Versioned executable user documentation for in-development science tools

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

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

[2]*LUTH - Observatoire de Paris, Paris, France*

[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 and beta-testers. This community is not always aware of the developments undertaken 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 beta-tester user community with versioned reproducible environments and executable documentation, in the form of Jupyter notebooks tutorials and virtual environments technologies that are versioned coupled with the code base.* We believe this set-up greatly improves the user experience for a software in prototyping phase, as well as providing 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 morphology, flux and position of γ-ray sources.

Gammapy has been proposed as a prototype for the Cherenkov Telescope Array (CTA) science tools, and it has been used to simulate and analyse data from the main IACT (Imaging Air Cherenkov Telescopes) facilities like H.E.S.S., MAGIC, and VERITAS, as well as Fermi-LAT data and simulated observations for CTA.

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

The Gammapy package is a software still in evolution, rapidly changing, where many developments are ongoing. As such it is a place for Python-coding γ-ray astronomers to share their code, contribute and collaborate. So far, developer coding sprints have been scheduled to happen with the same regularity of the user hands-on tutorials held in CTA

[1]https://gammapy.org

consortium meetings. Users and potential contributors benefit from the documentation of the Gammapy API, as well as from a collection of tutorials in the form of Jupyter notebooks (Kluyver et al. 2016), where both need to be updated frequently. The rise in the contributor and user base together with a high development activity, have revealed the need for reproducible executable tutorials, versioned coupled with the code base and integrated into the published documentation. In this way, users could easily bring to life, and re-use for their own purposes, the different versions of the published tutorials.

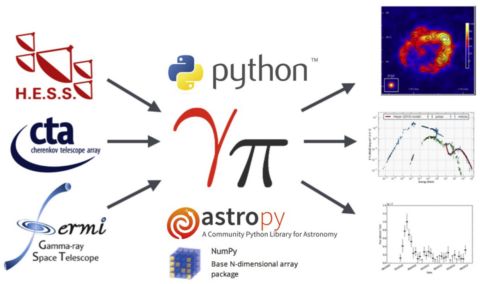

Figure 1. Gammapy is a Python package for high-level γ-ray data analysis. Using event lists, exposures and point spread functions as input you can use it to generate science results such as images, spectra, light curves or source catalogues.

3. Integrating versioned and executable user documentation

The source documentation, the tutorials, and the code base are maintained in a single versioned Github repository[2].The Jupyter notebooks are stored trimmed of their output cells, which greatly helps in difference comparisons during the contribution review. The code base and tutorials are continuously built and tested in different virtual environments using the continuous integration service Travis CI.

The documentation is generated using Sphinx, where the empty notebooks are previously executed and filled with *jupyter nbconvert*, and then integrated into the documentation with the *nbsphinx* extension. The web published tutorials[3], versioned coupled with the code base, provide links to versioned spaces of myBinder (Project Jupyter et al. 2018), where they may be executed in virtual execution environments hosted in the myBinder infrastructure. These virtual environments are built with the help of versioned *Dockerfiles* placed at the top level of the Github repository.

A small collection of versioned *tutorial bundles* composed of Jupyter notebooks, a *conda* file environment, and the datasets needed, may be downloaded to the user desktop using the `gammapy download` command. For each of this 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.

---

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

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

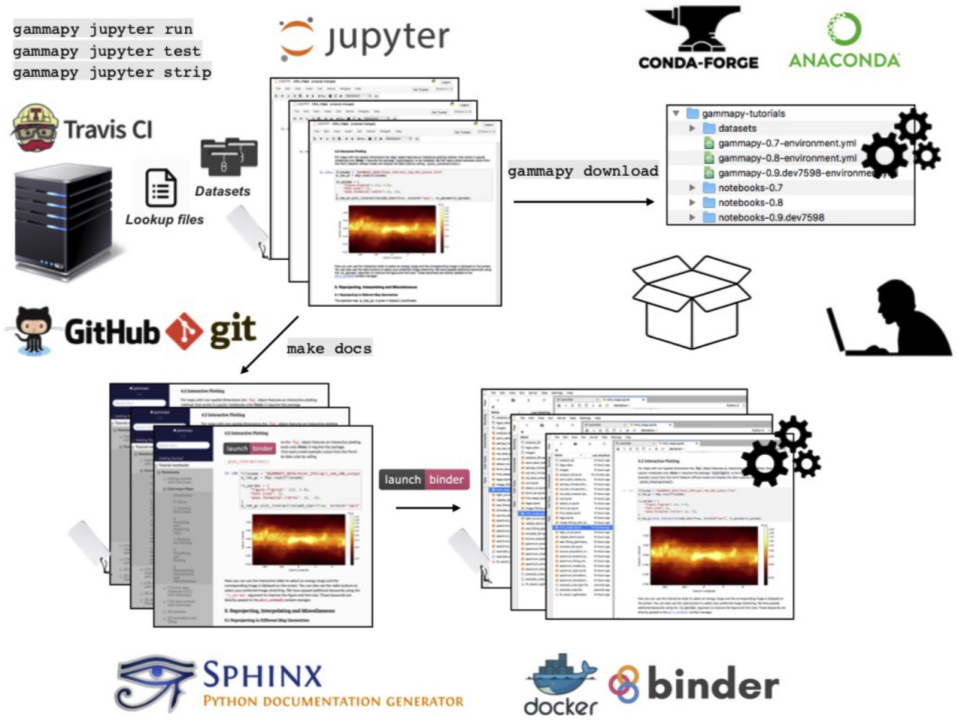

Figure 2. Technical set-up for building and shipping versioned executable user documentation. The maintainers build the documentation from empty-output Jupyter notebooks using Sphinx, and publish the generated files in a versioned Github repository. The published tutorials provide links to access myBinder spaces where the notebooks may be executed. These Jupyter notebooks may be also downloaded to the user desktop, together with the reproducible *conda* virtual environments and the datasets needed.

4. Command line tools and functionalities

We have developed a set of command line tools for users to download the versioned executable *tutorial bundles*, and for documentation maintainers to prepare and validate them. The `gammapy download` command provides users with the means to retrieve the *tutorial bundles* or any tutorial-related asset (i.e. dataset or Jupyter notebook) for a specific version of the Gammapy package.

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 straight forward execution of the notebooks. We have implemented our own solution for execution validation of 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. In the documentation building process, it is possible to add or skip the notebooks integration step.

```
$ 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 3. Users may retrieve versioned executable tutorials, or specific related digital assets like notebooks and datasets, using the `gammapy download` command.

Acknowledgments. The authors would like to thank the communities involved in the following open-source projects: *Git*[4], *Travis CI*[5], *Anaconda*[6], *conda*[7], *conda-forge*[8], *Sphinx*[9], and *nbsphinx*[10]. The exposed work heavily relies on these software.

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://travis-ci.com`

[6] `https://www.anaconda.com`

[7] `https://conda.io`

[8] `https://conda-forge.org`

[9] `https://www.sphinx-doc.org`

[10] `https://github.com/spatialaudio/nbsphinx`