Reproducible Research Environments with repo2docker

Jessica Forde
Project Jupyter
jzf2101@columbia.edu

Tim Head
Wild Tree Tech

Chris Holdgraf, Yuvi Panda, Fernando Perez
UC Berkeley

M. Pacer
Netflix

Gladys Nalvarte, Benjamin Ragan-Kelley
Simula Research Laboratory

Erik Sundell
IT-Gymnasiet Uppsala

Abstract

Reproducibility challenges in machine learning often center on questions of software engineering practices. Researchers struggle to reproduce another scientist's work because they cannot translate a paper into code with similar results or run an author's code. repo2docker provides a simple tool for checking the minimum requirements to reproduce a paper by building a Docker image based on a repository path or URL. Its goal is to minimize the effort needed to convert a static repository into a working software environment. By inspecting a repository for standard configuration files used in contemporary software engineering and leveraging containerization methods, repo2docker deterministically reproduces the environment of the author so the researcher can reproduce the author's experiments.

1 Introduction

Contemporary scientific workflows "depend on chains of computer programs that generate data, and clean up data, and plot data, and run statistical models on data" [16]. Researchers have reported difficulty in reproducing the work of other researchers and even their own work [1]. A study of papers published in *Science* in 2011 found that only 26% were reproducible [17]. Machine learning researchers are also examining the research practices of their discipline by replicating the results of Baker [1] within their own community [10]. There are efforts to report the level of reproducibility of machine learning experiments [5, 11] and to critique machine learning methodologies for their lack of reproducibility [6]. At the same time, the growing popularity of open-source software has made it easier to access the tools of other researchers [16]. Notable projects include IPython [9] and Jupyter Notebooks, CodaLab [7], and GitHub. Additionally, developments in containerization technology, such as Docker, have made it possible to more easily create reproducible, lightweight software environments. Docker uses Dockerfiles to set up its virtual environment, a Docker image. Tools like Docker are primarily used by the dev-ops community, and many scientists do not currently write their own Dockerfiles (see Figure 2a). While there exist tools such as Heroku build packs and source2image to automate the creation of configuration files, these tools were designed primarily for software engineers and require additional effort to work with typical software used by researchers [8].

repo2docker [12] is a simple open-source tool to containerize environments for scientific reproducibility. It uses a simple command-line interface to test the reproducibility of a repository's software environment and enables the reproducibility of the repository in a language- and platform-agnostic manner. repo2docker takes as input a path or URL to a repository with standard configuration files and builds a Docker image containing the repository's files with an environment built from the dependencies indicated in the config files. These configuration files are standard tools to reproduce a

Preprint. Work in progress.

software environment, and we find that research repositories that already use these file formats to describe their software environment are more popular on GitHub (see Section 4).

2 Creating Docker Images with repo2docker

The core feature of repo2docker is to fetch a repository at an arbitrary URL, inspect the repository for configuration files that define the environment needed to run its content, and build a container image based on the files in the repository. After installing Docker and repo2docker, one may call the command-line interface with the path to the repository:

```
jupyter-repo2docker https://github.com/dtak/rrr --ref master/d2bce99
```

repo2docker expects the path or URL to point to a git repository such as GitHub, GitLab, or the path to a local git repository. repo2docker accepts named git branches and commit hashes with `--ref branch-name/commit-hash`, allowing researchers to build specific versions of a repository. Optionally, it launches a local Jupyter server so a researcher can interact with the repository and registers the container image with an image registry. In this case, repo2docker will return:

```
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://0.0.0.0:36511/?token=f94f8fabb92e22f5bfab116c382b4707fc2cade56ad1ace0
```

The URL directs the user to a Jupyter Notebook interface running the environment contained in the Docker image. The working space is populated with all files specified in the repository and all dependencies specified in the configuration files are installed. The user may also use JupyterLab by opening the original URL and then navigating to `http://0.0.0.0:36511/lab`.

3 Language-agnostic Software Environments

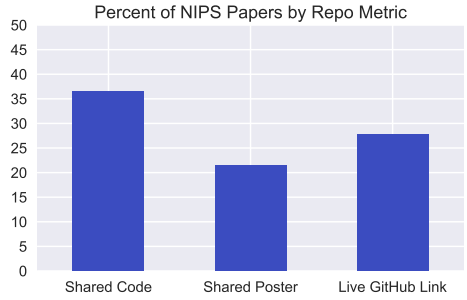
repo2docker uses standard file formats used by a variety of package managers and installation tools to deterministically recreate the software environment of a repository, similar to Heroku build packs, conda, pip, and apt-get. repo2docker works best with repositories primarily written in Python, Julia, and R, though it has been used to create reproducible environments for other languages such as Go[18], C++[14, 3, 2] and Haskell[4] by using combinations of configuration files. repo2docker detects the following configuration file formats:

- `Dockerfile`
- `environment.yml`
- `requirements.txt`
- `REQUIRE`
- `apt.txt`
- `postBuild`
- `runtime.txt`
- `setup.py`
- `install.R`

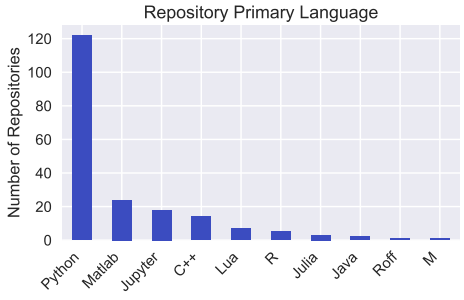
Dockerfiles receive precedence over other file types. Some files are language-specific, such as `setup.py` for Python, `REQUIRE` for Julia, and `install.R` for R. `apt.txt` installs Debian packages from an Ubuntu environment, and `postBuild` is a shell script that runs at the end of the build process for additional customization. Many of these configuration files are composable such that combinations of environments can be defined in a single repository. The GitHub organization binder-exanples contains example repositories using these various files with Python, Julia, LaTeX, and R. They can be explored interactively using a service called Binder [13]. In addition, members of the machine learning community have already used repo2docker to create reproducible environments for their work. To visit a live example of Ross et al. [15], visit the binder of the repository.

4 Engineering Practices in Machine Learning Research

To demonstrate how configuration files used by repo2docker increase the reproducibility of machine learning research, we analyzed publication data from NIPS. In 2017, NIPS included URLs for the papers, code, and posters of conference publications on its schedule. We collected the URLs from the schedule to examine the reproducibility of the software environments of NIPS 2017 papers through their published code. Results are published on mybinder.org with repo2docker using the GitHub repo jzf2101/r2d_study .

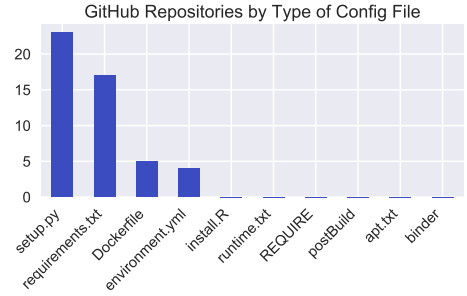


(a) Percent of NIPS 2017 papers (679) with links to code, poster, and live GitHub repository.

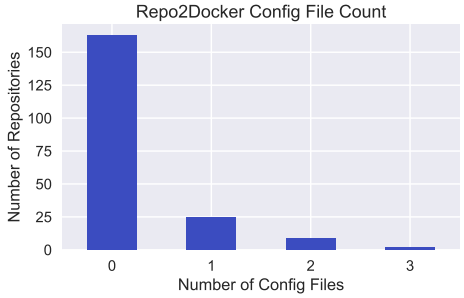


(b) Number of repositories by primary language.

Figure 1: Conference wide metrics for NIPS 2017 papers. Only 36.5% include links to code. The majority of these links are to GitHub repositories in Python, which repo2docker config files support.



(a) Number of repositories with each repo2docker configuration file.

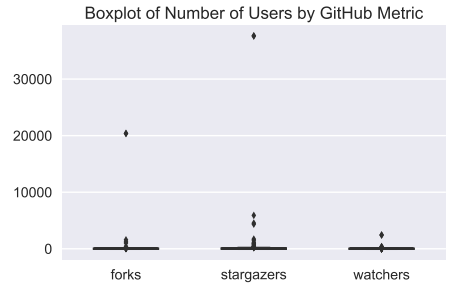


(b) Number of repositories by number of configuration files in each repository.

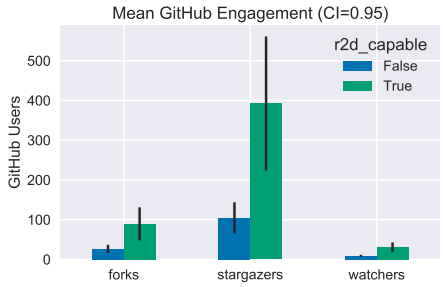
Figure 2: Presence of repo2docker-ready configuration files among 197 GitHub repositories of NIPS 2017 papers. The majority do not have configuration files to recreate the environment of the paper. Most papers that include these files use `setup.py` or `requirements.txt`. Few supply a Dockerfile.

Figure 1 shows overall reproducibility results for NIPS 2017 papers. While all 679 papers published a link to the paper, 36.5% and 21.5% included links to code and poster, respectively. 27.8% provided links to GitHub repositories that we tested and found to be live. We investigated this subset of papers with code on GitHub to determine if their they included configuration files compatible with repo2docker (n=197). All but 30 repositories used a language largely supported by repo2docker (Python, Julia, or R). In fact, 160 repositories contained Python code, and 121 repositories were written primarily in Python. The majority of repositories, therefore, could likely parameterize their software environments with repo2docker's standard configuration files. Nevertheless, the majority of repositories do not supply the files to deterministically reproduce the environment of the paper. Five explictly provided a Dockerfile. Only 36 used at least one of files repo2docker uses that define the environment. The most popular file type was `setup.py`, followed by `requirements.txt`.

To correlate the inclusion of these configuration files with the reproducibility of a paper, we use GitHub engagement metrics as a proxy for ease of software use. Because these configuration files are used by repo2docker and other tools to deterministically reproduce the software environment of a paper's experiments, we consider the presence of these files a minimum level of reproducibility for a paper. Users on GitHub can fork, stargaze, or watch another user's repository, and the number of users who have performed each action reflect a repository's engagement with other users. Notably, forking allows a user to modify a copy of another user's repository, and may indicate attempts to reproduce the author's work locally. We exclude papers that are connected to larger deep learning libraries (i.e. TensorFlow models, DyNet). These are papers whose URL is to a folder of a larger repository and often have outsize GitHub engagement (as high as 36,531 stargazers).



(a) Boxplot of all repositories by GitHub metric. Papers with code maintained by deep learning frameworks are outliers.



(b) Mean GitHub metrics with 95% CI for repositories that are and are not repo2docker ready.

Figure 3: Papers whose code is maintained by deep learning frameworks (TensorFlow, DyNet) receive orders of magnitude more forks and stargazers and were excluded from analysis. Compared to other paper repositories, repo2docker-ready repositories have significantly higher GitHub engagement.

We find that repositories that use configuration files to reproduce the environment of the paper (n=36) are between 2.35 and 2.75 times as popular across all metrics as those that only provide code (n=158). The difference in these average metrics are statistically significant: p-values of the independent two-sample t-tests of forks, stargazers, and watchers are 0.037, 0.015, and 0.010, respectively. These statistical differences suggest that users on GitHub prefer to work with software written by authors who include these files to install the environment. In particular, the greater number of forks suggest users are more willing to reproduce or extend the work of these repo2docker-ready papers.

5 Binder: Using repo2docker in Production

While repo2docker can be used to build Docker images locally, it can also be used to flexibly generate software environments as a part of a pipeline. Binder is a free open-source service that lets users share a live, reproducible version of their repository. It uses repo2docker to generate Docker images of these repositories and register them so that others may access them with Binder. Binder connects JupyterHub, a scalable multi-user Juptyer server, to repo2docker to automate the building, registering, and deploying pipeline for users. Users have free access to a deterministically configured environment with the code as built by repo2docker. They can run the software immediately in the browser. Because repo2docker, JupyterHub, and BinderHub are open-source software, one can deploy Binder on their own servers to customize storage and compute power. We have shared our analysis on Binder to encourage the reproducibility of our work and to explicitly describe how configuration files can recreate the environment of our analysis with the GitHub repo jzf2101/r2d_study.

6 Conclusion

To reproduce the results of a machine learning paper, one must be able to run code or write code that reproduces the paper's experiments and results. Typically, the process begins with reproducing the author's software environment. Including configuration files in a research repository provides researchers with the specifics of experimental setup. Tools like apt-get, pip, and conda take their preferred configuration file and install listed dependencies. repo2docker gives authors the ability to combine these configuration files to deterministically replicate an experimental software environment in a Docker image with a single command. We have found that few canonical research repositories use these files in practice. Anecdotally, we observed that some describe dependencies in the README, but these notes are not standardized such that software could deterministically parse these descriptions and install all software. Additional effort is required to setup repositories without these files based on the README alone. By highlighting the significant increase in engagement from GitHub users, we encourage researchers to provide these configuration files in their repositories so that other GitHub users can more easily use their code. We also encourage conference organizers to publish a digest of available code resources from accepted papers to facilitate the reproduction of presented experiments.

References

[1] M. Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, May 2016. URL http://dx.doi.org/10.1038/533452a.

[2] CERN PH-SFT. rootbinder. URL https://github.com/cernphsft/rootbinder.

[3] diana-hep. pyhf. URL https://github.com/diana-hep/pyhf.

[4] A. Gibiansky. IHaskell. URL https://github.com/gibiansky/IHaskell.

[5] O. E. Gundersen and S. Kjensmo. State of the art: Reproducibility in artificial intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17248/15864.

[6] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. Sept. 2017. URL http://arxiv.org/abs/1709.06560.

[7] P. Liang and E. Viegas. CodaLab worksheets for reproducible, executable papers, Dec. 2015. URL https://nips.cc/Conferences/2015/Schedule?showEvent=5779.

[8] Y. Panda. Why repo2docker? why not s2i? http://words.yuvi.in/post/why-not-s2i/, Dec. 2017. URL http://words.yuvi.in/post/why-not-s2i/. Accessed: 2018-6-21.

[9] F. Perez and B. E. Granger. IPython: A system for interactive scientific computing. *Computing in Science Engineering*, 9(3):21–29, May 2007. URL http://dx.doi.org/10.1109/MCSE.2007.53.

[10] J. Pineau. Reproducibility in deep reinforcement learning and beyond, Dec. 2017. URL https://twitter.com/xtimv/status/938917013086380032.

[11] J. Pineau, G. Fried, R. N. Ke, and H. Larochelle. ICLR 2018 reproducibility challenge. https://www.cs.mcgill.ca/~jpineau/ICLR2018-ReproducibilityChallenge.html, 2017. URL https://www.cs.mcgill.ca/~jpineau/ICLR2018-ReproducibilityChallenge.html. Accessed: 2018-6-10.

[12] Project Jupyter Contributors. repo2docker, 2017. URL https://github.com/jupyter/repo2docker/.

[13] Project Jupyter Contributors. Introducing binder 2.0 — share your interactive research environment. *eLife*, Nov. 2017. URL https://elifesciences.org/labs/8653a61d/introducing-binder-2-0-share-your-interactive-research-environment.

[14] QuantStack. xeus-cling. URL https://github.com/QuantStack/xeus-cling.

[15] A. S. Ross, M. C. Hughes, and F. Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages Pages 2662–2670., Mar. 2017. URL https://www.ijcai.org/proceedings/2017/371.

[16] J. Somers. The scientific paper is obsolete. *The Atlantic*, Apr. 2018. URL https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/.

[17] V. Stodden, J. Seiler, and Z. Ma. An empirical analysis of journal policy effectiveness for computational reproducibility. *Proc. Natl. Acad. Sci. U. S. A.*, 115(11):2584–2589, Mar. 2018. URL http://dx.doi.org/10.1073/pnas.1708290115.

[18] Y. Watanabe. lgo. URL https://github.com/yunabe/lgo.