Astronomical Data Analysis Software and Systems XXVII
ASP Conference Series, Vol. 522
Pascal Ballester, Jorge Ibsen, Mauricio Solar, and Keith Shortridge, eds.

Stingray Software The Code of the Spectral-timing Revolution: Black Holes, a Library, and a GUI

Simone Migliari,[1,2] Paul Balm,[2] Ricardo Vallés,[2] Matteo Bachetti,[3] Daniela Huppenkothen,[4] Abigai Stevens,[5] Matteo Guainazzi,[6] and Erik Kuulkers[6]

[1]*ESA/ESAC, Madrid, Spain.* smigliari@sciops.esa.int

[2]*Timelab Technologies Ltd, London, UK.*

[3]*INAF-AOC, Cagliari, Italy*

[4]*University of Washington, Seattle, WA, USA*

[5]*UvA, Amsterdam, the Netherlands*

[6]*ESA/ESTEC, Noordwijk, the Netherlands*

Abstract. The use by the astrophysical community of novel high-frequency time-series analysis techniques, more recently brought outstanding advancements especially in the study of relativistic systems, such as black holes and neutron stars. One of the obstacles of the several fast-moving research groups in this field is the lack of shared, publicly available tools for analysis. Many individual tools and libraries exist, but they are typically private, self-made implementations in a variety of different languages, without a clear, independent check of reliability. Furthermore, the lack of any GUI makes the tools (more) difficult to use. We initiated an open source project that implemented timing techniques in such a way that scientists with or without good knowledge of Python can exploit the full power of X-ray data. Under the sometimes used single umbrella name of *Stingray Software*, we in fact built three: *Stingray the library*, that provides a robust, well tested and clear API in Python to perform all most common (and some advanced) timing and spectral timing analysis techniques; *HENDRICS*, a shell script interface; *DAVE*, an interactive interface in Javascript and Python that aims to implement data exploration and also reduce the learning curve for newcomers and non-programmers. Our Open Source approach involves the interested astronomical community as well as non-astronomy developers.

1. The Science Context: Black Holes and Compact Objects

One of the main goals of high energy astrophysics is to understand how matter behaves under strong gravity, near black holes where relativistic corrections to Newtonian physical laws - as predicted by Einstein - are important. The most important window to observe the regions near black holes is X-rays. X-ray radiation is produced when matter heats up while falling towards the event horizon. Since X-rays are shielded by the Earth atmosphere, our research uses data from satellites. By studying the fast-variability of this X-ray radiation (variability in the range of second to fraction-of-millisecond), we can have tremendous insights on the geometry and nature of the phenomena that occur

in the proximity of relativistic compact objects, therefore constraining the properties of the space-time in extreme-field gravity. In more recent years, new ideas about how to analyze astronomy data with new but also old, consolidated time-series analysis techniques has initiated what we call *the spectral-timing revolution*, leading to a hotbed of new developments.

2. Problem: Current Software Tools

One of the obstacles of the several fast-moving research groups in this field is the lack of shared, publicly available tools for analysis. Many individual tools and libraries exist, and some are even publicly available, but what is lacking is a coherent set for a complete analysis. Also, the lack of any GUI makes the tools (more) difficult to use. This situation hinders collaboration and mobility of researchers between groups. It is also a barrier for other researchers or students to join this field that is now living an exciting period of new discoveries. In this project, we implement new timing techniques in such a way that scientists with or without good knowledge of Python can exploit the full power of X-ray astrophysical data: 1) a well-documented API with classes and methods that can be used and modified to improve the analysis pipeline; command-line scripts to launch the analysis in batch mode; 2) a GUI for data exploration, detailed analysis and interactive adjustments.

Existing tools that we use (directly or as a source of information) to build our product are Astropy, Sherpa, SciPy and Carma. On one hand, the fact that there are many tools already available, highlights the problem that many research groups tend to use their own tools, which hampers collaboration between the groups. This project has the goal to provide a single, well-tested and well-documented package, that all groups can use together to collaborate. On the other hand, the existing tools provide an excellent source of information about which functionalities are needed and how scientists want to use them.

3. Requirements Overview

The four main requirements are:

The software has to suit a broad range of problem domains within astronomy: The software needs to be flexible and adaptable, using a programming language that is both popular among the potential users and suitable for the computational algorithmic tasks. This was the reason for choosing Python as the language for implementing the algorithms.

The software must permit data exploration: Existing software tools for analysis of variable events fall short in the area of data exploration, generally because they lack a graphical user interface. One very important aspect of the GUI is to contribute to a modern user experience.

Access to existing software is limited: Access to existing analysis tools is limited for a variety of political, legal and technical reasons. We implemented an open source software that invites improvements and development from anyone who wishes to use it. There is no limits to the access to the software, and all documentation and software will be available. The software is released under the open source license Apache v2 License, allowing the highest degree of flexibility to benefit from the software.

Existing software tools have outdated user interfaces and work-flows: Another concern with existing software packages is that these tools have grown slowly, often over decades, and that they were built by scientist users, not software engineers. This leads to software that is difficult to modify for anyone, let alone someone outside of the initial research group. It also tends to lead to software with a less than intuitive user interface.

4. Open Source development

Two imperatives in the development of the software are: 1) involvement of specialized astronomical community in the definition of the software functionalities, 2) involvement of professional developers and software engineers in the architectural design and coding of the software. These characteristics lead naturally to the development of the software as an open source. Among the benefits of developing an open source software, we highlight here six: **1. Security.** Ǵiven enough eyeballs, all bugs are shallowq́uotes what is often referred to as the Linus Law, after Linus Torvalds, the creator of Linux. Bugs in open source software also tend to get fixed immediately. **2. Quality.** Open source software gets closest to what users want because those users can have a hand in making it so. **3. Customizability.** Users can take a piece of open source software and change it and/or expand it to add the functionality they needs, therefore making it useful to a larger community. **4. Freedom.** Since one of the primary goals of the software is to become widely used as standard in the community, turning to open source means that there are no barriers in use and development manpower worldwide. **5. Flexibility.** Open source software is typically much less resource-intensive than a proprietary software, meaning that it can be run well even on older hardware. **6. Support Options.** Behind an open source software there is often support through the vibrant communities surrounding each piece of software.

The use of open source license assures that the source code will always be available and its value can be maintained. The use of a popular and suitable programming language for algorithmic tasks helps the adaptability to future requirements. A modern user interface will permit data exploration in ways that current tools do not. The open source license applicable to the software will permit free access and maximizes its use.

5. Results: the Stingray Software

This software was kick-started thanks to funding from the European Space Agency (ESA) through the project *Data Analysis of Variable Events*, or DAVE, that we also kept as the name of the GUI. Under the sometimes used single umbrella name *Stingray Software*, we in fact refer to three software: a library, a shell scripting interface and a GUI. At the time of writing, three coordinated papers with details on the three software are in preparation. We herewith show some headlines:

Stingray, the library, is a spectral-timing software package for astrophysical X-ray (and more) data. Stingray merges existing efforts for a (spectral-)timing package in Python, and is structured with the best guidelines for modern open-source programming, following the example of Astropy. Stingray aims not only at becoming a standard timing package, but at extending the implementation to the most advanced spectral

timing techniques available in the literature. Reference: D. Huppenkothen et al. in preparation. Github link: `https://github.com/StingraySoftware/stingray`

HENDRICS, the shell scripting interface, is a set of command-line scripts based on Stingray is designed to do correctly and fairly easily a quick-look (spectral-)timing analysis of X-ray data. Originally, its development as MaLTPyNT - Matteo's Libraries and Tools in Python for NuSTAR Timing - was driven by the need of performing aperiodic timing analysis on NuSTAR data. Today, this set of command line scripts is much more complete and it is capable of working with the data of many more satellites. The analysis done in HENDRICS is compatible with the graphical user interface DAVE (below), so that users will have the choice to analyze datasets with an easy interactive interface, and continue the analysis in batch mode with HENDRICS. Reference: M. Bachetti et al. in preparation. Github link: `https://github.com/StingraySoftware/HENDRICS`

DAVE, the GUI, is built on top of the Stingray library. It is intended to be used by astronomers for time-series analysis of variable sources. The goal of the GUI is to enable scientific exploration of astronomical (X-ray, but not only) observations and to do detailed analysis of data in a graphical environment. Reference: S. Migliari et al. in preparation. Github link: `https://github.com/StingraySoftware/dave`

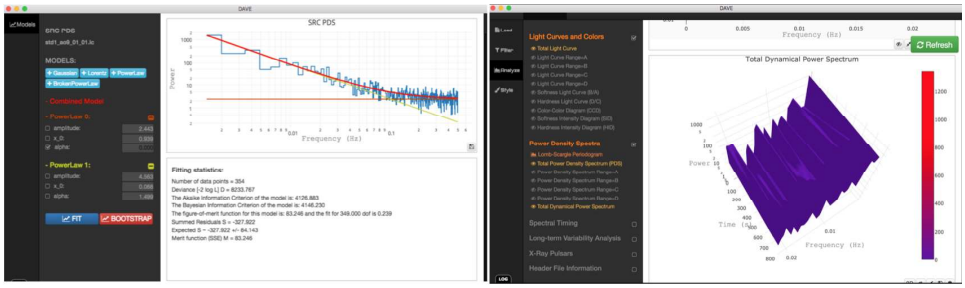

Figure 1. Examples of the DAVE GUI: *Left:* Fitting a PDS with power law models; *Right:* 3D view of the dynamical PDS.