ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

The Scikit-HEP Project

Matthieu MARINANGELI, on behalf of the Scikit-HEP community

École Polytechnique Fédérale de Lausanne, Switzerland

PYHEP Workshop, 7th of July 2018, Sofia

What is Scikit-HEP?

A reunion of two worlds that seem apart.

HEP, largely ROOT-based

- ROOT for almost everything
- Toolkit for modelling / fitting Roofit
- Statistics: RooStats
- Machine learning: TMVA
- Etc ...

Scientific Computing in Python

- The father of them all: Scipy
- Data manipulation: Numpy, Pandas
- Plotting: matplotlib, seaborn, Bokeh
- Machine learning: scikit-learn, TensorFlow
- Etc ...

Various initiatives already exist to link the two worlds, but only tackling specific issues.

Need for a more generalised effort. Others did it: Astropy, biopython

What is Scikit-HEP?

A reunion of two worlds that seem apart.

HEP, largely ROOT-based

- ROOT for almost everything
- Toolkit for modelling / fitting Roofit
- Statistics: RooStats
- Machine learning: TMVA
- Etc ...

Scientific Computing in Python

- The father of them all: Scipy
- Data manipulation: Numpy, Pandas
- Plotting: matplotlib, seaborn, Bokeh
- Machine learning: scikit-learn, TensorFlow
- Etc ...

Various initiatives already exist to link the two worlds, but only tackling specific issues.

Need for a more generalised effort. Others did it: Astropy, biopython

The Scikit-HEP project

The *Scikit-HEP project* is a community-driven and community-oriented project with the aim of providing Particle Physics at large with a toolset of Python packages containing core and common tools.

scikit-hep.org

What it is NOT ...

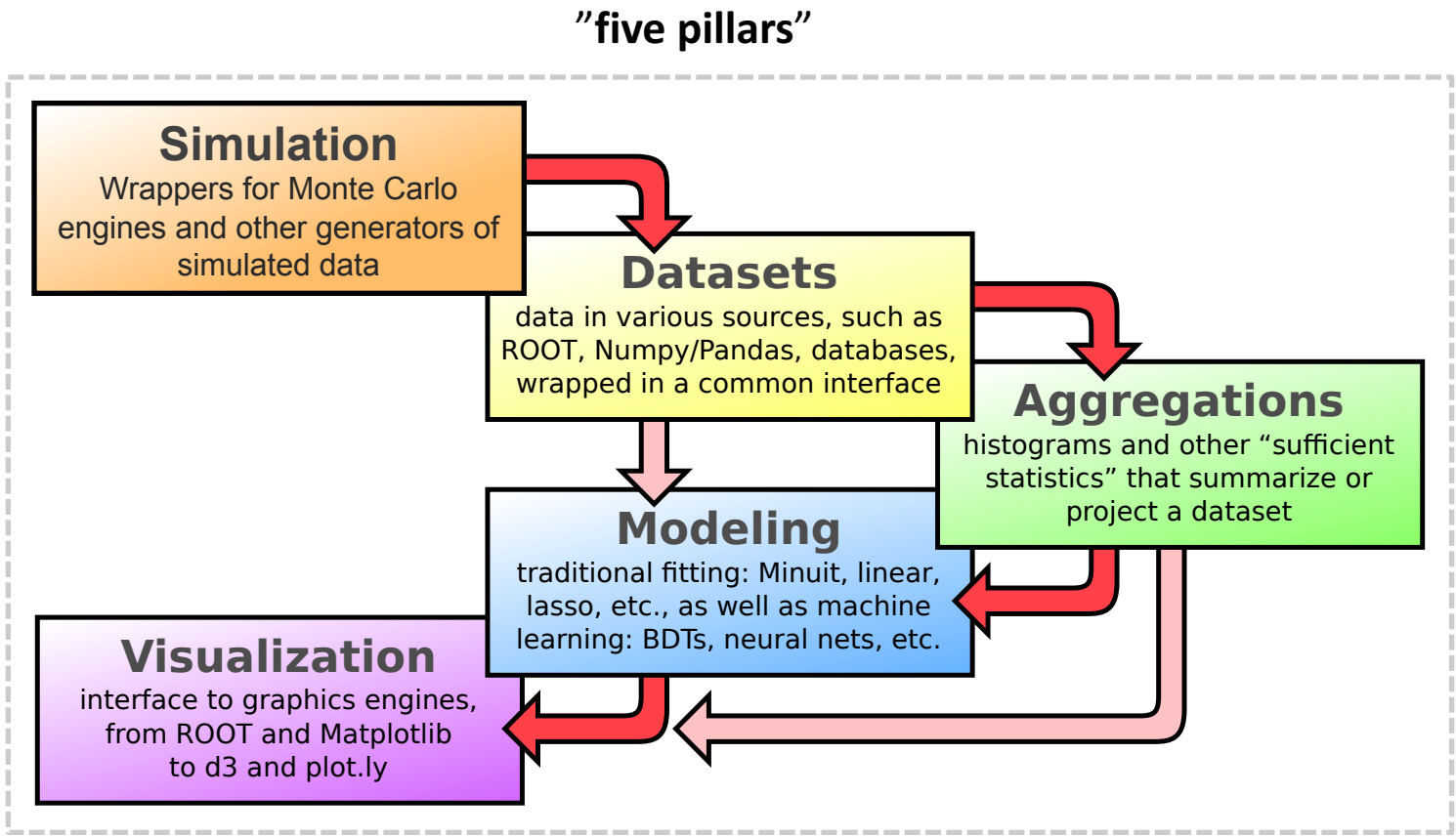
- A replacement for ROOT
- A replacement for the Python ecosystem based on Numpy, scikit-learn etc..

and what it IS ...

- An initiative to improve the **interoperability between HEP tools and the Python scientific ecosystem:**
  - Expand typical toolkit of HEP physicists
  - Common definitions and APIs to ease “cross-talk”
- Similar to the Astropy project (learning from good examples)

The Scikit-HEP toolset structure

The project aims to provide tools categorized in the following groups:



- Modules for units and constants
- Maths and statistics
- Conversion between style of expressions
  - ROOT to numexpr
- Provenances
- Etc...

The Scikit-HEP packages

- **scikit-hep:** starting point of the project. Containing tools for maths, units, constants, visualization and modelling and a unified dataset interface.
- **uproot:** minimalist ROOT I/O in pure Python and Numpy.
- **histbook:** versatile, high-performance histogram toolkit for Numpy.
- **formulate:** easy conversions between different styles of expressions (currently between ROOT and numexpr).
- **numpythia:** the interface between PYTHIA and Numpy.
- **pyjet:** the interface between FastJet and Numpy.
- **vegascope:** minimal viewer of Vega and Vega-Lite gaphics from python.

The packages will be presented in a **small analysis example.**

Small analysis example: read a ROOT file

Let's explore a ntuple of $Z \rightarrow \mu^+\mu^-$ events and see what we can already do. I decide to use **numpy**.

ROOT I/O in pure Python and Numpy. https://github.com/scikit-hep/uproot

uproot (originally μproot, for "micro-Python ROOT") is a reader and (someday) a writer of the ROOT file format using only Python and Numpy. Unlike the standard C++ ROOT implementation, uproot is only an I/O library, primarily intended to stream data into machine learning libraries in Python. Unlike PyROOT and root_numpy, uproot does not depend on C++ ROOT. Instead, it uses Numpy calls to rapidly cast data blocks in the ROOT file as Numpy arrays.

```
In [1]: rootfile = "Zmumu.root"
        import uproot
        zmumu = uproot.open(rootfile)["events"]
        zmumu.arrays(["px1","px2","py1","py2","M"])

Out[1]: {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])}
```

name of the tree

dictionary of arrays

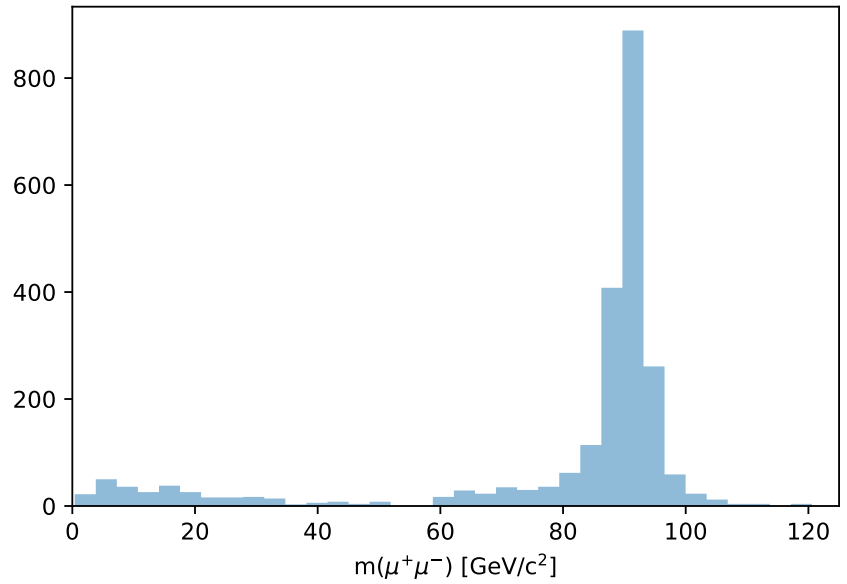
No need to have ROOT installed.

Small analysis example: dataset and visualization

Let's play with datasets. Scikit-HEP aims to provide a **common interface** for datasets from various sources.
Currently only **Numpy** arrays, ROOT **TTree** under development.

```
In [2]: from skhep.dataset.numpydataset import *

        zmumu_dataset = NumpyDataset(zmumu.arrays(["px1","px2","py1","py2","M"]))
        zmumu_dataset

Out[2]: NumpyDataset([(-41.19528764,  34.14443725,  17.4332439 , -16.11952457, 82.46269156),
                      ( 35.11804977, -41.19528764, -16.57036233,  17.4332439 , 83.62620401),
                      ( 35.11804977, -40.88332344, -16.57036233,  17.29929704, 83.30846467),
                      ...,
                      ( 32.37749196, -68.04191497,   1.19940578, -26.10584737, 95.96547966),
                      ( 32.37749196, -68.79413604,   1.19940578, -26.39840043, 96.49594381),
                      ( 32.48539387, -68.79413604,   1.2013503 , -26.39840043, 96.65672765)],
                     dtype=[('px1', '<f8'), ('px2', '<f8'), ('py1', '<f8'), ('py2', '<f8'), ('M', '<f8

In [3]: zmumu_dataset.M

Out[3]: SkhepNumpyArray([82.46269156, 83.62620401, 83.30846467, ..., 95.96547966,
                         96.49594381, 96.65672765])
```

```
In [4]: from matplotlib import pyplot as plt
        %matplotlib inline
        from skhep.visual import MplPlotter as skh_plt
        plt.rcParams['figure.figsize'] = (8,6)

In [5]: _ = skh_plt.hist(zmumu_dataset.M, bins=50)
        plt.xlim(0,125)
        plt.xlabel("m($\mu^{+}\mu^{-}$) [GeV/c$^2$]")
        plt.ylabel("events")

Out[5]: <matplotlib.text.Text at 0x10d336a90>
```

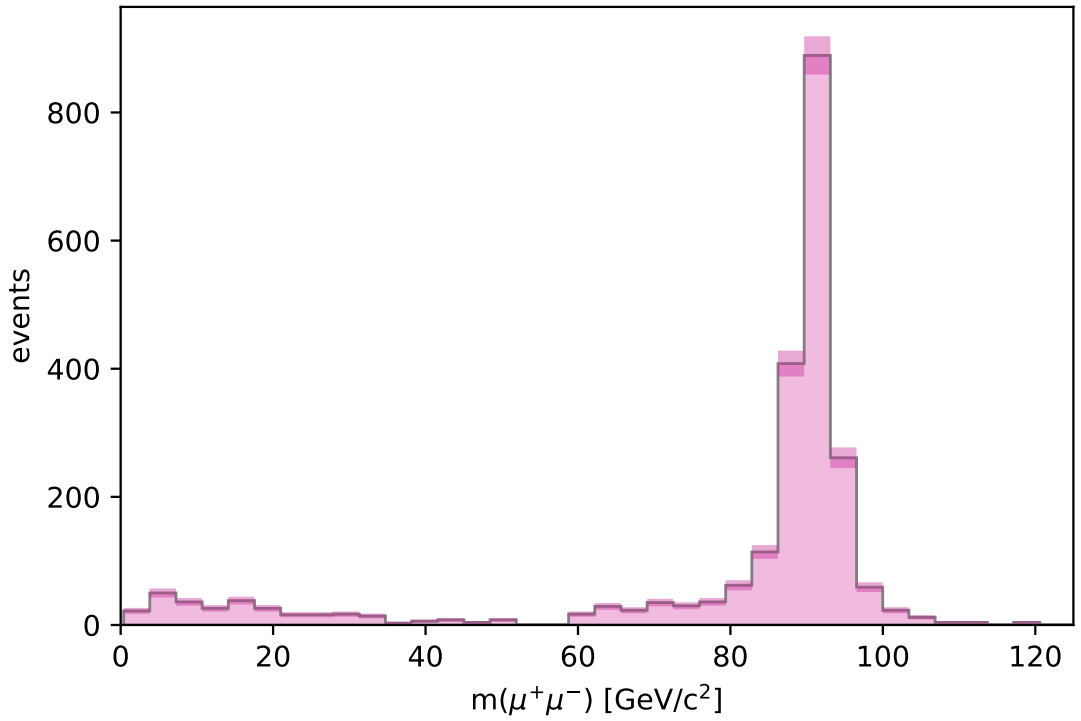


Plotting the invariant mass of the two muons using the **visual** module of Scikit-HEP

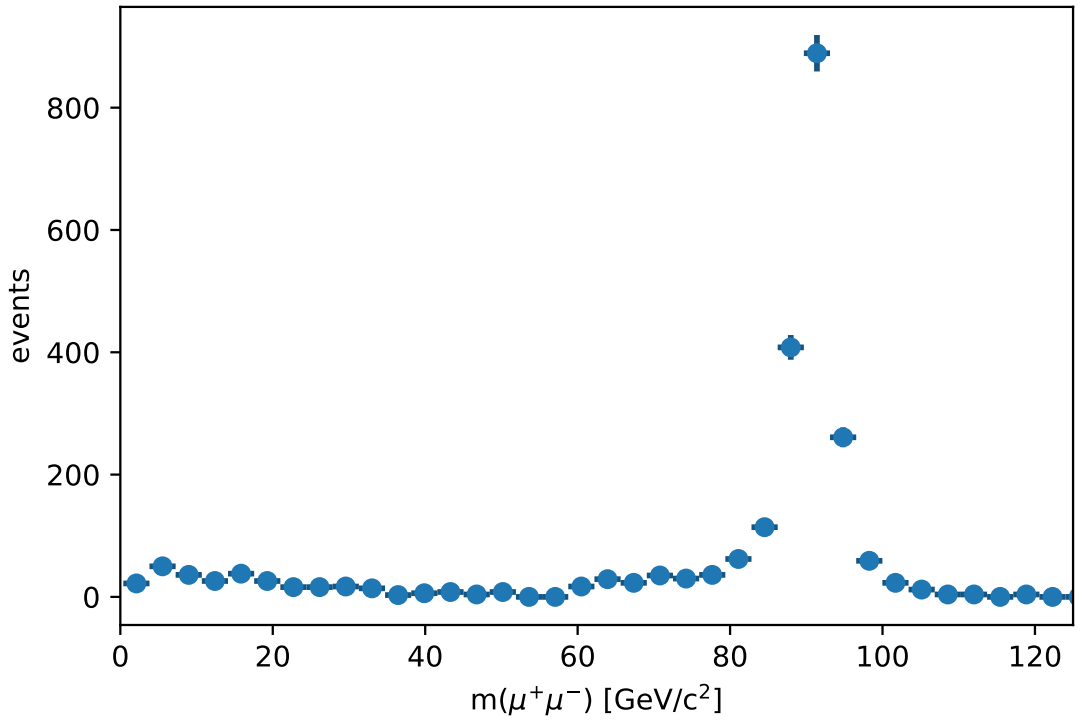
Small analysis example: dataset and visualization

Other plotting options.

```
_ = skh_plt.hist(zmumu_dataset.M, bins=50,
errorbars=True, color='C6')
```




```
_ = skh_plt.hist(zmumu_dataset.M, bins=50,
errorbars=True, histtype='marker')
```



Only matplotlib as plotting backend in skhep module for now. Other will come (ROOT, etc ...)

Small analysis example: aggregation

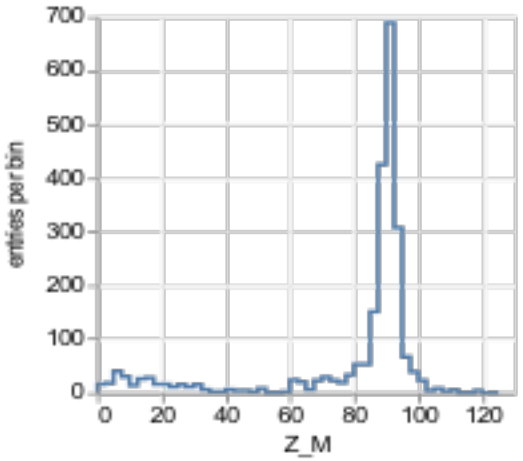
histbook

https://github.com/scikit-hep/histbook

Versatile, high-performance histogram toolkit for Numpy

Relatively new additions to Scikit-HEP project. Provides ways of aggregating and visualizing you data through **Vega**.

```
from histbook import *
from vega import VegaLite as canvas
histogram = Hist(bin("Z_M", 50, 0, 125))
M = zmumu_dataset.M.view(np.ndarray)
histogram.fill(Z_M = M)
histogram.step("Z_M").to(canvas)
```



Can be modified in Vega Editor

Can be converted to ROOT histograms or as Pandas DataFrame

```
In [15]: histogram.root()

Out[15]: <ROOT.TH1D object at 0x7fd656670f20>

In [16]: histogram.pandas()

Out[16]:
```

Z_M	count()	err(count())
[-inf, 0.0)	0.0	0.000000
[0.0, 2.5)	16.0	4.000000
[2.5, 5.0)	18.0	4.242641
[5.0, 7.5)	40.0	6.324555
[7.5, 10.0)	30.0	5.477226
[10.0, 12.5)	14.0	3.741657
[12.5, 15.0)	26.0	5.099020
[15.0, 17.5)	28.0	5.291503
[17.5, 20.0)	16.0	4.000000
[20.0, 22.5)	16.0	4.000000
[22.5, 25.0)	11.0	3.316625
[25.0, 27.5)	15.0	3.872983
[27.5, 30.0)	11.0	3.316625

More fun to come …

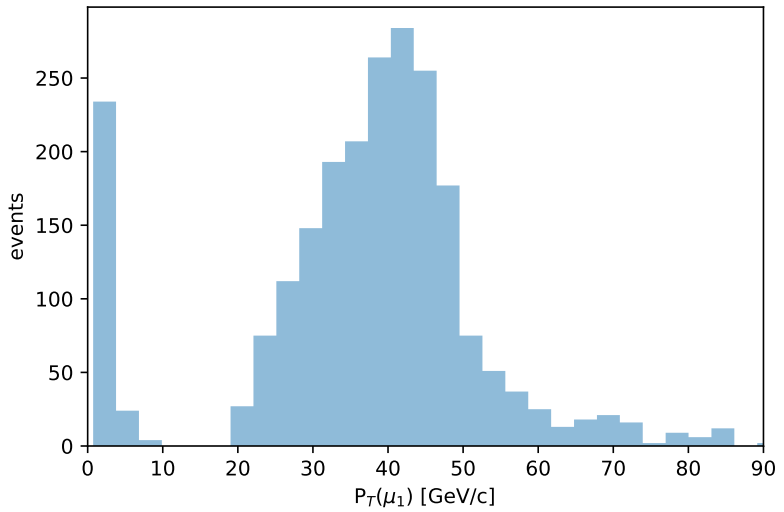
Small analysis example: construct variables

Back to the dataset:

I want pt1 and pt2

```
zmumu.arrays(["px1","px2","py1","py2","M"])
```

```
In [22]: zmumu_dataset.pt1 = np.sqrt(zmumu_dataset.px1**2 + zmumu_dataset.py1**2)

In [23]: zmumu_dataset.pt1

Out[23]: SkhepNumpyArray([44.7322, 38.8311, 38.8311, ..., 32.3997, 32.3997,
                          32.5076])

In [24]: _ = skh_plt.hist(zmumu_dataset.pt1, bins=60)
         plt.xlim(0,90)
         plt.ylabel("events")
         plt.xlabel("P$_T$($\mu_1$) [GeV/c]")
         plt.savefig("pt1.pdf",dpi = 1000)
```

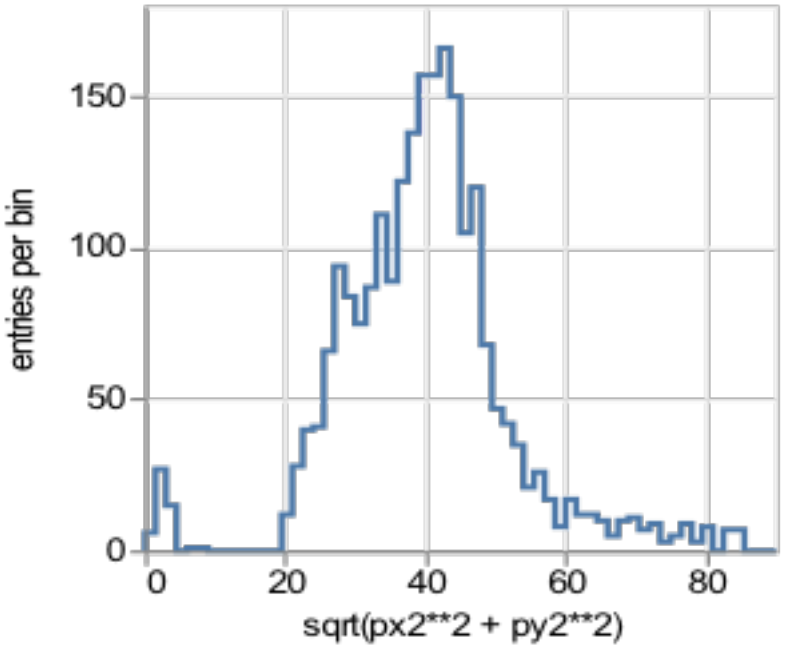


pt1 is now available in the dataset

NumpyDataset

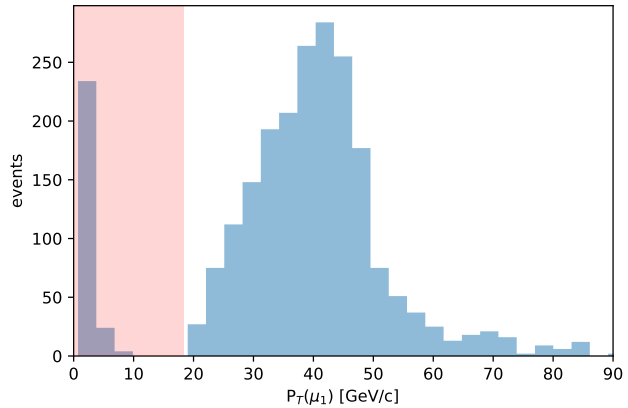
histbook

```
hist = Hist(bin("sqrt(px2**2 + py2**2)", 60, 0, 90))
hist.fill(px2=zmumu_dataset.px2.view(np.ndarray),
          py2=zmumu_dataset.py2.view(np.ndarray))
hist.step("sqrt(px2**2 + py2**2)").to(canvas)
```



Small analysis example: selection

Use the NumpyDataset for selection: reject low muon p_T events

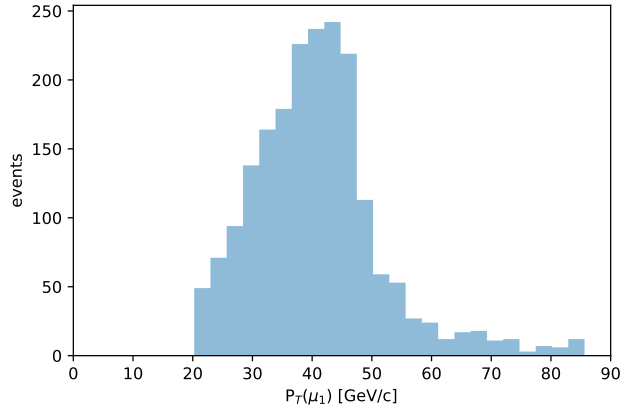



```
zmumu_dataset1 = zmumu_dataset[(zmumu_dataset.pt1 > 20) & (zmumu_dataset.pt2 > 20)]
```


With ROOT I would just do :

- tree.CopyTree(**"pt1 > 20 && pt2 > 20"**), Possible with the select method of NumpyDataset.

```
zmumu_dataset2 = zmumu_dataset.select("pt1 > 20 & pt2 > 20")
```



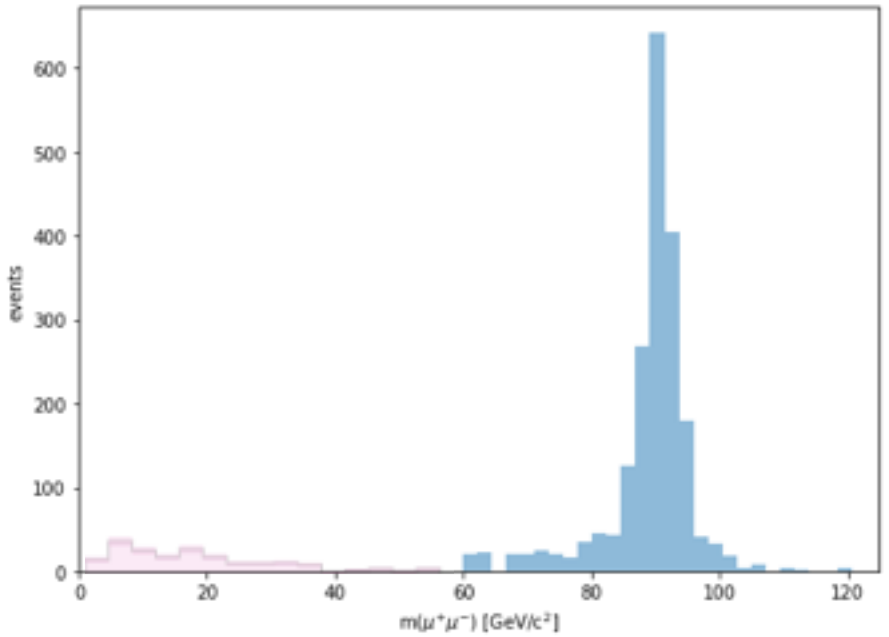
- tree.CopyTree(**"TMath::Sqrt(px1**2 + py1**2) > 20 && TMath::Sqrt(px2**2 + py2**2) > 20"**). Need to be converted to **numexpr style.**

```
import formulate  https://github.com/scikit-hep/formulate
### write the selection as a formula
pt1 = formulate.from_root('TMath::Sqrt(px1**2 + py1**2)')
pt2 = formulate.from_root('TMath::Sqrt(px2**2 + py2**2)')
cut = (pt1 > 20) & (pt2 > 20)
cut.to_numexpr()
```

```
'(sqrt(((px1 ** 2) + (py1 ** 2))) > 20) & (sqrt(((px2 ** 2) + (py2 ** 2))) > 20)'
```

```
zmumu_dataset3 =
zmumu_dataset.select(cut.to_numexpr())
```

Small analysis example: provenances



3 months after .... I have this "*zmumu_dataset3*" and
I want to redo a plot of the Z mass.

I remember some events at low mass, what happened?

History of operations (provenance) is saved in the NumpyDataset:

```
print(zmumu_dataset3.provenance)
```

```
0: <ObjectOrigin>
1: <Transformation(Array pt1 has been created)>
2: <Transformation(Array pt2 has been created)>
3: <Transformation(Selection, (sqrt(((px1 ** 2) + (py1 ** 2))) > 20) & (sqrt(((px2 ** 2) + (py2 ** 2))) > 20), applied)
```

```
print(zmumu_dataset3.provenance[0].detail)
```

```
{b'px1': array([-41.19528764,  35.11804977,  35.11804977, ...,  3
        32.37749196,  32.48539387]), b'px2': array([ 34.14443725,
       -68.79413604, -68.79413604]), b'py1': array([ 17.4332439 ,
         1.19940578,   1.2013503 ]), b'py2': array([-16.11952457,
       -26.39840043, -26.39840043]), b'M': array([82.46269156, 83
       96.49594381, 96.65672765])}
```

```
print(zmumu_dataset3.provenance[1].detail)
```

```
Array pt1 has been created (0: <ObjectOrigin>
1: <Transformation(px1 has been squared)>
2: <Transformation(py1**2 has been added to px1**2)>
3: <Transformation(px1**2 + py1**2 has been raised to the power of 0.5)>)
```

Provenances are however not saved into files... (yet ?)

Small analysis example: summary

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

Visualization
Interface to graphics engines, from ROOT and Matplitlib to d3 and plot.ly

- Reading ROOT files with uproot to use numpy arrays
  - possibility to be read as pandas DataFrame as well
- Manipulate datasets with a common interface:
  - all examples shown with NumpyDataset will work for other sources (RootDataset ...)
  - common selection system for each kind of dataset. Easy conversion between different styles of expressions (formulate)
  - navigate through history of transformations
- Use histbook to create histograms and fill them from datasets:
  - visualized with Vega. If you work with bare python use vegascope.
- Use scikit-hep matplolib backend for plotting variables.
  More backends to come...

- **Modules for units and constants**
- **Maths and statistics**
- **Conversion between style of expressions**
  - ROOT to numexpr
- **Provenances**

Should add provenances in plots, histograms ... and eventually save them into files.

Wasn't it easy to use? It is suitable for beginners in HEP (undergraduate, MSc students).

Simulation

- **numpythia**: interface between Pythia and Numpy (https://github.com/scikit-hep/numpythia)

```
from numpythia import Pythia, hepmc_write, hepmc_read
from numpythia import STATUS, HAS_END_VERTEX, ABS_PDG_ID

params = {"Beams:eCM": 13000, "WeakSingleBoson:ffbar2gmZ": "on",
          "23:onMode": "off" ,"23:onIfAny": "13", "WeakZ0:gmZmode": 2}

pythia = Pythia(params=params)
selection = ((STATUS == 1) & ~HAS_END_VERTEX)

for event in pythia(events=100):
    array = event.all(selection)
    muplus  =  array[array["pdgid"] ==-13]
```

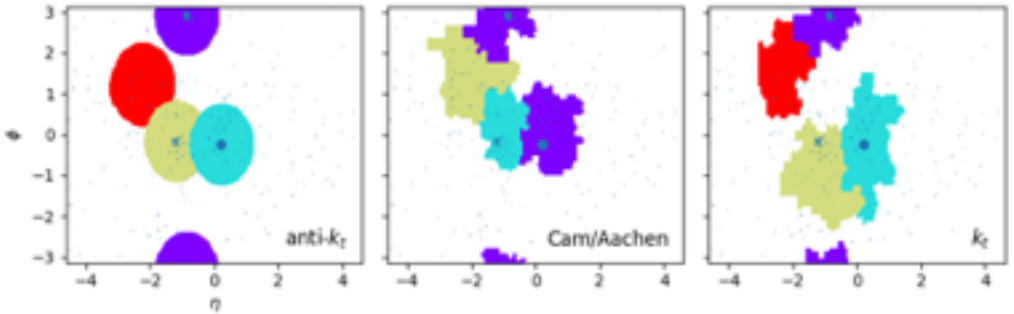
E, px, py, pz, mass, prodx, prody, prodz, prodt, pdgid, status

- very easy to use
  - used by master students at LHCb - easier than pythia standalone—good as a starting point for new students
- could have more variables. Only 4-momenta, production vertices ... but no decay information (vertices, products ...)
- Pythia8 already provides a python interface which is still under development

- **pyjet:** interface between FastJet and Numpy (https://github.com/scikit-hep/numpythia)

```
from pyjet import cluster
from pyjet.testdata import get_event

vectors = get_event()
sequence = cluster(vectors, R=0.1, p=-1)
jets = sequence.inclusive_jets() # list of PseudoJets
```

Jet clustering



Pyjet explored as an option to generate some LHCb phase-2 prospects for the LLP community white paper - great tool for multi-collaboration community works.

Will be updated in the next months and also new features to come such as jet substructure tools, vertex finders etc...

Others

Modeling:

- Bayesian Blocs algorithm
- Maybe include the hep_ml package, a ML library with miscellaneous tools for HEP

Units and constants:

```
>>> 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
```

Maths:

- Geometry, vectors, 4-vectors.
- Kinematics functions

example:

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

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

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

Example

Calculate in the rest frame of a particle of mass M decaying to 2 particles labeled 1 and 2, $P(M) \to p1(m1) + p2(m2)$, the momenta of 1 and 2 given by $p = |\mathbf{p1}| = |\mathbf{p2}|$:

```
>>> 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*M)
>>> print p / GeV   # print the CMS momentum in GeV
2.61453580221
```

Try it!

The toolset is currently distributed trough **pip. :**

- pip install scikit-hep, uproot, histbook, formulate ....

WE NEED FEEDBACK!

Want to know more?

Star 77

Navigation

About
Installation
Get help
Documentation
Contributing
Affiliated packages

Welcome to the Scikit-HEP project!

You can find a little introduction to the project in the About page. To get started, first install the skhep module and then read the main documentation.

If you have ideas concerning the development of the project, or if there is a feature missing you'd like to see, you are most welcome to contribute to Scikit-HEP. Please also check out the list of affiliated packages.

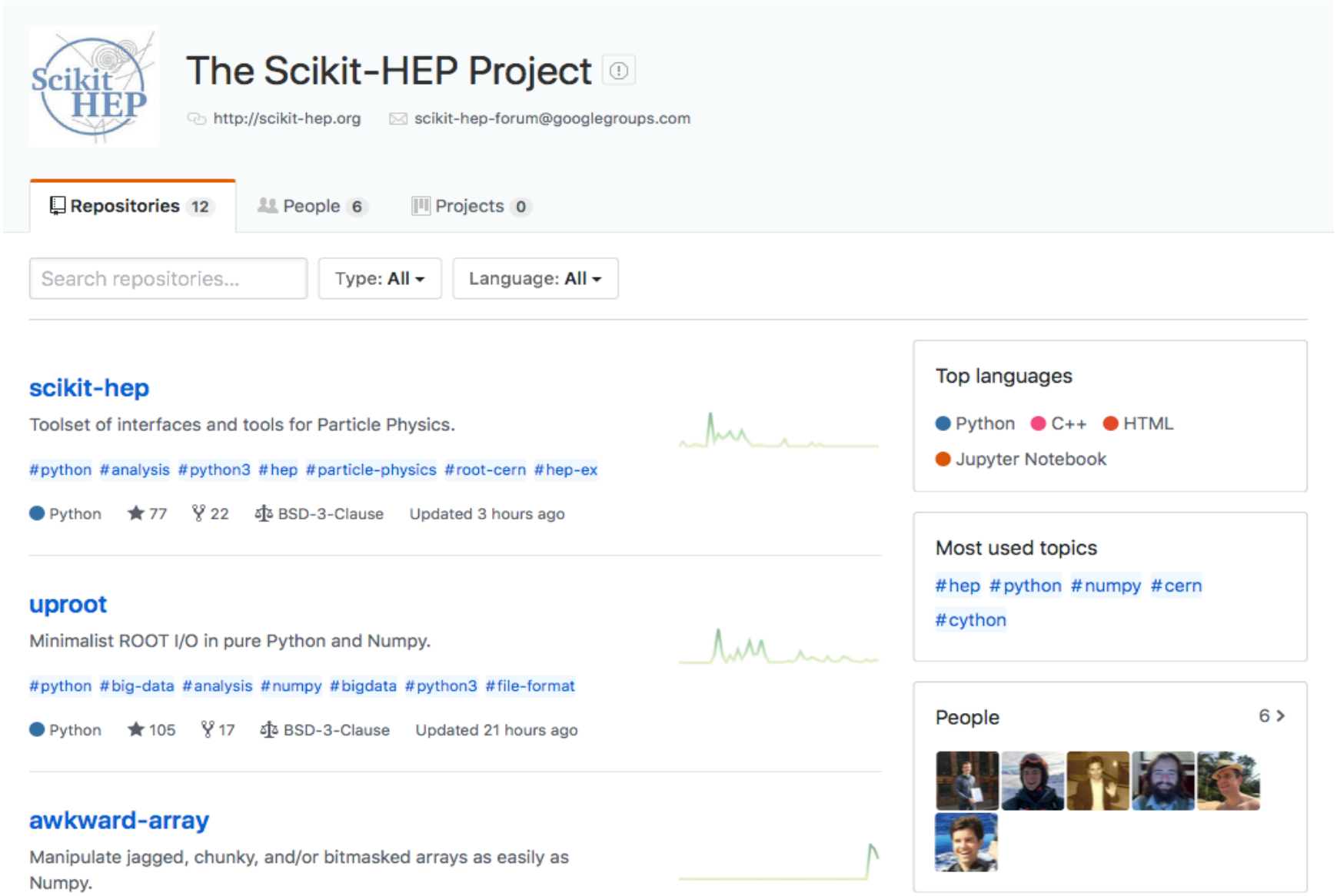
Indices and tables

- Index
- Module Index
- Search Page

©2016-2017, The Scikit-HEP Developers. | Page source

Want to know more?

https://github.com/scikit-hep



Want to contribute?

There is still a lot to develop.

We are a community. Everybody is welcome to contribute.

We have a **forum of project ideas** page on our website. You are welcome to bring yours.

We would like to have more people from various experiments:

- Eg. LHC, neutrino community, simulation community, Belle-II, FCC, SHIP, CLIC etc ...

Forum on project ideas

Fork me on GitHub

This page collects project ideas coming from either the Scikit-HEP team or more broadly from anyone from the Scikit-HEP community. Do get in touch if you would like to add an idea to this forum. It will be the best way to raise awareness about your idea and attract contributors!

Google Summer of Code 2017

Scikit-HEP is participating in the Google Summer of Code 2017 program with CERN as an organization, and under the umbrella of the HEP Software Foundation, see the direct link.

We have put forward 2 successful proposals! The direct links to the project proposals are the following:

- Python bindings for the Hydra C++ library for analysis on massively multi-threaded platforms.
  Student: Deepanshu Thakur. Miscellaneous done
- Visualization tools for Scikit-HEP. Visualization open

Data aggregation

- Data aggregation largely means histogramming. An interesting idea would be to exploit the Histogrammar package implementation in Python as a powerful way of dealing with data aggregation in Scikit-HEP. Aggregations open

Datasets

- Datasets are central in HEP and Scikit-HEP too. This important package needs further development and lack parts of the implementation. Get in touch with *Jim* and *Eduardo* if the topic is of interest to you. Datasets open

Math modules

- The 3D and Lorentz vector classes need to be improved to exploit NumPy arrays. There is also functionality to be implemented. | Contact *Eduardo* if such a development is of interest to you. Math ongoing
- Now that 3D and Lorentz vector classes are available, though in development, it would be handy to build on them. Needed are a 3D point class and then a set of mathematical functions implementing handy geometry-related calculations commonly used in HEP. Contact *Vanya* and *Eduardo* if such a development is of interest to you. Math ongoing

Thank you.

```
tsf-492-wpa-6-100:scikit-hep matthieumarinangeli$ ./scripts/skhep-info
```


```
Homepage http://scikit-hep.org
GitHub   https://github.com/scikit-hep/scikit-hep
PyPI     https://pypi.python.org/pypi/scikit-hep
```