Draft version October 21, 2022
Typeset using LaTeX default style in AASTeX631

show your work!: a workflow for open source scientific articles

Rodrigo Luger and Others TBD

ABSTRACT

This paper introduces `showyourwork`, a workflow that enables the creation and distribution of fully reproducible and open source scientific articles.

1. THREE ARGUMENTS FOR REPRODUCIBILITY

Short intro here. I can think of three big arguments for reproducibility, but there are likely many more.

1.1. *Ensuring the validity of results*

As astronomical research software becomes increasingly more complex, and as research results become increasingly more interdependent, it becomes ever more challenging to ensure the validity and correctness of results published in the literature. Unfortunately, the current peer review system in astronomy is simply not set up to do this. Checking all of the results in a paper would require the painstaking and methodical review of all of the paper's methods—which usually means scrutinizing all of the code used to generate the figures, tables, and other quantities in the paper. Any stats on how much time a typical referee spends on reviewing a paper? Would be nice to back all this up with some numbers. We could also look for papers with erroneous results that propagated outwards in the literature because some big mistake made it past peer review. Anyone know of any classic examples?

In practice, this is virtually impossible for three reasons:

1. Modern codebases can be very large and often require deep familiarity with the software to use—not to mention review them. Volunteer referees rarely have the time to invest in learning new software in order to provide a comprehensive review.

2. Writing a paper in astronomy is rarely ever done in a linear, procedural fashion: the codebase is constantly changing, and the state of the code when (say) Figure 1 was produced may be very different from that when (say) Figure 2 was made. Moreover, many results depend on the execution of lengthy pipelines with intermediate steps, each potentially requiring manual tinkering that is not always documented and may be difficult to replicate exactly.

3. The majority of astronomical code is not open source and simply cannot be vetted by third parties (citation needed). While there has been a marked increase in the number of open source astronomical tools in recent years (e.g., `astropy`, `exoplanet`, `emcee`, `exofast`...), most code associated with the generation of the results in individual papers is not open source (citation needed); readers are often expected to take it on faith that there are no bugs in that code, or that the code works exactly as described in the text, with no pitfalls or missing details. Even when the code is made publicly available, e.g., by being published on `GitHub`, it is often not documented sufficiently to enable one to execute it and reproduce the paper's results out-of-the-box. And even with proper documentation, the code may require external dependencies, custom virtual environments, or access to closed-source datasets that make it difficult or impossible for a third party to replicate it.

A few sentences here on how an easily reproducible article could help with all this.

1.2. *Reducing duplication of effort*

We shouldn't have to ever reinvent the wheel. Astronomy should be collaborative. Time saved on duplicated effort can be invested in moving the field forward rather than always playing catch-up. Perhaps some stats on CPU usage in the field, and how reproducibility can help? Can also talk about open data here; e.g., telescope time is super competitive; we should make it easy for others to use our data, both raw and processed.

1.3. *Promoting equity and inclusivity*

Proprietary legacy code is antithetical to DEI. "Source code available upon request" is antithetical to DEI. "My code is on my website" is antithetical to DEI. It's 2022, FFS. We need to do everything we can to make astronomy open to everyone.

2. CURRENT APPROACHES TO REPRODUCIBILITY

Discuss code links in Luger et al. (2019, 2021c,b), and how they have caught on, e.g., Paillas et al. (2022). Give a detailed overview of reproducibility in astronomy; this page has some nice resources. Look at reviews by Harlan Krumholz in medicine. Talk about `maneage`.

3. THIS PAPER: THE show your work! WORKFLOW

This paper introduces `showyourwork`, a workflow designed to make it easy for authors to develop, publish, and distribute open and reproducible research papers in astronomy and other scientific disciplines. At the highest level, `showyourwork` is a command-line tool that builds papers from a set of instructions contained in a version-controlled repository and organized into manuscript files, scripts, pipeline files, and configuration/specifications files (see Figure 1). Every time the user makes changes to any of the files in the repository, the article is automatically re-built on the cloud and made publicly available on the remote repository. The build step—which installs the required software, generates all figures and other procedural output from scratch (with intelligent caching), and compiles the final article in an isolated build environment—acts as a unit test for the paper. If it passes, the paper is (by definition) reproducible.

The ideas presented here are not new; nor are the ways in which they are implemented. Instead, the main goal of `showyourwork` is to make developing—and using—reproducible scientific articles as easy as possible. `showyourwork` works out of the box for simple projects, in which each figure in the article can be generated by running a given script. But it also works for more complicated pipelines, such as projects that depend on many intermediate steps or those that require running expensive simulations. The workflow interfaces directly with `Zenodo`, allowing users to automatically upload the results of simulations so that expensive build steps can be bypassed on the cloud. In fact, most of the features under the hood are there to make the workflow as flexible and customizable as possible.

Papers that use `showyourwork` can be reproduced by downloading the associated repository and running the `showyourwork` command-line tool. Such papers (like this one!) include clickable icons next to each of their figures linking to (1) the exact version of the script used to generate them and (2) the exact version(s) of the dataset(s) used in their creation.

4. USING `showyourwork`

In this section we provide basic, high-level instructions on how to install and use `showyourwork`. These instructions are for version `0.3.1` of the code, the current version at the time of writing (September 2022) and are subject to change in future releases. For more details, including custom commands, settings, examples, and troubleshooting tips, please refer to the documentation for the specific version of `showyourwork`.

4.1. *Prerequisites*

`showyourwork` requires the `conda` package manager and is currently tested only on Unix-like operating systems (such as Linux, Ubuntu, or MacOS). Users must also have access to a `GitHub` account; other `git` platforms are not currently supported. Note that users do *not* need a local installation of a TeX distribution, as `showyourwork` uses the `conda`-managed `tectonic` package to compile articles.

4.2. *Installation*

`showyourwork` can be installed with the `Python` package manager `pip`:

```
pip install -U showyourwork
```

(recommended) or from source on `GitHub`:

```
git clone https://github.com/showyourwork/showyourwork
cd showyourwork
python setup.py install .
```

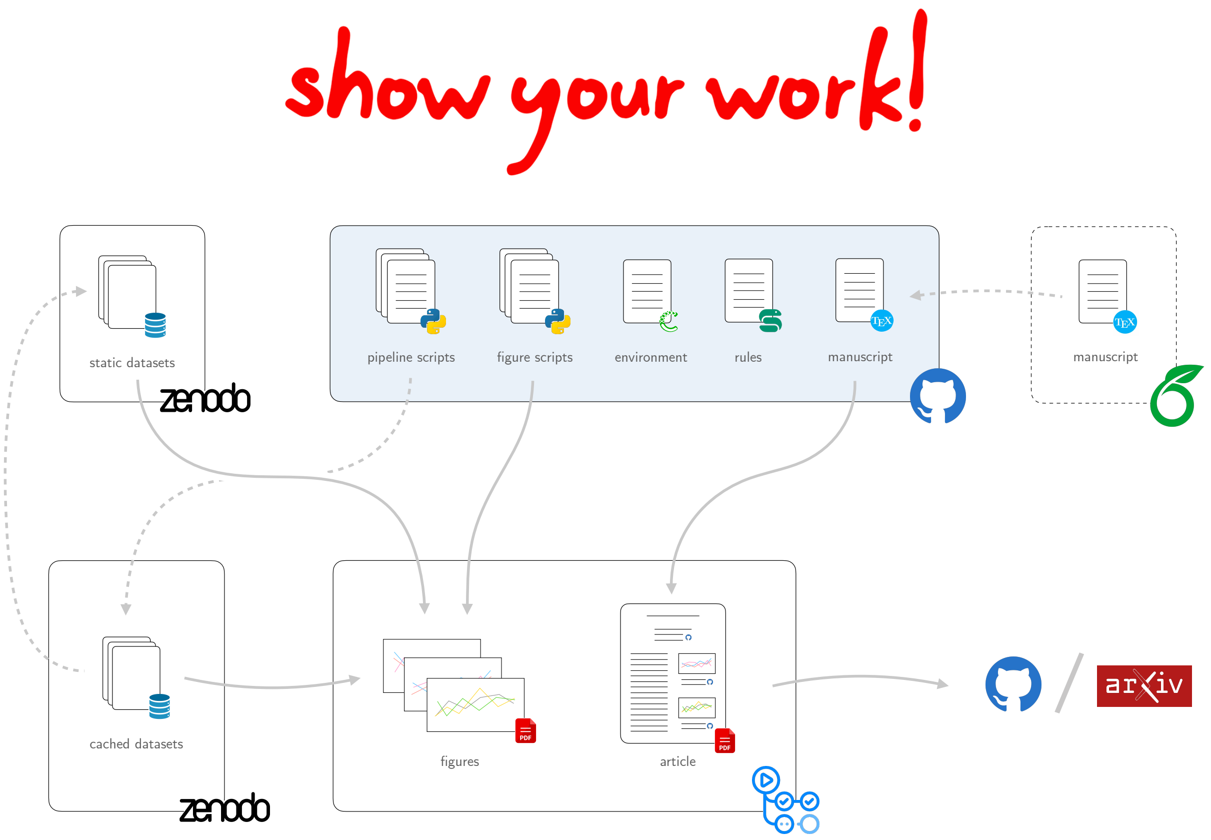

Figure 1. Overview of the showyourwork workflow for a scientific article. All code and instructions to compile the article exist within a GitHub repository (top center), optionally synced with an Overleaf project (top right). Upon every commit, a GitHub Action is triggered to build the article in an isolated environment. Scripts are executed to generate figures and other workflow output (bottom center), optionally downloading version-controlled datasets from Zenodo (top left), and optionally caching intermediate results on Zenodo Sandbox (bottom left). Finally, the compiled article .pdf file, alongside an arXiv-compatible tarball, are generated and pushed to a special branch on the GitHub repository.

4.3. *Reproducing an article*

Any project based on showyourwork can be reproduced by cloning its GitHub repository and running showyourwork. For example, to reproduce the exact version of this paper that you are currently reading, run:

```
git clone https://github.com/showyourwork/showyourwork-paper
cd showyourwork-paper
git checkout 45a1005e46c200954ecd6cba4b13312ad1d72acf
showyourwork
```

This will set up a custom conda environment for the workflow, download the required datasets from Zenodo, build all of the figures, and generate a .pdf file identical to this one.

4.4. *Creating a new article*

New projects may be created by running

```
showyourwork setup user/repo
```

where user and repo are the user's GitHub handle and the repository name, respectively. After the user answers a few prompts, showyourwork will set up a local git repository repo with the correct structure and placeholder files (see §4.5). The article may then be built by running

```
showyourwork
```

from the root of the repository.

4.5. *Repository structure*

Figure 2 shows the basic directory structure for a showyourwork article repository as of version 0.3.1. Repositories are comprised of configuration files at the root level and a directory called src containing the manuscript files and all of the the code and workflow scripts needed to render the final article .pdf file. More details here.

5. HOW IT WORKS

Discuss how things are actually implemented. Talk about the dependence on snakemake (Mölder et al. 2021) and tectonic. Discuss the integration with Zenodo for both static datasets and cached datasets. Discuss integration with Overleaf.

6. EXAMPLES

In this section, we showcase a few projects based on showyourwork to illustrate how the workflow can facilitate the development and dissemination of open source scientific articles.

The most fundamental feature of showyourwork is the automatic generation of the figures in a paper, such as Figure 3, which we reproduce from Wagg et al. (2022). This figure was generated from the script src/scripts/eccentricity.py and has only a single other dependency: the conda environment file environment.yml (see Figure 6).

To establish dependencies between scripts and figures included in the TeX manuscript, users may provide the custom \script{script} command within the figure environment, where script is the name of the script showyourwork should execute to generate the figure(s) within the current environment. Additional dependencies may be specified explicitly in the showyourwork.yml configuration file or implicitly via custom rules in the Snakefile.

As a seconde example, consider Figure 4, which we reproduce from Luger et al. (2021a). Unlike Figure 3, this figure has an external dependency: a Zenodo-hosted dataset containing the *CRIRES* observations of the spectrum of a brown dwarf. Users simply specify the DOI of the dataset and the files they wish to make dependencies of a given figure in showyourwork.yml, and showyourwork will download the data as needed.

Figure 5 is an example of a figure depending on an intermediate result that is cached on Zenodo Sandbox. More details.

Discuss Table 1 and variable output. Variable output can also occur inline. For instance, the following sentence is programmatically generated from the same script: the value of the Stochastic Harmonic Oscillator kernel for $\beta = 0.50$, $\lambda = 0.50$, $\tau = 0.25$ is 1.14997. Note the icon in the margin that automatically links to the corresponding snakemake rule, where users can easily change the values of the kernel parameters.

Discuss Figure 6 and this paper as a whole as a meta-example.

Finally, cite a bunch of papers here that already use showyourwork and showcase its various features.

7. CURRENT ISSUES AND FUTURE WORK

This is a work in progress. Currently focused primarily on short-term reproducibility; discuss the various issues that arise from relying on GitHub, conda, pip, etc. Broken links will be an issue. Conda environments are not impermeable. Currently difficult (impossible?) to run stuff on a cluster. Doesn't play nice with journals. Anything else?

8. CONCLUSION

Stuff here. showyourwork is a community project – please contribute!

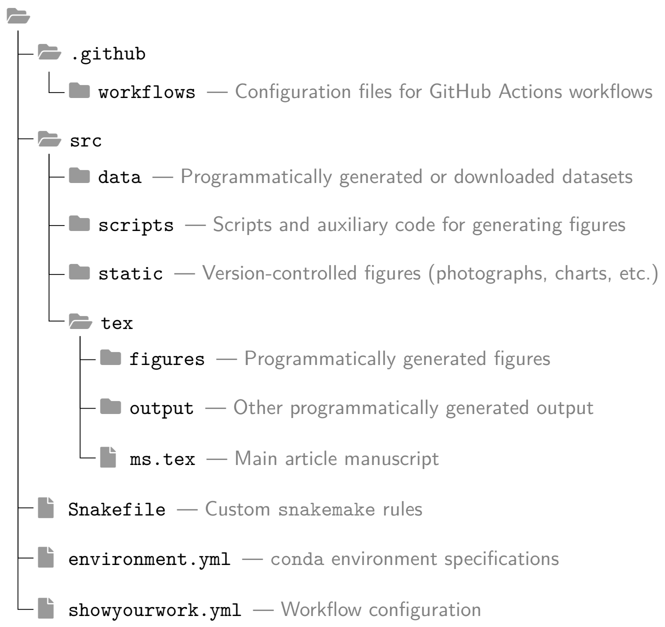
.github
workflows — Configuration files for GitHub Actions workflows
src
data — Programmatically generated or downloaded datasets
scripts — Scripts and auxiliary code for generating figures
static — Version-controlled figures (photographs, charts, etc.)
tex
figures — Programmatically generated figures
output — Other programmatically generated output
ms.tex — Main article manuscript
Snakefile — Custom snakemake rules
environment.yml — conda environment specifications
showyourwork.yml — Workflow configuration

Figure 2. The basic repository structure for an open source scientific article built with `showyourwork`. The article you are reading was generated from a repository with this exact structure; you can check it out here. This figure was automatically generated from the TikZ code in `src/scripts/tree.tex` by specifying a custom command in `showyourwork.yml`.

Name	Representation
Constant	α^2
Squared Exponential	$e^{-\frac{\tau^2}{2\lambda^2}}$
Exponential	$e^{-\frac{\tau}{\lambda}}$
Matérn 3/2	$\left(1 + \frac{\sqrt{3}\tau}{\lambda}\right) e^{-\frac{\sqrt{3}\tau}{\lambda}}$
Matérn 5/2	$\left(1 + \frac{\sqrt{5}\tau}{\lambda} + \frac{5\tau^2}{3\lambda^2}\right) e^{-\frac{\sqrt{5}\tau}{\lambda}}$
Rational Quadratic	$\left(1 + \frac{\tau^2}{2\gamma\lambda^2}\right)^{-\gamma}$
Cosine	$\cos\left(\frac{2\pi\tau}{\lambda}\right)$
Sine Squared Exponential	$e^{-\Gamma \sin^2\left(\frac{\pi\tau}{\lambda}\right)}$
Stochastic Harmonic Oscillator	$\frac{\beta \sin\left(\frac{\tau\sqrt{1-\beta^2}}{\lambda}\right)}{\sqrt{1-\beta^2}} + \cos\left(\frac{\tau\sqrt{1-\beta^2}}{\lambda}\right)$

Table 1. The functional forms of various commonly used Gaussian process kernels; based on Table 1 in Aigrain & Foreman-Mackey (2022). The contents of this table are programmatically generated; the kernels are defined as `sympy` expressions and rendered into TEX form in the script `src/scripts/kernels.py`. The autogenerated `GitHub` icon in the margin links to the rule in the `Snakefile` that runs that script.

REFERENCES

Aigrain, S., & Foreman-Mackey, D. 2022, arXiv e-prints, arXiv:2209.08940. https://arxiv.org/abs/2209.08940

Crenshaw, J. F. 2022, in prep.

Crossfield, I. J. M., Biller, B., Schlieder, J. E., et al. 2014, Nature, 505, 654, doi: 10.1038/nature12955

Luger, R., Agol, E., Foreman-Mackey, D., et al. 2019, AJ, 157, 64, doi: 10.3847/1538-3881/aae8e5

Luger, R., Bedell, M., Foreman-Mackey, D., et al. 2021a, arXiv e-prints, arXiv:2110.06271. https://arxiv.org/abs/2110.06271

Luger, R., Foreman-Mackey, D., & Hedges, C. 2021b, AJ, 162, 124, doi: 10.3847/1538-3881/abfdb9

Luger, R., Foreman-Mackey, D., Hedges, C., & Hogg, D. W. 2021c, AJ, 162, 123, doi: 10.3847/1538-3881/abfdb8

Mölder, F., Jablonski, K., Letcher, B., et al. 2021, F1000Research, 10, doi: 10.12688/f1000research.29032.1

Paillas, E., Cuesta-Lazaro, C., Zarrouk, P., et al. 2022, arXiv e-prints, arXiv:2209.04310. https://arxiv.org/abs/2209.04310

Wagg, T., Breivik, K., & de Mink, S. E. 2022, ApJS, 260, 52, doi: 10.3847/1538-4365/ac5c52

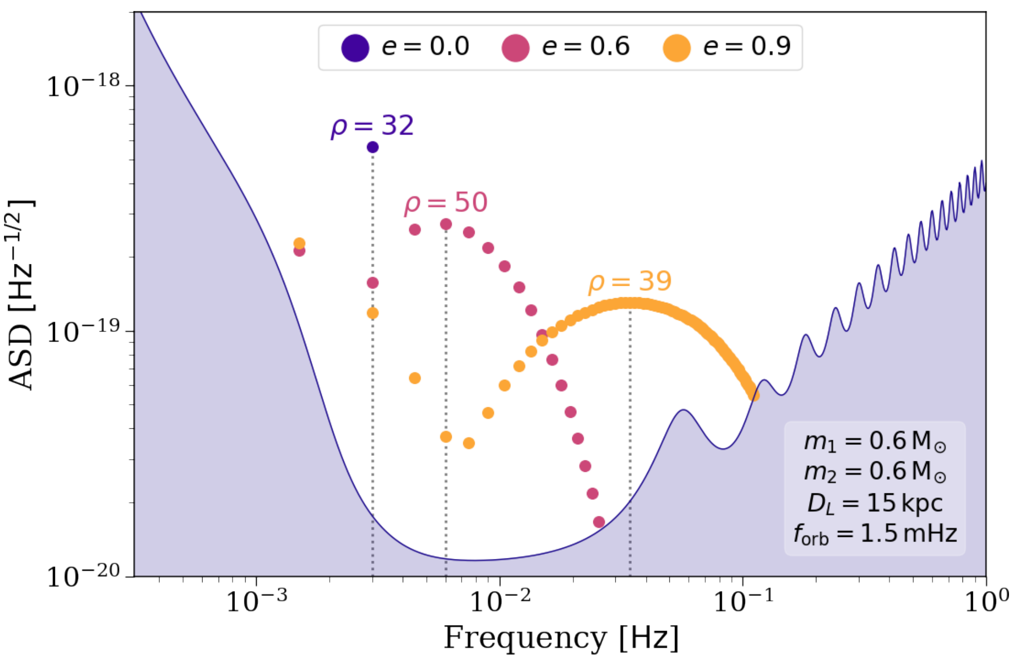

Figure 3. The effect of binary eccentricity on the detectability of a *LISA* gravitational wave source; reproduced from Figure 3 in Wagg et al. (2022). This figure was automatically generated from the script `src/scripts/eccentricity.py`. The `GitHub` icon in the margin is a clickable link pointing to the exact version of the script on `GitHub` used to produce this figure.

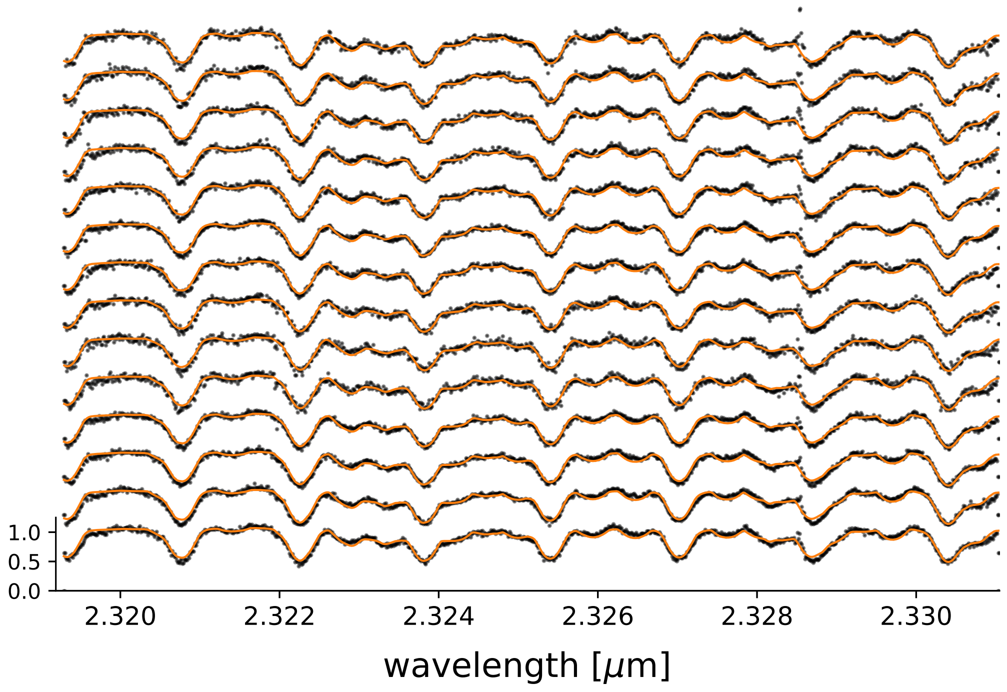

Figure 4. 16 *CRIRES* spectra of WISE 1049-5319B spanning a full rotation period of the brown dwarf; adapted from Figure 14 in Luger et al. (2021a) and based on data from Crossfield et al. (2014). This figure was automatically generated from the script `src/scripts/luhman16b.py` and a dataset downloaded from `Zenodo`. In addition to the `GitHub` icon linking to the script, this figure also has a dataset icon linking to the Zenodo deposit that hosts the data needed to generate it.

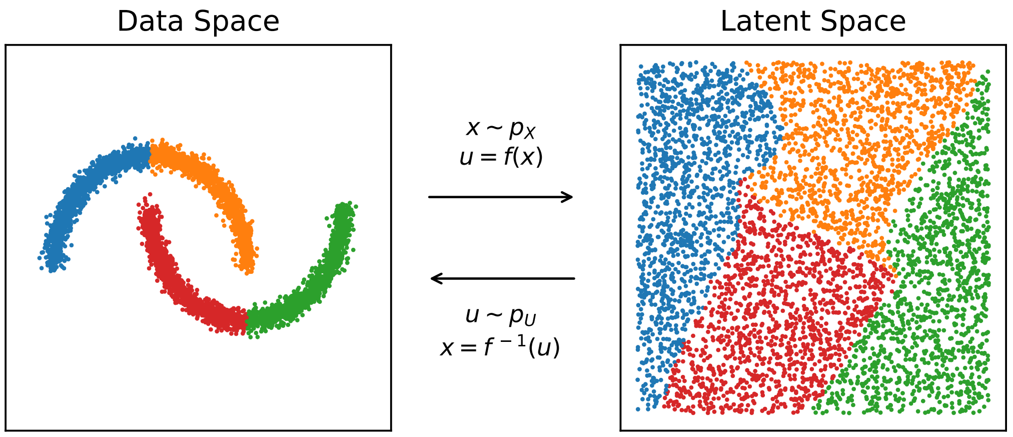

Figure 5. A normalizing flow demonstrated on the two moons data set from `scikit-learn`; reproduced from Figure 1 in Crenshaw (2022). This figure was automatically generated from the script `src/scripts/two_moons.py` and an intermediate dataset that was automatically cached on `Zenodo`.

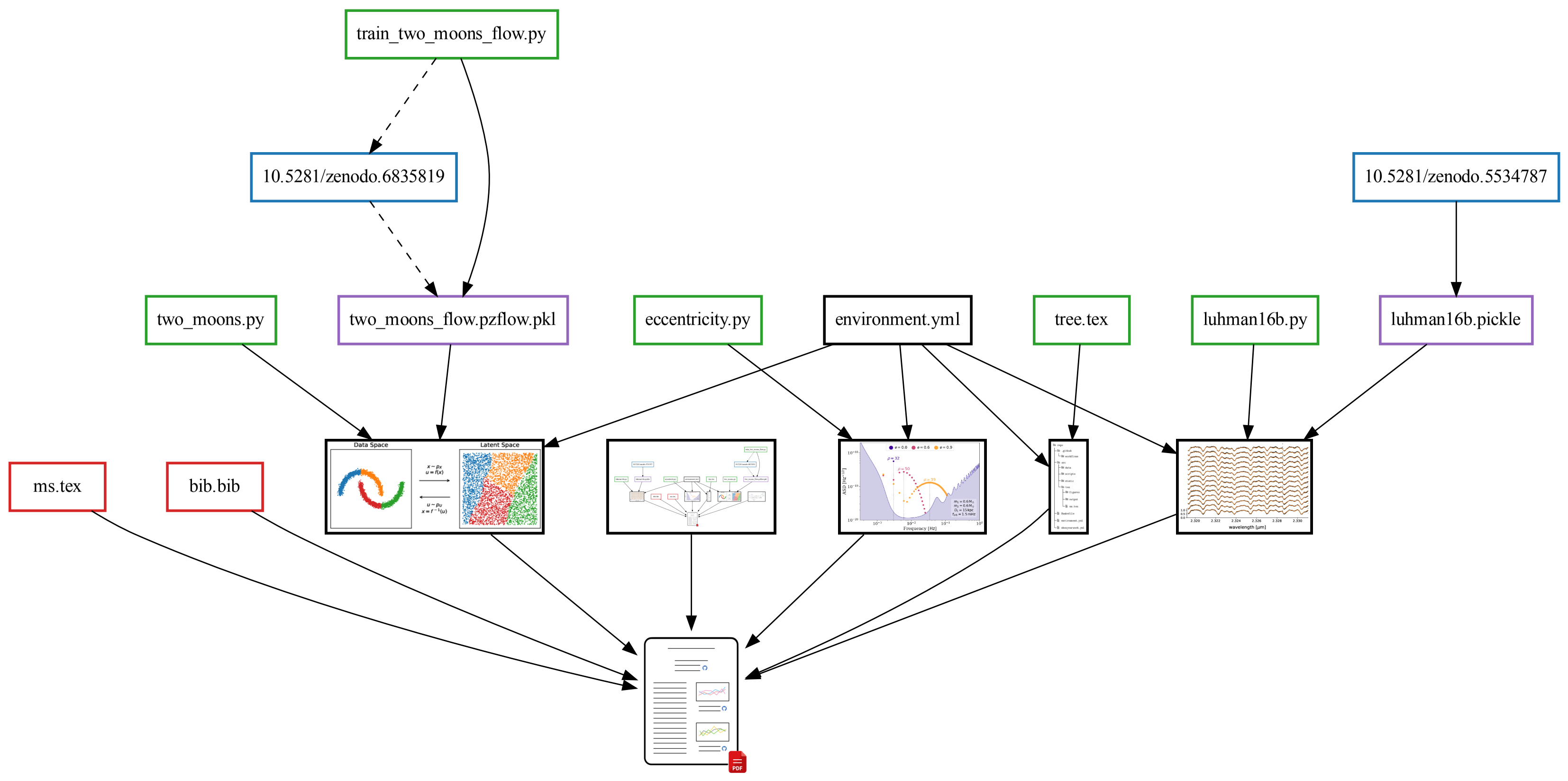

Figure 6. A directed acyclic graph (DAG) showing the complete list of dependencies for the article. Scripts are shown in green, `Zenodo` deposits in blue, datasets in purple, and TEX files in red. This figure is located in the `src/static` directory and, unlike the other figures in this article, is version controlled by `git`. The `src/static` directory is reserved for figures that are not programmatically generated and simply get copied over to the output directory at compile time.