Reproducible Research Environments with Repo2Docker

Jessica Forde
Project Jupyter
jzf2101@columbia.edu

Tim Head
Wild Tree Tech
tim@wildtreetech.com

Chris Holdgraf
UC Berkeley
choldgraf@berkeley.edu

Erik Sundell
IT-Gymnasiet Uppsala
erik.i.sundell@gmail.com

Abstract

Reproducibility challenges in machine learning center on questions of software engineering practices. In particular, a reproducible paper should have software that is easy to run to reproduce the results of the paper. repo2docker provides a simple tool for checking the minimum requirements to reproduce a paper by building a Docker image based on a GitHub repository URL. By inspecting a GitHub repository for standard configuration files used in contemporary software engineering practices, repo2docker leverages containerization methods to produce a deterministic code environment for the researcher that installs the dependencies of a given research repository. Researchers who use these standard configuration files in their work not only make their papers compatible to repo2docker, but also attract more engagement from other GitHub users, indicating their repository's ease of use.

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"[Somers, 2018]. Across various scientific disciplines, researchers have reported difficulty reproducing the work of other researchers and even their own work [Baker, 2016]. This year, a study of papers from the journal Science in 2011 found that only 26% of papers were able to be reproduced [Stodden et al., 2018]. Machine learning researchers are also examining the research practices of their own discipline by replicating the results of Baker [2016] in their own community [Pineau, 2017], documenting the level of reproducibility in their own publications [Gundersen and Kjensmo] [Pineau et al.], and critiquing methodologies in machine learning research for their lack of reproducibility [Henderson et al., 2017].

At the same time, with the growing popularity of open-source software, it has become easier to access the tools of other researchers [Somers, 2018]. A number open-source tools have been created to model, visualize, and share scientific ideas. Notable projects include IPython [Perez and Granger, 2007] and Jupyter Notebook, CodaLab [Liang and Viegas, 2015], and GitHub. At the same time, developments in containerization have allowed for reproducible, light-weight creation of software environments. repo2docker (r2d) [Project Jupyter Contributors, 2017] uses a simple command-line interface method of testing the reproducibility of a 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 the dependencies indicated in the config files installed. Because these configuration files are used in standard practice to reproduce a

Preprint. Work in progress.

software environment, we find that research repositories that already use these file formats to describe their software environment are more popular of GitHub.

2 Creating Docker Images with repo2docker

The core feature of repo2docker is to fetch a repo, build a container image based on the specifications found in the repo, and optionally launch a local Jupyter Notebook to explore 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/norvig/pytudes
```

Currently, repo2docker expects the local path or url to point to a git repository such as GitHub, GitHub Gists, GitLab, or a local git repository. repo2docker accepts named git branches with the `-ref origin/branchname` flag. When the repository is built, repo2docker will return the following message:

```
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 the Jupyter Notebook interface of the Docker image with the repository as the repository folder as the working directory the dependencies of the repository installed. To print, build and run the Dockerfile, one may use the `-debug` flag. repo2docker also deterministically produces Dockerfiles of the repository without building the image with the `-no-build` flag.

3 Language-agnostic Engineering Practices

repo2docker uses standard file formats used by various package managers and installation tools across a variety of platforms to deterministically recreate the software environment of a repository. It follows in the tradition of tools such as Heroku build packs, conda, pip, and apt-get. Currently, repo2docker works best with repositories primarily written in Python, Julia, and R. It accepts the following file formats:

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

Dockerfiles receive precedence over other file types. Many of these files are language specific installation files, such as `enviornment.yml` for conda, `REQUIRE` for Julia, and`install.R` for R. `apt.txt` installs various Debian packages, and `postBuild` allows for additional environmental customization by running the `postBuild` script at the end of the build process. The GitHub organization https://github.com/binder-examples/ contains example repositories using these various files with Python, Julia, LaTeX, and R. To visit a live example of a machine learning paper [Ross et al., 2017], visit the binder of the repo https://mybinder.org/v2/gh/dtak/rrr/master.

4 Engineering Practices in Machine Learning Research

To demonstrate how the standard files used by repo2docker increase the reproducibility of machine learning research, we have research publication data from NIPS 2017. In 2017, NIPS published URLs

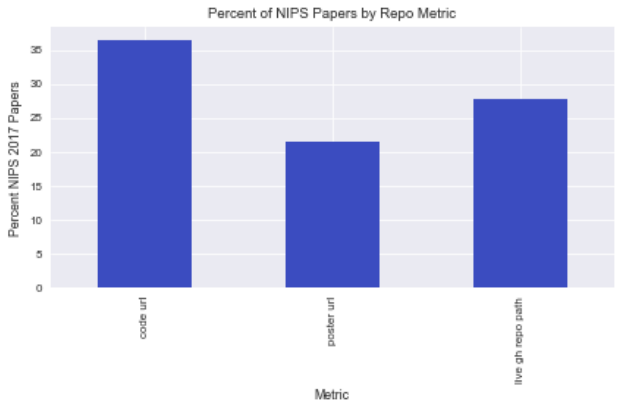

Figure 1: Percent of total NIPS 2017 papers (679) with links to code, poster pdf, and live GitHub repository

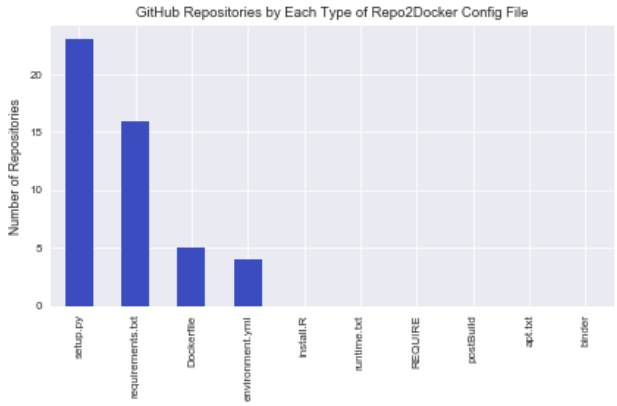

Figure 2: Number of repositories with each type of repo2docker configuration file. The majority of papers that include configuration files use standard Python files such as `setup.py` or `requirements.txt`. Few directly supply a Dockerfile.

for the paper, code, and poster of each conference publication on its schedule. We have collected the URLs from the schedule to examine the reproducibility NIPS 2017 papers and published code. Results are published on mybinder.org with repo2docker.

Figure 1 shows overall reproducibility results for NIPS 2017 papers. While all 679 papers published a link to the pdf of the paper, only 36.5% included links to code. 27.8% of papers provided links to GitHub repositories that we tested and found to be currently live. This number was greater than the number of percent of papers that included a poster link, 21.5%. Eight of the papers required manual review to locate the exact link of the paper.

In total, 197 repositories were examined for current compatibility with repo2docker through the presence of its accepted the configuration files. All but 30 repositories used a language largely supported by repo2docker, either Python, Julia, or R. In fact, 160 repositories contained Python, and Python was the primary language of 121 repositories. The majority or repositories, therefore, can be built with repo2docker's standard configuration files.

Nevertheless, the majority of repositories do not supply configuration files to deterministically recreate the environment of the paper. Only 36 use any type of file. Five directly provide a Dockerfile to the user to recreate the environment. The most popular file type is `setup.py`, to install the repository like a Python package, followed by `requirements.txt`.

To correlate the use of these configuration files to reproducibility, we have chosen GitHub engagement metrics as a proxy for ease of software use. Repositories on GitHub may be stargazed, watched, or

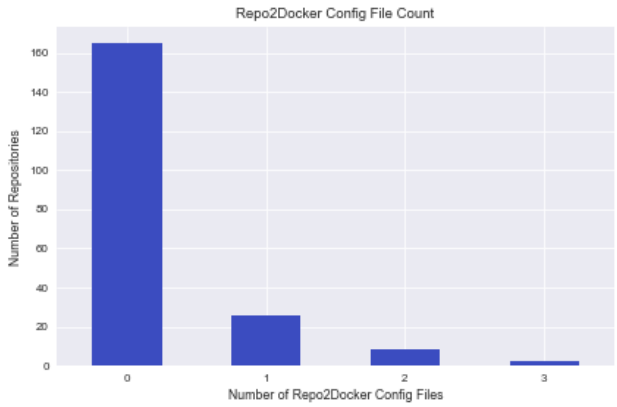

Figure 3: Number of repositories by number of configuration files in each repository. Of the 197 papers with GitHub repositories available online, the majority do not supply configuration files to recreate the environment of the paper.

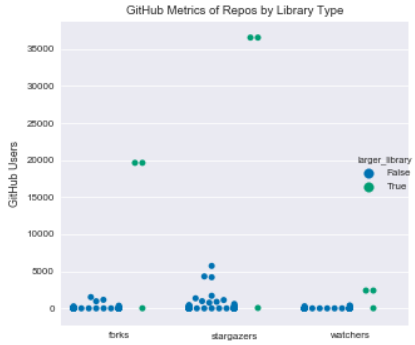

Figure 4: Gittered plot of repositories by GitHub metric. Note that papers that belong to larger libraries have order of magnitude more attention from the GitHub community.

forked. Most notably, forks allow a user to modify a copy of the code with their own GitHub account. In comparing these two repository populations we have found paper repositories that are connected to larger deep learning libraries. We identify these as having paths that are not to the home directory of the repo. These repositories have stargazers as high as 36,531. Excluding these repositories from our comparison, so that repositories only about that paper are compared, we find that repositories that use the configuration files used by repo2docker are significantly more popular across all metrics. P-values for forks, stargazers, and watchers are 0.04106162, 0.01712506, and 0.01100394, respectively.

Figure 5: Mean GitHub engagement across metrics with errorbars. Excluding outliers from larger libraries shows average popularity of repo2docker-ready repos to be higher.

References

James Somers. The scientific paper is obsolete. *The Atlantic*, April 2018. URL https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/.

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

Victoria Stodden, Jennifer Seiler, and Zhaokun Ma. An empirical analysis of journal policy effectiveness for computational reproducibility. *Proc. Natl. Acad. Sci. U. S. A.*, 115(11):2584–2589, March 2018. URL http://dx.doi.org/10.1073/pnas.1708290115.

Joelle Pineau. Reproducibility in deep reinforcement learning and beyond, December 2017. URL https://twitter.com/xtimv/status/938917013086380032.

Odd Erik Gundersen and Sigbjørn 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.

Joelle Pineau, Genevieve Fried, Rosemary Nan Ke, and Hugo Larochelle. ICLR 2018 reproducibility challenge. https://www.cs.mcgill.ca/~jpineau/ICLR2018-ReproducibilityChallenge.html. URL https://www.cs.mcgill.ca/~jpineau/ICLR2018-ReproducibilityChallenge.html. Accessed: 2018-6-10.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. September 2017. URL http://arxiv.org/abs/1709.06560.

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.

Percy Liang and Evelyne Viegas. CodaLab worksheets for reproducible, executable papers, December 2015. URL https://nips.cc/Conferences/2015/Schedule?showEvent=5779.

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

Andrew Slavin Ross, Michael C Hughes, and Finale 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., March 2017. URL https://www.ijcai.org/proceedings/2017/371.