Reproducibility, preservation, and access to research with ReproZip and ReproServer

Vicky Steeves[1], Rémi Rampin[2], Fernando Chirigati[3]

Abstract

The adoption of reproducibility remains low, despite incentives becoming increasingly common in different domains, conferences, and journals. The truth is, reproducibility is technically difficult to achieve due to the complexities of computational environments. To address these technical challenges, we created ReproZip, an open-source tool that automatically packs research along with all the necessary information to reproduce it, including data files, software, OS version, and environment variables. Everything is then bundled into an `rpz` file, which users can use to reproduce the work with ReproZip and a suitable unpacker (e.g.: using Vagrant or Docker). The `rpz` file is general and contains rich metadata: more unpackers can be added as needed, better guaranteeing long-term preservation. However, installing the unpackers can still be burdensome for secondary users of ReproZip bundles. In this paper, we will discuss how ReproZip and our new tool, ReproServer, can be used together to facilitate access to well-preserved, reproducible work. ReproServer is a web application that allows users to upload or provide a link to a ReproZip bundle, and then interact with/reproduce the contents from the comfort of their browser. Users are then provided a persistent link to the unpacked work on ReproServer which they can share with reviewers or colleagues.

Keywords

Research reproducibility, data management, data librarianship, digital preservation, access, reuse

1. Introduction

Libraries, museums, and archives are tasked with saving the world's knowledge and culture, and keeping it re-usable throughout time. With the advent of born-digital works, attaining this goal becomes more complicated. We see digital stewardship as the long-term active management of digital materials and their supporting information including, but not limited to, metadata, documentation, and provenance. The goal is preservation and unencumbered access in the long-term. The proliferation of bespoke, unique processes for conducting research and the technical requirements to access the outcomes are ever-evolving, often taking place in proprietary environments that have been difficult for digital stewards to preserve (Chassanoff and Altman, 2019).

Digital preservation practitioners are constantly looking for improved methods to reliably preserve the works of their communities, as well as more efficient and scalable means of providing access to materials. In the case of those working with research communities, this often entails verifying that research data can be opened in the future by forward-migrating formats, and ensuring that there is enough documentation about these datasets to be useful to other researchers (Johnston, 2014). This has since evolved to include preserving software alongside data, especially the project-specific software often developed by researchers for their work (Rios et al., 2017). Similarly, capturing and

 Steeves, Vicky; Rampin, Rémi; Chirigati, Fernando (2020) Reproducibility, preservation, and access to research with ReproZip and ReproServer, IASSIST Quarterly 44(1-2), pp. 1-11. DOI: https://doi.org/10.29173/iq969

describing these software dependencies and scripting workflows have been a growing necessity for researchers seeking to make their work reproducible. Reproducible research implies that when another researcher tries to rerun the same analysis—using the same data and code as the original creator— they will achieve the same result (Goodman et al., 2016). This is extremely difficult, however. Dependencies of a research process are not always known in full detail by the original researcher, much less the digital steward tasked with the long-term safety of the work (Marwick, 2015). Archivists and librarians who endeavor to accession and preserve reproducible research materials therefore must have a corresponding toolkit up to the task.

But why is it so important for reproducibility and archival access to access materials in the original computational environment? Given how much modern research practices rely on unique toolkits, the output from any analysis is highly dependent on the actual software in which the research happens (Pawlik et al., 2019). Results of research change depending on versions of software, on operating system changes, and other custom configurations (Gronenschild et al., 2012). Ensuring that the research process includes documentation about dependencies is important for the integrity of the scholarly record. Likewise in archival science, the idea of authenticity of access to materials remains an important facet of both analog and digital work. Authenticity refers to the idea that materials should be viewed/rendered exactly as they would have at the time of their creation, and has led to large-scale efforts in emulation services within digital archives (Espenschied and Lialina, n.d.). Just as web archivists often need an emulated browser to view WARC files (i.e. Webrecorder Player[4]), research archivists often need emulated computational environment to engage with original research materials and to confirm they are reproducible (Rhizome, 2015).

We envision an ecosystem of open infrastructure that makes reproducibility easy for both the originating researcher and those seeking to view, reproduce, and extend the original work. Ideally, a researcher should have access toa self-contained, distributable, and reproducible bundle of their work using as few commands as possible. Other researchers/reviewers should be able to reproduce and reuse experiments with a few mouse clicks, and without having to deal with all the complex chains of dependencies required to run the corresponding computational processes.

To that end, we have developed two open-source tools: ReproZip[5] and ReproServer[6]. ReproZip is a tool that can automatically and transparently capture a computational experiment, creating a single, distributable bundle, without the researcher having to specify all of the required dependencies. Other researchers can then use ReproZip to automatically unpack this bundle and set up the environment in order to reproduce the research, even if the operating systems are different (Chirigati et al., 2016). Our second tool, ReproServer, is a web application that provides easier access to ReproZip bundles: instead of having to install different software together with ReproZip to set up and reproduce a bundle, ReproServer allows researchers to reproduce and reuse other people's research from the comfort of their browser (Rampin et al., 2018a).

There is existing work in providing authentic access to materials at scale, though with varied results for reproducibility. Binder[7] is one such system. Binder can provide access to R scripts and Jupyter Notebooks directly from a URL, DOI, or GitLab or GitHub repository, allowing users to interact with these notebooks in a browser-based live environment. However, this is a solution specific to Jupyter

Notebooks and R scripts, and users must provide a file that describes the dependencies for the notebooks, i.e., users must manually capture such dependencies, which can be a burdensome task given there are chains of dependencies that can be unknowable to the user without proper documentation (Jupyter et al., 2018). Emulation as a Service Infrastructure[8] (EaaSI) is another system that provides access to operating systems in-browser and at scale. EaaSI excels at emulation and provides the user access to older systems. That said, users' emulated environment is a sandbox in which to do work, and while it can certainly help in efforts to recreate research projects from the past, users still must manually declare their environment (Cochrane, 2018). This presents similar problems as with Binder.

We posit that core digital stewardship goals intersect with computational reproducibility, and that applying tools for computational reproducibility in an archival context can help ensure that materials preserved as key parts of the scholarly record can be verified, reused, reproduced, and accessed with integrity in the future. In this paper, we will describe how ReproZip and ReproServer can be used in tandem to: 1) capture a research process/workflow with all its dependencies and components in a preservation-ready format, the ReproZip bundle, and 2) give patrons easy access to view and execute the contents of those bundles in the original computational environments, via their browser, with ReproServer.

2. Preservation with ReproZip

2.1 ReproZip Infrastructure

ReproZip is an open-source tool that automatically captures all the dependencies of a research product or application originally run in a Linux environment, and creates a single, distributable bundle that can be used to reproduce the entire experiment in another environment (e.g., on Linux, Windows, or macOS) (Chirigati et al., 2016). The tool works in two steps:

1. *Packing*. Given a research application that runs on a Linux OS, ReproZip automatically and transparently traces all of the system calls related to the execution of that application. It captures all dependencies at the OS level, including software, data files, databases, libraries, environment variables, parameters, and OS and hardware information. Using this information (which can optionally be customized by the user), ReproZip creates a compendium for it: an rpz file containing the dependencies. The packing step must be done via the command line; future work includes a GUI and support for non-Linux environments.

Unpacking. Given an rpz file, other users can use ReproZip to set up the application in their environment, even if their OS is different than the one used for the creation of the experiment. Users can choose among different *unpackers* (e.g.: Vagrant or Docker for an isolated reproduction). ReproZip then automatically sets up the environment and allows users to rerun the application. The unpacking step can be done either via the command line or using the provided GUI.

For instance, suppose that a user, Alice, has a Python script named prediction.py that represents her research. The script predicts the values of handwritten digits from an input image (using a variety of software packages) and outputs a new image with predictions. To pack this script with ReproZip, she

first prepends reprozip trace to the execution of her script (via the command line): reprozip trace python prediction.py. ReproZip then detects all the necessary dependencies in order to rerun this script in the future. She then can use reprozip pack to create an rpz bundle for it. If she shares this bundle with another researcher, Bob, he can use the reprounzip component to set up and reproduce the script. Because he works on macOS and the script was originally created on a Linux OS, Bob chooses the unpacker reprounzip-docker to let ReproZip automatically set up the experiment using Docker. ReproZip takes care of all the unpacking details, and Bob is able to reproduce the experiment even though he is not familiar with Docker.

Provided that ReproZiphas access to the original command-line execution, it can trace and reproduce a variety of research products, including Python/R/Julia/any language scripts, applications based on compiled source code (e.g.: JAR files, C/C++ binaries), Jupyter notebooks, interactive applications, GUI applications, and more involved scenarios such as client-server applications (e.g.: databases). ReproZip has supported research in a variety of disciplines, including neuroscience, physics, computer vision, computational humanities, data science, data visualization, and data journalism. Some users have contributed explanations for their research with steps to create the ReproZip bundle and to access the contents, collected on the ReproZip Examples website[9]. ReproZip is currently being used by ReproMan[10], a tool that helps create and manage computing environments in the neuroimaging domain, and by CoRR[11], a web platform from the National Institute of Standards and Technology (NIST) that stores and manages data from computational and experimental materials science. ReproZip has also been used to support reproducibility evaluation for research papers published in different venues, including: the Information Systems Journal[12]; ACM SIGMOD[13]; and conferences that follow the Artifact Evaluation Process guidelines[14].

2.2 Archival Benefits of ReproZip

Among the many benefits of ReproZip, *digital preservation* is the most relevant. Using ReproZip, digital archivists and librarians can meticulously trace computing environments in which research takes place, from data files and applications to complicated chains of software dependencies. This process allows patrons to reproduce environments over time (Steeves et al., 2018). The rpz file generated by ReproZip is a preservation-ready object for several reasons:

- *Flexibility.* The file is generalized, completely agnostic to the unpacking technology being used: it can be rerun using many different unpacker plugins through ReproZip. In addition, thanks to ReproZip's plugin model, unpackers can be added and removed as systems become more or less popular or usable, allowing long-term preservation. For instance, if Docker becomes obsolete, then a ReproZip unpacker can be written for another containerization system and the ReproZip packages remain usable.

Completeness. The rpz file contains all the necessary files to reproduce and reuse the packed research. In addition, it provides a configuration file that is machine-readable and lists all the information about the computational environment and dependencies necessary to recreate it If in the future there are no containers or virtual machines, the archivist/librarian can still use the robust technical and administrative metadata from the bundle.

Bundle Size. Because ReproZip captures only the necessary files for reproduction (i.e., the minimal set of files), the rpz file is often very small compared to the entire computational environment in which the research runs. This makes it easier to store and share these bundles.

As an example of how ReproZip can be used to streamline digital preservation, the tool was recently used as the core component in a recent project, *Saving Data Journalism*[15]. This Institute of Museum and Library Service[16] (IMLS) funded endeavor is aimed at preserving interactive news applications, i.e., dynamic websites that allows readers to fully interact with the stories (e.g.: Dollar for Docs[17]). Due to its technological complexity, dynamic web content cannot currently be fully archived or preserved by libraries, newsrooms, or cultural institutions. As part of the project, we built a prototype, called ReproZip-Web[18], that leverages ReproZip and Webrecorder to correctly preserve this type of application (Boss et al., 2019). Preliminary tests with the prototype showcased its usefulness in preserving and replaying news applications; a demonstration video is available online[19].

While ReproZip significantly reduces the barrier to reproducing and preserving different applications, to unpack and rerun rpz bundles, users must still download the reprounzip component and their unpacker of choice (e.g.: reprounzip-docker to use Docker[20], and reprounzip-vagrant to use Vagrant[21]), together with the software to be used by the unpacker (e.g.: Docker or Vagrant and VirtualBox[22]). Even if this is only required once, having to download and set up these tools can prove to be a heavy burden (and intrusive). Therefore, there is still a barrier to accessing rpz files, as one might not have the required software to run ReproZip and rerun the research. To address this limitation, we started implementing a web application called ReproServer, detailed next.

3. ReproServer — Where Preservation Meets Access

3.1 ReproServer Infrastructure

ReproServer is a cloud-native application designed to run ReproZip research bundles from the browser. It builds on one of ReproZip's unpacker plugins, reprounzip-docker, and provides a web interface through which users can interact with bundled research and software in their original environment (Rampin et al., 2018b). The system is functional and we maintain a public deployment at https://server.reprozip.org/.

When provided an rpz file, either via direct upload from the user's computer or via a link to a supported data repository, ReproServer builds a container of the preserved environment and allows the user to run the program with the option to use different parameters or input data. The program is then executed in a cluster, and the results are shown to the user with all the same interactions that they might have with the reprounzip component on their desktop, including the ability to download the output files and upload their own input files.

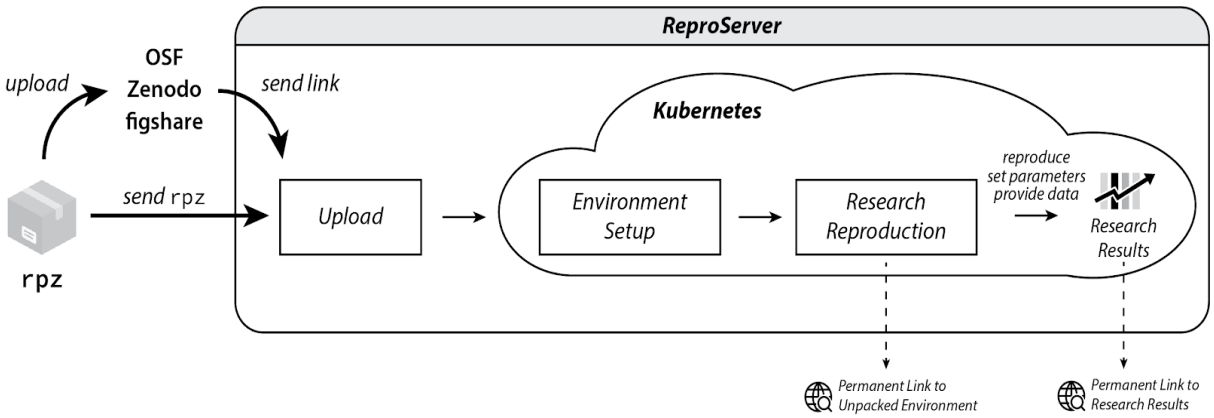

Figure 1. Overall architecture of ReproServer.

ReproServer is built on Docker and Kubernetes, which are common, well-supported technologies for the deployment of this type of applications. Kubernetes is an open-source container orchestration system, which can be deployed on many infrastructures, and it is supported by most cloud providers (e.g.: Google Cloud, Amazon Web Services, Microsoft Azure, Digital Ocean). It features automatic scaling, allowing for efficient medium-scale deployments accessible to the public.

ReproServer was designed with collaboration and interoperability in mind. We rely on existing third-party data repositories and we currently support the Open Science Framework[23], Zenodo[24], and Figshare[25]. ReproServer can use a reference to the file in the repository (e.g. file URL), which appears in the URL of the environment in ReproServer. Each execution of the unpacked research also gets a URL that can be sent to others to show the results and retrieve output files. In addition to batch experiments (i.e., non-interactive applications), ReproServer also supports interactive web-based experiments (e.g.: websites and Jupyter notebooks), providing the user with a persistent URL where they can access the program (which can be shared with others) while it is running. This URL is shareable with others so that they can easily access an unpacked work without having to rebuild it every time.

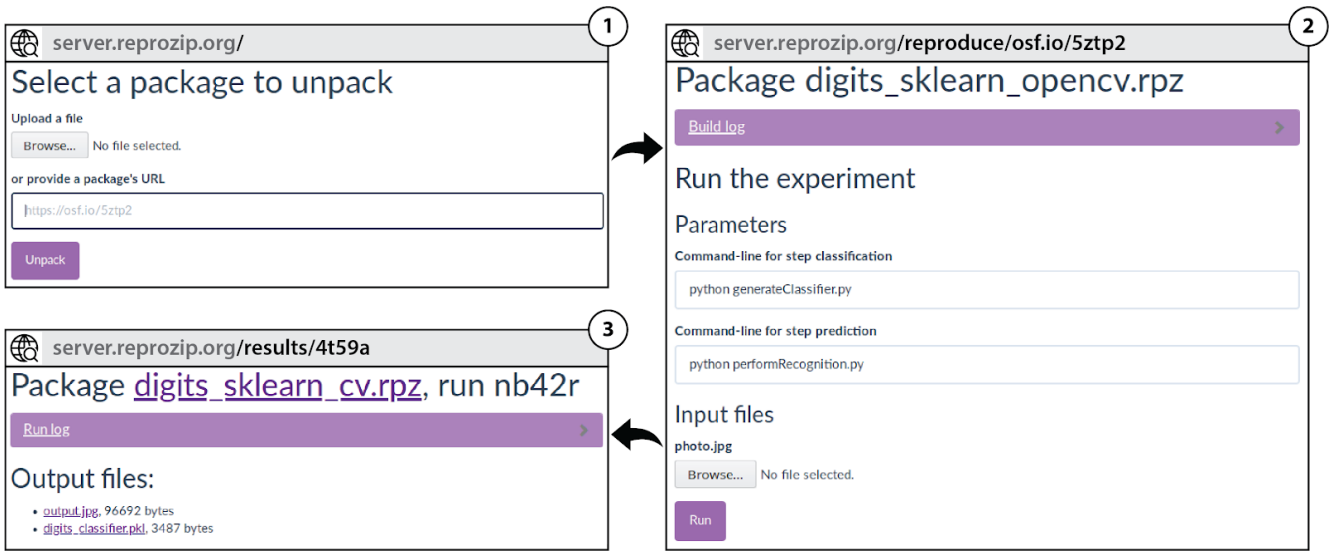

Figure 2. The workflow of using ReproServer to unpack and interact with an RPZ file hosted on the OSF.

 Steeves, Vicky; Rampin, Rémi; Chirigati, Fernando (2020) Reproducibility, preservation, and access to research with ReproZip and ReproServer, IASSIST Quarterly 44(1-2), pp. 1-11. DOI: https://doi.org/10.29173/iq969

For instance, suppose that Sarah is a reviewer on Alice's paper, and wants to look at her entire research process to verify the claims within the paper. She can use ReproServer to reproduce Alice's work, without having to install any additional software on her computer. She chooses to upload the rpz bundle to ReproServer and immediately reruns the prediction script, which returns consistent results. Given Alice's research is reproducible, she feels comfortable signing off on a positive review.

Sarah then decides to try the script with a few other images she often uses in her research because she finds the research that Alice has done is novel. She goes back to ReproServer, changes the input file of the execution, and reruns the script again for different image files. For one of Sarah's files, the prediction seems to be wrong. She then shares the link to the unpacked experiment with Alice so she can see and interact with these results also through ReproServer, and hopefully improve her model. Sarah would have neither been able to do that without the ease of access and use that goes into ReproServer, nor if Alice had not had been a great steward of her research and packed it for reproducibility.

3.2 Archival Access with ReproServer

Researchers and digital stewards alike can use ReproZip to create a compendium of work that is easily shareable, citable (if deposited in an institutional or subject-specific repository), and that they and the community-at-large can use. Accompanying that, ReproServer can act as a virtual, interactive reading room for those interested in exploring preserved reproducible research, for purposes such as studying the history of science (comparing workflows across time in a given niche, for instance), verifying results of research and building on it with in-browser interactions, and surely more ways that we have not envisioned — patrons are infinitely creative when approaching collections in libraries/archives.

For the librarian/archivist, the workflow for preserving these complex research processes and objects becomes easier to manage. They would straightforwardly trace and pack the given research with ReproZip, ingest into a repository or secure storage layer, and provide a button on the record page that links to ReproServer for easy access to patrons to allow them to fully reproduce the work. For archival researchers, interacting with primary source materials as authentically as possible is extremely important for understanding the complexities (Kim and Mennerich, 2015). If libraries/archives provide the primary source materials (the rpz bundle) in the collections via the web browser, archival researchers can view and interact with materials in their original environment, with detailed metadata and provenance information, which ensures high-fidelity, with respect to original functionality and utility.

Research bundles like ReproZip's rpz files can be hard to read at first blush -- in fact, there have been guides written such as *How to Read a Research Compendium* to educate even computationally-advanced patrons in understanding the materials captured in these packages (Nüst et al., 2018). By designing a simple interface with which to access and interact with these files, ReproServer can be an onramp for patrons who might want to explore different methods of research, but lack some of the computational literacy to fully understand how to read the bundle or use it on their own computer.

4. Future Work

To build on our current work with ReproServer, we would like to support interacting with more varied types of research processes from the browser. Right now, we only allow uploading new inputs, but we would like to, for instance, give users the chance to interact with the packed terminal applications via a terminal right on the webpage, or conversely interact with GUI applications from their browser. This would give greater freedom to those patrons who want more in-depth investigations of the unpacked works.

For patrons that use big data, we are also looking into integrating with High-Performance Computing (HPC) clusters and cloud providers directly, to allow the reproduction of research processes that are distributed, use large data, or need to be run on specific hardware (for example, specific GPUs for Artificial Intelligence research that might exist on one HPC cluster but not the current public ReproServer deployment). This means that patrons with credentials to specialized computing environments would be able to bridge ReproServer and their custom compute cluster, and use their own compute to reproduce work via ReproServer.

We also want to support a wider range of data repositories, especially those being supported and maintained by librarians and archivists at their institutions. ReproServer can currently obtain files from Zenodo, OSF, and Figshare, but it can be similarly integrated with more software by writing adapters for their APIs. We want to support all the repositories where researchers are likely to deposit their software, including the self-hostable options that might be used by institutions, such as Dataverse[26], DSpace[27], and Samvera[28].

Finally, a long-term goal of ReproZip is the support of more operating systems. We are investigating ways to capture and reproduce research on macOS and Microsoft Windows since those operating systems are prevalent in many research domains. We are also looking into creating Linux environments with the same dependencies that were used on the author's machine (for example, Python/R/Ruby packages are usually available for all operating systems), to alleviate the need to run virtual machines with alternative OSes (at substantial cost).

5. Conclusion

Ultimately, in a time when the initiatives of libraries and archives in particular are being cannibalized by for-profit and corporate entities, it's more important than ever to build out and support open infrastructure that can empower institutions to safely archive the work of their communities, and provide user-friendly means of accessing that work authentically. ReproZip and ReproServer were built with this in mind. There is no vendor lock-in, and no need to work within or port to commercial platforms for reproducibility. Any institution could instantiate ReproServer (with minimal computing infrastructure) and allow their users to reproduce the work held safely in their repositories. Likewise, ReproServer adds to the open-source ecosystem for authentically accessing research in the cloud. ReproZip and ReproServer increase open-source support for reproducibility in the long-term and ensure that researchers and the archivists/librarians they trust to steward their work will be able to safely and accurately maintain access to reproducible research materials.

Ultimately, ReproZip and ReproServer improves research by lowering the barriers to reproducibility, and the way in which the LIS community can preserve and make discoverable these research compendia. ReproServer builds on this important work by making access to these materials seamless; through an easily accessible web interface that allows for multiple ways of providing an rpz file (including integrations with repositories), multiple ways of interacting with the unpacked work, and open-sourcing this for use by anyone/any institution.

Acknowledgements

We would like to acknowledge Dr. Juliana Freire, the Principal Investigator of the ReproZip project, for her support in continuing to build ReproZip and now ReproServer. We would also like to thank Genevieve Milliken for providing feedback on this paper prior to submission. We would also like to acknowledge the support from the Gordon and Betty Moore Foundation as well as the Alfred P. Sloan Foundation. The Moore-Sloan Data Science Environment was vital to the development of ReproZip.

References

Boss, K.E., Steeves, V., Rampin, R., Chirigati, F., Hoffman, B., 2019. Saving Data Journalism: Using ReproZip-Web to Capture Dynamic Websites for Future Reuse (preprint). LIS Scholarship Archive. https://doi.org/10.31229/osf.io/khtdr

Chassanoff, A., Altman, M., 2019. Curation as "Interoperability With the Future": Preserving Scholarly Research Software in Academic Libraries. J. Assoc. Inf. Sci. Technol. 0. https://doi.org/10.1002/asi.24244

Chirigati, F., Rampin, R., Shasha, D., Freire, J., 2016. ReproZip: Computational Reproducibility With Ease, in: Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16. ACM, New York, NY, USA, pp. 2085–2088. https://doi.org/10.1145/2882903.2899401

Cochrane, E., 2018. Emulation as a Service Infrastructure (EaaSI). https://doi.org/None

Espenschied, D., Lialina, O., n.d. Authenticity/Access | One Terabyte of Kilobyte Age. URL https://blog.geocities.institute/archives/3214 (accessed 11.1.19).

Goodman, S.N., Fanelli, D., Ioannidis, J.P.A., 2016. What does research reproducibility mean? Sci. Transl. Med. 8, 341ps12-341ps12. https://doi.org/10.1126/scitranslmed.aaf5027

Gronenschild, E.H.B.M., Habets, P., Jacobs, H.I.L., Mengelers, R., Rozendaal, N., van Os, J., Marcelis, M., 2012. The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements. PLoS ONE 7, e38234. https://doi.org/10.1371/journal.pone.0038234

Johnston, L.R., 2014. A Workflow Model for Curating Research Data in the University of Minnesota Libraries: Report from the 2013 Data Curation Pilot (Report). University Digital of Minnesota Conservancy.

Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Holdgraf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., Willing, C., 2018. Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. Proc. 17th Python Sci. Conf. 113–120. https://doi.org/10.25080/Majora-4af1f417-011

Kim, J., Mennerich, D., 2015. Jeremy Blake's Time-Based Paintings: A Case Study – Electronic Media Review Four.
Marwick, B., 2015. How computers broke science – and what we can do to fix it [WWW Document]. The Conversation. URL http://theconversation.com/how-computers-broke-science-and-what-we-can-do-to-fix-it-49938 (accessed 3.27.17).
Nüst, D., Boettiger, C., Marwick, B., 2018. How to Read a Research Compendium. ArXiv180609525 Cs.
Pawlik, M., Hütter, T., Kocher, D., Mann, W., Augsten, N., 2019. A Link is not Enough – Reproducibility of Data. Datenbank-Spektrum. https://doi.org/10.1007/s13222-019-00317-8
Rampin, R., Chirigati, F., Steeves, V., Freire, J., 2018a. ReproServer: Making Reproducibility Easier and Less Intensive. ArXiv180801406 Cs.
Rampin, R., Chirigati, F., Steeves, V., Freire, J., 2018b. ReproServer: Making Reproducibility Easier and Less Intensive. ArXiv180801406 Cs.
Rhizome, 2015. Cyberspace, the old-fashioned way [WWW Document]. Rhizome. URL http://rhizome.org/editorial/2015/nov/30/oldweb-today/ (accessed 11.1.19).
Rios, F., Contaxis, N., Almas, B., Jabloner, P., Kelly, H., 2017. Exploring Curation-ready Software: Use Cases. URL http://www.softwarepreservationnetwork.org/exploring-curation-ready-software-use-cases/ (accessed 7.8.17).
Steeves, V., Rampin, R., Chirigati, F., 2018. Using ReproZip for Reproducibility and Library Services. IASSIST Q. 42, 14–14. https://doi.org/10.29173/iq18

End-notes

[1] Vicky Steeves is the Librarian for Research Data Management and Reproducibility, a dual appointment between New York University Division of Libraries and NYU Center for Data Science. She can be reached by email <vicky.steeves@nyu.edu> and her ORCID is 0000-0003-4298-168X.
[2] Rémi Rampin is a Research Engineer at the Tandon School of Engineering at New York University. He can be reached by email <remi.rampin@nyu.edu> and his ORCID is 0000-0002-0524-2282.
[3] Fernando Chirigati is a Postdoctoral Researcher at the Tandon School of Engineering at New York University. He can be reached by email <fchirigati@nyu.edu> and his ORCID is 0000-0002-9566-5835.
[4] https://webrecorder.io/
[5] https://reprozip.org
[6] https://server.reprozip.org
[7] https://mybinder.org/
[8] https://www.softwarepreservationnetwork.org/eaasi/
[9] https://examples.reprozip.org/
[10] https://github.com/ReproNim/reproman
[11] https://www.nist.gov/programs-projects/cloud-reproducible-records
[12] https://www.journals.elsevier.com/information-systems/
[13] http://db-reproducibility.seas.harvard.edu/
[14] https://www.artifact-eval.org/guidelines.html
[15] https://savingjournalism.reprozip.org/
[16] https://www.imls.gov/
[17] https://projects.propublica.org/docdollars/

[18] https://github.com/reprozip-news-apps/reprozip-web
[19] https://www.youtube.com/watch?v=_SFVMz-kNFo&list=PLjgZ3v4gFxpWc7xC9uKIyGY8A8rLgcvv5
[20] https://www.docker.com/
[21] https://www.vagrantup.com/
[22] https://www.virtualbox.org/
[23] https://osf.io/
[24] https://zenodo.org/
[25] https://figshare.com/
[26] https://dataverse.org/
[27] https://www.dspace.com/en/inc/home.cfm
[28] https://samvera.org/