SHORT: Evaluating Tools for Enhancing Reproducibility in Computational Scientific Experiments

Lázaro Costa
lazaro@fe.up.pt
University of Porto &
HASLab/INESC TEC, Portugal

Susana Barbosa
susana.a.barbosa@inesctec.pt
INESC TEC, Portugal

Jácome Cunha
jacome@fe.up.pt
University of Porto &
HASLab/INESC TEC, Portugal

ABSTRACT

Ensuring the reproducibility of computational scientific experiments is crucial for advancing research and fostering scientific integrity. However, achieving reproducibility poses significant challenges, particularly in the absence of appropriate software tools to help. This paper addresses this issue by comparing existing tools designed to assist researchers across various fields in achieving reproducibility in their work.

We were able to successfully run eight tools and execute them to reproduce three existing experiments from different domains. Our findings show the critical role of technical choices in shaping the capabilities of these tools for reproducibility efforts.

By evaluating these tools for replicating experiments, we contribute insights into the current landscape of reproducibility support in scientific research. Our analysis offers guidance for researchers seeking appropriate tools to enhance the reproducibility of their experiments, highlighting the importance of informed technical decisions in facilitating reproducibility across diverse domains.

CCS CONCEPTS

• **General and reference** → **Evaluation**; *Empirical studies.*

KEYWORDS

Reproducibility, Open Science, Empirical Evaluation

ACM Reference Format:
Lázaro Costa, Susana Barbosa, and Jácome Cunha. 2024. SHORT: Evaluating Tools for Enhancing Reproducibility in Computational Scientific Experiments. In *ACM Conference on Reproducibility and Replicability (ACM REP '24), June 18–20, 2024, Rennes, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3641525.3663623

1 INTRODUCTION

The pursuit of scientific knowledge hinges upon the ability to replicate and reproduce experimental results reliably [5]. Following the definition of the National Academies Report [39], "reproducibility refers to the computational ability to duplicate results using the same materials, data, methods, and/or analytical conditions" [8]. Reproducibility serves as a cornerstone of scientific inquiry, providing a means to validate findings, verify hypotheses, and build upon existing knowledge. In the realm of computational science, where data analysis [31] and modeling [3] play pivotal roles, ensuring the reproducibility of experiments is paramount [22]. However, the complexity of computational workflows, coupled with the rapid evolution of software technologies, has made achieving reproducibility a daunting task [14]. Without robust tools and methodologies to support reproducibility efforts, researchers face significant barriers in accurately documenting, sharing, and replicating computational experiments. Thus, the credibility and reliability of scientific research may be compromised, hindering the advancement of knowledge and innovation in various domains [37]. Therefore, addressing the challenges of reproducibility in computational science is not merely a technical endeavor but a fundamental requirement for fostering transparency and rigor in science [32].

In this paper, our focus lies on assessing existing tools designed to aid researchers from diverse fields in overcoming the hurdles associated with reproducibility. We were able to successfully run eight tools and utilize them to reproduce three existing experiments spanning different domains, thus providing empirical evidence of the performance of these tools (more in Section 3). By comparing and evaluating these tools, we seek to shed light on their technical capabilities and ease of use, providing valuable insights into their effectiveness in facilitating reproducibility [39]. We compare several features, such as the programming languages (PLs) supported or the capability to (automatically) detect and install dependencies (more in Section 3). Through our analysis, we conclude that the technical choices underlying each tool influence their efficacy in replicating experiments across various research areas (Section 4).

By conducting this comparative study, we endeavor to contribute to the ongoing discourse on reproducibility in scientific research. Our findings hold implications for researchers seeking suitable tools to bolster the reproducibility of their computational experiments, emphasizing the significance of informed technical decisions in enhancing scientific integrity in diverse fields.

2 METHODOLOGY

In this section, we detail the methodology to collect tools (Section 2.1) and the experiments used to compare them (Section 2.2).

2.1 Collection of Tools

To collect the existing reproducibility tools, we searched the four main scientific libraries: ACM DL [19], IEEE Xplore [25], Science Direct [17] and SpringerLink [45]. We used the search query "computational reproducibility" without any other limitation. Following that, we examined the title and abstract of each article.

Our focus is exclusively on tools for reproducing computational experiments. Thus, we exclude approaches with a broader scope,

ACM REP '24, June 18–20, 2024, Rennes, France

ACM ISBN 979-8-4007-0530-4/24/06
https://doi.org/10.1145/3641525.3663623

such as scientific workflows [33], and tools such as Kepler [34], VisTrails [20], YesWorkflow [24], popper [29], and WaveLab [7].

Moreover, our primary emphasis is on tools that can be used in any field. Thus, we exclude techniques tailored to specific fields such as machine learning (BEAT [2], DVC [15], Kaggle [30], OpenML [51], PapersWithCode [52] and Collective Mind (CM) [10]), artificial intelligence (Hugging Face [18]), or life sciences (Galaxy [21]).

Based on this initial set of papers, we used the snowball search technique to increase the number of works about reproducibility.

Based on the initial search and the snowballing, we found 18 tools: Binder [9], CARE [28], Code, Data, Environment (CDE) [23], Code Ocean[46], Encapsulator [40], FLINC [1], PARROT [49], Provenance-To-Use (PTU) [41], Prune [26], RenkuLab [42], Reprozip [47], reprozip-jupyter [43], ResearchCompendia [48], SciInc [53], Sciunit [50], SOLE [35], Umbrella [36], and Whole Tale [6].

We analyzed all the tools we could find online. However, some are not available for download, namely SOLE, SciInc, ResearchCompendia, and CARE. We contacted the authors of the corresponding papers, but we did not receive an answer, and thus those approaches were removed from our analysis.

Furthermore, some available approaches were impossible to install or use: Umbrella, PARROT, Prune, Encapsulator, and reprozip-jupyter. We published a reproducible package of this work at Zenodo [11], in which the "Failled" folder contains the dockerfile[1] we used to try to execute these tools. In addition, we have provided within each dockerfile the commands we used to run it. This allows others to verify our work, which is especially important in these failing cases. For CDE, the available repository link[2] was not available. Nevertheless, the PTU tool is built on top of CDE, where the authors extended the CDE tool for versioning and summarizing its provenance, and we were able to use PTU. For Umbrella, PARROT, and Prune approaches, we successfully installed the dependencies, but when we tried to use both approaches, an error was launched (*command not found* and *package not found*, respectively). In the case of Encapsulator, it was installed successfully. However, we could not use the approach, as an exception was launched during the execution. Regarding reprozip-jupyter, we could successfully install the Jupyter Notebook plugin, but when we tried to click on the trace button, an exception was launched. We contacted the authors of the papers on these tools, but we did not receive an answer, and thus, these approaches were not further considered.

The tools available and that we could use are: Binder, Code Ocean, FLINC, RenkuLab, Reprozip, Sciunit, and Whole Tale.

2.2 Experiments

To compare the different tools, we decided to use a set of existing experiments and try to reproduce each one with all the tools. For that, we collected all the published use case experiments in those approaches. From the works SciInc [53] and Sciunit [50] we collected three experiments: the Chicago Food Inspections Evaluation (FIE) [38], the Variable Infiltration Capacity (VIC) [4], and the Incremental Query Execution (IQE) [13]. FIE is an experiment written using the programming language R from the machine learning domain; IQE is written in Python from a domain we could not determine; and Lynne is a Jupyter Notebook that runs with Ipython Kernel from Earth and Space Science.

Table 1: Characterization of three experiments.

Experiment	FIE	IQE	Lynne
Programming Language	R	Python	Ipython
Project Size	141 MB	51.5 KB	7.64 MB
# Files	91	22	37
# Lines of Script Files	1975	188	3702

Since some tools only work with Jupyter Notebooks it was necessary to have an experiment of this type. We did not consider the Sciunit [44] examples because those are produced using the Sciunit tool, and the original experiment is unavailable. Thus, we collected a Jupyter Notebook from Zenodo using the string query "Jupyter Notebook", only considering repositories from "Software", choosing the "best match", and picking the first result.[3] The experiment we collected, which we term Lynne, is part of the work [16].

In the end, we had four experiments collected from the published works of the considered reproducibility approaches and a Jupyter Notebook example collected from Zenodo.

We downloaded the code and data of these experiments, however, after analyzing them, we verified that some files were missing in the VIC experiment. Thus, we did not consider this experiment. FIE, IQE, and Lynne are the experiments that we used to evaluate and characterize the selected approaches. In Table 1, we show the characterization of these experiments: the PL of the experiments, the size of the project, the number of files, and the number of lines in the script files of the original experiment.

3 RESULTS

This section presents the results of using each tool to reproduce each of the three experiments. Table 2 shows the reproduction of each experiment using each tool, and the package size (in MB) generated (Zip format) for each experiment. The labels are "Yes", "No" and "-", meaning "we could reproduce", "we could not reproduce" and "the platform does not support this type of experiment", respectively.

Reprozip, Sciunit, and PTU are approaches used through the command-line interface (CLI), so they cannot reproduce figures and tables of a Jupyter Notebook. FLINC is a Jupyter Notebook plugin that can only reproduce the Jupyter Notebook experiment.

We proposed a set of characteristics to classify each reproducibility platform, and we separated these characteristics into four groups: *i)* orthogonal characteristics, *ii)* experimental configuration, *iii)* reproducible procedure, and *iv)* interoperability/maintenance. These groups reflect the different stages of most computational experiments. The characteristics are based on the features described in the works presenting these tools.

The group "orthogonal characteristics" (C1-C5) includes:

C1 The type of the user interface;
C2 The operating systems where the tool runs;
C3 The supported PLs;

[1]We used Docker (http://docker.com/), a virtualization environment to run each tool.
[2]https://web.stanford.edu/~pgbovine/cde.html

[3]https://zenodo.org/search?q=Jupyternotebook&f=resource_type%3Asoftware&l=list&p=1&s=10&sort=bestmatch

Table 2: Reproduction of each experiment on each platform and the package size in MB, when applicable.

Experiment	FIE	IQE	Lynne
Binder	No (NA)	Yes (NA)	Yes (NA)
Code Ocean	Yes(144)	Yes(0.04)	Yes(8)
FLINC	-	-	Yes(91)
PTU	Yes(302)	Yes(5)	-
RenkuLab	Yes(291)	Yes(0.10)	Yes(10)
Reprozip	Yes(278)	Yes(6)	-
Sciunit	Yes(269)	Yes(17)	-
WholeTale	Yes(290)	Yes(0.53)	Yes(82)

C4 If the approach uses the provenance technique to track all input files and code with its dependencies;
C5 The environment used to execute the experiment: Containerization (CI); Self-Contained Packages (SCP); Application Virtualization (AV).

The "orthogonal characteristics" play a crucial role in broadly classifying the reproducibility platforms.

The second group, "experimental configuration" (C6-C11), includes characteristics to classify and define the experimental configuration procedure, such as:

C6 If it is possible to use a link to a remote repository to create the project and upload all the data/code of the remote repository to the reproducibility tool. Examples of data management repositories are GitHub (GH), GitLab (GL), Zenodo (Z), Figshare (F), HydroShare (HS), Dataverse (D), Gist (G), and DataONE (DO).
C7 If the tool integrates with a repository for data storage and management to save the newly created reproducibility project.
C8 If the (hierarchical) structure of files follows rigid rules;
C9 If it automatically detects the required dependencies;
C10 If the tool automatically installs the required dependencies;
C11 If it is possible to configure and save a file and parameters used in the experiment for future executions.

Emphasizing these criteria is crucial as they significantly impact the efficiency and reliability of reproducibility tools, ensuring a streamlined and standardized approach to handling experimental configurations.

The group "reproducible procedure" (C12-C16) includes all the characteristics of the procedure used to reproduce the experiment:

C12 If it supports the creation of a shareable and reproducible package with all the required information (code, data, dependencies) to reproduce the experiment;
C13 If only the required dependencies are shared to reduce the Zip package size;
C14 If it is possible to reproduce the experiment on an online platform;
C15 If it enforces the execution with the same PL version. Since RenkuLab, Whole Tale, and Code Ocean are development tools, to evaluate this functionality, we will consider developing the entire experiment on the platform itself.
C16 If it enforces the reproducibility of results. One way of allowing this, is by saving the result of the previous execution and compare with the next one. Non-deterministic experiments produce a different result with each run.

The "reproducible procedure" criteria is essential to characterize the creation of the reproducibility package.

The fourth group, "interoperability/maintenance" (C17-C19), includes characteristics such as:

C17 If the approach is interoperable across operating systems;
C18 If the approach is interoperable across platforms. For example, a reproducibility tool that builds a reproducible package that can be executed in another reproducibility tool;
C19 If the approach is currently maintained.

The "interoperability/maintenance" criteria comprise key characteristics vital for the approaches' sustainability and compatibility.

In Table 3, we classify all the tools based on these criteria.

4 DISCUSSION

The reproducibility of experiments varied across different tools. While some tools successfully reproduced all three experiments, others encountered limitations in certain scenarios. Notably, Code Ocean, RenkuLab, and WholeTable demonstrated high reproducibility across experiments, reproducing all three; all the others faced constraints in reproducing the experiments, at least in one case. Although Binder documents support R code, we were unable to use it to execute this type of experiment. We tried to execute the FIE experiment using different versions of R (4.0, 4.1, 4.3), but the environment building failed. We also tried to execute examples in R from the Binder website[4], but they also failed.

The technical choices behind each tool significantly influenced their reproducibility capabilities. Tools utilizing CLI (C1) such as Reprozip, Sciunit, and PTU demonstrated versatility in reproducing experiments, although with limitations in replicating figures and tables from Jupyter Notebooks. FLINC, as a Jupyter Notebook plugin, excelled in reproducing Jupyter Notebook experiments but encountered challenges in other scenarios.

Tools based on provenance, such as Reprozip, Sciunit, FLINC, and PTU, automatically detect (C9) and install (C10) dependencies. This facilitates the creation of a reproduction package and also its re-execution. On the other hand, these tools are limited to Linux OS (C2). Nevertheless, these kinds of tools tend to be useful, independently of the PL used, as they can cope with all PLs (C3).

Online tools, such as RenkuLab, WholeTale, Code Ocean, and Binder, can be used in any OS as they run on a browser. However, they require more input from the researchers when preparing the reproducibility package. Without exception, they are limited in the number of supported PLs. Moreover, the user has little control over the environment where the experiment is created and executed. For instance, we were not able to execute experiments written in R in the Binder platform as an error was raised when creating the environment, a process that we could not control.

The interoperability across operating systems (C17) and platforms (C18) varied among the approaches. While some platforms

[4]Example 1: https://mybinder.org/v2/gh/binder-examples/r/master?filepath=index.ipynb
Example 2: https://mybinder.org/v2/gh/pablobernabeu/Modality-switch-effects-emerge-early-and-increase-throughout-conceptual-processing/cd4ea149820fd48c9247191a4d5670c5fa34961d?urlpath=shiny/Shiny-app/

Table 3: Characteristics of reproducibility tools.

Charact. \ Tool	RenkuLab	WholeTale	Reprozip	Sciunit	Code Ocean	Binder	FLINC	PTU
C1	CLI, web	CLI, web	CLI, app	CLI	CLI, web	web	Jupyter plugin	CLI
C2	All	All	Linux	Linux	All	All	Linux	Linux
C3	R, Python, Julia, MATLAB	R, IPython, IRKernel, Julia, MATLAB	Unlimited PLs	Unlimited PLs	Python, R, Java, C, C++, Scala, Lua, MATLAB	IPython, IRKernel, IJulia	IR, IPython, (C, C++ and Fortran) Jupyter Kernels	Unlimited PLs
C4	No	No	Yes	Yes	No	No	Yes	Yes
C5	CI	CI	SCP	AV	CI	CI	AV	SCP
C6	GL, GH	GH, GL, Z, D, DO	No	No	GH	GH, GL, Z, F, HS, D, G	No	No
C7	Yes	Yes	No	No	Yes	No	No	No
C8	Yes	No	No	No	Yes	Yes	No	No
C9	No	No	Yes	Yes	No	No	Yes	Yes
C10	No	Yes	Yes	Yes	No	No	Yes	Yes
C11	No	No	Yes	Yes	Yes	No	No	Yes
C12	No	Yes	Yes	Yes	Yes	No	Yes	Yes
C13	No	No	Yes	Yes	No	No	Yes	Yes
C14	Yes	Yes	No	Yes	Yes	Yes	No	No
C15	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
C16	No	No	No	No	No	No	Yes	No
C17	Yes	Yes	Yes	No	Yes	Yes	No	No
C18	No	No	Yes	Yes	Yes	No	No	No
C19	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

demonstrated versatility in reproducibility across different environments, others exhibited limitations in cross-platform compatibility. Maintenance considerations (C19) emerged as a crucial factor influencing the sustainability and usability of reproducibility platforms.

5 THREATS TO VALIDITY

A limitation of our study stems from the challenge of identifying all relevant reproducibility tools. To mitigate this threat, we conducted extensive searches in the largest libraries and utilized snowballing.

The scarcity of experiments available in the papers analyzed is also a threat to the robustness and generalizability of our conclusions. However, we used all existing experiments and added another from Zenodo, a highly-used science repository.

6 RELATED WORK

Despite the progress in existing tools for facilitating reproducibility, numerous authors struggle to replicate their results even after a year. Ivie et al. [27] address technical and social barriers in reproducibility, considering the technical challenges of reproducing from single commands to complex workflows. However, in their work, they do not evaluate or compare existing approaches or tools.

Sciunit [50] used two experiments, VIC and FIE, to compare the efficiency of their approach against PTU in terms of the time needed to execute, the creation of the reproducibility package, and its reproducibility. Their comparison is very specific and with just a single tool, while in our work we evaluate all the existing tools.

Davis et al. [12] interviewed 26 researchers to understand the challenges researchers face when conducting programmer user studies. Executing user studies can pose challenges, and researchers might encounter difficulties in devising effective strategies to overcome these hurdles, leading to a tendency to avoid engaging in user studies. While this is not a reproducibility problem, it is related as the creation and (re-)execution of an experiment environment is also one of the challenges raised by several researchers.

7 CONCLUSION

The comparative analysis of reproducibility tools underscores the importance of informed technical choices and platform characteristics in facilitating reproducibility efforts. These findings provide valuable insights for researchers seeking suitable tools to enhance the reproducibility of their computational experiments, guiding future developments in reproducibility platforms and methodologies.

In future work, we intend to make the study broader by including more experiments, considering domain-specific approaches, and broader approaches such as scientific workflows.

ACKNOWLEDGMENTS

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project LA/P/0063/2020. DOI 10.54499/LA/P/0063/2020 | https://doi.org/10.54499/LA/P/0063/2020. The first author was also supported by the doctoral Grant SFRH/BD/1513 66/2021 financed by Portuguese Foundation for Science and Technology.

REFERENCES

[1] Raza Ahmad, Naga Nithin Manne, and Tanu Malik. 2022. Reproducible Notebook Containers using Application Virtualization. In *2022 IEEE 18th International*

Conference on e-Science (e-Science). 1–10. https://doi.org/10.1109/eScience55777.2022.00015

[2] André Anjos, Laurent El Shafey, and Sébastien Marcel. 2017. Beat: An open-science web platform. In *Thirty-fourth International Conference on Machine Learning*.

[3] Frank T. Bergmann, Richard Adams, Stuart Moodie, Jonathan Cooper, Mihai Glont, Martin Golebiewski, Michael Hucka, Camille Laibe, Andrew K. Miller, David P. Nickerson, Brett G. Olivier, Nicolas Rodriguez, Herbert M. Sauro, Martin Scharm, Stian Soiland-Reyes, Dagmar Waltemath, Florent Yvon, and Nicolas Le Novère. 2014. COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics* 15, 1 (14 Dec 2014), 369. https://doi.org/10.1186/s12859-014-0369-z

[4] Mirza M. Billah, Jonathan L. Goodall, Ujjwal Narayan, Bakinam T. Essawy, Venkat Lakshmi, Arcot Rajasekar, and Reagan W. Moore. 2016. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. *Environmental Modelling & Software* 78 (2016), 31–39. https://doi.org/10.1016/j.envsoft.2015.12.010

[5] Christine L Borgman. 2012. The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology* 63, 6 (2012), 1059–1078.

[6] Adam Brinckman, Kyle Chard, Niall Gaffney, Mihael Hategan, Matthew B. Jones, Kacper Kowalik, Sivakumar Kulasekaran, Bertram Ludäscher, Bryce D. Mecum, Jarek Nabrzyski, Victoria Stodden, Ian J. Taylor, Matthew J. Turk, and Kandace Turner. 2019. Computing environments for reproducibility: Capturing the "Whole Tale". *Future Generation Computer Systems* 94 (2019), 854–867. https://doi.org/10.1016/j.future.2017.12.029

[7] Jonathan B. Buckheit and David L. Donoho. 1995. *WaveLab and Reproducible Research*. Springer New York, New York, NY, 55–81. https://doi.org/10.1007/978-1-4612-2544-7_5

[8] Rosemary Bush, Andrea Dutton, Michael Evans, Rich Loft, and Gavin A. Schmidt. 2020. Perspectives on Data Reproducibility and Replicability in Paleoclimate and Climate Science. *Harvard Data Science Review* 2, 4 (dec 16 2020). https://hdsr.mitpress.mit.edu/pub/dijwtzza.

[9] Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroff, M Pacer, Yuvi Panda, Fernando Perez, Benjamin Ragan Kelley, and Carol Willing. 2018. Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. In *Proceedings of the 17th Python in Science Conference*, Fatih Akici, David Lippa, Dillon Niederhut, and M Pacer (Eds.). SciPy, Austin, Texas, 113–120. https://doi.org/10.25080/Majora-4af1f417-011

[10] ML Commons. 2024. *Collective Mind (CM)*. https://github.com/mlcommons/ck [Online; accessed April-2024].

[11] Lázaro Costa, Jácome Cunha, and Susana Barbosa. 2024. Reproducible package of Conference on Reproducibility and Replicability 2024. https://doi.org/10.5281/zenodo.10651131

[12] Matthew C. Davis, Emad Aghayi, Thomas D. Latoza, Xiaoyin Wang, Brad A. Myers, and Joshua Sunshine. 2023. What's (Not) Working in Programmer User Studies? *ACM Trans. Softw. Eng. Methodol.* 32, 5, Article 120 (jul 2023), 32 pages. https://doi.org/10.1145/3587157

[13] DBGroup. 2019. *Incremental Query Execution*. https://bitbucket.org/TonHai/iqe/src/master/ [Online; accessed February-2024].

[14] Paolo Di Tommaso, Maria Chatzou, Evan W. Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. 2017. Nextflow enables reproducible computational workflows. *Nature Biotechnology* 35, 4 (01 Apr 2017), 316–319. https://doi.org/10.1038/nbt.3820

[15] Data Version Control (DVC). 2024. *DVC - Data Version Control*. https://dvc.org/ [Online; accessed February-2024].

[16] Lynne J. Elkins and Marc Spiegelman. 2021. *pyUserCalc: A revised Jupyter notebook calculator for uranium-series disequilibria in basalts*. https://doi.org/10.5281/zenodo.5598074

[17] Elsevier. 2024. *ScienceDirect*. https://www.sciencedirect.com/ [Online; accessed February-2024].

[18] Hugging Face. 2024. *Hugging Face*. https://huggingface.co/ [Online; accessed February-2024].

[19] Association for Computing Machinery. 2024. *ACM Digital Library*. https://dl.acm.org/ [Online; accessed February-2024].

[20] Juliana Freire and Claudio T. Silva. 2012. Making Computations and Publications Reproducible with VisTrails. *Computing in Science & Engineering* 14, 4 (2012), 18–25. https://doi.org/10.1109/MCSE.2012.76

[21] Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 11, 8 (25 Aug 2010), R86. https://doi.org/10.1186/gb-2010-11-8-r86

[22] Odd Erik Gundersen. 2021. The fundamental principles of reproducibility. *Philosophical Transactions of the Royal Society A* 379, 2197 (2021), 20200210.

[23] Philip J. Guo and Dawson Engler. 2011. CDE: Using System Call Interposition to Automatically Create Portable Software Packages. In *2011 USENIX Annual Technical Conference (USENIX ATC 11)*. USENIX Association, Portland, OR. https://www.usenix.org/conference/usenixatc11/cde-using-system-call-interposition-automatically-create-portable-software

[24] James Hanken, Timothy McPhillips, Tianhong Song, Tyler Kolisnik, Steven Aulenbach, Khalid Belhajjame, R Kyle Bocinsky, Yang Cao, James Cheney, Fernando Chirigati, Saumen Dey, Juliana Freire, Christopher Jones, Keith Kintigh, Tim Kohler, David Koop, James Macklin, Paolo Misser, Mark Schildhauer, and Bertram Ludäscher. 2015. YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts. *International Journal of Digital Curation* 10 (02 2015), 298–313. https://doi.org/10.2218/ijdc.v10i1.370

[25] IEEE. 2024. *IEEE Explore*. https://ieeexplore.ieee.org/Xplore/home.jsp [Online; accessed February-2024].

[26] Peter Ivie and Douglas Thain. 2016. PRUNE: A preserving run environment for reproducible scientific computing. In *2016 IEEE 12th International Conference on e-Science (e-Science)*. 61–70. https://doi.org/10.1109/eScience.2016.7870886

[27] Peter Ivie and Douglas Thain. 2018. Reproducibility in Scientific Computing. *ACM Comput. Surv.* 51, 3, Article 63 (jul 2018), 36 pages. https://doi.org/10.1145/3186266

[28] Yves Janin, Cédric Vincent, and Rémi Duraffort. 2014. CARE, the Comprehensive Archiver for Reproducible Execution. In *Proceedings of the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering*. ACM, Article 1, 7 pages. https://doi.org/10.1145/2618137.2618138

[29] Ivo Jimenez, Michael Sevilla, Noah Watkins, Carlos Maltzahn, Jay Lofstead, Kathryn Mohror, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. 2017. The popper convention: Making reproducible systems evaluation practical. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 1561–1570.

[30] Kaggle. 2024. *Your Machine Learning and Data Science Community*. https://www.kaggle.com/ [Online; accessed February-2024],.

[31] Barbara A Kitchenham, Shari Lawrence Pfleeger, Lesley M Pickard, Peter W Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on software engineering* 28, 8 (2002), 721–734.

[32] Joseph Ledet, Alejandro Teran-Somohano, Zachary Butcher, Levent Yilmaz, Alice E Smith, Halit Oğuztüzün, Orçun Dayıbaş, and Bilge Kaan Görür. 2014. Toward model-driven engineering principles and practices for model replicability and experiment reproducibility. In *Proceedings of the symposium on theory of modeling & simulation-DEVS integrative*. 1–8.

[33] Bertram Ludäscher, Shawn Bowers, and Timothy McPhillips. 2009. *Scientific Workflows*. Springer US, Boston, MA, 2507–2511. https://doi.org/10.1007/978-0-387-39940-9_1471

[34] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. 2006. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 10 (2006), 1039–1065. https://doi.org/10.1002/cpe.994 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.994

[35] Tanu Malik, Quan Pham, Ian T Foster, F Leisch, and RD Peng. 2014. SOLE: towards descriptive and interactive publications. *Implementing reproducible research* 33 (2014).

[36] Haiyan Meng and Douglas Thain. 2015. Umbrella: A Portable Environment Creator for Reproducible Computing on Clusters, Clouds, and Grids. *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing* (2015). https://api.semanticscholar.org/CorpusID:8303136

[37] Viswanath Nandigam, Kai Lin, Manu Shantharam, Scott Sakai, and Subhashini Sivagnanam. 2020. Research Workflows - Towards reproducible science via detailed provenance tracking in Open Science Chain. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '20)*. Association for Computing Machinery, New York, NY, USA, 484–486. https://doi.org/10.1145/3311790.3399619

[38] City of Chicago. 2016. *Food Inspection Evaluation Predictions - Source Code*. https://github.com/Chicago/food-inspections-evaluation [Online; accessed February-2024].

[39] National Academies of Sciences, Policy, Global Affairs, Board on Research Data, Information, Division on Engineering, Physical Sciences, Committee on Applied, Theoretical Statistics, Board on Mathematical Sciences, et al. 2019. *Reproducibility and replicability in science*. National Academies Press.

[40] Thomas Pasquier, Matthew Lau, Xueyuan Han, Elizabeth Fong, Barbara S. Lerner, Emery R. Boose, Mercè Crosas, Aaron M. Ellison, and Margo Seltzer. 2018. Sharing and Preserving Computational Analyses for Posterity with encapsulator. *Computing in Science & Engineering* 20, 4 (2018), 111–124. https://doi.org/10.1109/MCSE.2018.042781334

[41] Quan Pham, Tanu Malik, and Ian Foster. 2013. Using provenance for repeatability. In *Proceedings of the 5th USENIX Conference on Theory and Practice of Provenance* (Lombard, IL) *(TaPP'13)*. USENIX Association, USA, 2.

[42] Chandrasekhar Ramakrishnan, Michele Volpi, Fernando Perez-Cruz, Lilian Gasser, Firat Ozdemir, Patrick Paitz, Mohammad Alisafaee, Philipp Fischer, Ralf Grubenmann, Eliza Jean Harris, et al. 2023. Renku: a platform for sustainable data science. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets*

and Benchmarks Track.

[43] ReproZip. 2016. *Making Jupyter Notebooks Reproducible with ReproZip*. https://docs.reprozip.org/en/1.x/jupyter.html [Online; accessed April-2024].

[44] sciunit Group. 2024. *Sciunit*. https://sciunit.run/examples/ [Online; accessed February-2024].

[45] Springer. 2024. *Springer Link*. https://link.springer.com/ [Online; accessed February-2024].

[46] Thomas Staubitz, Hauke Klement, Ralf Teusner, Jan Renz, and Christoph Meinel. 2016. CodeOcean - A versatile platform for practical programming excercises in online environments. In *2016 IEEE Global Engineering Education Conference (EDUCON)*. 314–323. https://doi.org/10.1109/EDUCON.2016.7474573

[47] Vicky Steeves, Rémi Rampin, and Fernando Chirigati. 2020. Reproducibility, preservation, and access to research with ReproZip and ReproServer. *IASSIST Quarterly* 44, 1-2 (Jun. 2020), 1–11. https://doi.org/10.29173/iq969

[48] Victoria Stodden, Sheila Miguez, and Jennifer Seiler. 2015. ResearchCompendia.org: Cyberinfrastructure for Reproducibility and Collaboration in Computational Science. *Computing in Science and Engg.* 17, 1 (jan 2015), 12–19. https://doi.org/10.1109/MCSE.2015.18

[49] Douglas Thain and Miron Livny. 2005. Parrot: An application environment for data-intensive computing. *Scalable Computing: Practice and Experience* 6, 3 (2005), 9–18.

[50] Dai Hai Ton That, Gabriel Fils, Zhihao Yuan, and Tanu Malik. 2017. Sciunits: Reusable Research Objects. In *2017 IEEE 13th International Conference on e-Science (e-Science)*. 374–383. https://doi.org/10.1109/eScience.2017.51

[51] Jan N. van Rijn, Bernd Bischl, Luis Torgo, Bo Gao, Venkatesh Umaashankar, Simon Fischer, Patrick Winter, Bernd Wiswedel, Michael R. Berthold, and Joaquin Vanschoren. 2013. OpenML: A Collaborative Science Platform. In *Machine Learning and Knowledge Discovery in Databases*, Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 645–649.

[52] Papers with Code. 2024. *Papers With Code*. https://paperswithcode.com/ [Online; accessed February-2024].

[53] Andrew Youngdahl, Dai-Hai Ton-That, and Tanu Malik. 2019. SciInc: A Container Runtime for Incremental Recomputation. In *2019 15th International Conference on eScience (eScience)*. 291–300. https://doi.org/10.1109/eScience.2019.00040