Appendix 2: Tools for Computational Reproducibility

This section outlines some open, scholar-led software projects that are aimed at helping researchers make their work computationally reproducible. While there are proprietary tools for computational reproducibility, they are not widely available and this resource focuses on openly available tools as a matter of ethics. The options discussed here are all free and open source, grassroots initiatives from scholars who are deeply invested in openness and reproducible research. Konkol et. al (2020), provides a wider survey of tools for computational reproducibility geared towards publishing computational research, which is inclusive of proprietary software as well as some open platforms described below.

There are four classes of computational reproducibility tools that will be discussed in this section:

1. ***Containers***: lightweight, portable, virtual operating systems.
2. ***Web-based integrated development environments*** (IDEs): which provide code editing and execution and often have additional features for reproducibility.
3. ***Web-based replay systems***: support for computational replay of materials that are hosted in a different place from the system.
4. ***Packaging tools***: software that automatically captures dependencies & computational environment used at time of executing a computational pipeline.

Containers

The research community has been increasingly using and sharing containers in service of reproducibility. Containers are a popular way to create virtual operating systems, like sandboxes, separate from the physical infrastructure and native operating system.[1] Two popular container systems, Singularity and Docker, are especially popular for research reproducibility.

Docker was made to "pack, ship and run any application as a lightweight container," specifically with the advantage of working in most computational environments.[2] It is

[1] Scott Hogg, "Software Containers: Used More Frequently than Most Realize," *Network World*, May 26, 2014, https://www.networkworld.com/article/2226996/software-containers--used-more-frequently-than-most-realize.html.
[2] Docker Inc., "Docker," accessed August 13, 2020, https://www.docker.com/.

widely used in software development to deploy software in the cloud as well as ensure a common development environment amongst programmers. There are several other tools that will be described below that rely on Docker in the back-end to remain reproducible.

Singularity was made for high performance computing (HPC) work because of security considerations that both allow users full flexibility within the container and keep them from accessing parts of the HPC environment that administrators do not want users to access. Starting a Singularity container "swaps'' out the host operating system environment for one the user controls without having root access, and allows the user to run that application in its native environment.[3] Singularity containers can then be shared to allow others to work in the same computational environment.

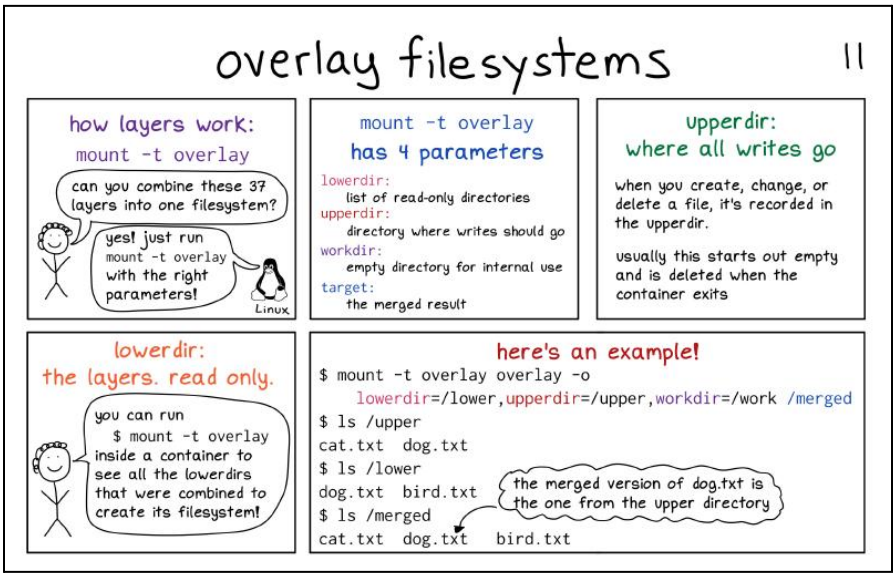
Containers, however, are best used for short-term reproducibility. There are several problems with their use for long-term sustainability. Containers have no idea of provenance or the computational pipeline used -- a container with code and data can be rendered virtually useless if not accompanied by extensive documentation about its inputs and workflow steps. In addition, learning how to use containers is also difficult as it is not always practical for researchers to create and use containers in their daily workflow.

Web-based Integrated Development Environments (IDEs)

An Integrated Development Environment (IDE) provides features for authoring, compiling, executing, and debugging code, as well as helpful functions like code completion, built-in support for version control, and syntax highlighting.[4] These are especially helpful for new programmers who benefit from the visual cues and prompts. IDEs can be either desktop or web-based applications.

The scholarly community has taken advantage of both containers and web-based IDEs to create a new type of this application geared for reproducible research. These systems often provide access to a coding environment in-browser, such as Jupyter notebooks or RStudio, or their own IDE.

[3] "Singularity," Sylabs.io, accessed August 13, 2020, https://sylabs.io/.
[4] "What Is an Integrated Development Environment (IDE)? - Definition from Techopedia," *Techopedia.com*, January 11, 2017, http://www.techopedia.com/definition/26860/integrated-development-environment-ide.

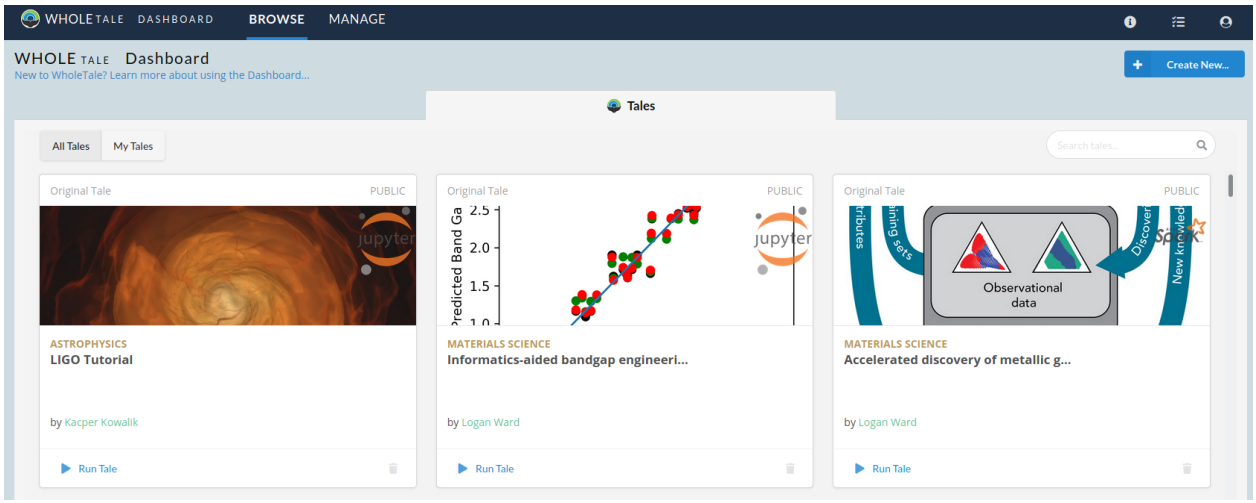


If you want to learn more about containers, this zine by Julia Evans is incredible: https://wizardzines.com/zines/containers/

and allow users to either export their work as a research compendium or allow sharing of these environments to bolster reproducibility.

One of these applications is WholeTale, which is an "NSF-funded Data Infrastructure Building Block (DIBBS) initiative to build a scalable, open source, web-based, multi-user platform for reproducible research enabling the creation, publication, and execution of tales - executable research objects that capture data, code, and the complete software environment used to produce research findings".[5] They define a tale as "an executable research object that combines data (references), code (computational methods), computational environment, and narrative (traditional science story)" -- which we know is also called a research compendium.[6]

[5] "Whole Tale," Whole Tale, 2019, https://wholetale.org/.
[6] "Whole Tale."



A beta version of the system is available at https://dashboard.wholetale.org.

When working in WholeTale, the users have the option to choose a type of environment from a list of options: RStudio, Jupyter Notebooks, OpenRefine 2.8, Jupyter Notebooks with Spark, and JupyterLab. Once within those environments, users can work as if they were on their local computer -- importing and installing new libraries, adding data, and even running high performance computing jobs. WholeTale will keep track of the version of any software dependencies and relevant environmental variables. Once a tale is complete, it can be published to a repository like Dataverse, with descriptive metadata and a research compendium that can later be rerun in WholeTale for reproducibility.[7]

However, the web-based IDEs for reproducibility require that researchers work within a specific online platform, and that can be untenable for those who need to be able to, for instance, work offline, or work across multiple types of environments for collaboration or compliance purposes. Most, if not all, of the proprietary tools that are marketed for computational reproducibility fall in this category of web-based IDEs.

Web-based Replay Systems

Given that researchers are hard pressed to change their workflows and tools, web-based replay systems were created. These are applications that take a link to research materials hosted elsewhere, build the computational environment in-browser, and displays to the user some method of interacting with the materials, such as an instance of JupyterLab.

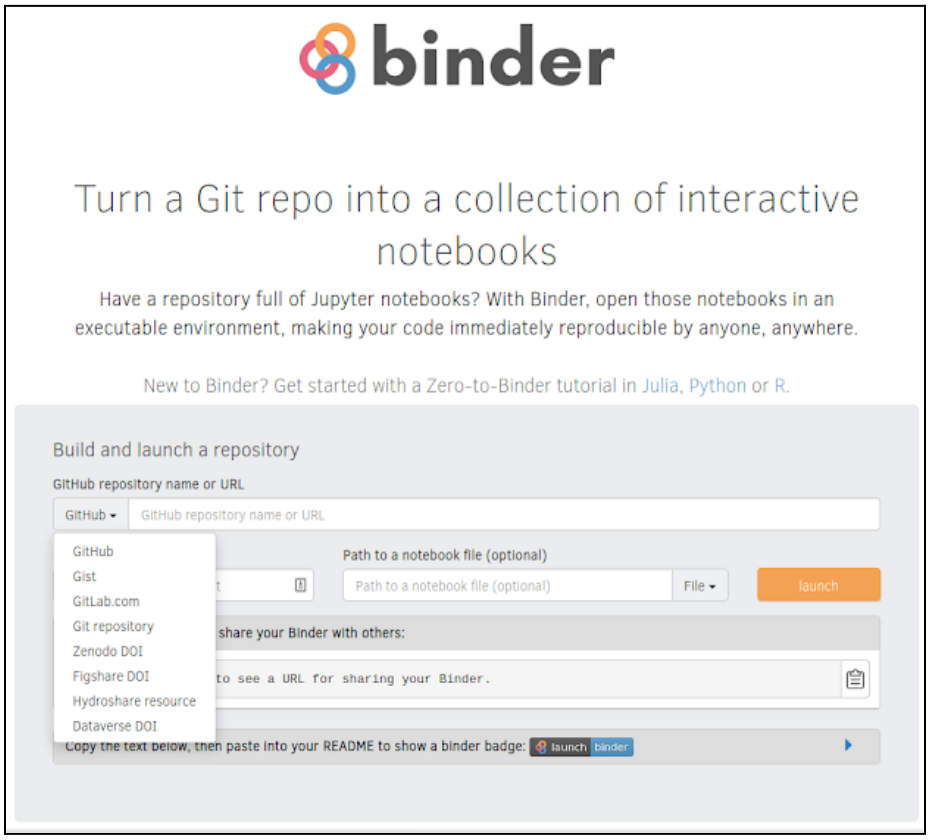
[7] Chard et al., "Application of BagIt-Serialized Research Object Bundles for Packaging and Re-Execution of Computational Analyses."

This offloads the responsibility for hosting materials to platforms devoted to that, and allows for the researchers to have flexibility in how they work.

Web-based replay systems allow for any user to interact with reproducible compendia in a 'sandbox', allowing users to modify input data or parameters, or even code, and re-execute it. They often ask the user to follow some structure for either the input or the directory structure in order to work properly, and use container systems in the backend to recreate the research compendia for researchers.

There are two large scale projects that allow for computational replay of research. One of those is Binder, from Project Jupyter. Binder uses repo2Docker to reproduce the computational environment of research hosted on Git hosting platforms (e.g. GitLab, GitHub) or repositories (e.g. Zenodo, Dataverse).[8] Users can replay materials in RStudio, Jupyter notebooks, JupyterLab, and Julia notebooks from Binder. When navigating to mybinder.org, the user is prompted to enter a URL or DOI that leads to a directory that contains Jupyter notebooks, RMarkdown files, or Julia notebooks. Binder will then look through the directory of files for something that will tell it about the computational dependencies, like a requirements.txt file for a Python project, or a Dockerfile. The user will then see the materials in the original computational environment, in the original interface.[9] Binder also provides a reusable link to this page with the live materials to others who want to reproduce the work.

[8] Project Jupyter et al., "Binder 2.0 - Reproducible, Interactive, Sharable Environments for Science at Scale," *Proceedings of the 17th Python in Science Conference*, July 15, 2018, 113–20, https://doi.org/10.25080/Majora-4af1f417-011.
[9] Jupyter et al.



The homepage of mybinder.org.

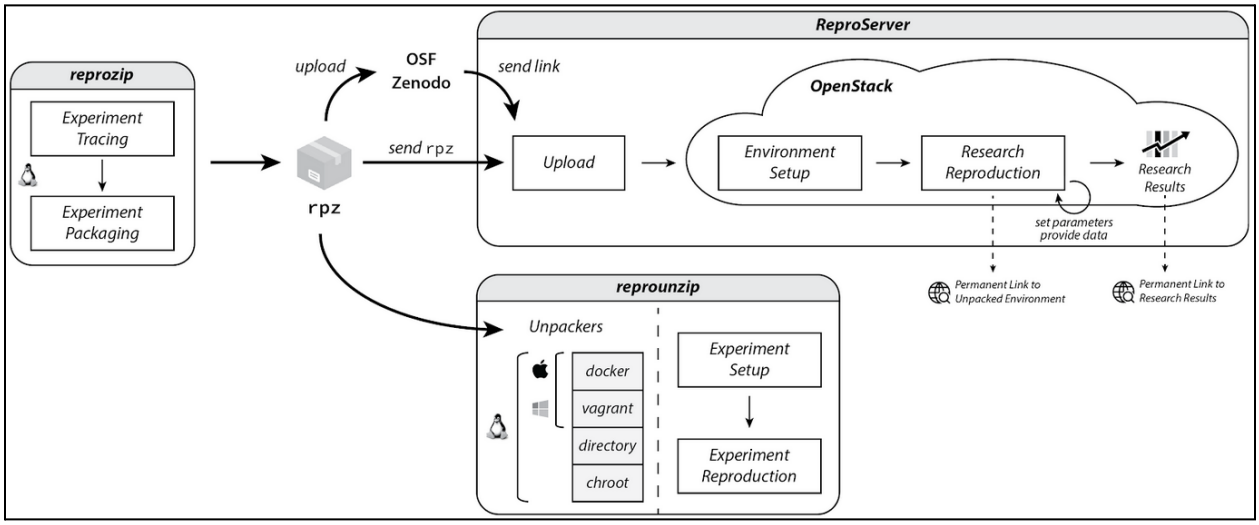
REANA is another example of a computational replay system, based in high energy physics (HEP). Made by a team at CERN (the European Organization for Nuclear Research) the goal of REANA is to help researchers "structure their input data, analysis code, containerised environments and computational workflows so that the analysis can be instantiated and run on remote compute clouds".[10] REANA relies heavily on the usage of the Common Workflow Language, "an open standard for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments".[11] This, in combination with the multiple container systems available on REANA, allow for computational replay of HEP workflows. This idea and process could, however, be generalized for other domains as well.

[10] "REANA - Reusable Analyses," accessed August 13, 2020, http://reanahub.io/.
[11] Peter Amstutz et al., "Common Workflow Language, v1.0," July 8, 2016, *5921760 Bytes*, https://doi.org/10.6084/M9.FIGSHARE.3115156.V2.

Packaging Systems

The final category of computational reproducibility tools we'll cover are packaging systems. Packaging systems are desktop or server-based tools that automatically capture dependencies & computational environments at time of executing a computational pipeline. The draw with packaging systems is the flexibility -- you don't have to go into a project thinking about reproducibility to be able to use a packaging tool to create a record of the computational environment. As long as your pipeline runs, the packaging tool will work.

One example is ReproZip. ReproZip works by running at the same time as a computational pipeline, tracing all the steps and dependencies while the pipeline runs like normal. Then it packages together input files, output files, parameters, environmental variables, executable code and steps, into a portable, generalized format: the RPZ (.rpz), or the ReproZip bundle. These bundles are small (size of the bundles really depends on the size of input and output data), portable (can be deposited into a repository or emailed!), and self-contained (everything needed to reproduce the pipeline is there!).[12]



ReproZip ecosystem, created by Fernando Chirigati. Used with permission.

ReproZip bundles can be replayed locally on any operating system (using ReproUnzip) or in-browser (using ReproServer). These tools will take a ReproZip bundle and

[12] Fernando Chirigati et al., "ReproZip: Computational Reproducibility With Ease," in *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16 (New York, NY, USA: Association for Computing Machinery, 2016), 2085–2088, https://doi.org/10.1145/2882903.2899401.

automatically unpack it, setting up all the dependencies and workflow steps for users, so they can reproduce the contents in the original computational environment. ReproUnzip operates on the plugin model, so users can choose which unpacker they can use to reproduce the work, for example Docker or Vagrant. However, this can be expanded to include any container or virtual machine systems in the future, because the extensive metadata ReproZip captures.[13]

ReproZip also has an ecosystem of other open tools: ReproZip-Web (combines ReproZip with web archiving technology to capture complex server-client applications), reprozip-jupyter (a ReproZip plugin for Jupyter notebooks, see example videos), ReproUnzip (a tool to replay and interact with the computational pipelines archived in ReproZip bundles), and ReproServer (a way to replay ReproZip bundles in-browser). Right now, ReproZip, ReproZip-Web, and reprozip-jupyter can only pack materials on Linux (because of the extensive information captured and the fact that the OS needs to be recreated at-will from ReproZip bundles), but users can install any of the other tools above on any operating system.

However, installing ReproUnzip and another piece of software can be a big ask for some researchers. To that end, ReproServer was created, which allows users to either upload a ReproZip bundle (.rpz) or provide a link to one, and then reproduce and interact with the contents of the RPZ file in-browser, drastically reducing the number of steps and complexity. What's more is that ReproServer integrates with repositories, such that users can create links like this: https://server.reprozip.org/osf.io/<5 character OSF link> to immediately begin reproducing the work or send to reviewers/collaborators for their input. ReproServer also provides a permanent URL to the unpacked environment and the results of rerunning the pipeline in the RPZ file.[14]

Summary

Different reproducibility tools will work for different researchers and workflows. For instance, when processing and analyzing research materials, many people tend to use containers or web-based IDEs because the rapid-prototyping capabilities are useful for the more exploratory and error-prone processing step. One key reason why they are especially useful

[13] Chirigati et al.
[14] Vicky Steeves, Rémi Rampin, and Fernando Chirigati, "Reproducibility, Preservation, and Access to Research with ReproZip and ReproServer," *IASSIST Quarterly* 44, no. 1–2 (June 29, 2020): 1–11, https://doi.org/10.29173/iq969.

in the analysis step is because they can also be ported to be compatible with web-based replay systems, which are useful in publishing your work.

When the work is done and nearing publication, people tend to prepare, structure, or export their research for web-based replay systems. These are useful because of the near-instant replay of computational research for reviewers of publications or presentations, members of promotion committees, or any interested party. This brings a wider accessibility to the reproducible work, which helps for post-publication review.

Lastly, packaging tools are the most sustainable for long-term reproducibility, especially when combined with emulation technology. Packaging tools are provenance-aware (e.g. it knows the order in which research pipelines run), automatically capture dependencies, automatically write in-depth technical and administrative metadata, and are interoperable (in that, they are built to work with a variety of other tools). These traits make them the most reliable for preservation and access purposes.

This section was meant to guide an understanding of the wider landscape of computational reproducibility tools. These four key classes of tools (containers, web-based IDEs, web-based replay systems, and packaging tools) and the examples discussed here reflect community-based efforts to scaffold the understandability and usability of their research, teaching, and learning. These tools can be used to both make one's own work reproducible, as well as help a designated community make their work more reproducible and sustainable in the long-term.

COI: Vicky Rampin contributes to the ReproZip project.

References

Amstutz, Peter, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, et al. "Common Workflow Language, v1.0," July 8, 2016, 5921760 Bytes. https://doi.org/10.6084/M9.FIGSHARE.3115156.V2.

Chard, Kyle, Niall Gaffney, Matthew B. Jones, Kacper Kowalik, Bertram Ludascher, Timothy McPhillips, Jarek Nabrzyski, et al. "Application of BagIt-Serialized

Research Object Bundles for Packaging and Re-Execution of Computational Analyses." In 2019 *15th International Conference on EScience* (EScience), 514–21. San Diego, CA, USA: IEEE, 2019. https://doi.org/10.1109/eScience.2019.00068.

Chirigati, Fernando, Rémi Rampin, Dennis Shasha, and Juliana Freire. "ReproZip: Computational Reproducibility With Ease." In *Proceedings of the 2016 International Conference on Management of Data*, 2085–2088. SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016. https://doi.org/10.1145/2882903.2899401.

Docker Inc. "Docker." Accessed August 13, 2020. https://www.docker.com/.

Hogg, Scott. "Software Containers: Used More Frequently than Most Realize." *Network World*, May 26, 2014. https://www.networkworld.com/article/2226996/software-containers--used-more-frequently-than-most-realize.html.

Jupyter, Project, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, et al. "Binder 2.0 - Reproducible, Interactive, Sharable Environments for Science at Scale." *Proceedings of the 17th Python in Science Conference*, July 15, 2018, 113–20. https://doi.org/10.25080/Majora-4af1f417-011.

Konkol, Markus, et al. "Publishing Computational Research -- A Review of Infrastructures for Reproducible and Transparent Scholarly Communication." *Research Integrity and Peer Review*, vol. 5, no. 1, Dec. 2020, p. 10. arXiv.org, https://doi.org/10.1186/s41073-020-00095-y.

Nüst, Daniel, Carl Boettiger, and Ben Marwick. "How to Read a Research Compendium." ArXiv:1806.09525 [Cs], June 11, 2018. http://arxiv.org/abs/1806.09525.

Nüst, Daniel, and Matthias Hinz. "Containerit: Generating Dockerfiles for Reproducible Research with R." *Journal of Open Source Software* 4, no. 40 (August 21, 2019): 1603. https://doi.org/10.21105/joss.01603.

"REANA - Reusable Analyses." Accessed August 13, 2020. http://reanahub.io/.

Sylabs.io. "Singularity." Accessed August 13, 2020. https://sylabs.io/.

Steeves, Vicky, Rémi Rampin, and Fernando Chirigati. "Reproducibility, Preservation, and Access to Research with ReproZip and ReproServer." *IASSIST Quarterly* 44, no. 1–2 (June 29, 2020): 1–11. https://doi.org/10.29173/iq969.

Techopedia.com. "What Is an Integrated Development Environment (IDE)? - Definition from Techopedia," January 11, 2017. http://www.techopedia.com/definition/26860/integrated-development-environment-ide.

Whole Tale. "Whole Tale," 2019. https://wholetale.org/.