Scikit-HEP project status

Eduardo Rodrigues
University of Cincinnati

DIANA/HEP Topical Meeting, CERN, 11 September 2017
Why Scikit-HEP?  (A bit o a recap)

- Python usage over time:
  - simple scripting tasks ➔ daily tasks ➔ an analysis framework

<table>
<thead>
<tr>
<th><strong>HEP, ROOT-based</strong></th>
<th><strong>Scientific Computing in Python</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT for almost everything</td>
<td>The father of them all: SciPy</td>
</tr>
<tr>
<td>Toolkit for modeling / fitting: RooFit</td>
<td>Data manipulation: NumPy, Pandas</td>
</tr>
<tr>
<td>Statistics: RooStats</td>
<td>Plotting: matplotlib, seaborn, Bokeh</td>
</tr>
<tr>
<td>Machine learning: TMVA</td>
<td>Machine learning: scikit-learn, TensorFlow</td>
</tr>
<tr>
<td>Etc.</td>
<td>Etc.</td>
</tr>
</tbody>
</table>

+ dedicated projects built atop the above:
  - Astropy, biopython, etc.

- This is where we start to strongly link with the scientific computing community
- We need ways to bridge between ROOT and the Python scientific ecosystem, and more
- Scope / need for a general(ised) effort
- A toolset rather than a toolkit seems the way forward (IMHO)
### The LHCb analysis software ecosystem (2/2)

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Software</th>
<th>Language of use</th>
<th>HEP ?</th>
</tr>
</thead>
</table>
| Data manipulation        | ROOT, numpy, pandas, bcolz, root_numpy, root_pandas | C++ & Python
                           |                   | Python       | Yes   |
|                          |                                 | Python           | No       |       |
|                          |                                 | Python           | No       |       |
| Machine learning         | TMVA, scikit-learn, NeuroBayes  | C++ & Python     | Yes     |       |
| (classification, regression) |                          | Python           | No       |       |
|                          |                                 | C++              | No       |       |
| Plotting                 | ROOT, matplotlib, seaborn, bokeh | C++ & Python     | Yes     |       |
|                          |                                 | Python           | No       |       |
|                          |                                 | Python           | No       |       |
| Fitting                  | RooFit, <Institute/user packages> | C++ & Python     | Yes     |       |
|                          |                                 | C++              | Yes     |       |
| Statistics               | CLs, RooStats                   | Python           | Yes     |       |
|                          |                                 | C++ & Python     | Yes     |       |
| Reweighting              | hep_ml                          | Python           | Yes & no|       |
| Error propagation        | uncertainties, mcerp            | Python           | No       |       |

**Other packages some analysts use**

- Docker for the management of resources and setting up a unified environment for defining the analysis pipeline
- jug for submitting jobs to the batch system

*Note: not claiming it to be a comprehensive list.*

**Numerous packages used!**

But is that really trivial to navigate between these? **Nope ...**
The Scikit-HEP project

The idea, in just one sentence

The Scikit-HEP project (http://scikit-hep.org/) is a community-driven and community-oriented project with the aim of providing Particle Physics at large with a Python package containing core and common tools.

What it is NOT …

- A replacement for ROOT
- A replacement for the Python ecosystem based on NumPy, scikit-learn & co.

… and what IT IS

- A non-monolithic Python toolset, a clearinghouse for doing HEP analysis in Python
- Emulate Scikit-Learn’s unified interface with Astropy’s embrace of third-party packages
- Bridge/glue between the ROOT-based and the Python scientific ecosystem
- We are building a community, engaging with (future) collaborators in various experiments
- Effort to improve discoverability of relevant tools
The Scikit-HEP project – 5 « pillars »

Simulation
Wrappers for Monte Carlo engines and other generators of simulated data

Datasets
data in various sources, such as ROOT, Numpy/Pandas, databases, wrapped in a common interface

Aggregations
histograms and other “sufficient statistics” that summarize or project a dataset

Modeling
traditional fitting: Minuit, linear, lasso, etc., as well as machine learning: BDTs, neural nets, etc.

Visualization
interface to graphics engines, from ROOT and Matplotlib to d3 and plot.ly

They cover all grand topics … !
The Scikit-HEP Project

Package structure on GitHub

scikit-hep
Toolkit of interfaces and tools for high energy physics (HEP).
- python
- analysis
- particle-physics

root_pandas
Forked from lalsr/root_pandas
A Python module for conveniently loading/saving ROOT files as pandas DataFrames.
- Python

root_numpy
The interface between ROOT and NumPy
- python
- numpy
- root
- hep

pyjet
The interface between FastJet and NumPy
- python
- numpy
- root
- hep
- cern

numpythia
The interface between PYTHIA and NumPy
- python
- numpy
- root
- hep
- cern

Interoperability
Collaboration
Reproducibility
Affiliated packages
Core package versus affiliated packages

*scikit-hep*

- Provides general “core” functionality (see next slide for examples)
- Does so with a unified interface
- Builds atop the affiliated packages providing bridges where relevant

*Affiliated packages*

- (Take good concept from Astropy of an affiliated package)
- Package not part of core scikit-hep but related to, and seen as part of, the community & project
- Bring-in functionality specific to certain topics/areas not of the widest community interest
  - Package can have a life of its own
- But the usage within the Scikit-HEP project should be profitable, since ability to interoperate more easily
scikit-hep core package (non-exhaustive list)

- Dataset
  - Common interface for data from/to various sources
  - Dealing with ROOT Ttree, Numpy arrays, etc.

- Aggregation
  - Summarise or project a dataset
  - Typically data aggregation = histogram

- Modeling
  - Data models and fitting utilities

- Visualization
  - Interface to graphics engines such as ROOT and matplotlib, among others

- Simulation
  - utilities, wrappers for Monte Carlo engines and other generators of simulated data

- Modules for units and constants

- Maths and statistics tools

Some modules are much more advanced than others
Affiliated packages

Part of Scikit-HEP

- root_numpy & root_pandas – ROOT-NumPy and ROOT-pandas interfaces
- numpythia – Pythia-NumPy interface
- pyjet – FastJet-NumPy interface

Planned and/or worth trying to get

- Histogrammar – histogramming in a more functional programming way  (http://histogrammar.org/)
- hep_ml package a ML library with miscellaneous tools for HEP  (https://arogozhnikov.github.io/hep_ml/)
- Linking module to Hydra(.Python), a library for data analysis in massively parallel platforms  (https://github.com/multithreadCorner/Hydra)
  - See the 2 following presentations!
- (There’s for sure more)
Some of the achievements so far

- The project has been defined as community-driven and community-oriented
  ⇒ the concept of a community is central!
- Community bonding work is time- and effort-consuming
- We do now have various contact persons in various experiments (Belle-II, CMS, DUNE, LHCb)

- We have a site page for a forum of project ideas … (needs to be updated BTW ;-))
- You are most welcome to bring your own ideas too!

- The scikit-hep package has numerous modules mostly ready for release

- Most of the affiliated packages are mature

- We had a Google Summer of Code project with outcome of relevance to this project
  (see next talks)
Planning

Next steps

- Finalise a few bits and pieces and make sure the code feels as uniform as possible
- Provide examples of how to perform typical simple-ish tasks, to lower threshold for users
- Bring test suite up to speed
- Test a distribution within LHCb for “guinea pigs”
- Development release end of October
- Further engage with Particle Physics community at large
  - Project presentations & tutorials

Releases

- End of October: development release
- End of 2017: 1st official release
Interested?

Links

- GitHub: https://github.com/scikit-hep/
- Website: http://scikit-hep.org/

Mailing lists

- Get in touch with the team “privately”: scikit-hep-admins@googlegroups.com
- Forum for anyone: scikit-hep-forum@googlegroups.com

Thank you
In HEP the standard set of basic units was originally defined by the [CLHEP] project:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Name</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>millimeter</td>
<td>mm</td>
</tr>
<tr>
<td>Time</td>
<td>nanosecond</td>
<td>ns</td>
</tr>
<tr>
<td>Energy</td>
<td>Mega electron Volt</td>
<td>MeV</td>
</tr>
<tr>
<td>Positron charge</td>
<td>eplus</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>kelvin</td>
<td>K</td>
</tr>
<tr>
<td>Amount of substance</td>
<td>mole</td>
<td>mol</td>
</tr>
<tr>
<td>Luminous intensity</td>
<td>candela</td>
<td>cd</td>
</tr>
<tr>
<td>Plane angle</td>
<td>radian</td>
<td>rad</td>
</tr>
<tr>
<td>Solid angle</td>
<td>steradian</td>
<td>sr</td>
</tr>
</tbody>
</table>
Module examples – constants

Constants (*skhep.constants*)

This package *skhep.constants* contains 2 sorts of constants:

- Physical constants.
- Common and/or handy constants.

All constants are computed in the HEP System of Units as defined in the *skhep.units* package.

Typical use case:

```python
>>> from skhep.constants import c_light
>>> from skhep.units import picosecond, micrometer
>>> tau_Bs = 1.5 * picosecond  # a particle lifetime, say the Bs meson's
>>> ctau_Bs = c_light * tau_Bs  # ctau of the particle, ~450 microns
>>> print ctau_Bs  # result in HEP units, so mm ;-)  
0.449688687
>>> print ctau_Bs / micrometer  # result in micrometers
449.688687
```
Module examples – simulation

- Trivial wrapper for the HepPID C++ library, using PyPDT

( More is coming on this front)

Standard use case:

```python
>>> from skhep.simulation import pdgid
>>> pdgid.isLepton(11)
True
>>> pdgid.charge(-4444)  # anti Omega_ccc^++
-2.0
```
Mathematical functions relevant to kinematics

```
skhep.math.kinematics.Kallen_function(x, y, z)
```

The Kallen function, aka triangle or lambda function, named after physicist Anders Olof Gunnar Kallen [Kallen].

**Definition:**

\[
\lambda(x, y, z) = x^2 + y^2 + z^2 - 2xy - 2yz - 2zx \\
= (x - y - z)^2 - 4yz \\
= [x - (\sqrt{y} + \sqrt{z})^2][x - (\sqrt{y} - \sqrt{z})^2] \quad \text{if } y, z > 0
\]

**Example:**

Calculate in the rest frame of a particle of mass \( M \) decaying to 2 particles labeled 1 and 2, \( P(M) \to p_1(m_1) + p_2(m_2) \), the momenta of 1 and 2 given by \( p = |p_1| = |p_2| \):

```python
>>> from skhep.math import Kallen_function
>>> from skhep.units import MeV, GeV
>>> from math import sqrt
>>> M = 5.279 * GeV; m1 = 493.7 * MeV; m2 = 139.6 * MeV
>>> p = sqrt(Kallen_function( M**2, m1**2, m2**2 ) / (2*2))
>>> print p / GeV  # print the CMS momentum in GeV
2.61453580221
```

**Reference:**

[Kallen] https://en.wikipedia.org/wiki/K%C3%A4llen_function