EPJ Web of Conferences **245**, 06028 (2020) https://doi.org/10.1051/epjconf/202024506028
CHEP 2019

The Scikit HEP Project - overview and prospects

Eduardo Rodrigues[1,*], *Benjamin* Krikler[2,**], *Chris* Burr[3,***], *Dmitri* Smirnov[4,****], *Hans* Dembinski[5,†], *Henry* Schreiner[6,‡], *Jaydeep* Nandi[7,§], *Jim* Pivarski[8,¶], *Matthew* Feickert[9,‖], *Matthieu* Marinangeli[10,**], *Nick* Smith[11,††], and *Pratyush* Das[12,‡‡]

[1]University of Liverpool
[2]University of Bristol
[3]CERN
[4]BNL
[5]Technical University Dortmund
[6]Princeton University
[7]National Institute of Technology, Silchar
[8]Princeton University
[9]University of Illinois at Urbana Champaign
[10]EPFL, Lausanne
[11]FNAL
[12]Institute of Engineering and Management, Kolkata

Abstract. Scikit-HEP is a community-driven and community-oriented project with the goal of providing an ecosystem for particle physics data analysis in Python. Scikit-HEP is a toolset of approximately twenty packages and a few "affiliated" packages. It expands the typical Python data analysis tools for particle physicists. Each package focuses on a particular topic, and interacts with other packages in the toolset, where appropriate. Most of the packages are easy to install in many environments; much work has been done this year to provide binary "wheels" on PyPI and conda-forge packages. The Scikit-HEP project has been gaining interest and momentum, by building a user and developer community engaging collaboration across experiments. Some of the packages are being used by other communities, including the astroparticle physics community. An overview of the overall project and toolset will be presented, as well as a vision for development and sustainability.

[*]e-mail: eduardo.rodrigues@liverpool.ac.uk
[**]e-mail: b.krikler@bristol.ac.uk
[***]e-mail: christopher.burr@cern.ch
[****]e-mail: dsmirnov@bnl.gov
[†]e-mail: hans.dembinski@tu-dortmund.de
[‡]e-mail: henry.fredrick.schreiner@cern.ch
[§]e-mail: nandi.jaydeep296@gmail.com
[¶]e-mail: pivarski@princeton.edu
[‖]e-mail: matthew.feickert@cern.ch
[**]e-mail: matthieu.marinangeli@epfl.ch
[††]e-mail: nick.smith@cern.ch
[‡‡]e-mail: reikdas@gmail.com

1 Introduction

Python is an ever more popular programming language across a broad range of communities, notably in Data Science. Outside High Energy Physics (HEP), the Python scientific ecosystem is built atop the "building blocks" of the SciPy ecosystem of open-source software for mathematics, science, and engineering [1]. Figure 1 provides a good visual illustration of the ecosystem, which grows from foundational libraries all the way to domain-specific projects such as Astropy [2]. The ecosystem provides tools for data manipulation, visualisation, statistics, machine learning, etc.

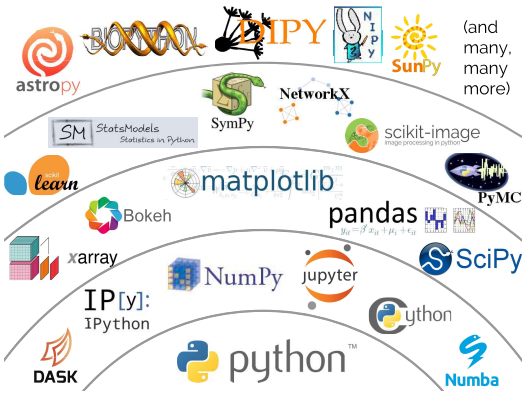

Figure 1. Schematic view of the Python scientific software ecosystem. Figure taken from Jake VanderPlas's presentation at the PyCon 2017 conference [3].

Traditionally, HEP has been evolving in a rather disjoint ecosystem based on the C++ ROOT data analysis framework [4]. Same as for the Python scientific ecosystem, it provides tools for data manipulation and modeling, for fitting, for statistics and machine learning applications. But it is a *toolkit* rather than a *toolset*, with an interface that is not that natural for Python — via bindings it provides.

Various initiatives tried to link both HEP and non-HEP worlds, at least for specific tasks. Unfortunately, the libraries were very largely developed by a single author and sustainability became quickly an issue, especially that many such authors left the field. Also, community adoption did not stick. We believed that a more generalised effort, domain-specific oriented, was the way forward, and this gave rise to the Scikit-HEP project in late 2016. It is only in 2018 that community adoption started to take up, with several project packages attracting much attention. We are proud to mention that several collider (Belle II, CMS) and non-collider (KM3NeT) experiments officially use some of Scikit-HEP in their external dependencies, as do other software projects (Coffea, zfit).

The project had been presented at the CHEP 2018 conference [5]. The present report superseeds Ref. [5] and presents the status of the project, which evolved considerably since CHEP 2018.

2 Scikit-HEP project overview

The Scikit-HEP project [6] is a community-driven and community-oriented effort with the aim of providing Particle Physics at large with a *toolset* ecosystem for data analysis in Python.

It does not attempt in any way to provide a replacement for the Python ecosystem based on the SciPy suite; it rather builds on its foundational libraries providing core and common tools for the HEP community. The grand plan of the project can be summarised in the following points:

- Create an ecosystem for particle physics data analysis in Python.
- Improve the interoperability between HEP tools and the scientific ecosystem in Python.
- Expand the typical toolset for particle physicists with high-standards, well-documented and easily installable domain-specific packages.
- Build a community of developers and users, having sustainability in mind.
- Improve discoverability of (domain specific) relevant tools.

The Scikit-HEP toolset is depicted (to a large extent) in figure 2. Some of the packages found in the GitHub organisation, such as the well-known packages `root_numpy` [7] and `root_pandas` [8], pre-dating the project, are not described in this report. They are nevertheless part of the project, but largely deprecated by the new and more versatile packages `uproot` [9] and `awkward-array` [10], see below. More importantly, it should be emphasised that most of the packages presently constituting the Scikit-HEP toolset are relatively new, having been released for the first time after the CHEP 2018 conference; these are marked as "new package" in figure 2.

The remainder of this report provides a whirlwind tour of the main packages.

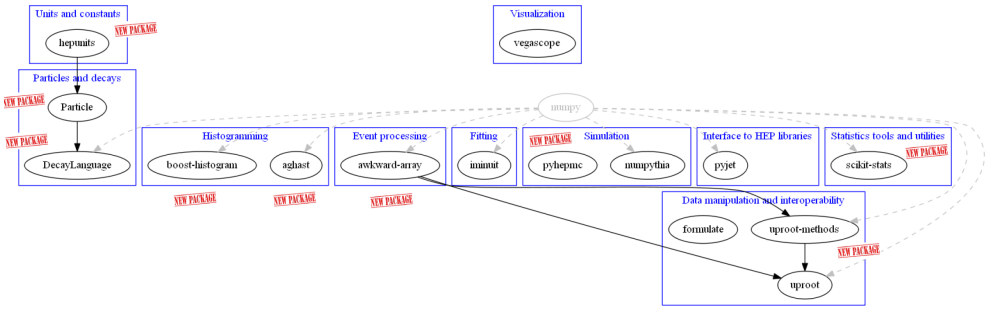

Figure 2. Overview of (most of) the packages making the Scikit-HEP toolset. For a sense of evolution all packages whose first release came out after the CHEP 2018 conference are marked as "new package". All GitHub repositories can be found at the location https://github.com/scikit-hep.

3 Whirlwind tour of Scikit-HEP packages

The obvious "point of entry" to the HEP ecosystem is via ROOT files. These can be natively and trivially read with the pure Python I/O `uproot` package [9], whose only dependencies are NumPy and Python libraries to deal with compression and decompression. The package is installable via `pip` or `conda` on virtually any computer, straighforwardly since it does *not* depend on ROOT. It has been a runaway success with over 15000 downloads per month.

`Uproot` bridges the ROOT and the NumPy-based ecosystems in its most simple incantation. Indeed a ROOT `TTree` object is read as a dictionary of arrays as follows:

```
>>> import uproot
>>> t = uproot.open("Zmumu.root")["events"]
>>> t.arrays(["px1","px2","py1","py2","M"])
{b'px1': array([-41.19528764,  35.11804977,  35.11804977, ...,  32.37749196,
         32.37749196,  32.48539387]),
 b'px2': array([ 34.14443725, -41.19528764, -40.88332344, ..., -68.04191497,
        -68.79413604, -68.79413604]),
 b'py1': array([ 17.4332439 , -16.57036233, -16.57036233, ...,   1.19940578,
          1.19940578,   1.2013503 ]),
 b'py2': array([-16.11952457,  17.4332439 ,  17.29929704, ..., -26.10584737,
        -26.39840043, -26.39840043]),
 b'M': array([82.46269156, 83.62620401, 83.30846467, ..., 95.96547966,
        96.49594381, 96.65672765])}
```

A plethora of other manipulations are available, such as iteration over several files and opening a remote file. Note that `uproot` is able since a few months to write simple ROOT objects such as `TTree`.

HEP analyses typically involve the manipulation of complex data structures that can be

- Variable length lists (jagged/ragged);
- Deeply nested (record structure);
- And of different data types in the same list (heterogeneous).

The `awkward-array` [10] package provides a way to analyse these variable-length tree-like data in Python by extending Numpy's idioms from flat arrays to arrays of data structures. The package is being reimplemented in C++, with a simpler interface and less limitations, based on acquired experience and user feedback; the developments are taking place at https://github.com/scikit-hep/awkward-1.0.

The analysis of datasets (processed *e.g.* with the two packages just described) typically involves data aggregations; these are most often in the form of one-dimensional histograms. Indeed, histogramming is central in any analysis workflow and has received much attention. The package `boost-histogram` [11] bundles the Python bindings for the performant C++14 multi-dimensional templated header-only library Boost.Histogram [12], albeit with a Pythonic API. Histograms can be defined in a very versatile way owing to the extensive types of axes (regular, variable, circular axes, etc.) and storages (interger, double, weighted values) defined, in multi dimensions. The package provides methods for selecting, rebinning, and projecting into lower-dimensional space. It is a high-performance histogramming package naturally talking to the NumPy ecosystem. Its interface is simple and user-friendly:

```
>>> import boost_histogram as bh
>>>
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>>
>>> hist = bh.Histogram(bh.axis.Regular(30, 0, 2*np.pi, circular=True))
>>> hist.fill(np.random.uniform(0, np.pi*4, size=300))
>>>
>>> plothist = lambda h: plt.bar(h.axes[0].centers, h, width=h.axes[0].widths
                                 )
>>> ax = plt.subplot(111, polar=True)
>>> plothist(hist);
```

This code snippet produces this output (figure 3):

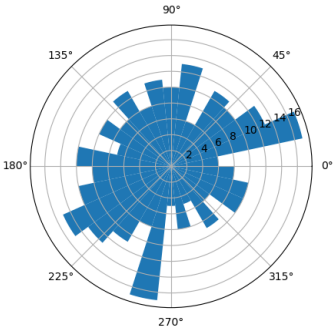

Figure 3. Example of a `boost-histogram` histogram with circular axes.

Continuing a common flow of work in a HEP analysis, it is important to mention the `iminuit` fitting package [13], the Python interface (bindings) to the `Minuit2` C++ package [14] used across particle physics. `Minuit2` is most commonly used for likelihood fits of models to data, and to get model parameter error estimates from likelihood profile analysis. The bindings constitute an important building block for sophisticated data modelling packages. It is used in other HEP packages and in various astroparticle physics packages.

The Scikit-HEP project contains other utility packages such as:

- The `Particle` package [15]: a Pythonic interface to the Particle Data Group (PDG) particle data table and Monte Carlo particle identification codes, with a multitude of goodies such as powerful and flexible searches as one-liners.
- The `DecayLanguage` package [16]: tools to parse decay files and programmatically manipulate them, query and display information; classes for a universal representation of particle decay chains.
- The `numpythia` [17] and `pyjet` [18] packages: they provide interfaces between NumPy and the popular Pythia [19] event generator and the FastJet [20] jet finding algorithm, respectively.

4 Outlook

The Scikit-HEP project has been gaining much interest and momentum in the last couple of years. Together with other projects, it is providing a modern and alternative ecosystem for HEP analysis, in Python. The project is community-driven and community-oriented. It is building a user and developer community engaging collaboration across experiments. This is crucial to ensure continuity and sustainability, with a culture where the users of today are meant to become the developers of tomorrow. Some of the project packages are being used by other communities, including the astroparticle physics community.

References

[1] Eric Jones, Travis Oliphant, Pearu Peterson *et al.*, SciPy: open source scientific tools for Python, http://www.scipy.org/

[2] The Astropy Collaboration, Thomas P Robitaille *et al.*, Astropy: a community Python package for astronomy, Astronomy & Astrophysics **A 33** (2013) 558, http://doi.org/10.1051/0004-6361/201322068

[3] Jake VanderPlas, *The Unexpected Effectiveness of Python in Science*, https://speakerdeck.com/jakevdp/the-unexpected-effectiveness-of-python-in-science, PyCon 2017

[4] The ROOT data analysis framework, https://root.cern.ch/

[5] Eduardo Rodrigues, The Scikit-HEP Project, EPJ Web of Conferences **214** (2019) 06005, https://doi.org/10.1051/epjconf/201921406005

[6] The Scikit-HEP Project, http://scikit-hep.org/

[7] Noel Dawe *et al.*, "scikit-hep/roo_numpy" [software], Zenodo, http://doi.org/10.5281/zenodo.592881

[8] Chris Burr, Igor Babuschkin *et al.*, "scikit-hep/root_pandas" [software], Zenodo, http://doi.org/10.5281/zenodo.593933

[9] Jim Pivarski *et al.*, "scikit-hep/uproot" [software], Zenodo, http://doi.org/10.5281/zenodo.1173083

[10] Jim Pivarski *et al.*, "scikit-hep/awkward-array" [software], Zenodo, http://doi.org/10.5281/zenodo.1472436

[11] Henry Schreiner, Hans Dembinski, "scikit-hep/boost-histogram" [software], Zenodo, http://doi.org/10.5281/zenodo.3492034

[12] Hans Dembinski, https://www.boost.org/doc/libs/1_72_0/libs/histogram/doc/html/index.html

[13] iminuit team, iminuit – A Python interface to Minuit, https://github.com/scikit-hep/iminuit

[14] F. James and M. Roos, Comput. Phys. Commun. **10** (1975) 343, http://doi.org/10.1016/0010-4655(75)90039-9

[15] Eduardo Rodrigues, Henry Schreiner, Tamas Gal, Matthew Feickert, "scikit-hep/particle" [software], Zenodo, http://doi.org/10.5281/zenodo.2552429

[16] Eduardo Rodrigues, Henry Schreiner, "scikit-hep/decaylanguage" [software], Zenodo, http://doi.org/10.5281/zenodo.3257423

[17] Noel Dawe, Eduardo Rodrigues, Henry Schreiner, "scikit-hep/numpythia" [software], Zenodo, http://doi.org/10.5281/zenodo.1471492

[18] Noel Dawe, Eduardo Rodrigues, Marcel R., "scikit-hep/pyjet" [software], Zenodo, http://doi.org/10.5281/zenodo.1197493

[19] Pythia, home.thep.lu.se/Pythia/

[20] M. Cacciari, G. P. Salam, G. Soyez, FastJet user manual, Eur. Phys. J. **C 72** (2012) 1896, https://doi.org/10.1140/epjc/s10052-012-1896-2