharing
	Raff and Farris (2023); Jiménez et al. (2017); Garcia-Silva et al. (2019); Niso et al. (2022); Shamir et al. (2013); Goble et al. (2013); Eckersley et al. (2003); Chue Hong (2018); Bontemps and Orozco (2021); Feger et al. (2020); Frery et al. (2020); Marek et al. (2018); Waltemath and Wolkenhauer (2016); Di Meglio et al. (2012); Dalle (2012); Isdahl and Gundersen (2019); Wilson et al. (2017); Robles (2010); Balz and Rocca (2020); Rozier and Rozier (2014); Cruz et al. (2018); Kalenkovich and Levchenko (2021); Ram et al. (2019); Bast (2019); Boettiger (2015); Widder et al. (2019); Milham and Klein (2019); Stodden and Miguez (2014); Tierney and Ram (2021); White et al. (2019); von Hahn and Mechefske (2022); Leek and Jager (2017); Fidler et al. (2017); Gille et al.; Hey and Payne (2015); Ibanez et al. (2018); Vilhuber (2020); Stodden et al. (2013); Morin et al. (2012); Kellogg et al. (2019); Gomes et al. (2022); Laine et al. (2007); Bell et al. (2017); Harrell et al. (2022); Gil et al. (2016); Orozco et al. (2020); Bell et al. (2017); Harrell et al. (2022); Skaggs et al. (2015); Pernet and Poline (2015); Kedron et al. (2021); Mauerer and Scherzinger (2022); LeVeque (2009); Peng (2011); Gil et al. (2015); Nüst et al. (2017); Chan and Schoch (2023)	
6.4	Lack of formal training and precise guidelines by institutions and journals on reproducibility-oriented practices(data and code sharing, exhaustive testing and documentation)	Lack of confidence in developed software for sharing and scrutiny, Possible adoption of ad-hoc software development practices
	Raff and Farris (2023); Botvinik-Nezer and Wager (2023); Jiménez et al. (2017); Scheliga et al. (2019); Levet et al. (2021); Wattanakriengkrai et al. (2022); Pernet and Poline (2015); Denaxas et al. (2017); Eckersley et al. (2003); Bontemps and Orozco (2021); Kedron et al. (2021); Bajpai et al. (2017); Feger et al. (2020); Kalenkovich and Levchenko (2021); Ram et al. (2019); Bast (2019); Irving (2016); Boettiger (2015); Lee et al. (2021); Widder et al. (2019); Milham and Klein (2019); Mccormick et al. (2014); Kim et al. (2018); Leek and Jager (2017); Hey and Payne (2015); Morin et al. (2012); Benthall and Seth (2020); Kellogg et al. (2019); Akhlaghi et al. (2021); Crouch et al. (2013); Gomes et al. (2022); Choi et al. (2023)	
6.5	Fragmented Collaboration in large-sized projects between domain researchers and developers	Inconsistent standards and practices, divided research community
	Orozco et al. (2020); Kedron et al. (2021); Di Meglio et al. (2012); Wu et al. (2011); Widder et al. (2019)	

Table 5 Continued

ID	Causes (RpD)	Effects (Interest)
6.6	Resources *(human, time, funding)* constraints and publication pressure	Incomplete documentation, Insufficient validation, Degraded research quality, Reduced collaborations
	Raghupathi et al. (2022); Huppmann et al. (2019); Anzt et al. (2019); Chue Hong (2018); Bajpai et al. (2017); Dalle (2012); Rozier and Rozier (2014); Widder et al. (2019); Leek and Jager (2017); Robinson et al. (2021); Bontemps and Orozco (2021); Waltemath and Wolkenhauer (2016); Balz and Rocca (2020); Cruz et al. (2018); Ram et al. (2019); Milham and Klein (2019); Hey and Payne (2015); Botvinik-Nezer and Wager (2023)	
6.7	Resistance to adopt reproducibility-oriented tools and practices by experienced researchers	Culture of irreproducible research
	Rollins et al. (2014); Gil et al. (2016); Frery et al. (2020)	
6.8	Lack of realisation of reproducible research's importance	Insufficient efforts to produce reproducible research
	Goble et al. (2013); Edmunds et al. (2017); Bell et al. (2017); Goswami et al. (2020); Lee et al. (2021); Stodden and Miguez (2014); Stodden et al. (2013); Kellogg et al. (2019); Crouch et al. (2013); Peng (2011)	
7.0	**Legal Issues**	
7.1	Intellectual property and ownership issues in Open Research Software and data	Limited re-usability of research artefacts, Complications in the public distribution of code and data over copyright concerns
	Ivie and Thain (2019); Morrison (2018); Bontemps and Orozco (2021); Feger et al. (2020); Balz and Rocca (2020); Rozier and Rozier (2014); Engel et al. (2017); Hutton and Henderson (2018); White et al. (2019); Ibanez et al. (2018); Gomes et al. (2022); Peng et al. (2006); Laine et al. (2007); Skaggs et al. (2015); Eckersley et al. (2003); Frery et al. (2020); Mcdougal et al. (2016); Koehler Leman et al. (2020); Stodden (2009)	
7.2	Fear of legal consequences over sharing of sensitive data involving human subjects	Limited data sharing and its reuse, Requiring additional effort on data anonymisation, Reduced understand-ability of data
	Ivie and Thain (2019); Mendez et al. (2020); Bontemps and Orozco (2021); Kedron et al. (2021); Feger et al. (2020); Pröell et al. (2015); Wilson et al. (2017); Rozier and Rozier (2014); Engel et al. (2017); Dorodchi et al. (2019); Gomes et al. (2022); Morrison (2018); Eckersley et al. (2003); Hutton and Henderson (2018)	
7.3	Proprietary software usage	Costly licence reproducibility
	Kanwal et al. (2017); Balz and Rocca (2020); LeVeque (2009); Alarid-Escudero et al. (2019); Mauerer et al. (2023); Föll et al. (2019)	

Another factor contributing towards RpD is the *lack of recognition, reward and incentives for reproducibility-oriented practices* (data and code sharing, exhaustive testing and documentation), which require researchers' time and effort, thus creating a culture of non-sharing, reduced priority of reproducibility practices and prolonged research cycles. Although reproducibility is encouraged by many institutions, journals and conferences, precise step-by-step guidelines to ensure reproducibility at every stage of scientific software development are still missing, leading to the adoption of ad-hoc practices.

Legal Issues: Intellectual property and ownership issues contributing towards RpD are multifaceted. The complex landscape of intellectual property law introduces uncertainties and challenges related to ownership rights of research artefacts. Open research often involves the use of various licensing agreements to govern the usage and distribution of software and data. Ambiguities or lack of clarity in these licenses can create confusion, hindering researchers' ability to confidently share their work while maintaining control over its use. Moreover, *fear of legal consequences over sharing sensitive data involving human subjects*, and lack of standardisation for determining and documenting ownership of research artefacts adds to the complexity. Additionally, the evolving nature of intellectual property laws and insufficient guidance collectively contribute to RpD.

4.4. RQ4:*What solutions are presented in the existing literature to tackle reproducibility problems in scientific software?*

Various scientists, during their course of action to resolve reproducibility issues, have proposed many solutions to manage them. In this regard, several guidelines and practices were proposed, specialised tools and frameworks were developed, or existing software engineering tools and technologies were suggested to ensure reproducibility. We, therefore, categorised them under three main headings for better understanding and systematic adoption.

4.4.1. Prevention Strategies

The guidelines or best practices to deal with reproducibility issues are coded as prevention strategies to manage RpD. To formulate them, we accumulated the existing guidelines and practices from the selected primary studies, induced the best out of them and combined them semantically, keeping in view the identified RpD items.

In total, 29 prevention strategies are crafted, shown in Table 6. The careful adoption of them in combination with each other can help researchers and scientific software developers in the management of RpD. For instance, a complete description of research methods, including data collection, processing, and analysis, can prevent RpD if documented from the start of the project and accurately linked to the fragments of code implementing the method. Similarly, data and code sharing in open trusted repositories (GitHub, Gitlab, Zenodo, Figshare) are supported by a large number of primary studies, as access to code and data enables reproducibility; however, useful sharing requires training to develop well-organised and documented code using version control systems. Moreover, the allocation of resources (human, time and funding) is required to support the sharing of code and data that is well-written, documented and developed using version control.

4.4.2. Existing Tools and Technologies

The existing SE tools, such as version control systems, containers and literate programming notebooks, can prevent RpD in scientific software projects.

Version Control is an established SE tool to track changes in any set of files, including source code, documentation and other scripts. Using version control in scientific software development can prevent RpD, allowing developers to keep track of all the changes made in scientific software code, data and supporting documentation. Moreover, Distributed Version Control Systems (DVCSs) allow researchers to collaborate and build upon each others' work. DVCSs also support code protection against unexpected deletion, overwrite or power failure; however, they are not permanent archiving solutions. We, therefore, recommend the use of DVCSs only during the scientific software development process and use open, trusted repositories such as Figshare and Zenodo for their long-term archiving and citation (See S1, S2, S9, S11, S16 in Table 6).

Containers 44 primary studies have suggested the use of virtualisation and containerisation technologies to ensure reproducibility (S8 in Table 6). Containers can be considered lightweight virtual machines that package a complete computational environment, including all required code, data, dependencies and configuration in a single *image*. Container images can be distributed publicly as a single executable file with an open-source licence. Moreover, documentation generated along with the image contains all necessary information about packaged software.

Careful engineering of containers makes them suitable for reproduction, allowing people with limited knowledge about a complex application to reproduce the software. Moreover, combining container technologies with continuous integration systems allows further validation and building upon existing software (Fernandez-Prades et al., 2018; Beaulieu-Jones and Greene, 2017). Different container and VM technologies are there; however the most suitable for reproducibility, as discussed in existing literature are Docker, Singularity (now Apptainer), Sciunit, Kubernetes, Mesos, VirtualBox and VMware. Moreover, integrating containers with cloud-based HPC infrastructure provides flexibility to develop and deploy scientific software in a portable and reproducible way (Vaillancourt et al., 2020).

Literate Programming Notebooks The use of literate programming notebooks to prevent RpD is supported by 16 primary studies. Literate programming notebooks combine chunks of code with human-readable text to create an executable document (Knuth, 1984). We recommend the use of notebooks (Jupyter, Markdown, Knitr) for small to medium-size analysis (S17 in Table 6), because as the project size increases the inherent problems of notebooks as highlighted by Pimentel et al. (2019) may accumulate over time resulting in growing debt.

4.4.3. Specialised Tools and Infrastructure

Version control, container, literate programming notebooks, documentation, and persistent sharing of code and data are pillars of scientific software reproducibility (Ziemann et al., 2023; Wang et al., 2020a; Clyburne-Sherin et al., 2019). Therefore, various scientists and research organisations have developed specialised tools, technologies and frameworks by using these existing SE tools to allow their easy integration with scientific software development processes to prevent RpD.

During the analysis, we identified 39 specialised platforms developed to proactively prevent RpD in scientific software.

Executable Electronic Document: The use of electronic documents for reproducible research was initially demonstrated by Claerbout and Karrenbach (1992). The idea was extended and refined with more sophisticated approaches. One approach is *Research Objects (ROs)* "semantically rich information units that aggregate and structure scientific assets such as research data, the computational resources and software processing the data in an experimental or observational context, and the scientific publications that communicate the subsequent findings to the wider scientific community". Transparency in ROs may prevent RpD (McPhillips et al., 2019). *Experiment Digital Object (EDO)* is a record published in an open repository as a collection of data and executable files used to reproduce computational experiments. *ReDoc* is based on the idea of an electronic document (Claerbout and Karrenbach, 1992), which allows readers to reproduce computational research using authors underlying data and program. *Research Compendium* is a " unit of scholarly communication which includes the research paper, the code, and the data" (Gentleman and Lang, 2007; Hinsen, 2011), which was further extended as Executable Research Compendium "a new standardise

Table 6
Prevention Strategies (Principal) to Manage RpD

ID	Prevention Strategies
S1	Use open trusted repositories for storing and publishing data as a package (data, metadata, provenance, documentation of processing-analysis steps and relevant scripts) Niso et al. (2022); Choi et al. (2021); Popp and Biskup (2022); Gil et al. (2016); Skaggs et al. (2015); Pernet and Poline (2015); Denaxas et al. (2017); Brunsdon and Comber (2020); Feinberg et al. (2020); Feger et al. (2020); Bugbee et al. (2020); Peng et al. (2006); Bánáti et al. (2015); Stodden et al. (2016); Stevens (2017); Mecum et al. (2018); Robles (2010); Stodden (2010); Fiore et al. (2018); Tatman et al. (2018); Peer et al. (2021); Rokem et al. (2017); Milham and Klein (2019); Stodden and Miguez (2014); Dylan Chapp (2020); Tierney and Ram (2021); Marwick (2017); White et al. (2019); Kim et al. (2018); Chen et al. (2019); Yu et al. (2016); Peng (2011); Ibanez et al. (2018); Barba (2019); Brito et al. (2020); Morin et al. (2012); Leipzig et al. (2021); Gentleman and Lang (2007); Gomes et al. (2022); Stodden et al. (2016); Lupelli et al. (2015); Botvinik-Nezer and Wager (2023); Maghami et al. (2023); Ziemann et al. (2023)
S2	Select repository that guarantees long-term storage provides versioning support and assigns DOIs with citation templates Garrett-Ruffin et al. (2021); Gil et al. (2016); Lefebvre and Spruit (2023); Eckersley et al. (2003); Huppmann et al. (2019); Isdahl and Gundersen (2019); Ram (2013); Rokem et al. (2017); Stodden and Miguez (2014); Tierney and Ram (2021); Marwick (2017); Leek and Jager (2017); Yu et al. (2016); Blinov et al. (2021); Barba (2019); Stodden et al. (2016); Peng (2011); Stodden (2010); Pernet and Poline (2015)
S3	Research sponsors and organisations should allocate sufficient funds to enable the long-term storage, archiving, and access of datasets Brinckman et al. (2019); Rollins et al. (2014); Balz and Rocca (2020); Cruz et al. (2018); Hutton and Henderson (2018); Milham and Klein (2019); Ibanez et al. (2018); Stodden et al. (2013)
S4	Data and metadata format standardisation and harmonisation Niso et al. (2022); Huber et al. (2021); Garrett-Ruffin et al. (2021); Gil et al. (2016); Zhao et al. (2018); Jenkins et al. (2016); Waltemath and Wolkenhauer (2016); Poldrack et al. (2019); Kalenkovich and Levchenko (2021); Mecum et al. (2018); Kim et al. (2018); Lowndes et al. (2017); Blinov et al. (2021); Barba (2019); Crick et al. (2017); Leipzig et al. (2021); Robles (2010)
S5	Provide Controlled access to sensitive data or grant data access to a designated third party to conduct certified reproductions Eckersley et al. (2003); Bontemps and Orozco (2021); Kedron et al. (2021); Lifschitz et al. (2011); Brito et al. (2020); Vilhuber (2020); Peng et al. (2006); Laine et al. (2007)
S6	Promote research data management education, support and attribution Feger et al. (2020); Waltemath and Wolkenhauer (2016); Wilson et al. (2017); Cruz et al. (2018); Leek and Jager (2017); Gomes et al. (2022); Cruz et al. (2018)
S7	Bringing computation to the data when dealing with big data Howe (2012)
S8	Use of Virtual Machine or Container technologies to encapsulate code and complete computational environment *Combine it with Continuous Integration (CI) Krafczyk et al. (2019); Essawy et al. (2020); Choi et al. (2021); Howison and Bullard (2016); Edmunds et al. (2017); Orzechowski et al. (2020); Canon and Younge (2019); Mauerer et al. (2023); Apostal et al. (2018); Fernandez-Prades et al. (2018); Waltemath and Wolkenhauer (2016); Mauerer and Scherzinger (2021); Di Meglio et al. (2012); Dalle (2012); Jansen et al. (2019); Balz and Rocca (2020); Nguyen et al. (2019); Canon (2020); Cruz et al. (2018); Cito and Gall (2016); Kalenkovich and Levchenko (2021); Tatman et al. (2018); Boettiger (2015); Stodden and Miguez (2014); Marwick (2017); Hale et al. (2017); Kim et al. (2018); Krafczyk et al. (2021); Bjorn et al. (2019); Brito et al. (2020); Beaulieu-Jones and Greene (2017); Vaillancourt et al. (2020); Benthall and Seth (2020); Kellogg et al. (2019); Howe (2012); Hinsen (2011); Bilke et al. (2019); Ivie and Thain (2019); Denaxas et al. (2017); Maghami et al. (2023); Choi et al. (2023); Ziemann et al. (2023); Piccolo and Frampton (2016); Bast (2019)
S9	Share clean, readable, well-organised, and documented code developed using version control system Niso et al. (2022); Spencer Smith et al. (2016); Orozco et al. (2020); Bilke et al. (2019); Fernandez-Prades et al. (2018); Mccormick et al. (2014); LeVeque (2009); Rozier and Rozier (2014); Peer et al. (2021); Wilson et al. (2014); Perkel (2020); Kim et al. (2018); Jiménez et al. (2017); Ram (2013); Gille et al.; McFee et al. (2018); Ibanez et al. (2018); Crick et al. (2017); Botvinik-Nezer and Wager (2023); Wagner et al. (2024); Ivimey-Cook et al. (2023); Ziemann et al. (2023)
S10	Provide training such as software carpentry to produce better quality code Denaxas et al. (2017); Gille et al.; Orchard and Rice (2014); Ram et al. (2019); Crick et al. (2017)

Table 6 Continued

ID	Prevention Strategies
S11	Use of code hosting services with integrated version control to publish code, associated workflows and link it to publication Bilke et al. (2019); Denaxas et al. (2017); Dalle (2012); Skaggs et al. (2015); Isdahl and Gundersen (2019); Mcdougal et al. (2016); LeVeque (2009); Rozier and Rozier (2014); Tatman et al. (2018); Rokem et al. (2017); Fehr et al. (2016); Marwick (2017); Kim et al. (2018); Jiménez et al. (2017); Mccormick et al. (2014); Krafczyk et al. (2021); Barba (2019); Crick et al. (2017); Morin et al. (2012); Kellogg et al. (2019); Gomes et al. (2022); Stodden et al. (2016); Crook et al. (2013); Peng (2011); Stodden (2010); Laine et al. (2007); Goble et al. (2013); Ihle et al. (2017); McFee et al. (2018); Frery et al. (2020); Piccolo and Frampton (2016); Kanwal et al. (2017); Lowndes et al. (2017); Waltemath and Wolkenhauer (2016); Mauerer and Scherzinger (2022); Millman and Pérez (2018); Orozco et al. (2020); Ziemann et al. (2023); Lee et al. (2021); Widder et al. (2019); Stodden and Miguez (2014); Wilson et al. (2014); Koehler Leman et al. (2020); Marwick (2017); Levet et al. (2021); Ibanez et al. (2018); Beaulieu-Jones and Greene (2017)
S12	Comprehensive testing (unit and integration) incorporating rigorous code reviews, automated testing, and continuous integration practices Robinson et al. (2021); Gil et al. (2016); Bilke et al. (2019); Pimentel et al. (2019); Fernandez-Prades et al. (2018); Cruz et al. (2018); Bast (2019); Alarid-Escudero et al. (2019); Tatman et al. (2018); Peer et al. (2021); Rokem et al. (2017); Fehr et al. (2016); Wilson et al. (2014); Koehler Leman et al. (2020); Perkel (2020); Scheliga et al. (2019); Millman and Pérez (2018); Kim et al. (2018); McFee et al. (2018); Lowndes et al. (2017); Ibanez et al. (2018); Beaulieu-Jones and Greene (2017); Mccormick et al. (2014); Nüst and Eglen (2021); Gomes et al. (2022); Bahaidarah et al. (2022); Ivimey-Cook et al. (2023)
S13	Use of DSA (Discrete Stochastic Arithmetic) to estimate rounding error propagation in simulation programs Jézéquel et al. (2015)
S14	Thoroughly document research methods, including data collection, processing and analysis from the start of a project and link them to the fragments of the code implementing the method Brinckman et al. (2019); Raghupathi et al. (2022); Stevens (2017); Hinsen (2011); Essawy et al. (2020); Orozco et al. (2020); Piccolo and Frampton (2016); Kedron et al. (2021); Waltemath and Wolkenhauer (2016); Vitek and Kalibera (2011); Irving (2016); Smith et al. (2016); Koehler Leman et al. (2020); Dylan Chapp (2020); Scheliga et al. (2019); Krafczyk et al. (2021); Chen et al. (2019); Yu et al. (2016); Ihle et al. (2017); Gomes et al. (2022); Botvinik-Nezer and Wager (2023); Zhu et al. (2023); Ivimey-Cook et al. (2023); Ziemann et al. (2023); Hutton and Henderson (2018); Raff and Farris (2023)
S15	Document computational environment as Read me— should indicate the operating system(s), software dependencies(packages, libraries, versions) and hardware used Piccolo and Frampton (2016); Chen et al. (2019); Tatman et al. (2018); Brito et al. (2020); Lee et al. (2021); Fehr et al. (2016); Dylan Chapp (2020); Tierney and Ram (2021); Santana-Perez and Pérez-Hernández (2015); Kim et al. (2018); Popp and Biskup (2022); Mendez et al. (2020); Zhu et al. (2023); Ivimey-Cook et al. (2023); Ziemann et al. (2023); Bast (2019)
S16	Store documentation in open trusted repositories with version control and provide separate link for documentation in publication McFee et al. (2018); Ibanez et al. (2018); Essawy et al. (2020); Wattanakriengkrai et al. (2022); Orozco et al. (2020); Piccolo and Frampton (2016); Knoll and Heedt (2020); Taylor et al. (2016); Bánáti et al. (2015); Stodden et al. (2016); Rokem et al. (2017); Dylan Chapp (2020); Ram (2013); Chen et al. (2019)
S17	Use of Literate programming notebooks for documenting medium-sized analysis Piccolo and Frampton (2016); Pimentel et al. (2019); Casseau et al. (2021); Fiore et al. (2018); Peng et al. (2006); Leek and Jager (2017); Kalenkovich and Levchenko (2021); Pernet and Poline (2015); Baiocchi (2007); Blinov et al. (2021); Isdahl and Gundersen (2019); Balz and Rocca (2020); Kluyver et al. (2016); Maghami et al. (2023); Ivimey-Cook et al. (2023); Ziemann et al. (2023)

packaging mechanism which combines data, software, text, and a user interface description" (Nüst et al., 2017).

Prickly Pear Archive (PPA) is a portable hypermedia for scholarly publications hosted on HPC resources. PPA allow researchers to test simulations using the production archive, and results can be incorporated into an executable paper (Bentley et al., 2019).

WAVELAB facilitate reproducibility by packaging all available code and figures with published wavelet articles (Buckheit and Donoho, 1995).

Whole Tale The Whole Tale presents an environment that facilitates the linkage of data and code with publications by strengthening the three layers of scholarly publication, i.e., scholarly process, data, and computational analysis. The Whole Tale architecture consists of micro-services that provide access to large amounts of data, create persistent identifiers, a containerised platform to conduct analysis and the ability to publish the entire methodology systematically and conveniently. Thus enabling readers of the manuscript to

Table 6 Continued

ID	Prevention Strategies
S18	Use modern programming languages and tools to generate documentation from code comments Doxygen and Sphinx, MKDocs Niso et al. (2022); McFee et al. (2018); Millman and Pérez (2018)
S19	A systematic and detailed approach for describing software and code reproducibility in research article Gil et al. (2016); Vitek and Kalibera (2011); Robles (2010); Rozier and Rozier (2014); Widder et al. (2019); Levet et al. (2021)
S20	Document complete testing environment Spencer Smith et al. (2016); Fehr et al. (2016)
S21	Define system scope and requirements Kanwal et al. (2017); Levet et al. (2021); Widder et al. (2019)
S22	Provide training and resources on reproducibility skills for developers and reviewers Chue Hong (2018); Kedron et al. (2021); Mauerer et al. (2023); Waltemath and Wolkenhauer (2016); LeVeque (2009); Cruz et al. (2018); Bast (2019); Scheliga et al. (2019); Kim et al. (2018); Parashar (2020); Gille et al.; Brito et al. (2020); Stodden et al. (2013); Benthall and Seth (2020); Kellogg et al. (2019); Crouch et al. (2013); Raff and Farris (2023)
S23	Create incentives and recognition systems as through citations, awards or promotions Brinckman et al. (2019); Eckersley et al. (2003); Chue Hong (2018); Bajpai et al. (2017); Feger et al. (2020); Marek et al. (2018); Di Meglio et al. (2012); Irving (2016); Milham and Klein (2019); Koehler Leman et al. (2020); von Hahn and Mechefske (2022); Scheliga et al. (2019); Leek and Jager (2017); Barba (2019); Crick et al. (2017); Stodden et al. (2013); Benthall and Seth (2020); Kellogg et al. (2019); Stodden et al. (2016); Raff and Farris (2023); Howison and Bullard (2016)
S24	Foster a culture of transparency and sharing while preserving the benefits of authors Garcia-Silva et al. (2019); Shamir et al. (2013); Bajpai et al. (2017); Harrell et al. (2022); Dalle (2012); Balz and Rocca (2020); Ram et al. (2019); Bast (2019); Lee et al. (2021); Stodden and Miguez (2014); Ibanez et al. (2018); Stodden et al. (2013); Kellogg et al. (2019); Gomes et al. (2022); Stodden (2009); Scheliga et al. (2019)
S25	Promote effective communication and collaboration within research teams and across disciplines Robinson et al. (2021); Huppmann et al. (2019); Anzt et al. (2019); Bilke et al. (2019); Pernet and Poline (2015); Frery et al. (2020); Wu et al. (2011); Cruz et al. (2018); Koehler Leman et al. (2020); Hutton and Henderson (2018); Kellogg et al. (2019)
S26	Allocate sufficient resources (time, human and funding) to support reproducibility efforts Brinckman et al. (2019); Edmunds et al. (2017); Bontemps and Orozco (2021); Milham and Klein (2019); Barba (2019); Stodden et al. (2013); Ihle et al. (2017); Laine et al. (2007); Botvinik-Nezer and Wager (2023)
S27	Develop and adopt community standards for reproducibility Orozco et al. (2020); Poldrack et al. (2019); Waltemath and Wolkenhauer (2016); Mcdougal et al. (2016); Tatman et al. (2018); Widder et al. (2019); Botvinik-Nezer and Wager (2023)
S28	Obtain and distribute appropriate open source licences (permissive or copyleft) to allow re-use of software and data Morrison (2018); Gil et al. (2016); Skaggs et al. (2015); Pernet and Poline (2015); Eckersley et al. (2003); Frery et al. (2020); Mcdougal et al. (2016); Tatman et al. (2018); Rokem et al. (2017); Koehler Leman et al. (2020); Stodden (2009); Scheliga et al. (2019); Levet et al. (2021); Jiménez et al. (2017); McFee et al. (2018); Barba (2019); Brito et al. (2020); Gomes et al. (2022); Stodden et al. (2016); Stodden (2010); Morin et al. (2012); Bast (2019); Tatman et al. (2018); Lee et al. (2021); Stodden and Miguez (2014); Tierney and Ram (2021); Yu et al. (2016); Peng (2011)
S29	Use of templates programs/simulated dataset/data anonymisation for private-sensitive data Ivie and Thain (2019); Kedron et al. (2021); Pröell et al. (2015); Rozier and Rozier (2014); Engel et al. (2017); Dorodchi et al. (2019); Krafczyk et al. (2021); Eckersley et al. (2003); Hutton and Henderson (2018); Ivimey-Cook et al. (2023)

view the entire method and reproduce the analysis (Brinckman et al., 2019).

N3phele is a cloud-based workbench accessed via a browser that allows researchers to perform complex analysis using only browser and cloud resources. It is a programming language and platform-independent framework, thus allowing developers to work independently and contribute to processing content. N3phele packages the scientific code by decoupling the desktop interface and orchestration environment from the cloud computing and storage resources. This reduces environmental dependencies in the analytic software, thus preventing RpD (Cook et al., 2012).

ReproZip enables researchers to make their experiments reproducible without making substantial efforts. Installing

Table 7
Specialised Tools and Infrastructure to Prevent RpD

Tool or Technology	Primary Study
Research Objects	Garcia-Silva et al. (2019); Nüst et al. (2017); McPhillips et al. (2019); Morrison (2018)
Experiment Digital Object	Orzechowski et al. (2020)
Research Compendium	Gentleman and Lang (2007); Nüst et al. (2017)
ReDoc	Schwab et al. (2000)
Prickly Pear Archive	Bentley et al. (2019)
WAVELAB	Buckheit and Donoho (1995)
Whole Tale	Brinckman et al. (2019)
N3phele	Cook et al. (2012)
ReproZip	Nüst et al. (2017); Pimentel et al. (2019)
ScienceCapsule	Ghoshal et al. (2021)
SemanticSCo Web	da Silva and Guareis de Farias (2019)
iEnviroment	Alencar et al. (2018)
PRISONER	Hutton and Henderson (2018)
Climate Analytics Hub	Fiore et al. (2018)
PopperCI	Jimenez et al. (2017a,b)
CodeOcean	Clyburne-Sherin et al. (2019)
Galaxy Framework	Föll et al. (2019); Leek and Jager (2017); Goecks et al. (2010); Piccolo and Frampton (2016)
AlgoRun	Hosny et al. (2016)
Invariant Framework	Meng et al. (2015)
Rang	Chan and Schoch (2023)
BIDS App	Niso et al. (2022); Garrett-Ruffin et al. (2021); Gorgolewski et al. (2017)
RE3	Bahaidarah et al. (2022)
Reproducible Experiment Descriptions	Jansen et al. (2019)
SwarmRob	Pörtner et al. (2018)
SciUnit	Essawy et al. (2020)
CERN Framework	Blomer et al. (2015); Chen et al. (2019)
Maneage	Akhlaghi et al. (2021)
Osiris	Wang et al. (2020b)
PyDFix	Mukherjee et al. (2021)
Automatic Control Knowledge Repository	Knoll and Heedt (2020)
Computer-Aided Reproducibility	Marek et al. (2018)
PyPHM	von Hahn and Mechefske (2022)
Omero Processing Extension	Taubert and Bücker (2017)
DataONE Data Package	Mecum et al. (2018)
ASpecD	Popp and Biskup (2022)
GigaDB	Edmunds et al. (2017)
CoMSES Net	Rollins et al. (2014)
OntoSoft	Essawy et al. (2020); Gil et al. (2016); Gil et al. (2015); Essawy et al. (2017)
Blockchain and Distributed Ledger Technologies	Wittek et al. (2021); Bell et al. (2017)

the Reprozip package can automatically capture the provenance of an entire experiment, including data dependencies, libraries, and configuration parameters, by tracking system calls. The captured information is combined into a lightweight, reproducible package that can be easily shared and reproduced in any computational environment (Nüst et al., 2017; Pimentel et al., 2019).

ScienceCapsule is a framework that supports reproducibility by automatically capturing and sharing scientific workflows. It captures and manages both computational and human elements of workflows. Moreover, it allows users to keep track of different workflow versions by creating workflow snapshots (Ghoshal et al., 2021).

SemanticSCo Web is a web-based, reproducibility-supported version of the SemanticSCo platform used for gene expression data analysis. The component Composition Manager of SemanticSCo Web records all information related to an analysis workflow in a relational database. This includes all analysis activities, input and output data and service execution parameters, thus allowing any developed workflow to be reproduced in a similar order in which all activities were designed and executed previously da Silva and Guareis de Farias (2019).

iEnviroment is a platform that facilitates access to open data resources and research collaboration. The integrated services enable secure storage, publication, discovery, reuse

and verification of scientific research data. It supports reproducibility by partially capturing the process (e.g., metadata, models, tools) that generates the research data (Alencar et al., 2018).

PRISONER is a framework which aims to support the execution of reproducible and privacy-respecting experiments using social media network data. It encapsulates underlying methodology details, such as data collection, processing, and participant consent, as a workflow which can be shared and reproduced (Hutton and Henderson, 2018).

Climate Analytics-Hub provides an open science environment for reproducible multi-model climate change data analytics experiments by combining big data approaches and parallel computing paradigms. However, it requires human intervention through the Ophidia analytics document (Fiore et al., 2013) to describe any missing information regarding the computational environment (Fiore et al., 2018).

PopperCI allows researchers to automate end-to-end execution and validation of their experiments. It is based on Popper, which is a manual DevOps approach for systematically implementing different stages of scientific workflow (Jimenez et al., 2017a,b).

Code Ocean is a platform that facilitates researchers to package code, data, results, metadata, and a computational environment into a single compendium called 'compute capsule' (Clyburne-Sherin et al., 2019).

Galaxy Framework is a web-based analysis platform for complex, interconnected data-driven experiments, providing users access to large-scale computational resources. It supports provenance tracking and version control, including the ability to switch between software and tool versions and to publish complete analysis, thus enabling reproducibility (Föll et al., 2019; Leek and Jager, 2017; Goecks et al., 2010; Piccolo and Frampton, 2016).

AlgoRun is a packaging system for implemented algorithms based on a Docker container technology. The packaged algorithms can be accessed and executed directly from a web browser, thus facilitating reproducibility (Hosny et al., 2016).

Invariant framework enables reproducibility by capturing and preserving the dependencies and configurations used by the program, including hardware, OS, and other static, dynamic, local and networked dependencies, source codes and data files. The resulting package is then stored and distributed through an open-source, public repository (Meng et al., 2015).

Rang is a reproducible research compendium based on Docker that automatically generates declarative descriptions of the computational environment to support reproducibility (Chan and Schoch, 2023).

BIDS Apps The Brain Imaging Data Structure (BIDS) is a community standard for organising, describing, and sharing neuroimaging research data. BIDS apps use BIDS standard and singularity containers to ensure the reproducibility of neuroimaging data analysis (Niso et al., 2022; Garrett-Ruffin et al., 2021; Gorgolewski et al., 2017).

RE3 is an open-source platform that uses the ML model (trained on a code readability survey) to assess the readability of R code. Reproducibility is then checked by using a Docker container that automatically bundles and executes the software, libraries, and configuration files together (Bahaidarah et al., 2022).

Reproducible Experiment Descriptions (RED) are JSON or YAML-based file formats for reproducible experiment descriptions. The programs are packaged and executed as Docker Image (Jansen et al., 2019).

SwarmRob is an orchestration solution to manage RpD in robotics research, using container images and services (Pörtner et al., 2018).

SciUnit is a lightweight solution for containerisation; it automatically traces and encapsulates dependencies (Essawy et al., 2020).

CERN analysis and preservation framework [15], [16] enable researchers to automatically record and preserve the entire workflow components, i.e., data, software, computing environment and associated documentation (Chen et al., 2019). GROW-FS and CERN-VM-FS are file systems developed for distributing complex software stacks across hundreds and thousands of machines (Blomer et al., 2015).

Maneage is designed to manage data lineage. The solution is proposed with the concept of a robust data management strategy from the start of a project to deal with the longevity issues in existing tools such as docker and Jupyter notebooks (Akhlaghi et al., 2021).

Osiris an automated approach to make Jupyter notebooks reproducible. Osiris take the notebook as input and reconstructs all possible execution orders that reproduce the exact notebook results without errors. On unsuccessful reproduction, it highlights the location of failure for understanding the root causes of non-reproducibility of Jupyter notebooks (Wang et al., 2020b).

PyDFix detects, and fixes RpD caused by dependency errors in Python builds (Mukherjee et al., 2021).

Automatic Control Knowledge Repository is more than a simple storage repository. It combines existing SE technologies, i.e., Distributed Version Control Systems and Automated Tests/Continuous Integration Services to prevent RpD (Knoll and Heedt, 2020).

Computer-Aided Reproducibility (CAR) is a tool that facilitates researchers in sharing their research data and results by partially automating the process of sharing. This prevents RpD by reducing the time and effort required for the manual data-sharing process (Marek et al., 2018).

PyPHM is a domain-specific tool designed to support Prognostics and Health Management researchers in conducting reproducible computational research (von Hahn and Mechefske, 2022).

Omero Processing Extension (OPE) prevent RpD by establishing a link between data processing and storage. It automatically generates a description of workflow execution

[15] https://github.com/cernanalysispreservation
[16] https://reanahub.io/

and annotates the results. A web-based interface allows easy access and usage (Taubert and Bücker, 2017).

DataONE Data Package DataOne [17] is a group of interoperable repositories that facilitate scientific data sharing and discovery. It links existing cyberinfrastructure to provide a distributed framework for long-term data preservation. Using the DataONE data package standard may prevent RpD by incorporating provenance information in datasets as part of the enclosing data package. Moreover, software and code can be included as a part of the data package using container technology to ensure a reproducible package (Mecum et al., 2018).

ASpecD is a modular framework written in Python programming language. It supports the analysis of complex spectroscopic data to allow reproducibility (Popp and Biskup, 2022). Similarly, *GigaDB* is an integrated database linked with the GigaScience journal. It hosts large-scale biological data and provides analysis tools and computing resources to support executable and reproducible publications (Edmunds et al., 2017).

CoMSES Net is a Computational Model Library where researchers can submit their model code and documentation for the certification process. The certified submissions are then assigned with persistent unique identifiers for citation (Rollins et al., 2014).

OntoSoft is a software registry that automatically captures the metadata of scientific software once linked to a code repository such as Git (Essawy et al., 2020; Gil et al., 2016; Gil et al., 2015; Essawy et al., 2017).

Blockchain and Distributed Ledger technologies can be used to ensure end-to-end integrity of scientific workflows (Wittek et al., 2021; Bell et al., 2017).

Mathematical Approach Bánáti et al. (2016) classified scientific workflows into four main categories based on the reproducibility cost of workflows. They employed decay parameters ("the type and the measure of the change of the given value") in scientific workflows to calculate the cost of reproducibility of a given workflow— an essential element of TD landscape.

5. Discussion and Implications

This SLR aimed to investigate the issues contributing towards RpD and the solution/guidelines to mitigate those issues, as evidenced by selected primary studies. For this, 214 primary studies were analysed to present a taxonomy of RpD items, i.e., a list of causes contributing towards the emergence and identification of RpD and a list of strategies, activities, and tools to prevent RpD in scientific software.

Although various studies exist on different types of TD Melo et al. (2022); Alves et al. (2016), reproducibility has yet to be mentioned as a type of TD. Therefore, we organised our emerged results with already established concepts of TD presented by Izurieta et al. (2016); Rios et al. (2018); Avgeriou et al. (2016) to characterise reproducibility as a type of TD as shown in Figure 6. Here, **Debt** may be a technical issue, developer challenge or sub-optimal activity having short-term benefits. **Debt items** may belong to any category, i.e., data-centric, code-centric, documentation-centric, human-centric, or tool-centric. **Interest** is the extra cost that needs to be paid because of Debt. **Principal** is the cost of developing or adopting a solution to prevent or remove RpD. **Indicators** are observable signs ('reproducibility smells') through which RpD becomes evident in scientific software projects. It represents the specific challenges, issues, or outcomes that result from factors contributing to RpD. From the obtained list of causes and effects, we hypothesised a few reproducibility smells shown in Figure 7, which we will validate as part of future work of this SLR in real scientific software projects.

As we collected primary studies from various scientific disciplines; we mapped all emerged issues under each category to the scientific domains presented in Figure 6. The mapping results presented in Table 8 give us an overview of problems specific to each domain, enabling their systematic identification and prevention. From this map, we identified some issues common to all scientific disciplines i.e., RpD1.4: *lack of an integrated and trusted infrastructure for storing, processing and distributing a growing volume of data*; RpD2.1: *complex algorithms and code* which results in *limited re-usability of code and inability to track code versions* and RpD4.1: missing or updated dependencies for complex software systems. Also, RpD1.1: *incomplete and selective reporting of datasets* and RpD5.1: *version upgradation in programming languages, libraries, operating systems, and other dependencies* is a common issue among nine disciplines. Moreover, all scientific domains show an agreement on RpD-6.2, 6.3 and 6.4 related to the *lack of training, guidelines and incentives for reproducible research*, which calls for future works in terms of training initiatives, formal guidelines and incentive schemes for reproducibility-oriented research.

From the cross-analysis of the pattern, we also observed that Information and Computing Sciences, Biomedical and Clinical Sciences, Economics, Mathematics and Biological Sciences have highlighted similar issues. Moreover, seven scientific disciplines, except Engineering and Earth Science, highlight legal issues related to open-source software. Similarly, all scientific disciplines highlight issues related to the tools and infrastructure, except psychology, which has only one occurrence in the category, i.e., RpD4.1: missing dependencies.

The mapping given in Table 8 also shows that RpD2.5, i.e., *lack of comprehensive testing (unit, integration, and regression tests)* is only highlighted by Information and Computing Science papers, which is an area of concern. In our view, lack of testing is a prominent factor contributing to RpD. However, it is not acknowledged by other scientific disciplines. There could be two possible reasons for this; either researchers from other domains do not know about the importance of testing for ensuring reproducibility, or they know about it but do not acknowledge it.

[17] https://www.dataone.org/

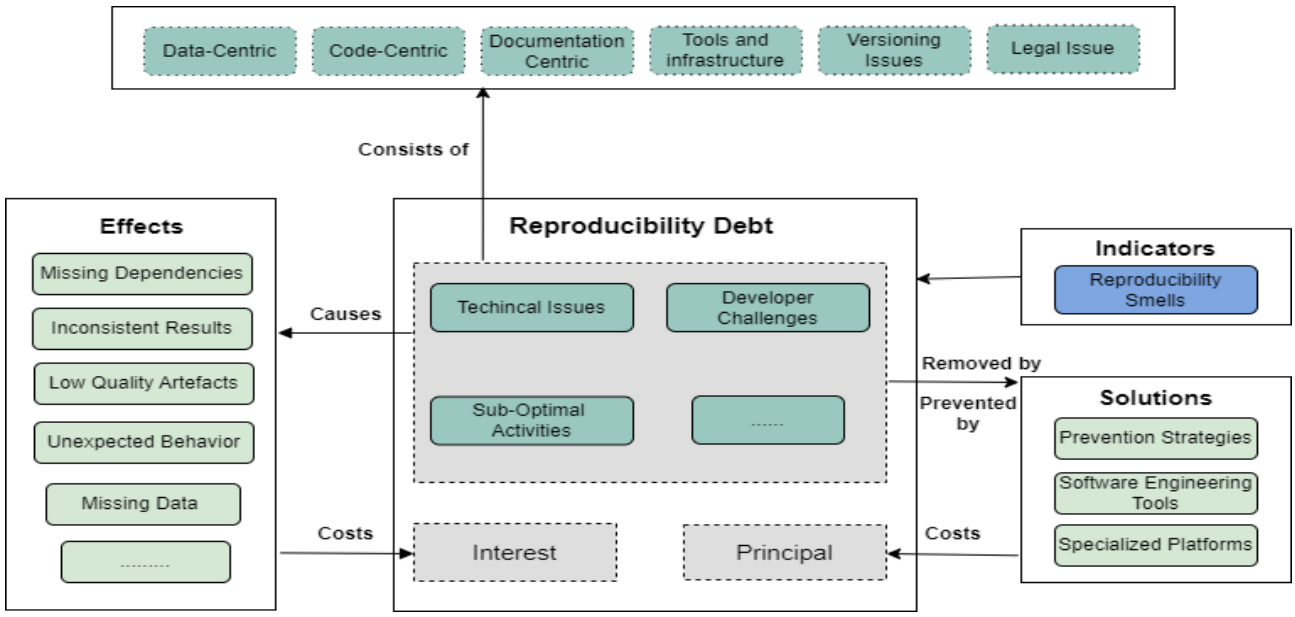

Figure 6: Reproducibility Debt and related concepts: **Debt** may be a technical issue, developer challenge or sub-optimal activity having short-term benefits. **Debt items** may belong to any category and have some effects. **Interest** is the extra cost that needs to be paid because of debt. **Principal** is the cost of developing or adopting a solution that can prevent or remove RpD.

Overall, the mapping presented in Table 8 indicates that the factors leading to RpD are distributed quite evenly across scientific domains, with only a few exceptions that have been discussed above. This suggests that each discipline faces similar or closely related issues and challenges that contribute to RpD and that a standardised solution could be implemented to manage it proactively. Our results also show a balance of efforts in highlighting the issues contributing towards RpD and proposing solutions (tools and practices) to mitigate those issues, as demonstrated by selected primary studies. However, there is a lack of empirical evidence on the state of adoption of those tools and suggested practices. This lack of evidence creates a gap between industry and academia on which tools and practices can be employed more effectively to prevent RpD.

In addition to all the above, we established a relationship between identified RpD items and existing TD types presented in Table 9. We concurrently analysed the RpD *causes and effects* that appeared under each category (as given in Table 5 and mapped them to the list of situations (smells) discussed by Alves et al. (2016); Rios et al. (2018); Sculley et al. (2015) where these existing types of debts can occur in software systems. For example, the code-centric issues contributing towards RpD are related to Code Debt (Unnecessary code duplication and complexity; Bad style that reduces the readability of code), Versioning Debt (Unnecessary code forks), Infrastructure Debt (Outdated components of an application's development environment), Build Debt (build process needs to run ill-defined dependencies) and Test Debt (Lack of tests e.g., unit tests, integration tests, and acceptance tests). Documentation issues are related to Documentation, Code and Data Debt (missing, incomplete or outdated documentation of scientific code and data; insufficient code comments).

Figure 7: Reproducibility smells hypothesised from the list of RpD causes and effects presented in Table 5

5.1. Implications

Although the primary goal of this SLR was to accumulate existing knowledge on reproducibility issues and proposed solutions, the answers to the RQs have implications

Table 8
Categories of RpD items exclusive to scientific software and their relation with existing TD types: Information and Computing (IC), Biomedical and Clinical (BC), Biological Science (BS), Physical Science (PS), Earth Science (ES), Economics and Mathematics (EM), Environmental Science (ES), Psychology (P), Engineering (En), Generic (Ge)

RpD Items	IC	BC	BS	PS	ES	EM	ES	P	En	Ge
Data-Centric										
RpD1.1	X	X	X		X	X	X	X	X	X
RpD1.2	X		X		X	X	X			X
RpD1.3	X	X	X		X	X	X			
RpD1.4	X	X	X	X	X	X	X	X	X	X
RpD1.5	X	X	X	X	X			X		X
Code-Centric										
RpD2.1	X		X	X	X	X	X	X	X	X
RpD2.2	X	X	X					X		
RpD2.3	X	X		X					X	
RpD2.4	X	X	X	X		X		X		X
RpD2.5	X									
Documentation										
RpD3.1	X	X	X	X	X	X	X			X
RpD3.2	X	X	X	X			X		X	X
RpD3.3	X	X	X		X	X			X	X
RpD3.4	X	X		X		X		X		
RpD3.5		X	X		X	X				X
RpD3.6	X	X	X		X	X	X			X
Infrastructure										
RpD4.1	X	X	X	X	X	X	X	X	X	X
RpD4.2	X	X	X	X	X		X		X	X
RpD4.3	X	X	X	X	X	X	X		X	X
RpD4.4			X	X		X	X			X
RpD4.5		X	X						X	X
RpD4.6	X	X	X	X	X	X			X	
RpD4.7	X									X
RpD4.8	X	X	X	X		X			X	X
Versioning										
RpD5.1	X	X	X	X	X	X		X	X	X
RpD5.2	X									
Human										
RpD6.1	X	X	X		X	X	X			
RpD6.2	X	X	X	X	X	X	X	X	X	X
RpD6.3	X	X	X	X	X	X		X	X	X
RpD6.4	X	X	X	X	X	X	X	X		X
RpD6.5	X	X	X	X		X	X	X	X	
RpD6.6	X			X	X	X				
RpD6.7			X	X	X					
RpD6.8	X	X			X	X				X
Legal Issues										
RpD7.1	X	X	X	X	X	X		X		X
RpD7.2	X	X	X	X	X	X		X		
RpD7.3	X	X	X	X		X				

for a broad range of audiences, including researchers[18], practitioners[19], policymakers, funding agencies, tool builders, educators, and editors/reviewers as discussed below:

1. The presented taxonomy of issues (RQ1) is the organised and structured overview of all major groups of problems attributed towards the emergence of RpD. It will give ***practitioners*** a quick understanding of RpD and the types of RpD items that could be present in their software.

[18]Individuals who aim to contribute to the area of RpD.

[19]Individuals or professionals involved in the scientific software development process such as domain researchers developing software for computing their results or software developers working for scientific research teams.

2. The 37 causes of RpD presented in this study would help ***practitioners*** to identify RpD in their scientific software projects. Additionally, the list of effects alongside each cause would guide prioritising each debt item for prevention or payment.

3. Out of 37 causes attributed to the emergence of the RpD, eight causes are associated with the people involved in developing and using scientific software, supported by 163 primary studies from all scientific domains. The lack of recognition, reward and incentives for reproducibility-oriented practices, i.e., data and code sharing, exhaustive testing and documentation, is one of the prominent causes mentioned by 56 primary studies. Based on this analysis, we recommend that ***funding agencies***, *research organisations and universities* need to pay more attention towards rewarding and incentive structures for extra reproducibility efforts.

4. Among the recorded causes, missing or updated dependencies for complex software systems are highlighted by 36 primary studies, which is the cause of the emergence of inconsistent results and sometimes even the inability to re-execute programs. Based on this analysis, we recommend that ***scientific software developers*** should pay more attention and effort to specify and document all required dependencies or use container technologies to package the complete software environment.

5. One of the RpD causes, i.e., resistance to adopting reproducibility practices by ***senior researchers***, will help them to self-analyse that although they are successful without adopting reproducibility practices as part of their research, it results in the culture of irreproducible research, as they train the new generation of scientist.

6. A list of 29 prevention strategies would guide ***practitioners*** towards systematically adopting SE tools (e.g., containers, literate programming notebooks and version control) and other reproducibility-oriented practices such as documentation, data sharing, and testing. Also, part of prevention strategies guides ***research organisations and universities*** to take necessary steps towards preventing human-centric issues.

7. We have identified 39 specialised platforms and technologies developed to ensure the reproducibility of scientific workflows presented in Table 7. Out of these 39 tools, six are based on the idea of executable documentation discussed in 10 primary studies, whereas 22 use container technology to encapsulate a complete computational environment discussed in 30 primary studies. Others provide storage support with integrated metadata capture and version control functions. Two primary studies also discuss the use of blockchain technologies to ensure the end-to-end reproducibility of scientific workflows. Thus, a decent number of automated solutions are developed that facilitate collaboration and dependency management to prevent RpD; however, relying on existing solutions is still a challenge because there is a lack of empirical evidence on the adoption, usability and trustworthiness of these technologies and platforms.

Moreover, all identified tools are designed to prevent RpD proactively (Wonsil et al., 2023) i.e., they should be used from the start of the project, which requires background knowledge of reproducibility and comfort with using these tools. The prioritisation of RpD needs specific strategies and guidelines to use these tools. Hence, much effort already put in by ***researchers and tool builders*** across domains can not be materialised to prevent RpD until a detailed exploration of RpD management activities.

8. The results of this SLR also have implications for ***researchers*** who want to investigate RpD across various scientific disciplines and those who want to investigate strategies for its management, i.e., (a) The proposed definition of RpD derived from the taxonomy of RpD items will allow researchers to share a common vocabulary; (b) As a result of this SLR, we formally acknowledge reproducibility as a type of TD. This calls for researchers in the TD area to explore existing TD management strategies in the context of RpD for its effective management; (c) The list of RpD items presented in this study is a starting point to characterise RpD, which requires further exploratory studies to investigate other technical issues, developer challenges and state-of-the-art solutions for RpD prevention and payment; (d) Out of 214 primary studies, only 1 study focused on the cost of reproducibility regarding jobs/tasks, which is another research gap; (e) In total, 56 primary studies highlighted the concern that reproducibility is not rewarded, which ultimately promotes a culture of non-reproducible research. Hence, investigating and analysing the effort and cost to ensure reproducibility is another research area.

9. For ***MSR researchers*** interested in empirical studies on reproducibility, the presented list of RpD items and reproducibility smells would serve as a guide to identify and measure RpD in existing open-source projects.

10. We identified that RpD is a type of debt which affects every aspect of the scientific software development process and is related to all tangible artefacts of the software development process, i.e., data, code and documentation, either directly or indirectly. This results in a large number of debt items present at each stage of the development process. The manual management of a large number of RpD items is, therefore, complex and requires the development of automated tools for RpD management, which is another research area for ***tool builders***.

11. As reproducibility is a newly acknowledged type of TD, our findings in Table 9, which show a relationship between RpD and existing TD types, would guide ***researchers in the TD area*** to explore already established strategies for TD management in relation to RpD. Prior work by Alves et al. (2016); Rios et al. (2018) demonstrates that most research regarding TD identification is towards code-related activities (code, design, and architecture debt), which could be applied to RpD issues related to these types of debt. i.e., code, dependency, build, versioning, and documentation issues.

6. Threats to Validity

We considered several threats that affect the validity of our results. These threats arise from the inherent challenges

Table 9
Categories of RpD items exclusive to scientific software and their relation with existing TD types

RpD Items	Code	Architecture	Documentation	Data	Versioning	Infrastructure	Build	Test	People
Data-Centric			X	X		X			
Code-Centric	X				X	X	X	X	
Tool-Centric		X				X	X		
Version Issues	X			X	X				
Documentation	X		X	X					
Human-Centric									X
Legal Issues									

and limitations of SLR methodology (Kitchenham et al., 2010). This section highlights those threats and the mitigation strategies opted to minimise their impact on the quality of the research process and results.

Descriptive Validity highlights the importance of accurately recording all data used for concluding results (Petersen et al., 2015). Qualitative studies are more susceptible to descriptive validity threats; therefore, to address this potential concern, we approached our study with a commitment to a systematic process and rigorous protocols throughout both the study selection and data extraction phases following the guidelines by Kitchenham and Charters (2007). All authors of the paper have validated our entire selection and data extraction process. For data extraction, all extraction fields were derived based on research objectives, and initially, data was extracted from a small group of control papers. Comprehensive discussions among all authors followed this to ensure that the recorded data and extraction fields aligned cohesively with our research objectives, thereby reinforcing the credibility of our approach. After reaching an agreement on extraction fields, the first author individually extracted the data from all selected papers, which was reviewed and validated by other authors.

Theoretical validity relies on our proficiency in accurately capturing what we intend to capture. Therefore, considerations such as biases and study selection play an essential role. During the study search, one threat is missing relevant literature. We used a systematic approach to formulate and test our search string to mitigate this threat. A group of control papers were obtained and analysed in detail to identify the *keywords* used in the search string. The search string was verified by running a pilot search with a criterion that *(at least five control papers should appear within 1-3 pages from top search results)*. Secondly, to avoid *publisher bias*, we searched the venues recommended by prior works in software engineering (Kitchenham and Charters, 2007) and the results were complemented by adding Google Scholar. Finally, we complemented the search with a forward and backward snowballing, applied to all selected studies after a complete reading of the manuscript (Wohlin, 2014).

Researcher biases may appear during study selection and extraction of data. To mitigate *selection bias*, the study selection process was carried out by two authors, during which the first author independently selected the primary studies and the second reviewed all selected and excluded papers. All disagreements were discussed in weekly meetings keeping in view the IECs and QC. To mitigate researcher bias during *data extraction and analysis*, all the extracted data and qualitative codes were carefully reviewed and discussed by all authors of this paper.

Reproducibility To allow reproducibility, (1) a methodology adopted for this study is carefully reported in the paper; (2) a raw data file is shared along with the Zotero library and qualitative analysis files to keep the transparency of the study selection, data extraction and analysis process; (3) artefacts related to the quantitative analysis are shared along with the calculated graphs and tables; (4) we used Jupyter Notebook, as it supports reproducibility for small to medium-size analysis; (5) we shared all research artefacts on Figshare for their long-term storage and archival, with attached license and citable DOI. All these steps may prevent RpD in our research process, thus allowing others to replicate, analyse and extend our results independently.

7. Conclusions and Future Works

When developers make low-quality technical choices to obtain short-term benefits, Technical Debt (TD) may be unwittingly incorporated, leading to an assortment of future problems. Multiple TD types have been identified, along with their occurrences and corresponding management strategies. Therefore, new types of TD need to be investigated to assist their management and control their impact. In this context, this work characterises a new type of TD, i.e., *Reproducibility Debt* (RpD).

The research focused on identifying the leading issues attributed to the emergence and identification of RpD and a list of solutions presented in the existing literature to mitigate those issues. A systematic literature review was conducted following rigorous protocols and evidence in existing literature was analysed in detail to present the results. In total, 214 primary studies highlighting reproducibility issues and their solutions in the context of scientific software were included after searching from renowned digital libraries. A three-stage filtering process was conducted using predefined protocols to select the studies published till January 2024.

The results of this systematic literature review show diverse issues contributing towards the emergence of RpD, involving issues related to scientific software code, data, documentation, issues related to the tools and technologies

used to develop scientific software, issues related to the people involved in scientific software development process and legal constraints. The taxonomy of issues led to the definition of RpD as "Reproducibility debt is a type of technical debt primarily impacting scientific software. It refers to the accumulative issues and challenges that hinder the ability to reproduce scientific outcomes, stemming from challenges inherent in scientific software code and data, including development, organisation, dissemination, and documentation."

Another contribution of this systematic review is a synthesised and comprehensive list of causes attributed towards the emergence of RpD and their corresponding effects. Moreover, a list of prevention strategies and existing platforms is presented, having implications for professionals involved in the scientific software development process. A consolidated list of implications is also presented for researchers who want to investigate RpD and its management strategies.

To conclude, the results provide considerable support for the objective of this work. The main contributions of this work are (1) a formal definition of RpD; (2) a taxonomy of issues contributing towards RpD; (3) a list of causes and effects having implications for software professionals to identify and measure RpD in their projects; (4) a list of strategies and tools to prevent or remove RpD; (5) the identification of gaps in existing research to guide future studies.

In future, we will continue our work with the gaps identified in this study. Also, the evidence from this systematic review will be triangulated with new empirical studies. Currently, our team is designing a Survey based on the InsightTD questionnaire to investigate reproducibility issues and challenges faced by scientific software developers in real scenarios. The survey investigations would be followed by interview-based case studies to identify and measure RpD in scientific software projects across different research organisations. This would result in a theoretical framework of RpD to serve as a guide for its effective management, i.e., identification, measurement, prioritisation, monitoring and payment.

References

Abubakar, H., Obaidat, M.S., Gupta, A., Bhattacharya, P., Tanwar, S., 2020. Interplay of machine learning and software engineering for quality estimations, in: International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), IEEE, USA. pp. 1–6.

Akhlaghi, M., Infante-Sainz, R., Roukema, B.F., Khellat, M., Valls-Gabaud, D., Baena-Galle, R., 2021. Toward long-term and archivable reproducibility. Computing in Science & Engineering 23, 82–91. doi:10.1109/mcse.2021.3072860.

Alarid-Escudero, F., Krijkamp, E.M., Pechlivanoglou, P., Jalal, H., Kao, S.Y.Z., Yang, A., Enns, E.A., 2019. A need for change! a coding framework for improving transparency in decision modeling. Pharmacoeconomics 37, 1329–1339.

Alencar, P., Cowan, D., Mulholland, D., 2018. The ienvironment platform: Developing an open science software platform for integrated environmental monitoring and modeling of surface water, in: 2018 IEEE International Conference on Big Data (Big Data), pp. 3201–3210. doi:10.1109/BigData.2018.8622373.

Alves, N.S., Mendes, T.S., de Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C., 2016. Identification and Management of Technical Debt: A Systematic Mapping Study. Information and Software Technology 70, 100–121.

Anchundia, C.E., Fonseca C., E.R., 2020. Resources for reproducibility of experiments in empirical software engineering: Topics derived from a secondary study. IEEE Access 8, 8992–9004. doi:10.1109/ACCESS.2020.2964587.

ANZSRC, . URL: https://www.abs.gov.au/statistics/classifications/australian-and-new-zealand-standard-research-classification-anzsrc/latest-release.

Anzt, H., Cojean, T., Kühn, E., 2019. Towards a new peer review concept for scientific computing ensuring technical quality, software sustainability, and result reproducibility. PAMM 19. doi:10.1002/pamm.201900490.

Apostal, S.F.J., Apostal, D., Marsh, R., 2018. Containers and reproducibility in scientific research, in: 2018 IEEE International Conference on Electro/Information Technology (EIT), pp. 0525–0530. doi:10.1109/EIT.2018.8500088.

Apptainer, . https://apptainer.org/.

ARC, . https://www.arc.gov.au/policies-strategies/policy/arc-open-access-policy.

ARDC, . https://ardc.edu.au/resources/working-with-research-software/.

Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C., 2016. Managing technical debt in software engineering (dagstuhl seminar 16162), in: Dagstuhl reports, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Bahaidarah, L., Hung, E., Oliveira, A.D.M., Penumaka, J., Rosario, L., Trisovic, A., 2022. Toward reusable science with readable code and reproducibility, in: 2022 IEEE 18th International Conference on e-Science (e-Science), IEEE Computer Society, Los Alamitos, CA, USA. pp. 437–439. doi:10.1109/eScience55777.2022.00079.

Baiocchi, G., 2007. Reproducible research in computational economics: guidelines, integrated approaches, and open source software. Computational Economics 30, 19–40. doi:10.1007/s10614-007-9084-4.

Bajpai, V., Kühlewind, M., Ott, J., Schönwälder, J., Sperotto, A., Trammell, B., 2017. Challenges with reproducibility, in: Proceedings of the Reproducibility Workshop, pp. 1–4.

Baldassari, B., 2013. Squore: a new approach to software project assessment, in: International Conference on Software & Systems Engineering and their Applications, ACM, New York, NY, USA.

Balz, T., Rocca, F., 2020. Reproducibility and replicability in sar remote sensing. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13, 3834–3843. doi:10.1109/JSTARS.2020.3005912.

Barba, L.A., 2019. Praxis of reproducible computational science. Computing in Science & Engineering 21, 73–78.

Bast, R., 2019. A fairer future. Nature Physics 15, 728–730.

Bavota, G., Russo, B., 2016. A Large-Scale Empirical Study on Self-Admitted Technical Debt, in: 13th International Conference on Mining Software Repositories, ACM, USA. p. 315–326. doi:10.1145/2901739.2901742.

Beaulieu-Jones, B.K., Greene, C.S., 2017. Reproducibility of computational workflows is automated using continuous analysis. Nature Biotechnology 35, 342–346.

Bell, J., LaToza, T.D., Baldmitsi, F., Stavrou, A., 2017. Advancing open science with version control and blockchains, in: 2017 IEEE/ACM 12th International Workshop on Software Engineering for Science (SE4Science), pp. 13–14. doi:10.1109/SE4Science.2017.11.

Benthall, S., Seth, M., 2020. Software engineering as research method: Aligning roles in econ-ark, in: Proceedings of the Python in Science Conference, Proceedings of the Python in Science Conference. doi:10.25080/majora-342d178e-015.

Bentley, M., Briggs, I., Gopalakrishnan, G., Ahn, D.H., Laguna, I., Lee, G.L., Jones, H.E., 2019. Multi-level analysis of compiler-induced variability and performance tradeoffs, in: Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing, Association for Computing Machinery, New York, NY,

USA. p. 61–72. doi:10.1145/3307681.3325960.

Bilke, L., Flemisch, B., Kalbacher, T., Kolditz, O., Helmig, R., Nagel, T., 2019. Development of open-source porous media simulators: Principles and experiences. Transport in Porous Media 130, 337–361. doi:10.1007/s11242-019-01310-1.

Bjorn, G., Olivier, S., Moreno, P., Hervé, M., Søndergaard, D., Hannes, R., Timo, S., O'Connor, B., Fábio, M., Victoria, D.D.A., et al., 2019. Recommendations for the packaging and containerizing of bioinformatics software. F1000Research 7.

Blinov, M.L., Gennari, J.H., Karr, J.R., Moraru, I.I., Nickerson, D.P., Sauro, H.M., 2021. Practical resources for enhancing the reproducibility of mechanistic modeling in systems biology. Current Opinion in Systems Biology 27, 100350. doi:10.1016/j.coisb.2021.06.001.

Blomer, J., Buncic, P., Meusel, R., Ganis, G., Sfiligoi, I., Thain, D., 2015. The evolution of global scale filesystems for scientific software distribution. Computing in Science Engineering 17, 61–71. doi:10.1109/MCSE.2015.111.

Boettiger, C., 2015. An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 71–79. doi:10.1145/2723872.2723882.

Bontemps, C., Orozco, V., 2021. Toward a FAIR Reproducible Research. p. 595–613. doi:10.1007/978-3-030-73249-3_30.

Botvinik-Nezer, R., Wager, T.D., 2023. Reproducibility in neuroimaging analysis: Challenges and solutions. Biological Psychiatry: Cognitive Neuroscience and Neuroimaging 8, 780–788. doi:https://doi.org/10.1016/j.bpsc.2022.12.006. reliability of Neurocognitive Measures for Mental Health.

Brinckman, A., Chard, K., Gaffney, N., Hategan, M., Jones, M.B., Kowalik, K., Kulasekaran, S., Ludäscher, B., Mecum, B.D., Nabrzyski, J., 2019. Computing environments for reproducibility: Capturing the "whole tale". Future Generation Computer Systems 94, 854–867.

Brito, J.J., Li, J., Moore, J.H., Greene, C.S., Nogoy, N.A., Garmire, L.X., Mangul, S., 2020. Recommendations to enhance rigor and reproducibility in biomedical research. GigaScience 9. doi:10.1093/gigascience/giaa056.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., Zazworka, N., 2010. Managing Technical Debt in Software-Reliant Systems, in: Workshop on Future of SE Research, Association for Computing Machinery, New York, NY, USA. p. 47–52. doi:10.1145/1882362.1882373.

Brunsdon, C., Comber, A., 2020. Opening practice: Supporting reproducibility and critical spatial data science. arXiv:2008.03256.

Buckheit, J.B., Donoho, D.L., 1995. WaveLab and Reproducible Research. Lecture Notes in Statistics. p. 55–81. doi:10.1007/978-1-4612-2544-7_5.

Bugbee, K., Ramachandran, R., Maskey, M., Barciauskas, A., Kaulfus, A., That, D.H.T., Virts, K., Markert, K., Lynnes, C., 2020. Advancing open science through innovative data system solutions: The joint ESA-NASA multi-mission algorithm and analysis platform (MAAP)'s data ecosystem, in: IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, pp. 3097–3100. doi:10.1109/IGARSS39084.2020.9323731.

Bánáti, A., Kacsuk, P., Kozlovszky, M., 2015. Four level provenance support to achieve portable reproducibility of scientific workflows, in: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 241–244. doi:10.1109/MIPRO.2015.7160272.

Bánáti, A., Kacsuk, P., Kozlovszky, M., 2016. Classification of scientific workflows based on reproducibility analysis, in: 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 327–331. doi:10.1109/MIPRO.2016.7522161.

Canon, R.S., 2020. The role of containers in reproducibility, in: 2020 2nd International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), pp. 19–25. doi:10.1109/CANOPIEHPC51917.2020.00008.

Canon, R.S., Younge, A., 2019. A case for portability and reproducibility of HPC containers, in: 2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), pp. 49–54. doi:10.1109/CANOPIE-HPC49598.2019.00012.

Casseau, C., Falleri, J.R., Blanc, X., Degueule, T., 2021. Immediate feedback for students to solve notebook reproducibility problems in the classroom, in: 2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 1–5. doi:10.1109/VL/HCC51201.2021.9576363.

Castleberry, D.G., Brandt, S.R., Löffler, F., Krishnan, H., 2012. The prickly pear archive: a portable hypermedia for scholarly publication, in: Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the EXtreme to the Campus and Beyond, Association for Computing Machinery, New York, NY, USA. doi:10.1145/2335755.2335840.

Chan, C.H., Schoch, D., 2023. rang: Reconstructing reproducible R computational environments. PLOS ONE 18, e0286761. doi:10.1371/journal.pone.0286761.

Chen, X., Dallmeier-Tiessen, S., Dasler, R., Feger, S., Fokianos, P., Gonzalez, J.B., Hirvonsalo, H., Kousidis, D., Lavasa, A., Mele, S., et al., 2019. Open is not enough. Nature Physics 15, 113–119.

Chirigati, F., Shasha, D., Freire, J., 2013. Reprozip: Using provenance to support computational reproducibility, in: Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, USENIX Association, USA.

Choi, Y.D., Goodall, J.L., Sadler, J.M., Castronova, A.M., Bennett, A., Li, Z., Nijssen, B., Wang, S., Clark, M.P., Ames, D.P., Horsburgh, J.S., Yi, H., Bandaragoda, C., Seul, M., Hooper, R., Tarboton, D.G., 2021. Toward open and reproducible environmental modeling by integrating online data repositories, computational environments, and model application programming interfaces. Environmental Modelling & Software 135, 104888. doi:https://doi.org/10.1016/j.envsoft.2020.104888.

Choi, Y.D., Roy, B., Nguyen, J., Ahmad, R., Maghami, I., Nassar, A., Li, Z., Castronova, A.M., Malik, T., Wang, S., Goodall, J.L., 2023. Comparing containerization-based approaches for reproducible computational modeling of environmental systems. Environmental Modelling Software 167, 105760. doi:https://doi.org/10.1016/j.envsoft.2023.105760.

Chue Hong, N., 2018. To achieve the goals of e-science, we must change research culture globally. Informatik Spektrum 41, 414–420. doi:10.1007/s00287-018-01134-1.

Cito, J., Gall, H.C., 2016. Using docker containers to improve reproducibility in software engineering research, in: 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), pp. 906–907.

Claerbout, J.F., Karrenbach, M., 1992. Electronic documents give reproducible research a new meaning: 62nd ann.

Clyburne-Sherin, A., Fei, X., Green, S., 2019. Computational reproducibility via containers in psychology. Meta-Psychology 3. doi:10.15626/MP.2018.892.

Codabux, Z., Vidoni, M., Fard, F., 2021. Technical Debt in the Peer-Review Documentation of R Packages: a rOpenSci Case Study, in: International Conference on Mining Software Repositories, IEEE, Madrid, Spain. pp. 1–11.

Cook, N., Milojicic, D., Kaufmann, R., Sevinsky, J., 2012. N3phele: Open science-as-a-service workbench for cloud-based scientific computing, in: 2012 7th Open Cirrus Summit, pp. 1–5. doi:10.1109/OCS.2012.30.

Corbin, J.M., Strauss, A., 2008. Basics of qualitative research (3rd ed.): Techniques and procedures for developing grounded theory. URL: https://api.semanticscholar.org/CorpusID:67424455.

Crick, T., Hall, B.A., Ishtiaq, S., 2017. Reproducibility in research: Systems, infrastructure, culture. Journal of Open Research Software 5, 32. doi:10.5334/jors.73.

Crook, S.M., Davison, A.P., Plesser, H.E., 2013. Learning from the past: Approaches for reproducibility in computational neuroscience.

Crouch, S., Hong, N.C., Hettrick, S., Jackson, M., Pawlik, A., Sufi, S., Carr, L., De Roure, D., Goble, C., Parsons, M., 2013. The software sustainability institute: Changing research software attitudes and practices. Computing in Science Engineering 15, 74–80. doi:10.1109/MCSE.2013.133.

Cruz, M.J., Kurapati, S., Turkyilmaz-van der Velden, Y., 2018. The role of data stewardship in software sustainability and reproducibility, in: 2018 IEEE 14th International Conference on e-Science (e-Science). doi:10.1109/eScience.2018.00009.

Cruzes, D.S., Dyba, T., 2011. Recommended steps for thematic synthesis in software engineering, in: 2011 International Symposium on Empirical Software Engineering and Measurement, IEEE, New York, NY, USA. pp. 275–284. doi:10.1109/ESEM.2011.36.

Cunningham, W., 1992. The WyCash portfolio management system, in: Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum), Association for Computing Machinery, New York, NY, USA. p. 29–30. URL: https://doi.org/10.1145/157709.157715, doi:10.1145/157709.157715.

Curtis, B., Sappidi, J., Szynkarski, A., 2012. Estimating the principal of an application's technical debt. IEEE Software 29, 34–42. doi:10.1109/MS.2012.156.

Cushing, J., Lach, D., Zanocco, C., Halama, J., 2018. Scientific visualization and reproducibility for open environmental science, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE Computer Society, Los Alamitos, CA, USA. pp. 3211–3216. doi:10.1109/BigData.2018.8622039.

Dalle, O., 2012. On reproducibility and traceability of simulations, in: Proceedings of the 2012 Winter Simulation Conference (WSC), pp. 1–12. doi:10.1109/WSC.2012.6465284.

Davis-Turak, J., Courtney, S.M., Hazard, E.S., Glen, W.B., Da Silveira, W.A., Wesselman, T., Harbin, L.P., Wolf, B.J., Chung, D., Hardiman, G., 2017. Genomics pipelines and data integration: challenges and opportunities in the research setting. Expert Review of Molecular Diagnostics 17, 225–237. doi:10.1080/14737159.2017.1282822.

Denaxas, S., Direk, K., Gonzalez-Izquierdo, A., Pikoula, M., Cakiroglu, A., Moore, J., Hemingway, H., Smeeth, L., 2017. Methods for enhancing the reproducibility of biomedical research findings using electronic health records. BioData Mining 10. doi:10.1186/s13040-017-0151-7.

Di Meglio, A., Estrella, F., Riedel, M., 2012. On realizing the concept study sciencesoft of the european middleware initiative: Open software for open science, in: 2012 IEEE 8th International Conference on E-Science, pp. 1–8. doi:10.1109/eScience.2012.6404450.

Docker, . https://www.docker.com/.

Dorodchi, M., Al-Hossami, E., Benedict, A., Demeter, E., 2019. Using synthetic data generators to promote open science in higher education learning analytics, in: 2019 IEEE International Conference on Big Data (Big Data), pp. 4672–4675. doi:10.1109/BigData47090.2019.9006475.

Dylan Chapp, Victoria Stodden, M.T., 2020. Building a vision for reproducibility in the cyberinfrastructure ecosystem: Leveraging community efforts. Supercomputing Frontiers and Innovations 7. doi:10.14529/jsfi200106.

Eckersley, P., Egan, G.F., De Schutter, E., Yiyuan, T., Novak, M., Sebesta, V., Matthiessen, L., Jaaskelainen, I.P., Ruotsalainen, U., Herz, A.V.M., et al., 2003. Neuroscience data and tool sharing. Neuroinformatics 1, 149–165. doi:10.1007/s12021-003-0002-1.

Edmunds, S.C., Li, P., Hunter, C.I., Xiao, S.Z., Davidson, R.L., Nogoy, N., Goodman, L., 2017. Experiences in integrated data and research object publishing using gigadb. International Journal on Digital Libraries 18, 99–111. doi:10.1007/s00799-016-0174-6.

Engel, F., Keary, A., Berwind, K., Bornschlegl, M.X., Hemmje, M., 2017. The role of reproducibility in affective computing, in: 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 2008–2014. doi:10.1109/BIBM.2017.8217969.

Erdemir, A., Guess, T.M., Halloran, J.P., Modenese, L., Reinbolt, J.A., Thelen, D.G., Umberger, B.R., 2016. Commentary on the integration of model sharing and reproducibility analysis to scholarly publishing workflow in computational biomechanics. IEEE Transactions on Biomedical Engineering 63, 2080–2085. doi:10.1109/TBME.2016.2602760.

Ernst, N., Kazman, R., Delange, J., 2021. Technical Debt in Practice: How to Find It and Fix It. MIT Press.

Essawy, B.T., Goodall, J.L., Voce, D., Morsy, M.M., Sadler, J.M., Choi, Y.D., Tarboton, D.G., Malik, T., 2020. A taxonomy for reproducible and replicable research in environmental modelling. Environmental Modelling & Software 134, 104753. doi:https://doi.org/10.1016/j.envsoft.2020.104753.

Essawy, B.T., Goodall, J.L., Xu, H., Gil, Y., 2017. Evaluation of the ontosoft ontology for describing metadata for legacy hydrologic modeling software. Environmental Modelling & Software 92, 317–329. doi:10.1016/j.envsoft.2017.01.024.

Feger, S.S., Wozniak, P.W., Lischke, L., Schmidt, A., 2020. 'Yes, I comply!': Motivations and practices around research data management and reuse across scientific fields. Proc. ACM Hum.-Comput. Interact. 4. doi:10.1145/3415212.

Fehr, J., Heiland, J., Himpe, C., Saak, J., 2016. Best practices for replicability, reproducibility and reusability of computer-based experiments exemplified by model reduction software. AIMS Mathematics 1, 261–281. doi:10.3934/math.2016.3.261.

Feinberg, M., Sutherland, W., Nelson, S.B., Jarrahi, M.H., Rajasekar, A., 2020. The new reality of reproducibility: The role of data work in scientific research. Proc. ACM Hum.-Comput. Interact. 4. doi:10.1145/3392840.

Fernandez-Prades, C., Vila-Valls, J., Arribas, J., Ramos, A., 2018. Continuous reproducibility in GNSS signal processing. IEEE Access 6, 20451–20463. doi:10.1109/access.2018.2822835.

Fernández-Sánchez, C., Garbajosa, J., Yagüe, A., Perez, J., 2017. Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. Journal of Systems and Software 124, 22–38. doi:https://doi.org/10.1016/j.jss.2016.10.018.

Fidler, F., Chee, Y.E., Wintle, B.C., Burgman, M.A., McCarthy, M.A., Gordon, A., 2017. Metaresearch for evaluating reproducibility in ecology and evolution. BioScience 67, 282–289. doi:10.1093/biosci/biw159.

Figshare, . https://figshare.com/.

Fiore, S., D'Anca, A., Palazzo, C., Foster, I., Williams, D., Aloisio, G., 2013. Ophidia: Toward big data analytics for escience. Procedia Computer Science 18, 2376–2385. doi:https://doi.org/10.1016/j.procs.2013.05.409. 2013 International Conference on Computational Science.

Fiore, S., Elia, D., Palazzo, C., D'Anca, A., Antonio, F., Williams, D.N., Foster, I., Aloisio, G., 2018. Towards an open (data) science analytics-hub for reproducible multi-model climate analysis at scale, in: 2018 IEEE International Conference on Big Data (Big Data), pp. 3226–3234. doi:10.1109/BigData.2018.8622205.

Flisar, J., Podgorelec, V., 2019. Identification of self-admitted technical debt using enhanced feature selection based on word embedding. IEEE Access 7, 106475–106494. doi:10.1109/ACCESS.2019.2933318.

Föll, M.C., Moritz, L., Wollmann, T., Stillger, M.N., Vockert, N., Werner, M., Bronsert, P., Rohr, K., Grüning, B.A., Schilling, O., 2019. Accessible and reproducible mass spectrometry imaging data analysis in galaxy. Gigascience 8, giz143.

Freire, J., Bonnet, P., Shasha, D., 2012. Computational reproducibility: State-of-the-art, challenges, and database research opportunities, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, New York, NY, USA. p. 593–596. URL: https://doi.org/10.1145/2213836.2213908, doi:10.1145/2213836.2213908.

Freire, S., Rios, N., Mendonça, M., Falessi, D., Seaman, C., Izurieta, C., Spínola, R.O., 2020. Actions and impediments for technical debt prevention: Results from a global family of industrial surveys, in: 35th Annual ACM Symposium on Applied Computing, ACM, USA. p. 1548–1555.

Frery, A.C., Gomez, L., Medeiros, A.C., 2020. A badging system for reproducibility and replicability in remote sensing research. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13, 4988–4995. doi:10.1109/JSTARS.2020.3019418.

Fucci, G., Cassee, N., Zampetti, F., Novielli, N., Serebrenik, A., Di Penta, M., 2021. Waiting Around or Job Half-Done? Sentiment in Self-Admitted Technical Debt, in: 18th International Conference on Mining Software Repositories, IEEE, Madrid, Spain. pp. 403–414. doi:10.1109/MSR52588.2021.00052.

Garcia-Silva, A., Gomez-Perez, J.M., Palma, R., Krystek, M., Mantovani, S., Foglini, F., Grande, V., De Leo, F., Salvi, S., Trasatti, E., Romaniello, V., Albani, M., Silvagni, C., Leone, R., Marelli, F., Albani, S., Lazzarini, M., Napier, H.J., Glaves, H.M., Aldridge, T., Meertens, C., Boler, F.,

Loescher, H.W., Laney, C., Genazzio, M.A., Crawl, D., Altintas, I., 2019. Enabling fair research in earth science through research objects. Future Generation Computer Systems 98, 550–564. doi:https://doi.org/10.1016/j.future.2019.03.046.

Garrett-Ruffin, S., Hindash, A.C., Kaczkurkin, A.N., Mears, R.P., Morales, S., Paul, K., Pavlov, Y.G., Keil, A., 2021. Open science in psychophysiology: An overview of challenges and emerging solutions. International Journal of Psychophysiology 162, 69–78. doi:https://doi.org/10.1016/j.ijpsycho.2021.02.005.

Geiger, R.S., Sholler, D., Culich, A., Martinez, C., Hoces de la Guardia, F., Lanusse, F., Ottoboni, K., Stuart, M., Vareth, M., Varoquaux, N., et al., 2018. Challenges of Doing Data-Intensive Research in Teams, Labs, and Groups: Report from the BIDS Best Practices in Data Science Series.

Gentleman, R., Lang, D.T., 2007. Statistical analyses and reproducible research. Journal of Computational and Graphical Statistics 16, 1–23.

Ghoshal, D., Bianchi, L., Essiari, A., Paine, D., Poon, S.S., Beach, M., N'Diaye, A.T., Huck, P., Ramakrishnan, L., 2021. Science capsule: Towards sharing and reproducibility of scientific workflows, in: 2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS), pp. 66–73. doi:10.1109/WORKS54523.2021.00014.

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 2016. Toward the Geoscience Paper of the Future: Best practices for Documenting and Sharing Research from Data to Software to Provenance. Earth and Space Science 3, 388–415. doi:10.1002/2015EA000136.

Gil, Y., Ratnakar, V., Garijo, D., 2015. Ontosoft: Capturing scientific software metadata, in: Proceedings of the 8th International Conference on Knowledge Capture, pp. 1–4.

Gille, S., Abernathey, R., Chereskin, T., Cornuelle, B., Heimbach, P., Mazloff, M., Rocha, C., Soares, S., Sonnewald, M., Boas, B.V., et al., . Open code policy for nasa space science: A perspective from nasa-supported ocean modeling and ocean data analysis .

GitHub, . https://github.com/.

Gitlab, . https://about.gitlab.com/.

Goble, C., De Roure, D., Bechhofer, S., 2013. Accelerating Scientists' Knowledge Turns. Communications in Computer and Information Science. p. 3–25. doi:10.1007/978-3-642-37186-8_1.

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, R86. doi:10.1186/gb-2010-11-8-r86.

Gomes, D.G.E., Pottier, P., Crystal-Ornelas, R., Hudgins, E.J., Foroughirad, V., Sánchez-Reyes, L.L., Turba, R., Martinez, P.A., Moreau, D., Bertram, M.G., et al., 2022. Why don't we share data and code? perceived barriers and benefits to public archiving practices. Proceedings of the Royal Society B: Biological Sciences 289. doi:10.1098/rspb.2022.1113.

González-Barahona, J.M., Robles, G., 2012. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. Empirical Software Engineering 17, 75–89.

Gorgolewski, K.J., Alfaro-Almagro, F., Auer, T., Bellec, P., Capotă, M., Chakravarty, M.M., Churchill, N.W., Cohen, A.L., Craddock, R.C., Devenyi, G.A., et al., 2017. Bids apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. PLOS Computational Biology 13, e1005209. doi:10.1371/journal.pcbi.1005209.

Goswami, P., Gupta, S., Li, Z., Meng, N., Yao, D., 2020. Investigating the reproducibility of npm packages, in: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 677–681. doi:10.1109/ICSME46990.2020.00071.

von Hahn, T., Mechefske, C.K., 2022. Computational reproducibility within prognostics and health management. arXiv:2205.15489.

Hale, J.S., Li, L., Richardson, C.N., Wells, G.N., 2017. Containers for portable, productive, and performant scientific computing. Computing in Science Engineering 19, 40–50. doi:10.1109/MCSE.2017.2421459.

Hannay, J.E., Dybå, T., Arisholm, E., Sjøberg, D.I., 2009. The effectiveness of pair programming: A meta-analysis. Information and software technology 51, 1110–1122.

Harrell, S.L., Michael, S., Maltzahn, C., 2022. Advancing adoption of reproducibility in hpc: A preface to the special section. IEEE Transactions on Parallel and Distributed Systems 33, 2011–2013. doi:10.1109/TPDS.2021.3128796.

Heaton, D., Carver, J.C., 2015. Claims about the use of software engineering practices in science: A systematic literature review. Information and Software Technology 67, 207–219. doi:10.1016/j.infsof.2015.07.011.

Hey, T., Payne, M.C., 2015. Open science decoded. Nature physics 11, 367–369.

Hidayetoğlu, M., Biçer, T., de Gonzalo, S.G., Ren, B., Gürsoy, D., Kettimuthu, R., Foster, I.T., Hwu, W.M.W., 2022. Memxct: Design, optimization, scaling, and reproducibility of x-ray tomography imaging. IEEE Transactions on Parallel and Distributed Systems 33, 2014–2031. doi:10.1109/TPDS.2021.3128032.

Hinsen, K., 2011. A data and code model for reproducible research and executable papers. Procedia Computer Science 4, 579–588. doi:https://doi.org/10.1016/j.procs.2011.04.061. proceedings of the International Conference on Computational Science, ICCS 2011.

Hosny, A., Vera-Licona, P., Laubenbacher, R.C., Favre, T., 2016. AlgoRun: a Docker-based packaging system for platform-agnostic implemented algorithms. Bioinformatics 32 15, 2396–8.

Howe, B., 2012. Virtual appliances, cloud computing, and reproducible research. Computing in Science & Engineering 14, 36–41.

Howison, J., Bullard, J., 2016. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. Journal of the Association for Information Science and Technology 67, 2137–2155. doi:10.1002/asi.23538.

Huber, R., D'Onofrio, C., Devaraju, A., Klump, J., Loescher, H.W., Kindermann, S., Guru, S., Grant, M., Morris, B., Wyborn, L., Evans, B., Goldfarb, D., Genazzio, M.A., Ren, X., Magagna, B., Thiemann, H., Stocker, M., 2021. Integrating data and analysis technologies within leading environmental research infrastructures: Challenges and approaches. Ecological Informatics 61, 101245. doi:https://doi.org/10.1016/j.ecoinf.2021.101245.

Huppmann, D., Gidden, M., Fricko, O., Kolp, P., Orthofer, C., Pimmer, M., Kushin, N., Vinca, A., Mastrucci, A., Riahi, K., Krey, V., 2019. The messageix integrated assessment model and the ix modeling platform (ixmp): An open framework for integrated and cross-cutting analysis of energy, climate, the environment, and sustainable development. Environmental Modelling Software 112, 143–156. doi:https://doi.org/10.1016/j.envsoft.2018.11.012.

Hutton, L., Henderson, T., 2018. Toward reproducibility in online social network research. IEEE Transactions on Emerging Topics in Computing 6, 156–167. doi:10.1109/tetc.2015.2458574.

Ibanez, L., Schroeder, W.J., Hanwell, M.D., 2018. Practicing open science, in: Implementing reproducible research. Chapman and Hall/CRC, pp. 241–280.

Ihle, M., Winney, I.S., Krystalli, A., Croucher, M., 2017. Striving for transparent and credible research: practical guidelines for behavioral ecologists. Behavioral Ecology 28, 348–354. doi:10.1093/beheco/arx003.

Irving, D., 2016. A minimum standard for publishing computational results in the weather and climate sciences. Bulletin of the American Meteorological Society 97, 1149–1158.

Isdahl, R., Gundersen, O.E., 2019. Out-of-the-box reproducibility: A survey of machine learning platforms. doi:10.1109/escience.2019.00017.

Ivie, P., Thain, D., 2019. Reproducibility in scientific computing. ACM Computing Surveys 51, 1–36. doi:10.1145/3186266.

Ivimey-Cook, E.R., Pick, J.L., Bairos-Novak, K.R., Culina, A., Gould, E., Grainger, M., Marshall, B.M., Moreau, D., Paquet, M., Royauté, R., et al., 2023. Implementing code review in the scientific workflow: Insights from ecology and evolutionary biology. Journal of Evolutionary Biology 36, 1347–1356. doi:10.1111/jeb.14230.

Izurieta, C., Ozkaya, I., Seaman, C.B., Kruchten, P.B., Nord, R.L., Snipes, W., Avgeriou, P., 2016. Perspectives on managing technical debt: A transition point and roadmap from dagstuhl, in: QuASoQ/TDA@APSEC.

Jalal Apostal, S.F., Apostal, D., Marsh, R., 2020. Improving numerical reproducibility of scientific software in parallel systems, in: 2020 IEEE

International Conference on Electro Information Technology (EIT), pp. 066–074. doi:10.1109/EIT48999.2020.9208338.

Jansen, C., Schilling, B., Strohmenger, K., Witt, M., Annuscheit, J., Krefting, D., 2019. Reproducibility and performance of deep learning applications for cancer detection in pathological images, in: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 621–630. doi:10.1109/CCGRID.2019.00080.

Jean-Paul, S., Elseify, T., Obeid, I., Picone, J., 2019. Issues in the reproducibility of deep learning results, in: 2019 IEEE Signal Processing in Medicine and Biology Symposium (SPMB), pp. 1–4. doi:10.1109/SPMB47826.2019.9037840.

Jenkins, J., Chang, L.C., Hutchinson, E., Irfanoglu, M.O., Pierpaoli, C., 2016. Harmonization of methods to facilitate reproducibility in medical data processing: Applications to diffusion tensor magnetic resonance imaging. doi:10.1109/bigdata.2016.7841086.

Jimenez, I., Arpaci-Dusseau, A., Arpaci-Dusseau, R., Lofstead, J., Maltzahn, C., Mohror, K., Ricci, R., 2017a. Popperci: Automated reproducibility validation, in: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 450–455. doi:10.1109/INFCOMW.2017.8116418.

Jimenez, I., Sevilla, M., Watkins, N., Maltzahn, C., Lofstead, J., Mohror, K., Arpaci-Dusseau, A., Arpaci-Dusseau, R., 2017b. The popper convention: Making reproducible systems evaluation practical, in: 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 1561–1570. doi:10.1109/IPDPSW.2017.157.

Jiménez, R.C., Kuzak, M., Alhamdoosh, M., Barker, M., Batut, B., Borg, M., Capella-Gutierrez, S., Chue Hong, N., Cook, M., Corpas, M., et al., 2017. Four simple recommendations to encourage best practices in research software. F1000Research 6, 876. doi:10.12688/f1000research.11407.1.

Johanson, A.N., Hasselbring, W., 2018. Software Engineering for Computational Science: Past, Present, Future. Computing in Science & Engineering 20, 90–109.

Jupyter, . https://jupyter.org/.

Jézéquel, F., Lamotte, J.L., Saïd, I., 2015. Estimation of numerical reproducibility on cpu and gpu, in: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 675–680. doi:10.15439/2015F29.

Kalenkovich, E., Levchenko, E., 2021. A reproducible MEEG data analysis workflow with Conda, Snakemake, and R Markdown. doi:https://doi.org/10.31234/osf.io/3k8nt.

Kanewala, U., Bieman, J.M., 2014. Testing scientific software: A systematic literature review. Information and Software Technology 56, 1219–1232. doi:https://doi.org/10.1016/j.infsof.2014.05.006.

Kanwal, S., Lonie, A., Sinnott, R.O., 2017. Digital reproducibility requirements of computational genomic workflows, in: 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1522–1529. doi:10.1109/BIBM.2017.8217887.

Kedron, P., Li, W., Fotheringham, S., Goodchild, M., 2021. Reproducibility and replicability: Opportunities and challenges for geospatial research. International Journal of Geographical Information Science 35, 427–445. doi:10.1080/13658816.2020.1802032.

Kellogg, L.H., Hwang, L.J., Gassmoller, R., Bangerth, W., Heister, T., 2019. The role of scientific communities in creating reusable software: Lessons from geophysics. Computing in Science Engineering 21, 25–35. doi:10.1109/mcse.2018.2883326.

Kim, Y.M., Poline, J.B., Dumas, G., 2018. Experimenting with reproducibility: a case study of robustness in bioinformatics. GigaScience 7, giy077. doi:10.1093/gigascience/giy077.

Kitchenham, B., Pretorius, R., Budgen, D., Pearl Brereton, O., Turner, M., Niazi, M., Linkman, S., 2010. Systematic literature reviews in software engineering – a tertiary study. Information and Software Technology 52, 792–805.

Kitchenham, B.A., Charters, S., 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001. Keele University and Durham University Joint Report.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S., et al., 2016. Jupyter notebooks-a publishing format for reproducible computational workflows. Elpub 2016, 87–90.

Knitr, . https://yihui.org/knitr/.

Knoll, C., Heedt, R., 2020. "automatic control knowledge repository" – a computational approach for simpler and more robust reproducibility of results in control theory, in: 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), pp. 130–136. doi:10.1109/ICSTCC50638.2020.9259657.

Knuth, D.E., 1984. Literate Programming. The Computer Journal 27, 97–111. doi:10.1093/comjnl/27.2.97.

Koehler Leman, J., Weitzner, B.D., Renfrew, P.D., Lewis, S.M., Moretti, R., Watkins, A.M., Mulligan, V.K., Lyskov, S., Adolf-Bryfogle, J., Labonte, J.W., et al., 2020. Better together: Elements of successful scientific software development in a distributed collaborative community. PLOS Computational Biology 16, e1007507. doi:10.1371/journal.pcbi.1007507.

Krafczyk, M., Shi, A., Bhaskar, A., Marinov, D., Stodden, V., 2019. Scientific tests and continuous integration strategies to enhance reproducibility in the scientific software context, in: Proceedings of the 2nd International Workshop on Practical Reproducible Evaluation of Computer Systems, Association for Computing Machinery, New York, NY, USA. p. 23–28. doi:10.1145/3322790.3330595.

Krafczyk, M.S., Shi, A., Bhaskar, A., Marinov, D., Stodden, V., 2021. Learning from reproducing computational results: introducing three principles and the reproduction package. Philosophical Transactions of the Royal Society A 379, 20200069.

Kubernetes, . https://kubernetes.io/.

Lacerda, G., Petrillo, F., Pimenta, M., Guéhéneuc, Y.G., 2020. Code smells and refactoring: A tertiary systematic review of challenges and observations. Journal of Systems and Software 167, 110610.

Laine, C., Goodman, S.N., Griswold, M.E., Sox, H.C., 2007. Reproducible research: Moving toward research the public can really trust. Annals of Internal Medicine 146, 450–453.

Langlois, P., Nheili, R., Denis, C., 2015. Numerical reproducibility: Feasibility issues, in: 2015 7th International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5. doi:10.1109/NTMS.2015.7266509.

Lee, G., Bacon, S., Bush, I., Fortunato, L., Gavaghan, D., Lestang, T., Morton, C., Robinson, M., Rocca-Serra, P., Sansone, S.A., et al., 2021. Barely sufficient practices in scientific computing. Patterns 2, 100206. doi:10.1016/j.patter.2021.100206.

Leek, J.T., Jager, L.R., 2017. Is most published research really false? Annual Review of Statistics and Its Application 4, 109–122. doi:10.1146/annurev-statistics-060116-054104.

Lefebvre, A., Spruit, M., 2023. Laboratory forensics for open science readiness: an investigative approach to research data management. Information Systems Frontiers 25, 381–399. doi:10.1007/s10796-021-10165-1.

Leipzig, J., Nüst, D., Hoyt, C.T., Ram, K., Greenberg, J., 2021. The role of metadata in reproducible computational research. Patterns 2, 100322. doi:https://doi.org/10.1016/j.patter.2021.100322.

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., Arcelli Fontana, F., 2021. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools. Journal of Systems and Software 171, 110827. URL: https://www.sciencedirect.com/science/article/pii/S016412122030220X, doi:https://doi.org/10.1016/j.jss.2020.110827.

LeVeque, R.J., 2009. Python tools for reproducible research on hyperbolic problems. Computing in Science & Engineering 11.

Levet, F., Carpenter, A.E., Eliceiri, K.W., Kreshuk, A., Bankhead, P., Haase, R., 2021. Developing open-source software for bioimage analysis: opportunities and challenges. F1000Research 10, 302. doi:10.12688/f1000research.52531.1.

Li, Z., Avgeriou, P., Liang, P., 2015. A systematic mapping study on technical debt and its management. Journal of Systems and Software 101, 193–220.

Li, Z., Liang, P., Avgeriou, P., 2014. Chapter 9 - architectural debt management in value-oriented architecting, in: Mistrik, I., Bahsoon, R., Kazman, R., Zhang, Y. (Eds.), Economics-Driven Software Architecture. Morgan Kaufmann, Boston, pp. 183–204. doi:https://doi.org/10.

1016/B978-0-12-410464-8.00009-X.

Lifschitz, S., Gomes, L., Rehen, S.K., 2011. Dealing with reusability and reproducibility for scientific workflows, in: 2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), pp. 625–632. doi:10.1109/BIBMW.2011.6112441.

Lim, E., Taksande, N., Seaman, C., 2012. A balancing act: What software practitioners have to say about technical debt. IEEE Software 29, 1–10.

Lima, J., Júnior, M.A., Moya, A., Almeida, R., Anjos, P., Lencastre, M., Fagundes, R., Alencar, F., 2019. As metodologias ativas e o ensino em engenharia de software: uma revisão sistemática da literatura, in: Proceedings of the 25th Workshop on Computing at School, SBC, Porto Alegre, RS, Brasil. pp. 1014–1023. doi:10.5753/cbie.wie.2019.1014.

Liu, J., Huang, Q., Xia, X., Shihab, E., Lo, D., Li, S., 2020. Is using deep learning frameworks free? characterizing technical debt in deep learning frameworks, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society, Association for Computing Machinery, New York, NY, USA. p. 1–10. doi:10.1145/3377815.3381377.

Lowndes, J.S.S., Best, B.D., Scarborough, C., Afflerbach, J.C., Frazier, M.R., O'Hara, C.C., Jiang, N., Halpern, B.S., 2017. Our path to better science in less time using open data science tools. Nature ecology & evolution 1, 0160.

Lupelli, I., Muir, D., Appel, L., Akers, R., Carr, M., Abreu, P., 2015. Provenance metadata gathering and cataloguing of efit++ code execution. Fusion Engineering and Design 96-97, 835–839. doi:https://doi.org/10.1016/j.fusengdes.2015.04.016. proceedings of the 28th Symposium On Fusion Technology (SOFT-28).

Maghami, I., Van Beusekom, A., Hay, L., Li, Z., Bennett, A., Choi, Y., Nijssen, B., Wang, S., Tarboton, D., Goodall, J.L., 2023. Building cyberinfrastructure for the reuse and reproducibility of complex hydrologic modeling studies. Environmental Modelling Software 164, 105689. doi:https://doi.org/10.1016/j.envsoft.2023.105689.

Maldonado, E., Shihab, E., 2015. Detecting and Quantifying Different Types of Self-Admitted Technical Debt, in: 7th International Workshop on Managing Technical Debt, IEEE, Bremen, Germany. pp. 9–15. doi:10.1109/MTD.2015.7332619.

Marek, M., Teymoori, P., Welzl, M., Gjessing, S., 2018. Computer-aided reproducibility. doi:10.1109/iccnc.2018.8390274.

Markdown, R., . R Markdown — rmarkdown.rstudio.com. https://rmarkdown.rstudio.com/.

Marrone, S., Olivieri, S., Piantadosi, G., Sansone, C., 2019. Reproducibility of deep cnn for biomedical image processing across frameworks and architectures, in: 2019 27th European Signal Processing Conference (EUSIPCO), pp. 1–5. doi:10.23919/EUSIPCO.2019.8902690.

Marwick, B., 2017. Computational reproducibility in archaeological research: Basic principles and a case study of their implementation. Journal of Archaeological Method and Theory 24, 424–450. doi:10.1007/s10816-015-9272-9.

Mauerer, W., Klessinger, S., Scherzinger, S., 2023. Beyond the badge: Reproducibility engineering as a lifetime skill, in: Proceedings of the 4th International Workshop on Software Engineering Education for the Next Generation, Association for Computing Machinery, New York, NY, USA. p. 1–4. doi:10.1145/3528231.3528359.

Mauerer, W., Scherzinger, S., 2021. Nullius in verba: Reproducibility for database systems research, revisited, in: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 2377–2380. doi:10.1109/ICDE51399.2021.00270.

Mauerer, W., Scherzinger, S., 2022. 1-2-3 reproducibility for quantum software experiments, in: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 1247–1248. doi:10.1109/SANER53432.2022.00148.

McConnell, S., 2008. Managing technical debt. Construx Software Builders, Inc , 1–14.

Mccormick, M., Liu, X., Jomier, J., Marion, C., Ibanez, L., 2014. Itk: enabling reproducible research and open science. Frontiers in Neuroinformatics 8. doi:10.3389/fninf.2014.00013.

Mcdougal, R.A., Bulanova, A.S., Lytton, W.W., 2016. Reproducibility in computational neuroscience models and simulations. IEEE Transactions on Biomedical Engineering 63, 2021–2035. doi:10.1109/tbme.2016.2539602.

McFee, B., Kim, J.W., Cartwright, M., Salamon, J., Bittner, R.M., Bello, J.P., 2018. Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. IEEE Signal Processing Magazine 36, 128–137.

McPhillips, T., Willis, C., Gryk, M.R., Nuñez-Corrales, S., Ludäscher, B., 2019. Reproducibility by other means: Transparent research objects, in: 2019 15th International Conference on eScience (eScience), pp. 502–509. doi:10.1109/eScience.2019.00066.

Mecum, B., Jones, M.B., Vieglais, D., Willis, C., 2018. Preserving reproducibility: Provenance and executable containers in dataone data packages, in: 2018 IEEE 14th International Conference on e-Science (e-Science), pp. 45–49. doi:10.1109/eScience.2018.00019.

Melo, A., Fagundes, R., Lenarduzzi, V., Santos, W.B., 2022. Identification and measurement of requirements technical debt in software development: A systematic literature review. Journal of Systems and Software 194, 111483. doi:https://doi.org/10.1016/j.jss.2022.111483.

Mendez, D., Graziotin, D., Wagner, S., Seibold, H., 2020. Open Science in Software Engineering. p. 477–501. doi:10.1007/978-3-030-32489-6_17.

Méndez Fernández, D., Monperrus, M., Feldt, R., Zimmermann, T., 2019. The open science initiative of the empirical software engineering journal. Empirical Software Engineering 24, 1057–1060.

Meng, H., Kommineni, R., Pham, Q., Gardner, R., Malik, T., Thain, D., 2015. An invariant framework for conducting reproducible computational science. Journal of Computational Science 9, 137–142. doi:https://doi.org/10.1016/j.jocs.2015.04.012. computational Science at the Gates of Nature.

Mesos, . https://mesos.apache.org/.

Milham, M.P., Klein, A., 2019. Be the change you seek in science. BMC Biology 17. doi:10.1186/s12915-019-0647-3.

Miller, J., 2005. Replicating software engineering experiments: a poisoned chalice or the holy grail. Inf. Softw. Technol. 47, 233–244.

Millman, K.J., Pérez, F., 2018. Developing open-source scientific practice, in: Implementing reproducible research. Chapman and Hall/CRC, pp. 149–183.

Morin, A., Urban, J., Adams, P.D., Foster, I., Sali, A., Baker, D., Sliz, P., 2012. Shining light into black boxes. Science 336, 159–160. doi:10.1126/science.1218263.

Morrison, R., 2018. Energy system modeling: Public transparency, scientific reproducibility, and open development. Energy Strategy Reviews 20, 49–63.

Mukherjee, S., Almanza, A., Rubio-González, C., 2021. Fixing dependency errors for python build reproducibility. doi:10.1145/3460319.3464797.

Nguyen, B., Berger, C., Benderius, O., 2019. Systematic benchmarking for reproducibility of computer vision algorithms for real-time systems: The example of optic flow estimation, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5264–5269. doi:10.1109/IROS40897.2019.8968066.

Niso, G., Krol, L.R., Combrisson, E., Dubarry, A.S., Elliott, M.A., François, C., Héjja-Brichard, Y., Herbst, S.K., Jerbi, K., Kovic, V., Lehongre, K., Luck, S.J., Mercier, M., Mosher, J.C., Pavlov, Y.G., Puce, A., Schettino, A., Schön, D., Sinnott-Armstrong, W., Somon, B., Šoškić, A., Styles, S.J., Tibon, R., Vilas, M.G., van Vliet, M., Chaumon, M., 2022. Good scientific practice in EEG and MEG research: Progress and perspectives. NeuroImage 257, 119056.

NSF, . https://www.nsf.gov/pubs/2018/nsf18053/nsf18053.jsp.

Nüst, D., Eglen, S.J., 2021. Codecheck: an open science initiative for the independent execution of computations underlying research articles during peer review to improve reproducibility. F1000Research 10, 253. doi:10.12688/f1000research.51738.2.

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. doi:10.1045/january2017-nuest.

Orchard, D., Rice, A., 2014. A computational science agenda for programming language research. Procedia Computer Science 29, 713–727. doi:https://doi.org/10.1016/j.procs.2014.05.064. 2014 International Conference on Computational Science.

Orozco, V., Bontemps, C., Maigné, E., Piguet, V., Hofstetter, A., Lacroix, A., Levert, F., Rousselle, J., 2020. How to make a pie: Reproducible research for empirical economics and econometrics. Journal of Economic Surveys 34, 1134–1169. doi:10.1111/joes.12389.

Orzechowski, M., Bali's, B., Slota, R.G., Kitowski, J., 2020. Reproducibility of computational experiments on kubernetes-managed container clouds with hyperflow, in: Computational Science – ICCS 2020, Springer International Publishing, Cham. pp. 220–233.

Parashar, M., 2020. Leveraging the national academies' reproducibility and replication in science report to advance reproducibility in publishing 2. doi:10.1162/99608f92.b69d3134.

Peer, L., Orr, L.V., Coppock, A., 2021. Active maintenance: A proposal for the long-term computational reproducibility of scientific results. PS: Political Science & Politics 54, 462–466. doi:10.1017/S1049096521000366.

Peng, R.D., 2011. Reproducible research in computational science. Science 334, 1226–1227. doi:10.1126/science.1213847.

Peng, R.D., Dominici, F., Zeger, S.L., 2006. Reproducible epidemiologic research. American journal of epidemiology 163, 783—789. doi:10.1093/aje/kwj093.

Perkel, J., 2020. Challenge to scientists: does your ten-year-old code still run? Nature 584, 656–658. doi:10.1038/d41586-020-02462-7.

Pernet, C., Poline, J.B., 2015. Improving functional magnetic resonance imaging reproducibility. GigaScience 4. doi:10.1186/s13742-015-0055-8.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology 64, 1 – 18.

Piccolo, S.R., Frampton, M.B., 2016. Tools and techniques for computational reproducibility. GigaScience 5. doi:10.1186/s13742-016-0135-4.

Pimentel, J.F., Murta, L., Braganholo, V., Freire, J., 2019. A large-scale study about quality and reproducibility of jupyter notebooks, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pp. 507–517. doi:10.1109/MSR.2019.00077.

Pinto, G., Wiese, I., Dias, L.F., 2018. How Do Scientists Develop Scientific Software? An External Replication, in: IEEE 25th International Conference on Software Analysis, Evolution and Reengineering, IEEE, Campobasso, Italy. pp. 582–591.

Poldrack, R.A., Feingold, F., Frank, M.J., Gleeson, P., De Hollander, G., Huys, Q.J.M., Love, B.C., Markiewicz, C.J., Moran, R., Ritter, P., et al., 2019. The importance of standards for sharing of computational models and data. Computational Brain Behavior 2, 229–232. doi:10.1007/s42113-019-00062-x.

Popp, J., Biskup, T., 2022. Aspecd: A modular framework for the analysis of spectroscopic data focussing on reproducibility and good scientific practice**. Chemistry–Methods 2. doi:10.1002/cmtd.202100097.

Potdar, A., Shihab, E., 2014. An Exploratory Study on Self-Admitted Technical Debt, in: International Conference on Software Maintenance and Evolution, IEEE, Victoria, Canada. pp. 91–100. doi:10.1109/ICSME.2014.31.

Pröell, S., Mayer, R., Rauber, A., 2015. Data access and reproducibility in privacy sensitive escience domains, in: 2015 IEEE 11th International Conference on e-Science, pp. 255–258. doi:10.1109/eScience.2015.20.

Pörtner, A., Hoffmann, M., Zug, S., Knig, M., 2018. SwarmRob: A Docker-based toolkit for reproducibility and sharing of experimental artifacts in robotics research, in: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 325–332. doi:10.1109/SMC.2018.00065.

Raff, E., Farris, A.L., 2023. A siren song of open source reproducibility, examples from machine learning, in: Proceedings of the 2023 ACM Conference on Reproducibility and Replicability, Association for Computing Machinery, New York, NY, USA. p. 115–120. doi:10.1145/3589806.3600042.

Raghupathi, W., Raghupathi, V., Ren, J., 2022. Reproducibility in computing research: An empirical study. IEEE Access 10, 29207–29223.

Ram, K., 2013. Git can facilitate greater reproducibility and increased transparency in science. Source Code for Biology and Medicine 8, 7. doi:10.1186/1751-0473-8-7.

Ram, K., Boettiger, C., Chamberlain, S., Ross, N., Salmon, M., Butland, S., 2019. A community of practice around peer review for long-term research software sustainability. Computing in Science Engineering 21, 59–65. doi:10.1109/MCSE.2018.2882753.

ReSA, . ReSA — researchsoft.org. https://www.researchsoft.org/. [Accessed 07-02-2024].

Revol, N., Théveny, P., 2014. Numerical reproducibility and parallel computations: Issues for interval algorithms. IEEE Transactions on Computers 63, 1915–1924. doi:10.1109/TC.2014.2322593.

Rios, N., Mendes, L., Cerdeiral, C., Magalhães, A.P.F., Perez, B., Correal, D., Astudillo, H., Seaman, C., Izurieta, C., Santos, G., Oliveira Spínola, R., 2020. Hearing the voice of software practitioners on causes, effects, and practices to deal with documentation debt, in: Madhavji, N., Pasquale, L., Ferrari, A., Gnesi, S. (Eds.), Requirements Engineering: Foundation for Software Quality, Springer International Publishing, Cham. pp. 55–70.

Rios, N., de Mendonça Neto, M.G., Spínola, R.O., 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Information and Software Technology 102, 117–145.

Robinson, A.C., Drake, R.R., Swan, M.S., Bennett, N.L., Smith, T.M., Hooper, R., Laity, G.R., 2021. A software environment for effective reliability management for pulsed power design. Reliability Engineering & System Safety 211, 107580. doi:https://doi.org/10.1016/j.ress.2021.107580.

Robles, G., 2010. Replicating msr: A study of the potential replicability of papers published in the mining software repositories proceedings, in: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp. 171–180. doi:10.1109/MSR.2010.5463348.

Rocha, J.C., Zapalowski, V., Nunes, I., 2017. Understanding Technical Debt at the Code Level from the Perspective of Software Developers. Association for Computing Machinery, New York, NY, USA. SBES'17, p. 64–73. doi:10.1145/3131151.3131164.

Rodríguez-Pérez, G., Robles, G., González-Barahona, J.M., 2018. Reproducibility and credibility in empirical software engineering: A case study based on a systematic literature review of the use of the szz algorithm. Information and Software Technology 99, 164–176. doi:https://doi.org/10.1016/j.infsof.2018.03.009.

Rokem, A., Marwick, B., Staneva, V., 2017. Assessing Reproducibility. University of California Press, Berkeley. pp. 1–18. doi:doi:10.1525/9780520967779-004.

Rollins, N.D., Barton, C.M., Bergin, S., Janssen, M.A., Lee, A., 2014. A computational model library for publishing model documentation and code. Environmental Modelling Software 61, 59–64. doi:https://doi.org/10.1016/j.envsoft.2014.06.022.

Rougier, N.P., Hinsen, K., Alexandre, F., Arildsen, T., Barba, L.A., Benureau, F.C., Brown, C.T., De Buyl, P., Caglayan, O., Davison, A.P., et al., 2017. Sustainable computational science: The rescience initiative. PeerJ Computer Science 3, e142. doi:10.7717/peerj-cs.142.

Rozier, K.Y., Rozier, E.W.D., 2014. Reproducibility, correctness, and buildability: The three principles for ethical public dissemination of computer science and engineering research, in: 2014 IEEE International Symposium on Ethics in Science, Technology and Engineering, pp. 1–13. doi:10.1109/ETHICS.2014.6893384.

Saarimaki, N., Baldassarre, M.T., Lenarduzzi, V., Romano, S., 2019. On the accuracy of sonarqube technical debt remediation time, in: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE Xplore, New York, NY, USA. pp. 317–324. doi:10.1109/SEAA.2019.00055.

Samuel, S., König-Ries, B., 2022. End-to-end provenance representation for the understandability and reproducibility of scientific experiments using a semantic approach. Journal of Biomedical Semantics 13, 1. doi:10.1186/s13326-021-00253-1.

Santana-Perez, I., Pérez-Hernández, M.S., 2015. Towards reproducibility in scientific workflows: An infrastructure-based approach. Scientific Programming 2015, 1–11. doi:10.1155/2015/243180.

Scheliga, K.S., Pampel, H., Konrad, U., Fritzsch, B., Schlauch, T., Nolden, M., zu Castell, W., Finke, A., Hammitzsch, M., Bertuch, O., Denker,

M., 2019. Dealing with research software: Recommendations for best practices.

Schwab, M., Karrenbach, M., Claerbout, J., 2000. Making scientific computations reproducible. Computing in Science and Engg. 2, 61–67.

Sciunit, . https://sciunit.run/.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F., Dennison, D., 2015. Hidden Technical Debt in Machine Learning Systems, in: NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, ACM, Montreal Canada. p. 2503–2511.

Shamir, L., Wallin, J.F., Allen, A., Berriman, B., Teuben, P., Nemiroff, R.J., Mink, J., Hanisch, R.J., DuPrie, K., 2013. Practices in source code sharing in astrophysics. Astronomy and Computing 1, 54–58. doi:https://doi.org/10.1016/j.ascom.2013.04.001.

Shull, F., Carver, J.C., Vegas, S., Juzgado, N.J., 2008. The role of replications in empirical software engineering. Empirical Software Engineering 13, 211–218.

Sierra, G., Shihab, E., Kamei, Y., 2019. A survey of self-admitted technical debt. Journal of Systems and Software 152, 70–82.

da Silva, W.D., Guareis de Farias, C.R., 2019. Support for accessibility, reproducibility and transparency in a service-oriented gene expression analysis platform, in: 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), pp. 477–482. doi:10.1109/CBMS.2019.00098.

da Silva Maldonado, E., Shihab, E., Tsantalis, N., 2017. Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt. IEEE Transactions on Software Engineering 43, 1044–1062. doi:10.1109/TSE.2017.2654244.

Skaggs, T., Young, M., Vrugt, J., 2015. Reproducible research in vadose zone sciences. Vadose Zone Journal 14. doi:10.2136/vzj2015.06.0088.

Smith, S., Jegatheesan, T., Kelly, D., 2016. Advantages, disadvantages and misunderstandings about document driven design for scientific software, in: 2016 Fourth International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering (SE-HPCCSE), IEEE. pp. 41–48.

Spencer Smith, W., Adam Lazzarato, D., Carette, J., 2016. State of the practice for mesh generation and mesh processing software. Advances in Engineering Software 100, 53–71. doi:https://doi.org/10.1016/j.advengsoft.2016.06.008.

Stevens, J., 2017. Replicability and reproducibility in comparative psychology. Frontiers in Psychology 8. doi:10.3389/fpsyg.2017.00862.

Stodden, V., 2009. Enabling reproducible research: Open licensing for scientific innovation. Science 13.

Stodden, V., 2010. Reproducible research: Addressing the need for data and code sharing in computational science. Computing in Science and Engineering 12, 8–13. doi:10.1109/MCSE.2010.113.

Stodden, V., Bailey, D.H., Borwein, J.M., LeVeque, R.J., Rider, W.J., Stein, W.A., 2013. Setting the default to reproducible reproducibility in computational and experimental mathematics. URL: https://api.semanticscholar.org/CorpusID:2142765.

Stodden, V., McNutt, M., Bailey, D.H., Deelman, E., Gil, Y., Hanson, B., Heroux, M.A., Ioannidis, J.P., Taufer, M., 2016. Enhancing reproducibility for computational methods. Science 354, 1240–1241.

Stodden, V., Miguez, S., 2014. Best practices for computational science: Software infrastructure and environments for reproducible and extensible research. Journal of Open Research Software 2, e21. doi:10.5334/jors.ay.

Tan, J., Feitosa, D., Avgeriou, P., 2022. Does it matter who pays back technical debt? an empirical study of self-fixed td. Information and Software Technology 143, 106738. doi:10.1016/j.infsof.2021.106738.

Tang, Y., Khatchadourian, R., Bagherzadeh, M., Singh, R., Stewart, A., Raja, A., 2021. An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems. IEEE, New York, NY, USA. pp. 238–250. doi:10.1109/icse43902.2021.00033.

Tatman, R., Vanderplas, J., Dane, S., 2018. A practical taxonomy of reproducibility for machine learning research.

Taubert, F., Bücker, H.M., 2017. On the reproducibility of biological image workflows by annotating computational results automatically, in: 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1538–1545. doi:10.1109/BIBM.2017.8217889.

Taufer, M., Padron, O., Saponaro, P., Patel, S., 2010. Improving numerical reproducibility and stability in large-scale numerical simulations on gpus, in: 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), pp. 1–9. doi:10.1109/IPDPS.2010.5470481.

Taylor, S.J.E., Fabiyi, A., Anagnostou, A., Barbera, R., Torrisi, M., Ricceri, R., Becker, B., 2016. Demonstrating open science for modeling simulation research, in: 2016 IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 191–192. doi:10.1109/DS-RT.2016.35.

Tierney, N.J., Ram, K., 2021. Common-sense approaches to sharing tabular data alongside publication. Patterns 2, 100368. doi:https://doi.org/10.1016/j.patter.2021.100368.

Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of Technical Debt. Journal of Systems and Software 86, 1498–1516. doi:https://doi.org/10.1016/j.jss.2012.12.052.

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:2005.02985.

Tsoukalas, D., Chatzigeorgiou, A., Ampatzoglou, A., Mittas, N., Kehagias, D., 2022. Td classifier: Automatic identification of java classes with high technical debt, in: 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), IEEE, New York, NY, USA. pp. 76–80.

Tsoukalas, D., Mittas, N., Chatzigeorgiou, A., Kehagias, D.D., Ampatzoglou, A., Amanatidis, T., Angelis, L., 2021. Machine learning for technical debt identification. IEEE Transactions on Software Engineering , 1–1doi:10.1109/TSE.2021.3129355.

Vaillancourt, P.Z., Coulter, J.E., Knepper, R., Barker, B., 2020. Self-scaling clusters and reproducible containers to enable scientific computing, in: 2020 IEEE High Performance Extreme Computing Conference (HPEC), IEEE. pp. 1–8.

Vegas, S., Juristo, N., Moreno, A., Solari, M., Letelier, P., 2006. Analysis of the influence of communication between researchers on experiment replication, in: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, Association for Computing Machinery, New York, NY, USA. p. 28–37. doi:10.1145/1159733.1159741.

Vidoni, M., 2021. Self-admitted technical debt in r packages: An exploratory study, in: IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE Computer Society, Los Alamitos, CA, USA. pp. 179–189. doi:10.1109/MSR52588.2021.00030.

Vilhuber, L., 2020. Reproducibility and replicability in economics 2. doi:10.1162/99608f92.4f6b9e67.

VirtualBox, .

Vitek, J., Kalibera, T., 2011. Repeatability, reproducibility, and rigor in systems research. doi:10.1145/2038642.2038650.

VMware, . https://www.vmware.com.

Wagner, S.J., Matek, C., Shetab Boushehri, S., Boxberg, M., Lamm, L., Sadafi, A., Winter, D.J., Marr, C., Peng, T., 2024. Built to last? reproducibility and reusability of deep learning algorithms in computational pathology. Modern Pathology 37, 100350. doi:https://doi.org/10.1016/j.modpat.2023.100350.

Waltemath, D., Wolkenhauer, O., 2016. How modeling standards, software, and initiatives support reproducibility in systems biology and systems medicine. IEEE Transactions on Biomedical Engineering 63, 1999–2006. doi:10.1109/tbme.2016.2555481.

Wang, J., KUO, T.Y., Li, L., Zeller, A., 2020a. Assessing and restoring reproducibility of jupyter notebooks, in: 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 138–149.

Wang, J., Kuo, T.y., Li, L., Zeller, A., 2020b. Restoring reproducibility of jupyter notebooks, in: 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), pp. 288–289.

Wattanakriengkrai, S., Chinthanet, B., Hata, H., Kula, R.G., Treude, C., Guo, J., Matsumoto, K., 2022. Github repositories with links to academic papers: Public access, traceability, and evolution. Journal of

Systems and Software 183, 111117. doi:https://doi.org/10.1016/j.jss.2021.111117.

White, L., Togneri, R., Liu, W., Bennamoun, M., 2019. Datadeps.jl: Repeatable data setup for reproducible data science. Journal of Open Research Software 7, 33. doi:10.5334/jors.244.

Widder, D.G., Sunshine, J., Fickas, S., 2019. Barriers to reproducible scientific programming, in: 2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE. pp. 217–221.

Wilson, G., Aruliah, D.A., Brown, C.T., Chue Hong, N.P., Davis, M., Guy, R.T., Haddock, S.H.D., Huff, K.D., Mitchell, I.M., Plumbley, M.D., et al., 2014. Best practices for scientific computing. PLOS Biology 12, e1001745. doi:10.1371/journal.pbio.1001745.

Wilson, L., Colborne, A., Smit, M., 2017. Preparing data managers to support open ocean science: Required competencies, assessed gaps, and the role of experiential learning, in: 2017 IEEE International Conference on Big Data (Big Data), pp. 3984–3993. doi:10.1109/BigData.2017.8258412.

Wittek, K., Wittek, N., Lawton, J., Dohndorf, I., Weinert, A., Ionita, A., 2021. A blockchain-based approach to provenance and reproducibility in research workflows, in: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1–6. doi:10.1109/ICBC51069.2021.9461139.

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, Association for Computing Machinery, New York, NY, USA. doi:10.1145/2601248.2601268.

Wonsil, J., Boufford, N., Agrawal, P., Chen, C., Cui, T., Sivaram, A., Seltzer, M., 2023. Reproducibility as a service. Software: Practice and Experience 53, 1543–1571. doi:10.1002/spe.3202.

Wu, W., Zhang, H., Li, Z., 2011. Open social based collaborative science gateways, in: 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 554–559. doi:10.1109/CCGrid.2011.43.

Xavier, L., Ferreira, F.F., Brito, R., Valente, M.T., 2020. Beyond the code: Mining self-admitted technical debt in issue tracker systems. Proceedings of the 17th International Conference on Mining Software Repositories MSR 20, 10.

Xavier, L., Montandon, J.E., Valente, M.T.O., 2022. Comments or issues: Where to document technical debt? IEEE Software 39, 84–91. doi:10.1109/ms.2022.3170825.

Yu, X., Duffy, C.J., Rousseau, A.N., Bhatt, G., Pardo Álvarez, Á., Charron, D., 2016. Open science in practice: Learning integrated modeling of coupled surface-subsurface flow processes from scratch. Earth and Space Science 3, 190–206. doi:10.1002/2015ea000155.

Zazworka, N., Vetro', A., Izurieta, C., Wong, S., Cai, Y., Seaman, C., Shull, F., 2014. Comparing four approaches for technical debt identification. Software Quality Journal 22, 403–426.

Zenodo, . https://zenodo.org/.

Zhao, Y., Wang, Y., Wang, H., Yan, B., Shen, F., Peterson, K.J., Rocca, W.A., Sauver, J.S., Liu, H., 2018. Annotating cohort data elements with ohdsi common data model to promote research reproducibility, in: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1310–1317. doi:10.1109/BIBM.2018.8621269.

Zhu, Z., Chen, M., Qian, Z., Li, H., Wu, K., Ma, Z., Wen, Y., Yue, S., Lü, G., 2023. Documentation strategy for facilitating the reproducibility of geo-simulation experiments. Environmental Modelling Software 163, 105687. doi:https://doi.org/10.1016/j.envsoft.2023.105687.

Ziemann, M., Poulain, P., Bora, A., 2023. The five pillars of computational reproducibility: bioinformatics and beyond. Briefings in Bioinformatics 24. doi:10.1093/bib/bbad375.

Śliwerski, J., Zimmermann, T., Zeller, A., 2005. When do changes induce fixes?, in: Proceedings of the 2005 International Workshop on Mining Software Repositories, Association for Computing Machinery, New York, NY, USA. p. 1–5. URL: https://doi.org/10.1145/1083142.1083147, doi:10.1145/1083142.1083147.