Draft version December 23, 2021
Typeset using LaTeX/show your work! **twocolumn** style in AASTeX631

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

Rodrigo Luger and Others TBD

ABSTRACT

Abstract coming soon.

1. INTRODUCTION

This paper introduces show your work!, a workflow that enables the creation and distribution of fully reproducible and open source scientific articles. As astronomical software becomes increasingly more complex, and as research results become increasingly more interdependent, it is crucial to ensure the validity and correctness of papers published in the field. However, the current peer review system is simply not set up to do this, since 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. 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.

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 exactly replicate.

3. The majority of astronomical code is not open source and simply cannot be vetted by third parties. 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; 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.

show your work! was designed to tackle these three issues, making it easy to develop, publish, and distribute truly open and reproducible research papers in astronomy. It exists as a `GitHub` template repository, which can be cloned at the click of a button to set up a new article. Users then add their LaTeX manuscript, bibliography, scripts used in the generation of the paper's figures, an anaconda environment specification, and instructions on how to download any external datasets required by the figures. Every time the user pushes a new commit to `GitHub`, the article is automatically built on the cloud using `GitHub Actions` and the resulting PDF is pushed to a separate branch of the repository. The build step—which sets up the conda environment, generates all figures from scratch (with intelligent caching), and compiles the PDF—acts as a unit test for the paper. If it passes, the paper is (by definition) reproducible.

The workflow works out of the box for simple projects, in which each figure is generated by running a given `Python` 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 on clusters. 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 stuff under the hood is there to make the workflow as flexible and customizable as possible; see the documentation for a list of examples of custom workflows.

Papers that use this workflow can be reproduced by cloning the repository and running `make`. Furthermore, these papers 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 `Zenodo`-hosted dataset(s) used in their creation.

2. EXAMPLES

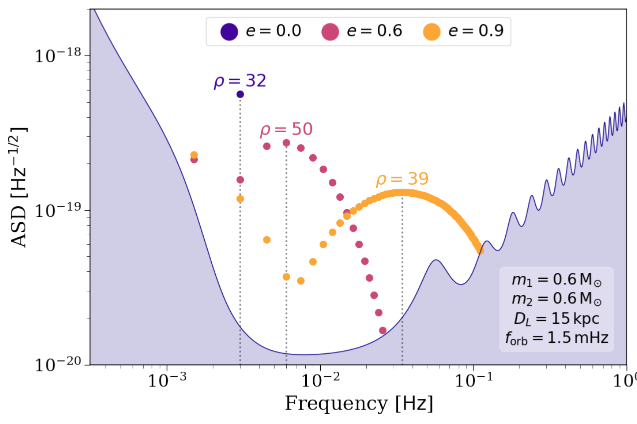

Figure 1. The effect of binary eccentricity on the detectability of a *LISA* gravitational wave source; reproduced from Figure 3 in Wagg et al. (2021). This figure was automatically generated from the script `src/figures/eccentricity.py`.

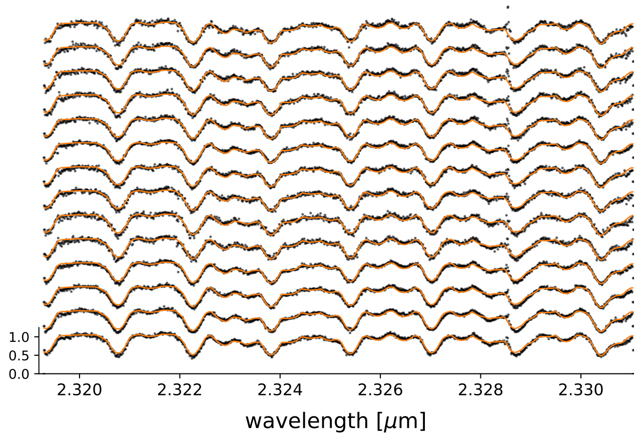

Figure 2. 16 *CRIRES* spectra of WISE 1049-5319B spanning a full rotation period of the brown dwarf; adapted from Figure 14 in Luger et al. (2021) and based on data from Crossfield et al. (2014). This figure was automatically generated from the script `src/figures/luhman16b.py` and a dataset downloaded from `Zenodo`.

3. PREREQUISITES

Conda. Linux/UNIX/Mac. GitHub account.

4. STARTING A PROJECT

Users can start a new project by creating a fresh repository based on the `showyourwork-template`. This will create a new repository under the user's `GitHub` account and trigger a `GitHub Action` that will finish the setup process and build a skeleton version of the paper. After a few minutes, a banner will appear on the repository's main page with links to the build logs and the compiled article PDF (see Figure 6). At this point, the repository can be cloned locally, e.g.,

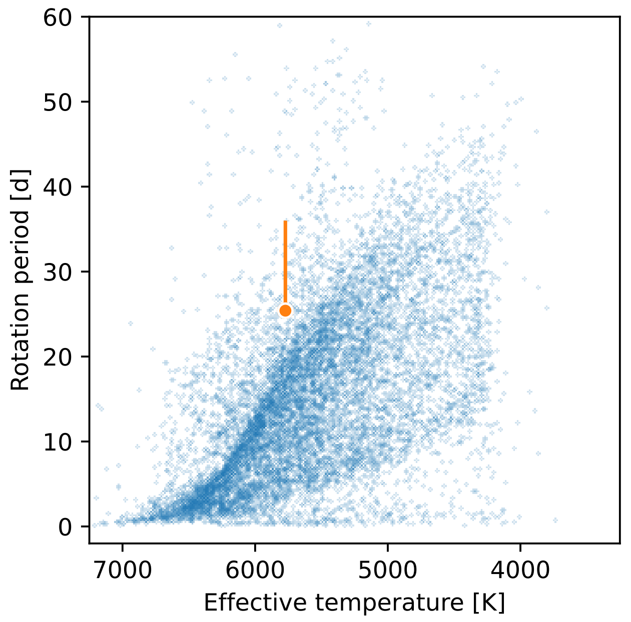

Figure 3. A pile-up of stars in the rotation period-temperature space at slightly faster rotation than the Sun (orange dot); adapted from Figure 1 in David et al. (in prep). This figure was automatically generated from the script `src/figures/rossbyridge.py`, a dataset downloaded from `Zenodo`, and the helper script `src/figures/helpers/sun.py`.

```
git clone https://github.com/user/repo
cd repo
```

where `user` and `repo` are the `GitHub` user name and repository name, respectively. The user may now edit the manuscript (`src/ms.tex`), add figure scripts to the `src/figures` directory, etc. (see §8). The article may be built locally by running `make` in the top level directory (see §6). Upon committing and pushing the changes to the remote repository, the `GitHub Action` will be triggered automatically, generating a new article PDF in sync with the latest changes (see §7).

5. REPRODUCING A PROJECT

Any project based on *show your work!* can be reproduced by cloning its `GitHub` repository and running `make`:

```
git clone https://github.com/user/repo
cd repo
make
```

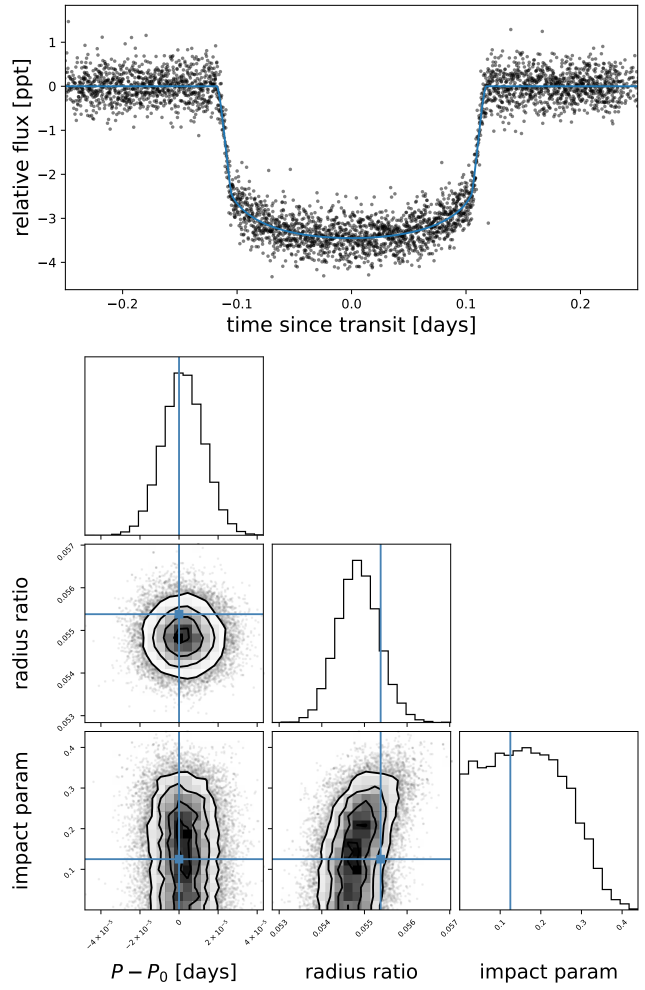

Figure 4. The phase-folded transit of HD 118203b in *TESS* (*top*) and the inferred joint posterior distributions over its period, radius, and impact parameter (*bottom*); adapted from the `exoplanet` documentation (Foreman-Mackey et al. 2021). Both figures were generated from the script `src/figures/HD118203.py`. They both depend on an intermediate result (a dataset containing the MCMC posterior samples), which can either be generated from scratch by running `src/analysis/HD118203.py` or by downloading a cached version from `Zenodo`.

This will create a fresh `conda` environment and run the `Snakemake` workflow to build the article from scratch, which may entail the execution of computationally-intensive scripts/tasks. These may sometimes be skipped by instead running

```
make fast
```

which will download any available intermediate files from `Zenodo` instead of generating them from scratch. See §6 for details.

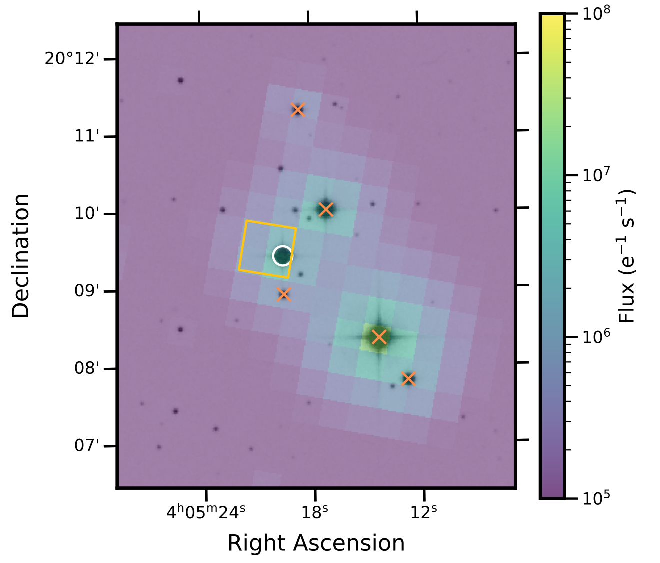

Figure 5. *TESS* target pixel file (TPF) of V1298 Tau overlaid with an r-band sky image from the Digitized Sky Survey (DSS); reproduced from Figure 1 in Feinstein et al. (2021). This figure was downloaded directly from the `GitHub` repository for that paper by providing a custom rule in the `Snakefile`. By default, margin icons are not added to figures with custom rules. Here we manually add an icon linking to the original figure using the `\marginicon` command.

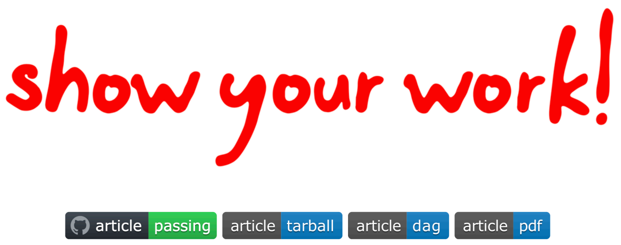

Figure 6. The default `README.md` banner in a repository instantiated from the `showyourwork-template`, with links to the `GitHub Action` build logs, a tarball containing the `TeX` source for the article, a directed acyclic graph (DAG) of the build process, and the compiled article PDF, respectively. Note that since this figure is a screenshot, we place it in the `src/static` folder and `git`-commit it. The workflow skips the generation step for any figures in that directory; note the absence of a margin icon next to this caption.

6. LOCAL BUILDS

Coming soon.

7. REMOTE BUILDS

Coming soon.

8. REPOSITORY STRUCTURE

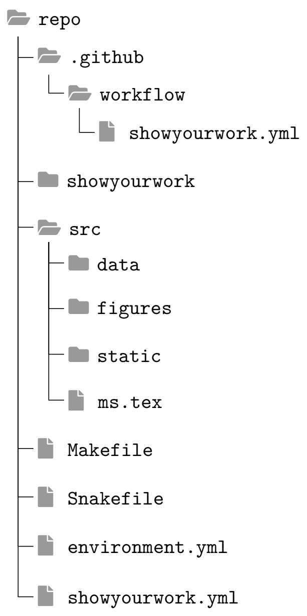

Figure 7. The basic repository structure for an open source scientific article based on `showyourwork-template`. See text for details.

Figure 7 shows the basic directory structure for a repository instantiated from `showyourwork-template`. The main components are:

- `.github/workflow/showyourwork.yml`: The configuration file for the `GitHub Actions` workflow, with instructions on how to set up the virtual environment, checkout the repository, and invoke the `showyourwork-action` to build and publish the article PDF.
- `showyourwork`: The `git` submodule containing the main workflow, which manages the entire build process for the article.
- `src`: A directory containing all of the scripts needed to generate the article.
- `src/data`: A directory containing programmatically generated (or downloaded) datasets and dependencies that are not tracked by `git`; this directory should be empty when the repository is first cloned.
- `src/figures`: A directory containing all of the scripts needed to generate the article figures. When the article is built, figure (output) files will be stored here, but they should not be tracked by `git`.
- `src/static`: A directory containing miscellaneous files (usually figures) that are tracked by `git`. These may include photographs, flowcharts, or other figures that cannot be programmatically generated.
- `src/ms.tex`: The main `TeX` article manuscript.
- `Makefile`: A read-only UNIX/Linux makefile that enables users to build the article by running `make` in the command-line terminal.
- `Snakefile`: A file containing the rules for the `Snakemake` workflow; see §13.1 By default, it simply imports all of the rules defined in `showyourwork/workflow/Snakefile`, but users can edit this file to add new rules or customize existing rules.
- `environment.yml`: The `conda` environment file specifying all of the direct software dependencies of the workflow; see §13.3
- `showyourwork.yml`: The main configuration file for the workflow, where users can specify figure and dataset dependencies, instructions for downloading datasets from `Zenodo`, etc.; see §13.2

9. AUTOMATIC FIGURE GENERATION

The workflow automatically determines the relationship between a figure (e.g., a `*.pdf` image) and the script that generated it (e.g., a `*.py` file) via inspection of the `figure` environment in the `TeX` file. Specifically, it inspects the `\includegraphics` and `\label` commands to infer the name of the figure and the script that generated it, respectively. Consider the following snippet, used to generate Figure 1:

```
\begin{figure}
  \begin{centering}
    \includegraphics{figures/eccentricity.pdf}
    \caption{...}
    \label{fig:eccentricity}
  \end{centering}
\end{figure}
```

The convention in show your work! is to infer the parent script for a figure referenced in an `includegraphics` call (in this case, `figures/eccentricity.pdf`) from the figure `label`. Specifically, if a label starts with `fig:`, the remainder of the label (e.g., `eccentricity`) is interpreted as the name of a script in the `figures/` directory which should be executed to produce that figure. By default, figure scripts are expected to be `Python` scripts, so in this case the workflow will attempt to run

```
python eccentricity.py
```

from within the `figures/` directory to generate `eccentricity.pdf`.

This behavior may be customized in several ways. For instance, the figure script need not be a `Python` script; see §10 for information on how to include scripts in any language. Another common scenario involves a script that generates multiple figures. These can all be included (using `\includegraphics`) in the same figure environment with a single shared label (denoting the name of the script), or in multiple different figure environments by appending an arbitrary suffix to each figure label to make them all unique (i.e., `fig:eccentricity:one`, `fig:eccentricity:two`, etc.) And finally, this behavior may be overridden entirely by either placing the figure in the `src/static` directory (and committing it directly to the `GitHub` repository) or by supplying a label that instead begins with `fig*:`, which instructs the workflow to not attempt to programmatically generate the figure. In the latter case, a custom rule must be provided in the `Snakefile` to generate it (see, e.g., Figure 5).

10. SUPPORT FOR OTHER LANGUAGES

Scripts in languages other than `Python` are supported via the inclusion of entries under the `scripts` key in the `showyourwork.yml` config file. Consider Figure 7, which was generated from the `TeX` file `figures/tree.tex`, which contains a `TikZ` picture. In the main manuscript, we include the figure in the usual way, i.e.,

```
\begin{figure}
  \begin{centering}
    \includegraphics{figures/tree.pdf}
    \caption{...}
    \label{fig:tree}
  \end{centering}
\end{figure}
```

By default, as we saw in §9, this instructs the workflow to execute a file called `figures/tree.py` to generate the corresponding figure. However, in the present case, that file does not exist; instead, we have a file called `figures/tree.tex`, which we would like to compile into a `PDF` using te `tectonic` engine. We can instruct the workflow to do this by specifying the following in `showyourwork.yml`:

```
scripts:
  tex:
    cd {script.path} && tectonic {script.name}
```

Each entry under the `scripts` key should be a file extension, and under each one, a string containing a shell command that instructs the workflow how to execute a given `script` to produce a given `output`. For convenience, the following placeholders are recognized and expand as follows at runtime:

- `{script}`: The full path to the input script.
- `{script.path}`: The full path to the directory containing the input script.
- `{script.name}`: The name of the input script (without the path).
- `{output}`: The full path to the output file.
- `{output.path}`: The full path to the directory containing the output file.
- `{output.name}`: The name of the output file (without the path).

If additional customization is needed, such as the need to provide command-line arguments that are specific to individual figures, users should instead provide custom rules in the `Snakefile` (§11).

11. ARBITRARY RULES

Coming soon.

12. ZENODO INTERFACE

Coming soon.

13. THIS PAPER

13.1. *The `Snakefile`*

```
# User config
configfile: "showyourwork.yml"

# Import the showyourwork module
module showyourwork:
    snakefile:
        "showyourwork/workflow/Snakefile"
    config:
        config

# Use all default rules
use rule * from showyourwork

# Custom rule to download the V1298 figure
rule v1298tau:
    output:
        "src/figures/v1298tau.pdf"
    shell:
        "curl -L https://github.com/afeinstein20/
            v1298tau_tess/raw/c670e0/src/static/
            TESSaperture.pdf --output {output}"
```

13.2. *The `showyourwork.yml` file*

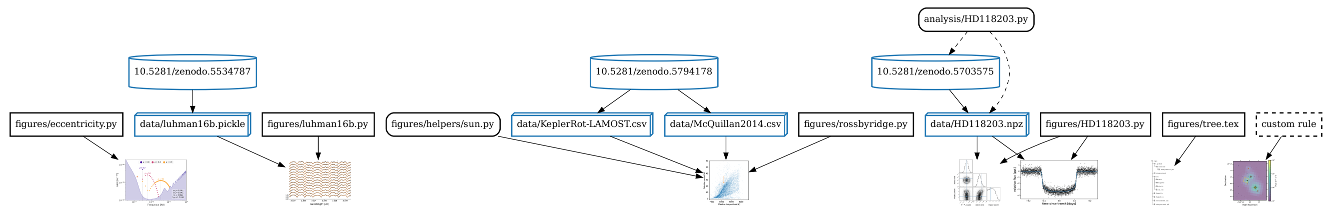

Figure 8. A directed acyclic graph (DAG) showing the build process for each of the (other) figures in this article. Figure scripts are represented by black rectangles; helper scripts (such as ones imported by figure scripts or used in dataset generation) are similar, but have rounded edges. Blue cylinders correspond to Zenodo records; blue boxes correspond to datasets.

```
# Generate a DAG of the build
dag: true

# Dependices of each script
dependencies:
  src/figures/HD118203.py:
    - src/data/HD118203.npz

  src/figures/luhman16b.py:
    - src/data/luhman16b.pickle

  src/figures/rossbyridge.py:
    - src/figures/helpers/sun.py
    - src/data/McQuillan2014.csv
    - src/data/KeplerRot-LAMOST.csv

  src/ms.tex:
    - src/style.tex

# Zenodo dataset metadata
zenodo:

  # A showyourwork-managed dataset
  src/data/HD118203.npz:
    id: 5703575
    script: src/analysis/HD118203.py
    title: HD118203 transit analysis
    description: >-
      Analysis of the transits of HD118203 in TESS
      based on arxiv.org/abs/1911.05150 and
      gallery.exoplanet.codes/tutorials/quick-tess
    creators:
      - Luger, Rodrigo

  # A static download-only dataset
  src/data/luhman16b.pickle:
    id: 5534787

  # A static download-only dataset
  src/data/McQuillan2014.csv:
    id: 5794178

  # A static download-only dataset
  src/data/KeplerRot-LAMOST.csv:
    id: 5794178

# Instructions on how to execute custom scripts
scripts:
  tex:
    cd {script.path} && tectonic {script.name}
```

13.3. *The* `environment.yml` *file*

```
channels:
  - defaults
  - conda-forge
dependencies:
  - numpy=1.19.2
  - pip=21.0.1
  - python=3.9
  - tectonic=0.8.0
  - pip:
      - matplotlib==3.4.3
      - lightkurve==2.0.11
      - celerite2==0.2.0
      - exoplanet==0.5.1
      - pymc3_ext==0.1.1
      - corner==2.2.1
      - legwork==0.1.6
```

REFERENCES

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

Feinstein, A. D., David, T. J., Montet, B. T., et al. 2021, arXiv e-prints, arXiv:2111.08660. https://arxiv.org/abs/2111.08660

Foreman-Mackey, D., Luger, R., Agol, E., et al. 2021, arXiv e-prints, arXiv:2105.01994. https://arxiv.org/abs/2105.01994

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

Wagg, T., Breivik, K., & de Mink, S. E. 2021, arXiv e-prints, arXiv:2111.08717. https://arxiv.org/abs/2111.08717