Astronomical Data Analysis Software and Systems XXVIII
ASP Conference Series, Vol. 523
P.J. Teuben, M.W. Pound, B.A. Thomas, and E.M. Warner, eds.

Versioned Executable User Documentation for In-development Science Tools

C. Boisson[1], J. E. Ruiz[2], C. Deil,[3] A. Donath,[3] and B. Khelifi[4]

[1]*LUTH, Observatoire de Paris, Paris, France;* catherine.boisson@obspm.fr

[2]*Instituto de Astrofísica de Andalucía - CSIC, Granada, Spain;* jer@iaa.es

[3]*Max-Planck-Institut für Kernphysik, Heidelberg, Germany*

[4]*APC - AstroParticule et Cosmologie, Université Paris Diderot, Paris, France*

Abstract. One key aspect of software development is feedback from users. This community is not always aware of the modifications made in the code base, neither they use the tools and practices followed by the developers to deal with a non-stable software in continuous evolution. The open-source Python package for gamma-ray astronomy Gammapy, provides its user community with versioned computing environments and executable documentation, in the form of Jupyter notebooks and virtual environment technologies that are versioned coupled with the code base. We find that this set-up greatly improves the user experience for a software in prototyping phase, as well as provides a good workflow to maintain an up-to-date documentation.

1. The Gammapy package

Gammapy[1] (Deil et al. 2017) is a community-developed, open-source Python package for high-level γ-ray data analysis, built on Numpy (Oliphant 2006) and Astropy (Greenfield et al. 2013). It provides functionalities to create sky images, spectra, light curves and source catalogs from event lists and instrument response information, determining the position, morphology and flux of γ-ray sources. Gammapy is a prototype for the Cherenkov Telescope Array (CTA) science tools. It has been used to simulate and analyze data for the CTA, as well as for the main IACT (Imaging Air Cherenkov Telescopes) facilities like H.E.S.S., MAGIC, VERITAS, and the Fermi-LAT telescope.

2. The user role for *in-development* software

The Gammapy package is a software in evolution, rapidly changing, where many developments are ongoing. As such, it is a place for γ-ray astronomers to share their code and contribute. So far, developer coding sprints have been scheduled to happen with the same regularity of user hands-on sessions held in CTA consortium meetings. A small collection of tutorials, in the form of Jupyter notebooks (Kluyver et al. 2016), was originally provided for learning purposes as a complement to the web published

[1]https://gammapy.org

documentation. The documentation and notebooks had to be updated frequently and separately. With the rise in the contributor and user base, as well as in the development activity, it emerged the need to have the notebooks integrated in the documentation, and versioned coupled with the code base. Users should easily bring to life the different versions of the published tutorials, so they could be able reproduce and re-use them for their own goals.

3. Integrating versioned and executable user documentation

A graphic description of the technical set-up may be found in Figure 1. The source files for the documentation, the Jupyter notebooks, and the code base are maintained in a single GitHub repository[2]. The code base and notebooks are continuously built and tested in different environments using the continuous integration service Travis CI.

The notebooks are stored stripped of their output cells, which greatly helps in identifying differences in the contribution review. We have implemented our own solution for execution validation of the notebooks: these are executed with *jupyter nbconvert*, the output cells parsed and, in case an error is found in one of the output cells, validation fails. Code formatting of their input cells is also done parsing their content with the help of the *Black* package.

During the documentation building process, the stripped notebooks are executed, so to refill their output cells, and later merged into the documentation with Sphinx and the *nbsphinx* extension. It is possible to skip the notebooks integration, since the execution step may be quite time consuming. In this sense, only fast simple notebooks are chosen to be part of the tutorials. The resulting web published tutorials[3] provide links to versioned *playground* spaces in myBinder (Project Jupyter et al. 2018), where they may be executed on-line in virtual environments hosted in the myBinder infrastructure. These environments are built with the help of a *Dockerfile* placed at the top level of the Github repository.

The user may retrieve specific *tutorial bundles* using the `gammapy download` command. One *tutorial bundle* is composed of a *conda* file environment, Jupyter notebooks, and the datasets needed to reproduce them. For each of these bundles we specify, in centralized index lookup files, the required computing environment, which tutorials and datasets to provide, and where to fetch them from. Deterministic computing environments are defined in the form of *conda* configuration files, with pinned version numbers for each dependency package.

4. Command line tools and functionalities

We have developed a set of tools for users to download the *tutorial bundles*, and for documentation maintainers to prepare and validate them. The `gammapy download` command shown in Figure 2, provides users with the means to retrieve a *tutorial bundle* or any tutorial-related asset (i.e. dataset or Jupyter notebook) for a specific version of the Gammapy package, so they can activate and use it at their will.

---

[2] `https://github.com/gammapy/gammapy`

[3] `https://docs.gammapy.org/0.8/tutorials.html`

On the other side, the `gammapy jupyter` command provides documentation maintainers with a tool to seamlessly manage and integrate Jupyter notebooks into the documentation. This command provides functionalities for execution validation, code formatting, output cells stripping and straightforward execution of the notebooks.

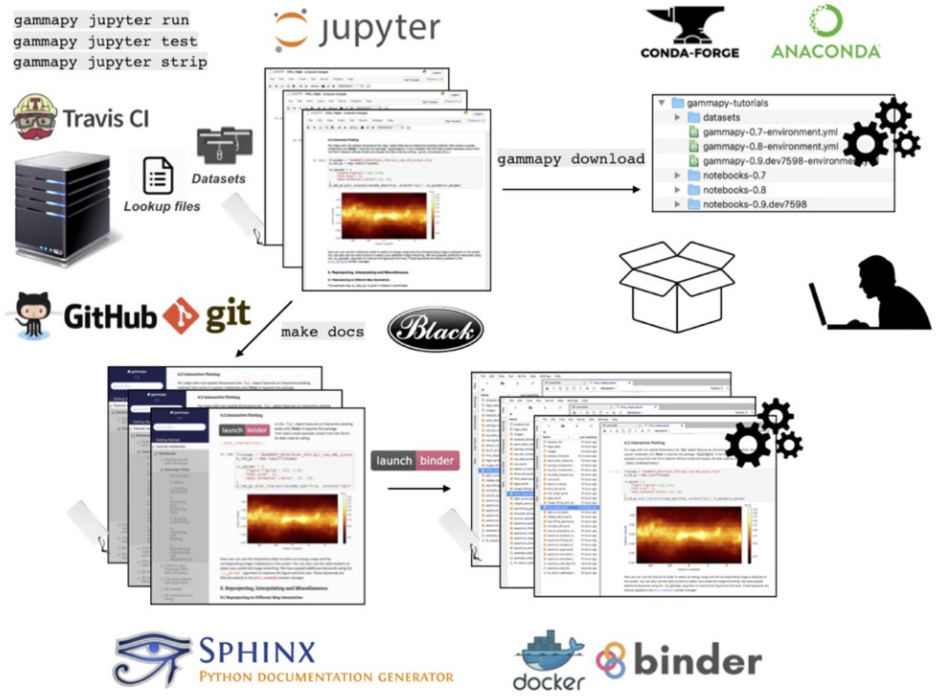

Figure 1. Technical set-up for building and shipping *tutorial bundles*. The maintainers build the documentation from stripped Jupyter notebooks using Sphinx. The resulting published tutorials provide links to access myBinder *playground* spaces, where the notebooks may be executed on-line. They may be also downloaded to the user desktop, together with the *conda* virtual environment and the datasets needed.

```
$ gammapy download tutorials --release 0.8
INFO:gammapy.scripts.downloadclass:Content will be downloaded in gammapy-tutorials/notebooks-0.8
Downloading files [==================================================] 100%
INFO:gammapy.scripts.downloadclass:Content will be downloaded in gammapy-tutorials/datasets
Downloading files [==================================================] 100%


***** Enter the following commands below to get started with Gammapy
cd gammapy-tutorials
conda env create -f gammapy-0.8-environment.yml
conda activate gammapy-0.8
export GAMMAPY_DATA=/Users/jer/Desktop/gammapy-tutorials/datasets
jupyter lab
```

Figure 2. Users may retrieve versioned *tutorial bundles*, or specific related digital assets like notebooks and datasets, using the `gammapy download` command.

5. Conclusions

We have presented a novel set-up to publish and distribute executable documentation version coupled with the code base of the Gammapy package. These *tutorial bundles* are composed of a *conda* virtual environment, Jupyter notebooks, and the datasets needed to reproduce them. The web published tutorials may be also executed on-line for learning purposes in myBinder *playground* spaces. We find that this set-up greatly improves the user experience for a software in prototyping phase, where maintenance and delivery of the *tutorial bundles* are done with on purpose command line tools.

Acknowledgments. The authors would like to thank the following projects and services that are used in the exposed work: *Git*[4], *GitHub*[5], *Travis CI*[6], *Anaconda*[7], *conda*[8], *conda-forge*[9], *Sphinx*[10], *nbsphinx*[11], and *Black*[12]. This work was partially funded by ASTERICS (http://www. asterics2020.eu/), a project supported by the European Commission Framework Programme Horizon 2020 Research and Innovation action under grant agreement n. 653477

References

Deil, C., Donath, A., Owen, E., Terrier, R., Bühler, R., & Armstrong, T. 2017, Gammapy: Python toolbox for gamma-ray astronomy, Astrophysics Source Code Library. 1711.014

Greenfield, P., Robitaille, T., Tollerud, E., Aldcroft, T., Barbary, K., Barrett, P., Bray, E., Crighton, N., Conley, A., Conseil, S., Davis, M., Deil, C., Dencheva, N., Droettboom, M., Ferguson, H., Ginsburg, A., Grollier, F., Moritz Günther, H., Hanley, C., Hsu, J. C., Kerzendorf, W., Kramer, R., Lian Lim, P., Muna, D., Nair, P., Price-Whelan, A., Shiga, D., Singer, L., Taylor, J., Turner, J., Woillez, J., & Zabalza, V. 2013, Astropy: Community Python library for astronomy, Astrophysics Source Code Library. 1304.002

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, edited by F. Loizides, & B. Schmidt (IOS Press), 87

Oliphant, T. 2006, Guide to NumPy (Trelgol Publishing)

Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroff, Pacer, M., Yuvi Panda, Fernando Perez, Benjamin Ragan Kelley, & Carol Willing 2018, in Proceedings of the 17th Python in Science Conference, edited by Fatih Akici, David Lippa, Dillon Niederhut, & M. Pacer, 113

---

[4] https://git-scm.com

[5] https://github.com

[6] https://travis-ci.com

[7] https://www.anaconda.com

[8] https://conda.io

[9] https://conda-forge.org

[10] https://www.sphinx-doc.org

[11] https://github.com/spatialaudio/nbsphinx

[12] https://github.com/ambv/black