UNIVERSITY OF LIVERPOOL

The Scikit-HEP project overview and future

Eduardo Rodrigues, for the Scikit-HEP project admins
University of Liverpool

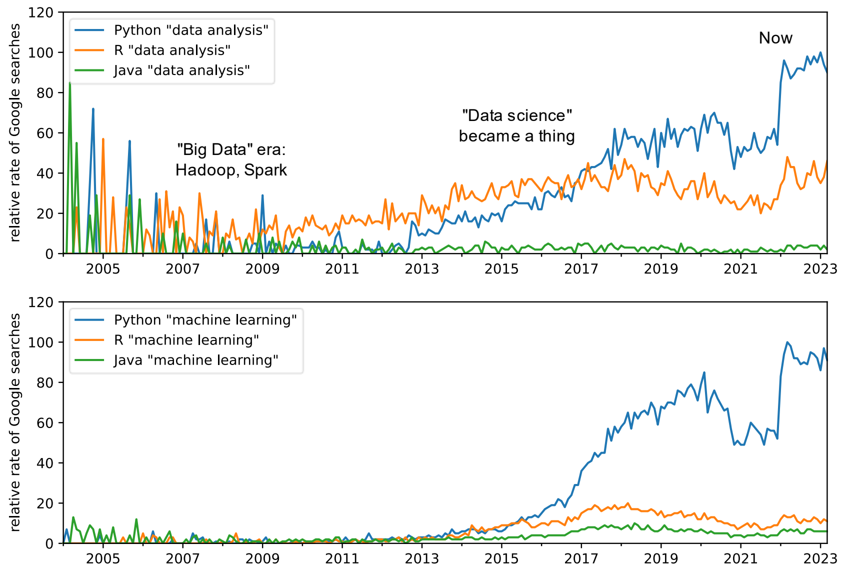
WLCG/HSF Workshop, DESY, 13-17th May 2024

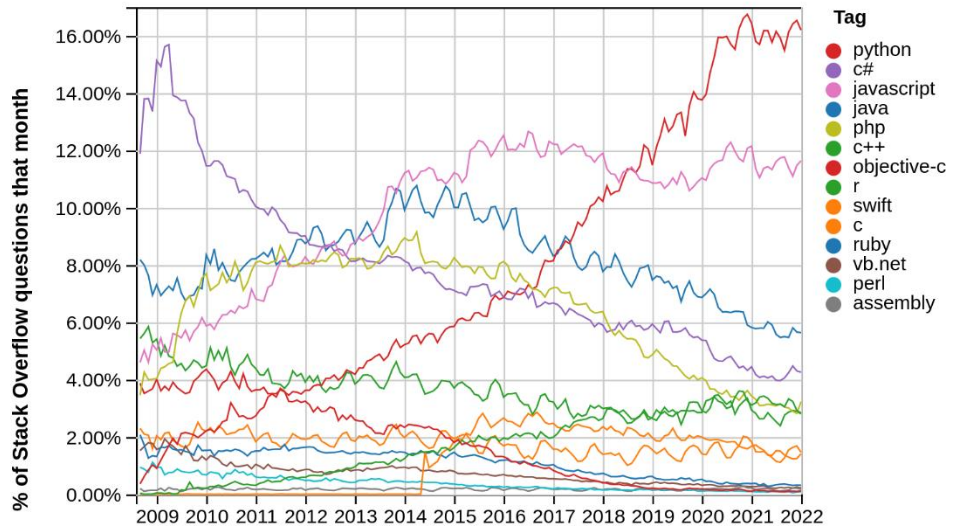
Historical perspective and project grand idea

- **Python and Pythonic physics analysis in the 2010s**
- **The very early days of Scikit-HEP**

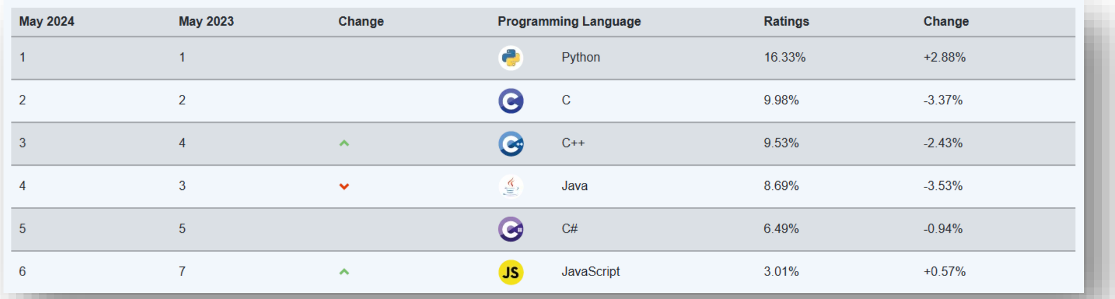
Recap – some motivation as to why many looked seriously at Python

- ❑ **Very well-known these days – Python for data analysis was the thing back in the days !**





- ❑ **Python continues to dominate a decade down the line !**
  - **Example from the TIOBE index:**

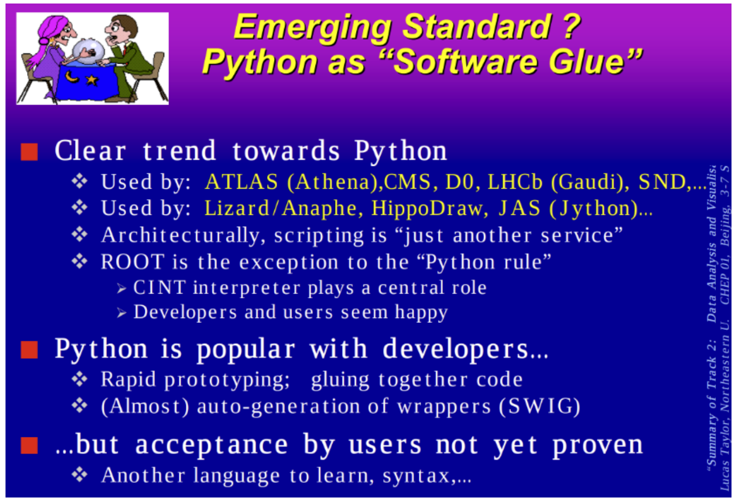


May 2024	May 2023	Change	Programming Language	Ratings	Change
1	1		Python	16.33%	+2.88%
2	2		C	9.98%	-3.37%
3	4	▲	C++	9.53%	-2.43%
4	3	▼	Java	8.69%	-3.53%
5	5		C#	6.49%	-0.94%
6	7	▲	JavaScript	3.01%	+0.57%

Python and Pythonic physics analysis in the 2000-2010s

- In HEP, Python was being used
  by some early and enthusiastic adopters
- "Python emerging as standard glue“

- PyROOT official bindings coming to life around 2004 …
- Though the scientific Python ecosystem around NumPy
  was already reasonably mature,
  not much of that was popular with HEP analysts

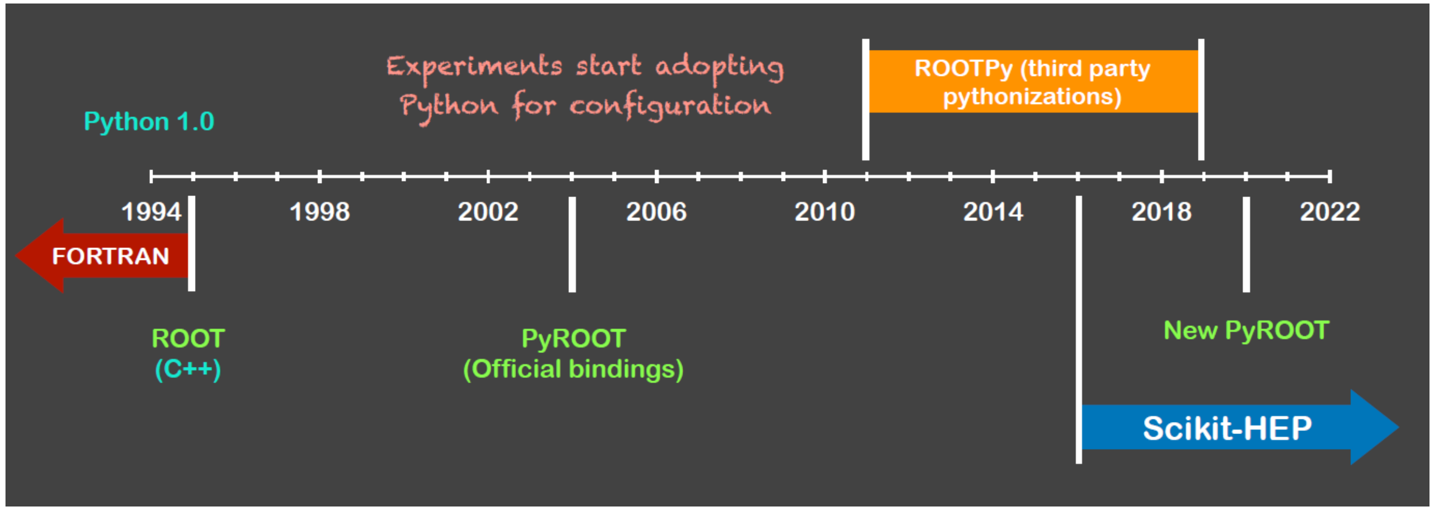


Emerging Standard ?
Python as “Software Glue”

- Clear trend towards Python
  - Used by: ATLAS (Athena),CMS, D0, LHCb (Gaudi), SND,...
  - Used by: Lizard/Anaphe, HippoDraw, JAS (Jython)...
  - Architecturally, scripting is “just another service”
  - ROOT is the exception to the “Python rule”
    - CINT interpreter plays a central role
    - Developers and users seem happy
- Python is popular with developers...
  - Rapid prototyping; gluing together code
  - (Almost) auto-generation of wrappers (SWIG)
- ...but acceptance by users not yet proven
  - Another language to learn, syntax,...

"Summary of Track 2: Data Analysis and Visualis
Lucas Taylor, Northeastern U. CHEP 01, Beijing, 3-7 S

CHEP 2001 !



Experiments start adopting Python for configuration
ROOTPy (third party pythonizations)
Python 1.0
1994
1998
2002
2006
2010
2014
2018
2022
FORTRAN
ROOT (C++)
PyROOT (Official bindings)
New PyROOT
Scikit-HEP

Python and Pythonic physics analysis in the 2010s

[Some figures have active links.]

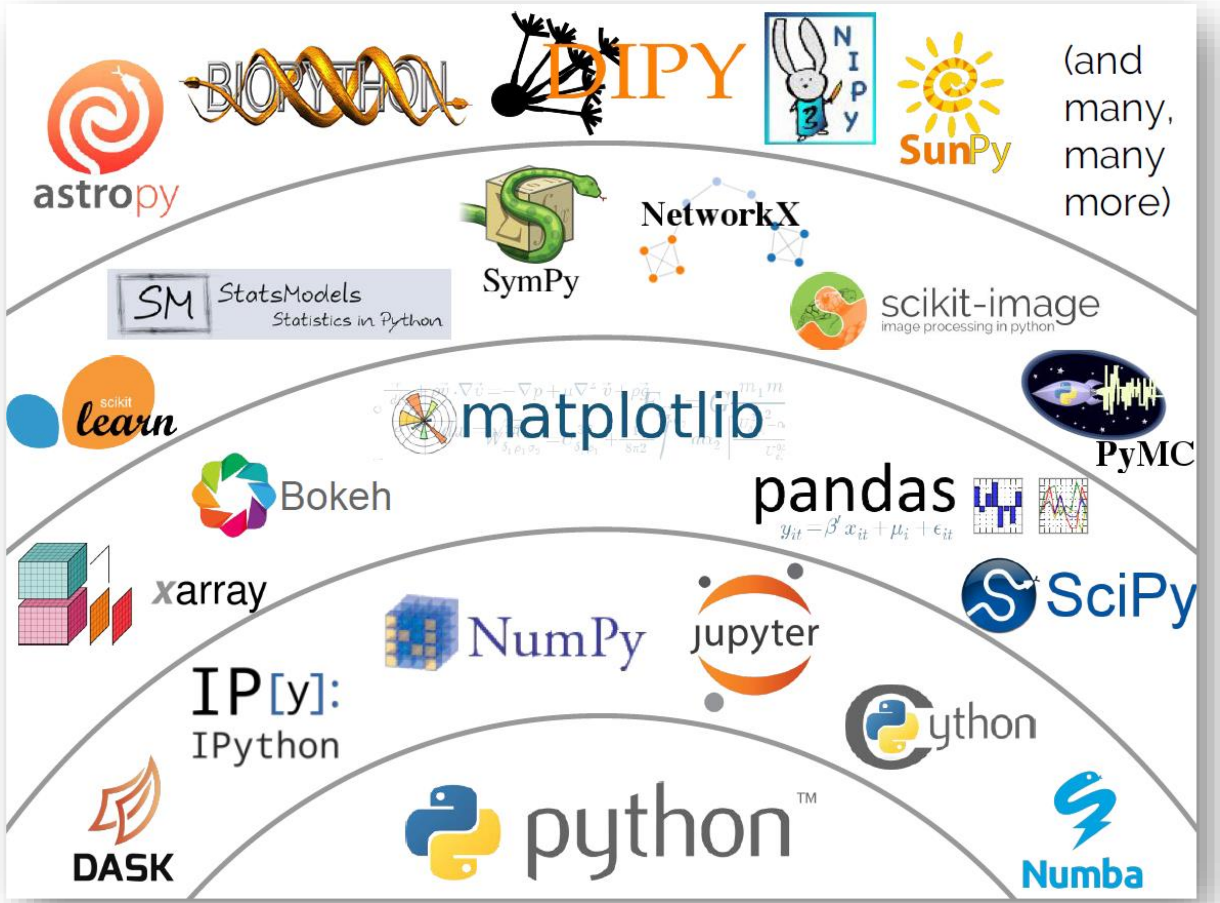
- ❑ Famous "shell diagram" of the Python scientific ecosystem outside HEP, back in the days:
- ❑ What about HEP? Why was HEP not present back in 2017?
- ❑ Sort of a philosophical question at this point. But there was a niche …

Domain-specific

Python's

Scientific

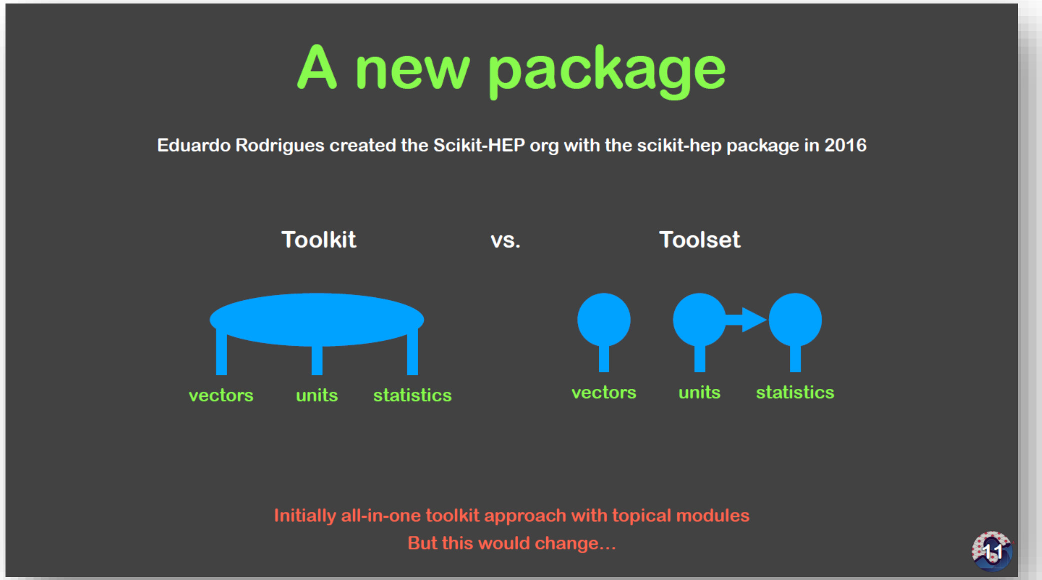
stack



Jake VanderPlas,
The Unexpected Effectiveness of Python in Science,
PyCon 2017

The very early days of Scikit-HEP

- ❑ Scikit-HEP started with the goal to:
  - - Improve the interoperability between HEP tools and the scientific Python ecosystem
  - - Build a community of developers and users – be community-driven and community-oriented
  - - Improve the discoverability of relevant tools
  - - …
- ❑ Project started around concept of pillars: datasets, data aggregation, modelling, visualisation and simulation (and "utilities")
- ❑ Initial approach quickly evolved towards a toolset, not a toolkit:

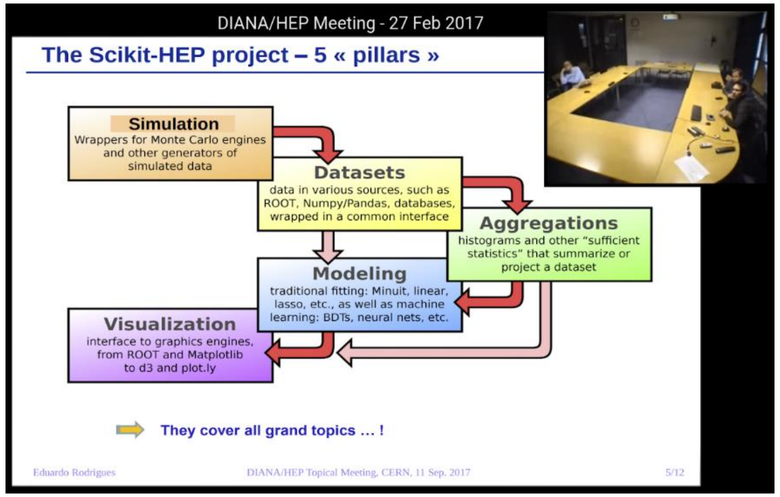


A new package

Eduardo Rodrigues created the Scikit-HEP org with the scikit-hep package in 2016

Initially all-in-one toolkit approach with topical modules
But this would change…

Henry Schreiner et al., "Awkward Packaging: building Scikit-HEP" ,SciPy 2022



DIANA/HEP Meeting - 27 Feb 2017

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

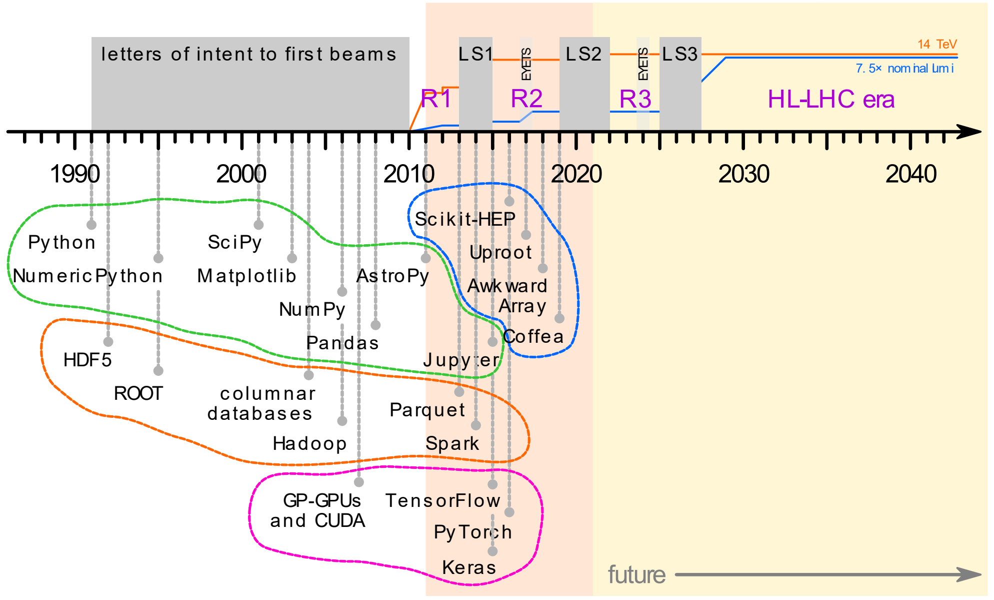
➡ They cover all grand topics … !

Eduardo Rodrigues DIANA/HEP Topical Meeting, CERN, 11 Sep. 2017 5/12

From first public talk on Scikit-HEP, in 2017

Some biased aspects of the timeline of Scikit-HEP (more @ https://scikit-hep.org/news)

- ❑ Much of the initial challenge / work was (1) attract people and (2) attract relevant packages
- ❑ Started off with inspiration from LHCb's Ostap package and Noel Dawe's rootpy, and putting together in a GitHub organisation packages root_numpy, then numpythia, pyjet, iminuit, etc.
- ❑ Uproot came out of conversations with Noel about root_numpy's installation difficulties
- ❑ uproot got showcased at CHEP 2018 and was a game-changer – for Python analysis in HEP and Scikit-HEP also
  - Loading/saving HEP data was *the* technical bottleneck that required a full ROOT installation. uproot made the barrier the lightest possible
- ❑ So was awkward array, as iminuit a few years before (used a lot in Astro)
- ❑ By CHEP 2018 (July 2018), the GitHub org was 8 people and 12 repositories
- ❑ Early 2019 – key developments on histogramming (boost-histogram)
- ❑ End of 2019 Belle II and CMS were both customers Also Flavio, a Flavour Physics phenomenology software project
- ❑ Community projects Coffea and zfit had been created in the meantime
- ❑ Many packages got archived along the way
  - No worries, they got superseded by better products !
- ❑ Fast-forward to 2024 – around 30 packages maintained



Scikit-HEP project overview

- Not a description of all packages
- Not attempting to be comprehensive in any aspect
- The following catches your attention?
  Feel encouraged to look further at the repos … ☺

Topics and overview of main packages

Full list at https://scikit-hep.org/packages

Basics
Awkward(array)
Vector
hepunits

Data manipulation & interoperability
uproot
uproot-browser
hepconvert

Histogramming
boost-histogram
Hist
histoprint
UHI

HEP specific libraries and interfaces to HEP libraries
Particle
DecayLanguage
fastjet
pylhe
pyhepmc

Fitting and Statistics
iminuit
pyhf
cabinetry
resample
hepstats

Visualisation
mplhep

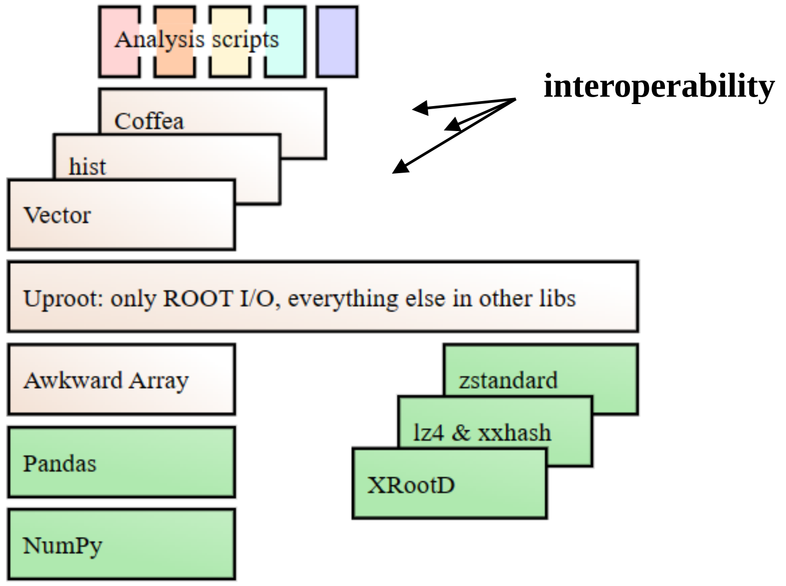
Basics, data manipulation, interoperability

Basics
Awkward(Array)
Vector
hepunits

Data manipulation & interoperability
uproot
uproot-browser
hepconvert

- Data manipulation is key
- So is interoperability to leverage efficiently both HEP specific tools and tools from the wider scientific ecosystem (based on NumPy)
- Uproot then awkward "filled the gap" and acted as an enabler / catalyst - uproot got showcased at CHEP 2018

- uproot:
  library for reading and writing ROOT files in pure Python and NumPy
- Awkward:
  library for nested, variable-sized data, including arbitrary-length lists, records, mixed types, and missing data, using NumPy-like idioms
- Vector:
  arrays of 2D, 3D, and Lorentz vectors
  (very recently: added a SymPy backend atop "standard backends")

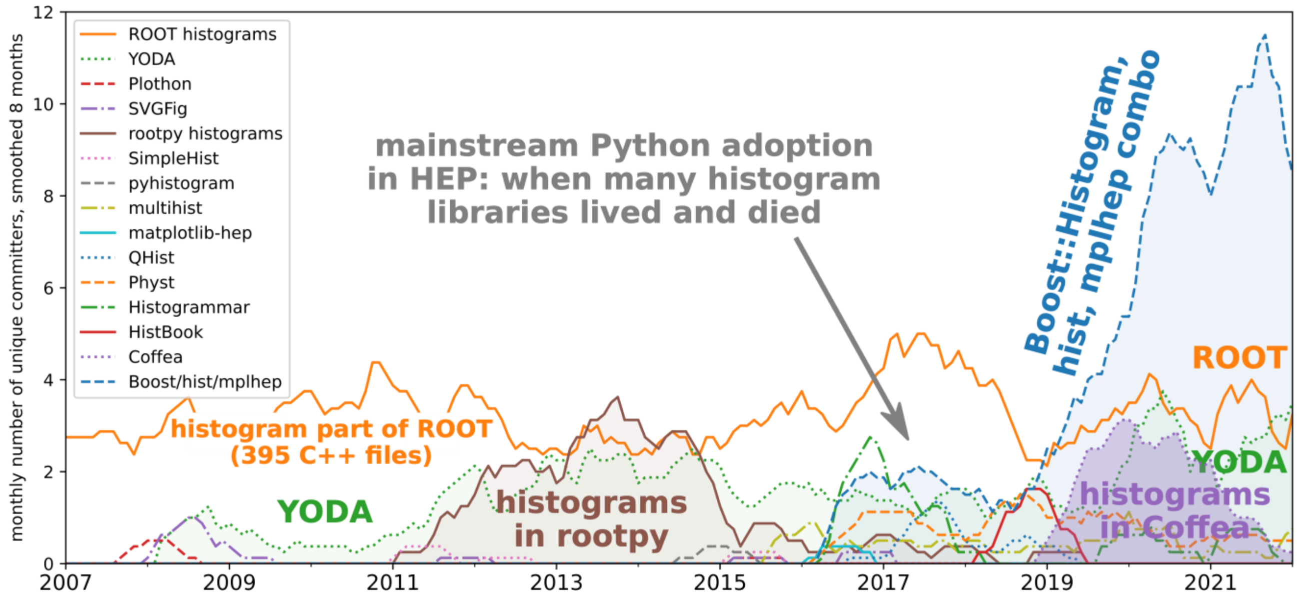


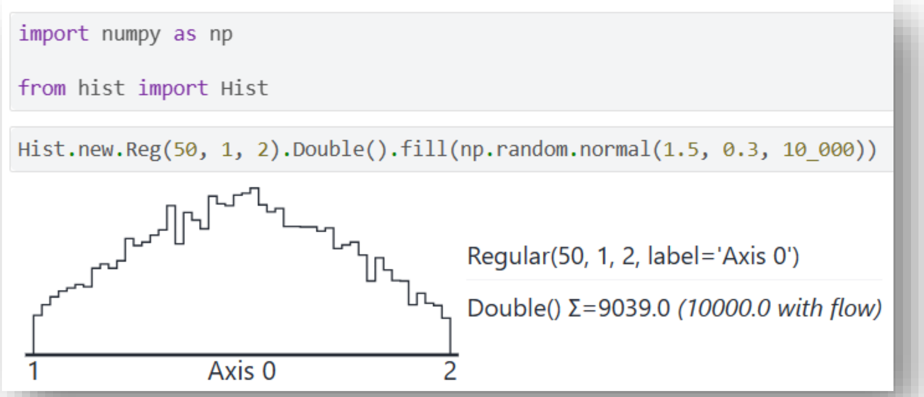
Histogramming

Histogramming
boost-histogram
Hist
histoprint
UHI

- No HEP analysis without histogramming ;-) !
- boost-histogram – Python bindings to C++ Boost.Histogram triggered a lot of activity …
- Came:
  Hist, an analyst-friendly front-end for boost-histogram
  UHI, a protocol package with interoperability in mind (see next slide)

- The "histogramming suite" has been one of the major successes
  - Illustration of the level of developer activity:





```
import numpy as np

from hist import Hist
```

```
Hist.new.Reg(50, 1, 2).Double().fill(np.random.normal(1.5, 0.3, 10_000))
```

```
Regular(50, 1, 2, label='Axis 0')
Double() Σ=9039.0 (10000.0 with flow)
```

Histogramming

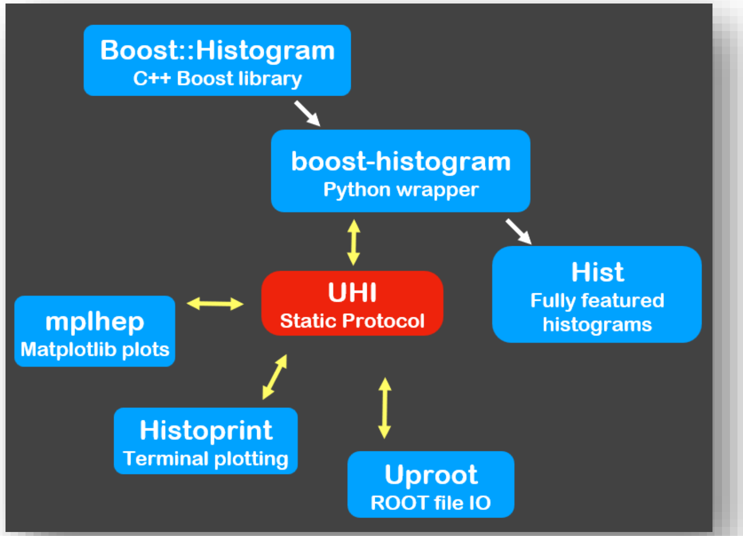
- **UHI – Universal Histogram Interface:**
  Primarily for documenting histogram indexing and the PlottableHistogram Protocol and any future cross-library standards
  ⇒ packages can collaborate thanks to agreeing on a shared protocol !

- **Protocol followed by other org packages:**
  boost-histogram, Hist, mplhep, histoprint, uproot

PlottableHistogram

Defines expected behaviors for producers

Consumers expect these behaviors



Fitting and Statistics

iminuit

- ❑ **Builds on the Minuit2 C++ code in ROOT**
- ❑ **Extensively used by other packages such as zfit, and also outside HEP, in Astro in particular**
- ❑ **Thorough unit testing has been revealing bugs in the C++ Minuit2, for which fixes upstream to ROOT have been submitted – iminuit also a gain for ROOT**

resample

- ❑ **Originally developed by David Saxton who works in the private sector as a statistician, "taken over" by us**
- ❑ **One of few libraries that implements the "extended bootstrap" technique**

Some future directions

- ❑ **Working to adopt and contribute to the HS^3 statistical representation to try to really unite the whole field on a common open statistical format**
- ❑ **Excellent to see a collaboration from**

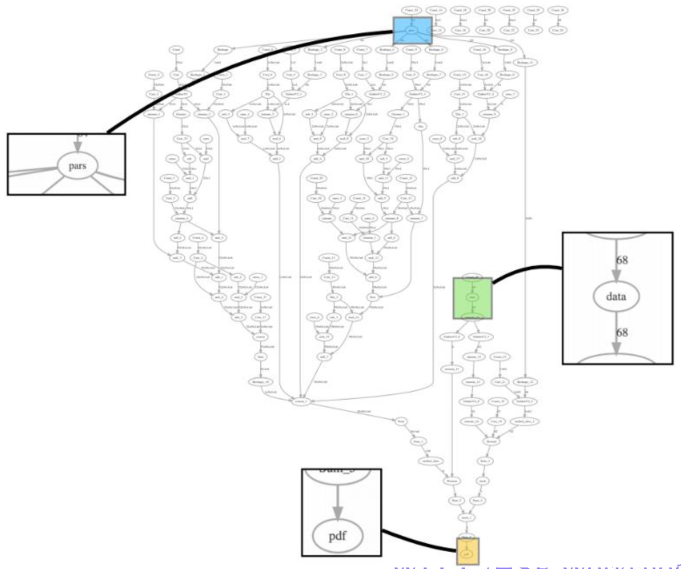
HS^3
High Energy Physics
Statistics Serialization Standard

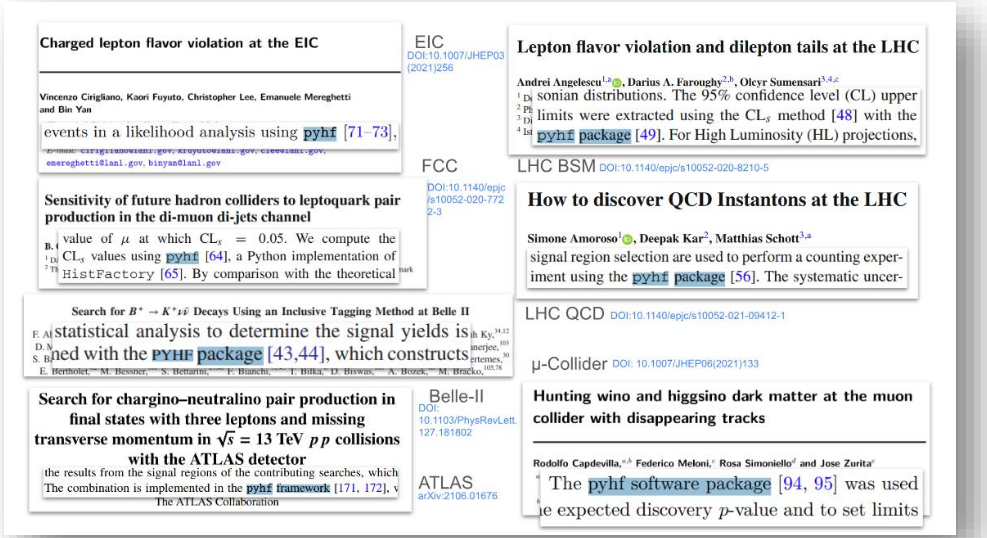
Fitting and Statistics

pyhf, cabinetry

- ❑ **pyhf has been a big hit ! The more that 100 citations attest this**
- ❑ **Also, widespread adoption of its use across multiple subfields (1) given it being Python and (2) given it provides the ability for open statistical model publication**

Example of a full HistFactory DAG visualisation:





Charged lepton flavor violation at the EIC

Vincenzo Cirigliano, Kaori Fuyuto, Christopher Lee, Emanuele Mereghetti and Bin Yan

events in a likelihood analysis using pyhf [71–73],

emereghetti@lanl.gov, binyan@lanl.gov

EIC DOI:10.1007/JHEP03(2021)256

Sensitivity of future hadron colliders to leptoquark pair production in the di-muon di-jets channel

value of μ at which $CL_s = 0.05$. We compute the CL_s values using pyhf [64], a Python implementation of HistFactory [65]. By comparison with the theoretical

FCC DOI:10.1140/epjc/s10052-020-7722-3

Search for $B^+ \to K^+ \nu \bar{\nu}$ Decays Using an Inclusive Tagging Method at Belle II

statistical analysis to determine the signal yields is ned with the PYHF package [43,44], which constructs

E. Bertholet, M. Bessner, S. Bettarini, F. Bianchi, T. Bilka, D. Biswas, A. Bozek, M. Bračko,

Belle-II DOI: 10.1103/PhysRevLett.127.181802

Search for chargino–neutralino pair production in final states with three leptons and missing transverse momentum in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector

the results from the signal regions of the contributing searches, which The combination is implemented in the pyhf framework [171, 172],

The ATLAS Collaboration

ATLAS arXiv:2106.01676

Lepton flavor violation and dilepton tails at the LHC

Andrei Angelescu[1,a], Darius A. Faroughy[2,b], Olcyr Sumensari[3,4,c]

sonian distributions. The 95% confidence level (CL) upper limits were extracted using the CL_s method [48] with the pyhf package [49]. For High Luminosity (HL) projections,

LHC BSM DOI:10.1140/epjc/s10052-020-8210-5

How to discover QCD Instantons at the LHC

Simone Amoroso[1], Deepak Kar[2], Matthias Schott[3,a]

signal region selection are used to perform a counting experiment using the pyhf package [56]. The systematic uncer-

LHC QCD DOI:10.1140/epjc/s10052-021-09412-1

μ-Collider DOI: 10.1007/JHEP06(2021)133

Hunting wino and higgsino dark matter at the muon collider with disappearing tracks

Rodolfo Capdevilla,[a,b] Federico Meloni,[c] Rosa Simoniello[d] and Jose Zurita[e]

The pyhf software package [94, 95] was used e expected discovery p-value and to set limits

Fitting and Statistics

Fitting and Statistics
iminuit
pyhf
cabinetry
hepstats

Future of pyhf as as “building block”

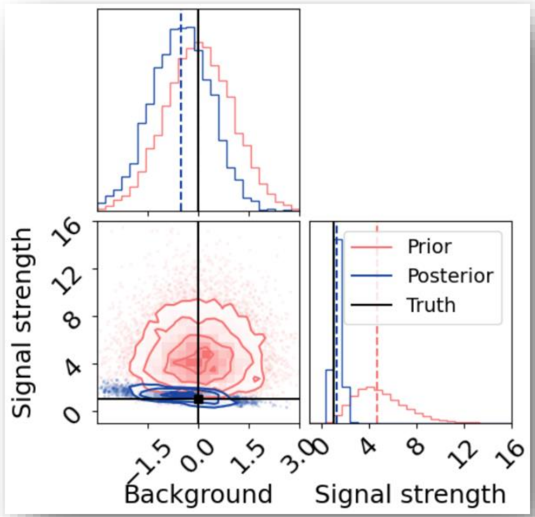
- ❑ **Working to become a tool that integrates well with others and/or can be leveraged**
- ❑ **Specific examples:**

- Paper "Bayesian Methodologies with pyhf“:

“bayesian_pyhf is a Python package that allows for the parallel Bayesian and frequentist evaluation of multi-channel binned statistical models. The Python library pyhf is used to build such models according to the HistFactory framework …”

- spey-pyhf plugin for inference in phenomenology

Smooth inference for reinterpretation studies



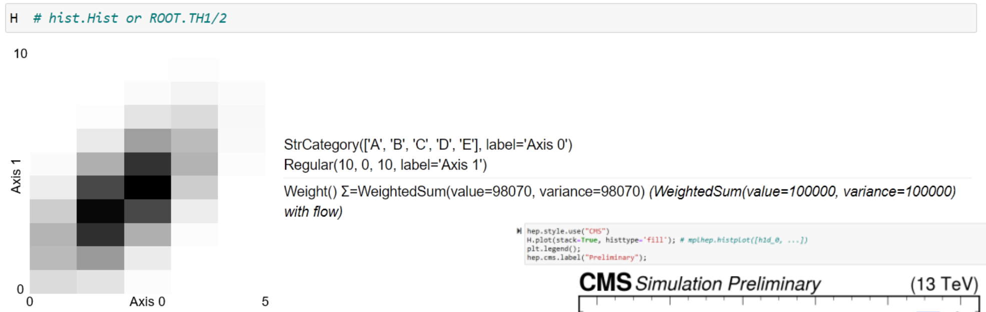
(a) Correlation for signal and background distribution. Indicated in black is the underlying truth.

Taken from Paper "Bayesian Methodologies with pyhf“

Visualisation

❑ **mplhep is (only!) about easy matplotlib histogramming / styling for HEP plots**

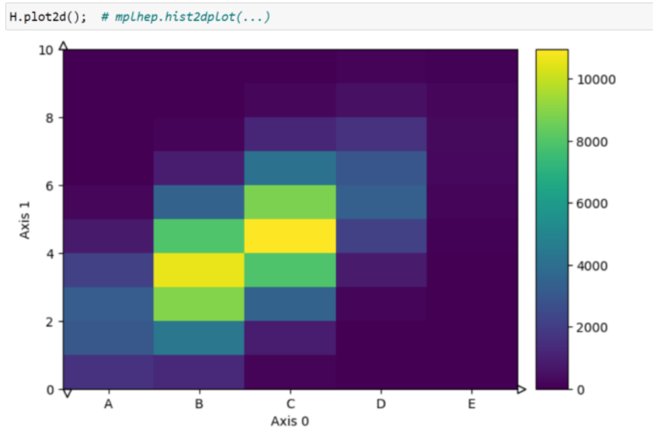
- **Convenient user interface for histogram plotting**
- **Interface to matplotlib, fully integrated with Hist**
- **Can also ingest ROOT TH1/TH2**



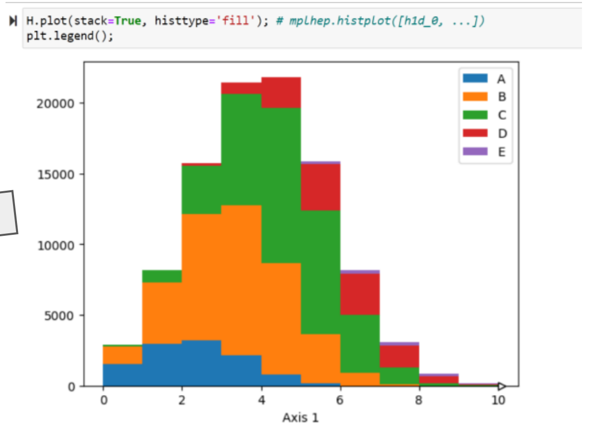
```
H  # hist.Hist or ROOT.TH1/2
```

StrCategory(['A', 'B', 'C', 'D', 'E'], label='Axis 0')
Regular(10, 0, 10, label='Axis 1')

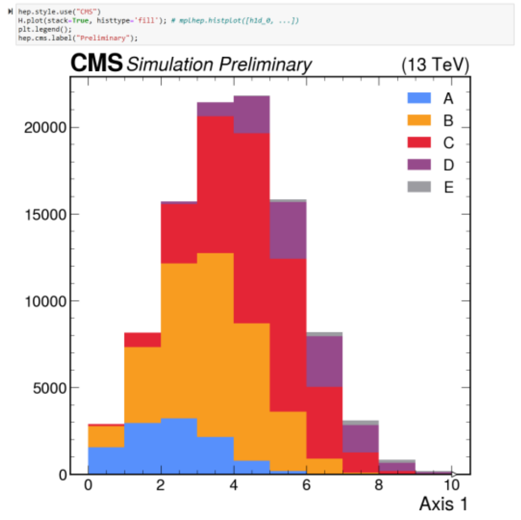
Weight() Σ=WeightedSum(value=98070, variance=98070) *(WeightedSum(value=100000, variance=100000) with flow)*



```
H.plot2d();  # mplhep.hist2dplot(...)
```



```
H.plot(stack=True, histtype='fill'); # mplhep.histplot([h1d_0, ...])
plt.legend();
```



```
hep.style.use("CMS")
H.plot(stack=True, histtype='fill'); # mplhep.histplot([h1d_0, ...])
plt.legend();
hep.cms.label("Preliminary");
```

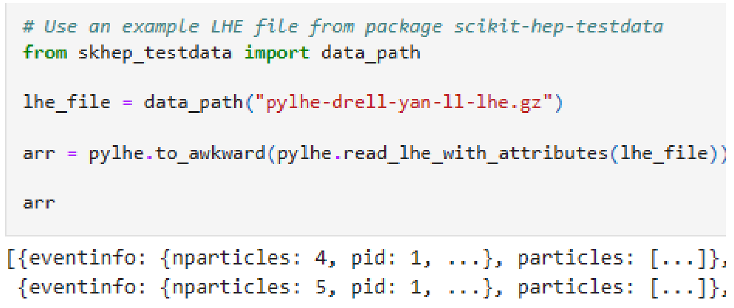
Publication quality plots conforming to collaboration requirements (ATLAS, CMS, LHCb, ALICE) can be produced with 2 lines of code

For CMS - CVD friendly color scheme (M. Petroff colors)

HEP domain-specific libraries

HEP specific libraries and interfaces to HEP libraries
Particle
DecayLanguage
fastjet
pylhe
pyhepmc

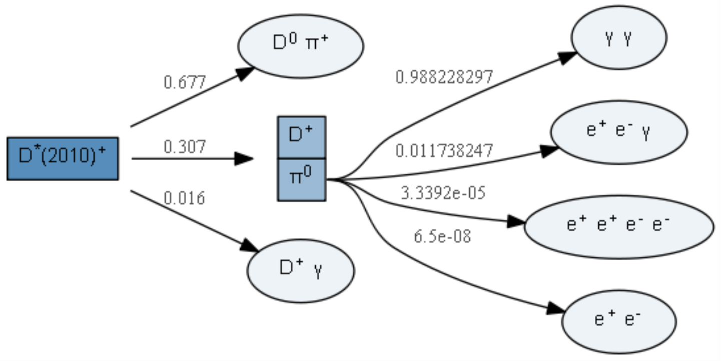
pylhe



```
# Use an example LHE file from package scikit-hep-testdata
from skhep_testdata import data_path

lhe_file = data_path("pylhe-drell-yan-ll-lhe.gz")

arr = pylhe.to_awkward(pylhe.read_lhe_with_attributes(lhe_file))

arr
```

```
[{eventinfo: {nparticles: 4, pid: 1, ...}, particles: [...]},
 {eventinfo: {nparticles: 5, pid: 1, ...}, particles: [...]},
```

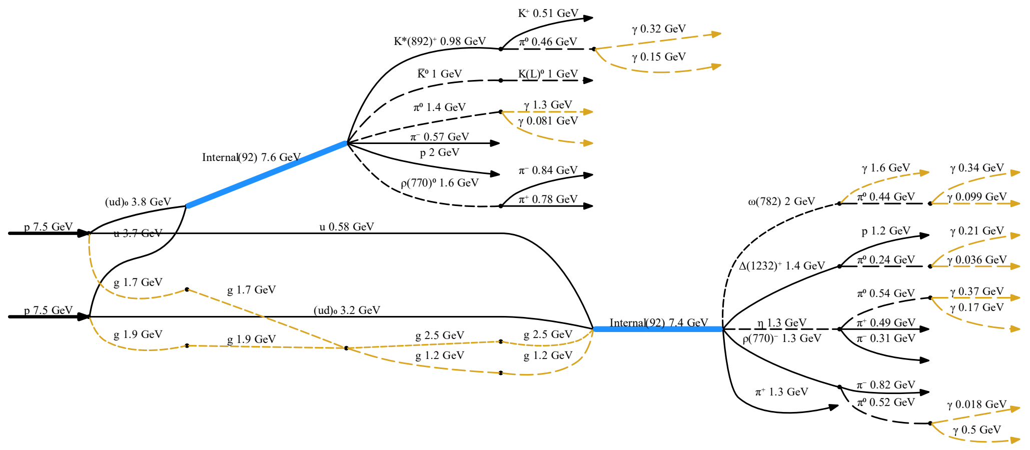
DecayLanguage



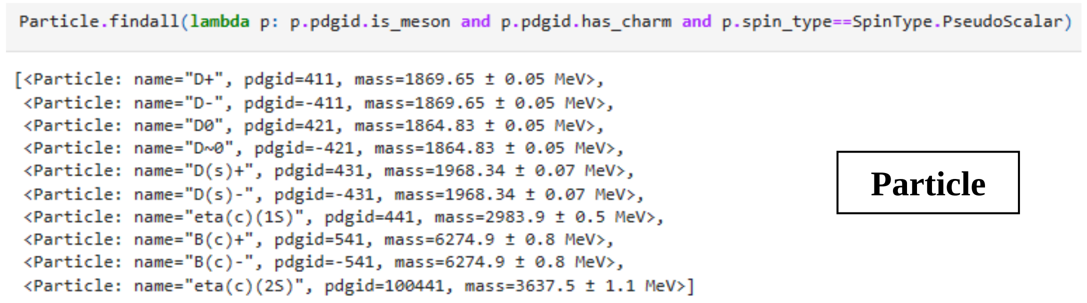
fastjet

Python bindings for the C++ FastJet library.
Contains 2 interfaces – the Classic and the Awkward(array)

pyhepmc



Find all pseudoscalar charm mesons known to the PDG:



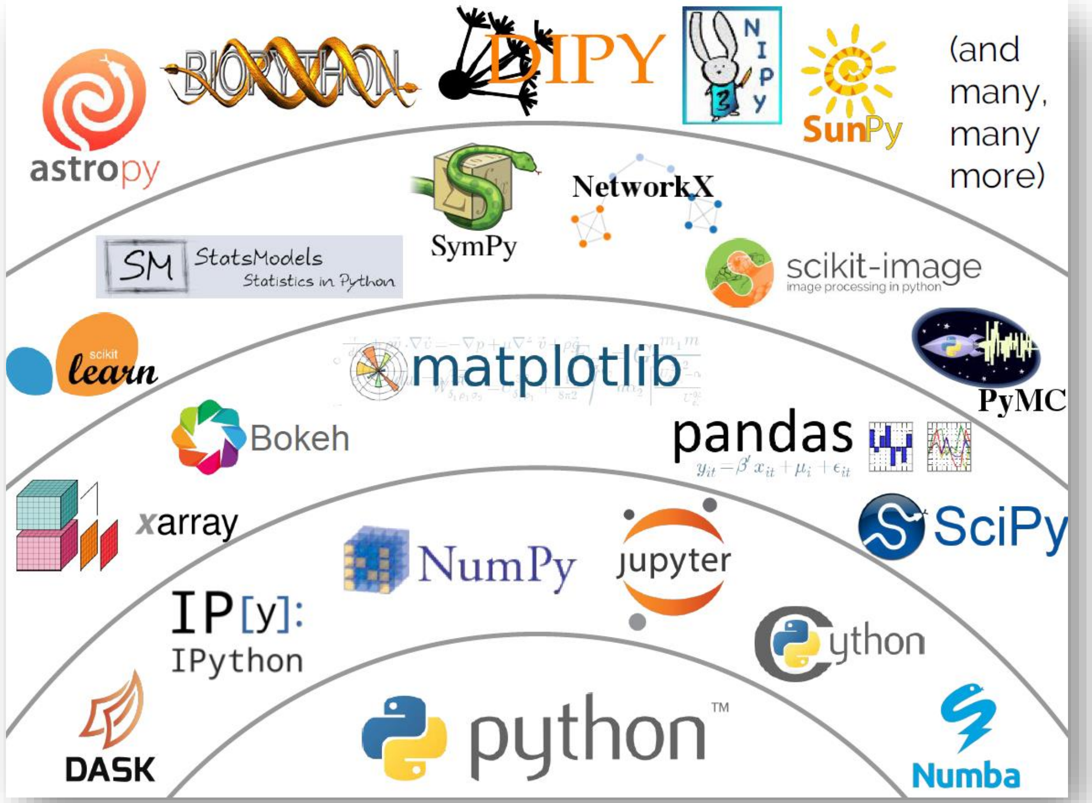
```
Particle.findall(lambda p: p.pdgid.is_meson and p.pdgid.has_charm and p.spin_type==SpinType.PseudoScalar)
```

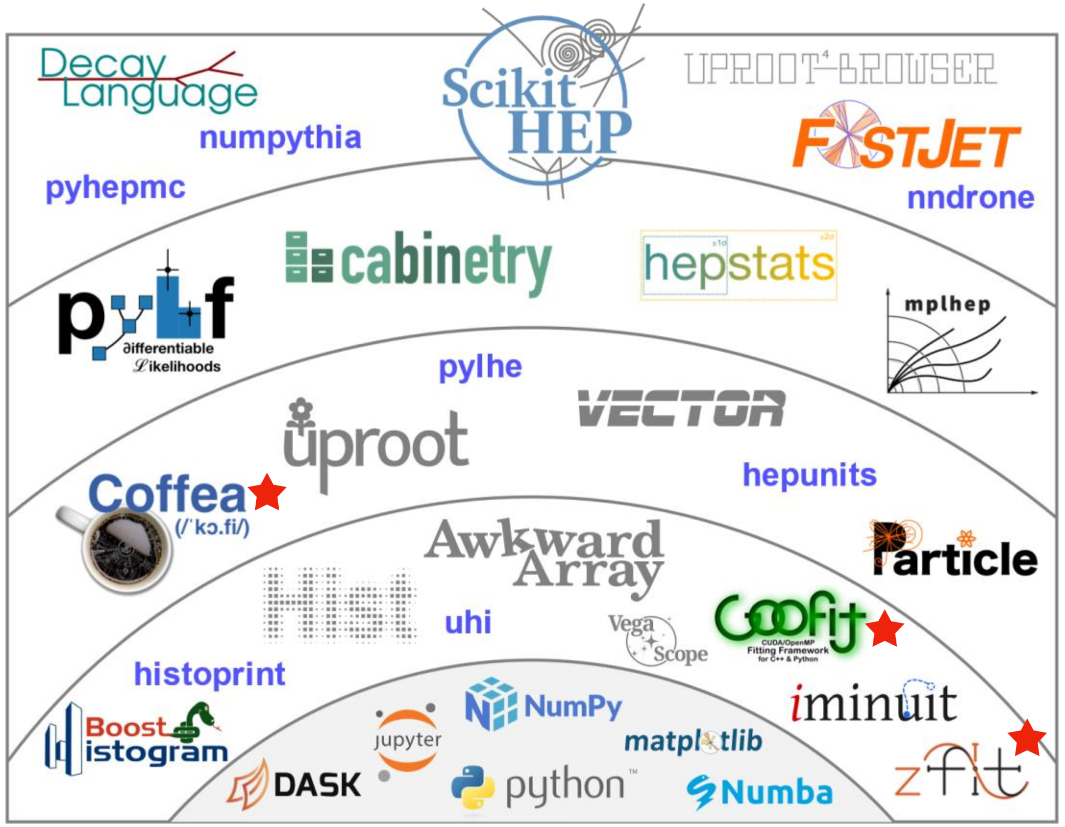
```
[<Particle: name="D+", pdgid=411, mass=1869.65 ± 0.05 MeV>,
 <Particle: name="D-", pdgid=-411, mass=1869.65 ± 0.05 MeV>,
 <Particle: name="D0", pdgid=421, mass=1864.83 ± 0.05 MeV>,
 <Particle: name="D~0", pdgid=-421, mass=1864.83 ± 0.05 MeV>,
 <Particle: name="D(s)+", pdgid=431, mass=1968.34 ± 0.07 MeV>,
 <Particle: name="D(s)-", pdgid=-431, mass=1968.34 ± 0.07 MeV>,
 <Particle: name="eta(c)(1S)", pdgid=441, mass=2983.9 ± 0.5 MeV>,
 <Particle: name="B(c)+", pdgid=541, mass=6274.9 ± 0.8 MeV>,
 <Particle: name="B(c)-", pdgid=-541, mass=6274.9 ± 0.8 MeV>,
 <Particle: name="eta(c)(2S)", pdgid=100441, mass=3637.5 ± 1.1 MeV>]
```

Particle

Our contribution to the HEP domain-specific Python analysis ecosystem

- ❑ **After 7-ish years we can proudly splash our “NumPy fueled” shell ecosystem side-by-side ☺ !**
  - **- The full HEP ecosystem is of course wider, ROOT being prominent, in particular**



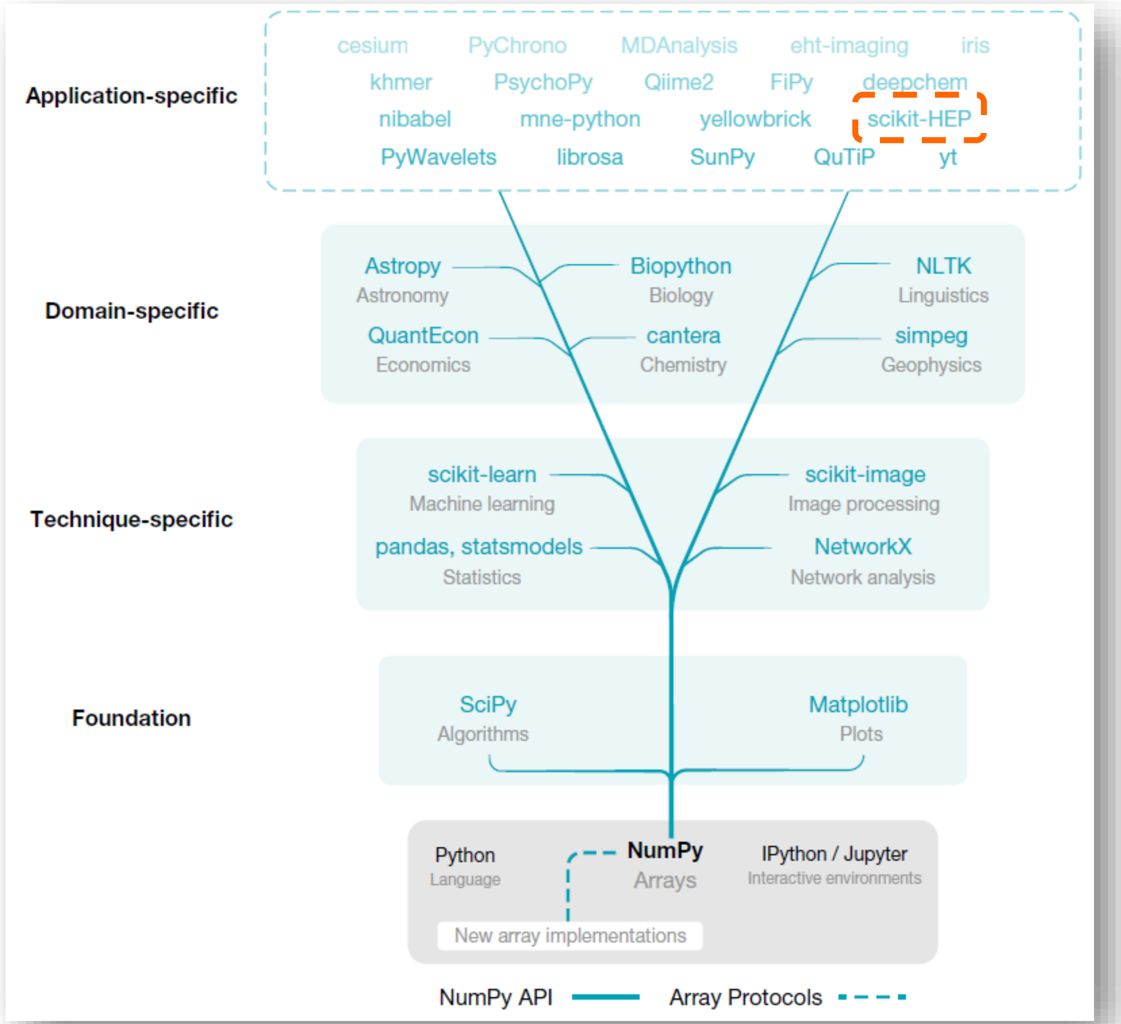


Affiliated package

Our contribution to the HEP domain-specific Python analysis ecosystem

- It was a good surprise to see Scikit-HEP in the "Array programming with NumPy" Nature paper in 2020!
  - Recognition that "HEP is out there as well"

(It should be under "Domain-specific", but fine ☺.)



Array programming with NumPy, Nature 585, 357 (2020)

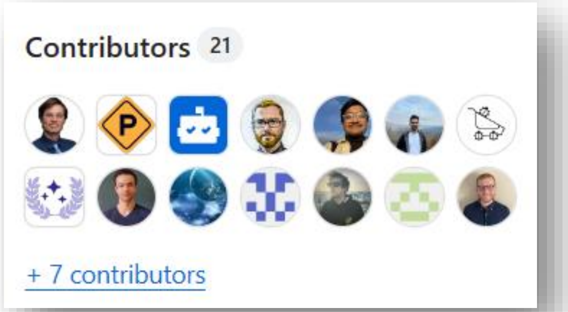
HEP community engagement

- Recognition of contributions and contributors
- Training material
- Community events

Engagement with community also means recognition of contributors !

- That seems totally natural to be “bijective” in matters of recognition and citation
- Seems effectively compulsory for community projects !

- Contributions to software are easily found in GitHub and GitLab repositories
  - Example in https://github.com/scikit-hep/boost-histogram



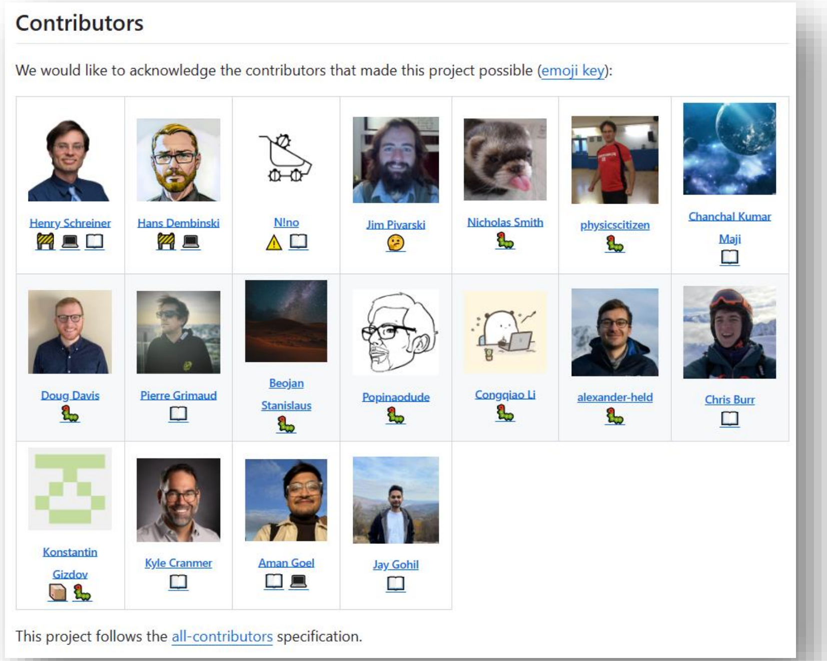
Contributors 21

+ 7 contributors

- But there are means to highlight contributions even more
- Our guidelines – use the all-contributors specification

All Contributors

- Caveat: only works on Markdown (README) files, not ReST



Contributors

We would like to acknowledge the contributors that made this project possible (emoji key):

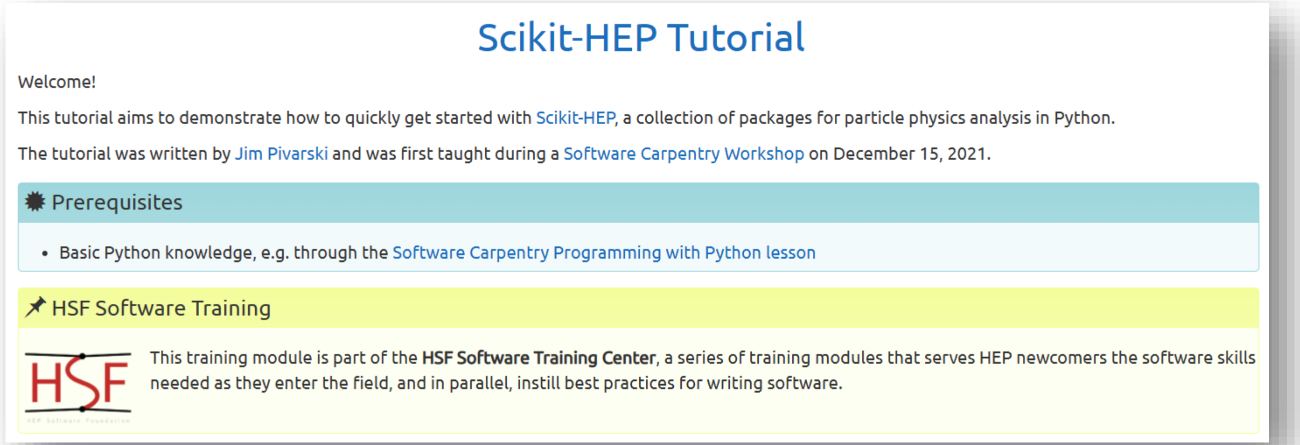
Henry Schreiner 🚧 💻 📖	Hans Dembinski 🚧 💻	N!no ⚠️ 📖	Jim Pivarski 🤔	Nicholas Smith 🐛	physicscitizen 🐛	Chanchal Kumar Maji 📖
Doug Davis 🐛	Pierre Grimaud 📖	Beojan Stanislaus 🐛	Popinaodude 🐛	Congqiao Li 🐛	alexander-held 🐛	Chris Burr 📖
Konstantin Gizdov 📦 🐛	Kyle Cranmer 📖	Aman Goel 📖 💻	Jay Gohil 📖			

This project follows the all-contributors specification.

Engagement via training and training material

- ❑ Training material on Scikit-HEP packages contributed to the HSF (Training WG) courses
- ❑ More recently, engagement also in training in general with an HSF Training WG convenership

- ❑ Next event "HSF/IRIS-HEP Python for Analysis Training" to take place on June 5th - https://indico.cern.ch/event/1408846/
  - Worth emphasising fact – it's community colleagues who take time to deliver these courses on ROOT, Scikit-HEP, etc. !



Scikit-HEP Tutorial

Welcome!

This tutorial aims to demonstrate how to quickly get started with Scikit-HEP, a collection of packages for particle physics analysis in Python.

The tutorial was written by Jim Pivarski and was first taught during a Software Carpentry Workshop on December 15, 2021.

Prerequisites

- Basic Python knowledge, e.g. through the Software Carpentry Programming with Python lesson

HSF Software Training

This training module is part of the HSF Software Training Center, a series of training modules that serves HEP newcomers the software skills needed as they enter the field, and in parallel, instill best practices for writing software.

https://github.com/hsf-training/hsf-training-scikit-hep-webpage

Engagement via workshops / conferences

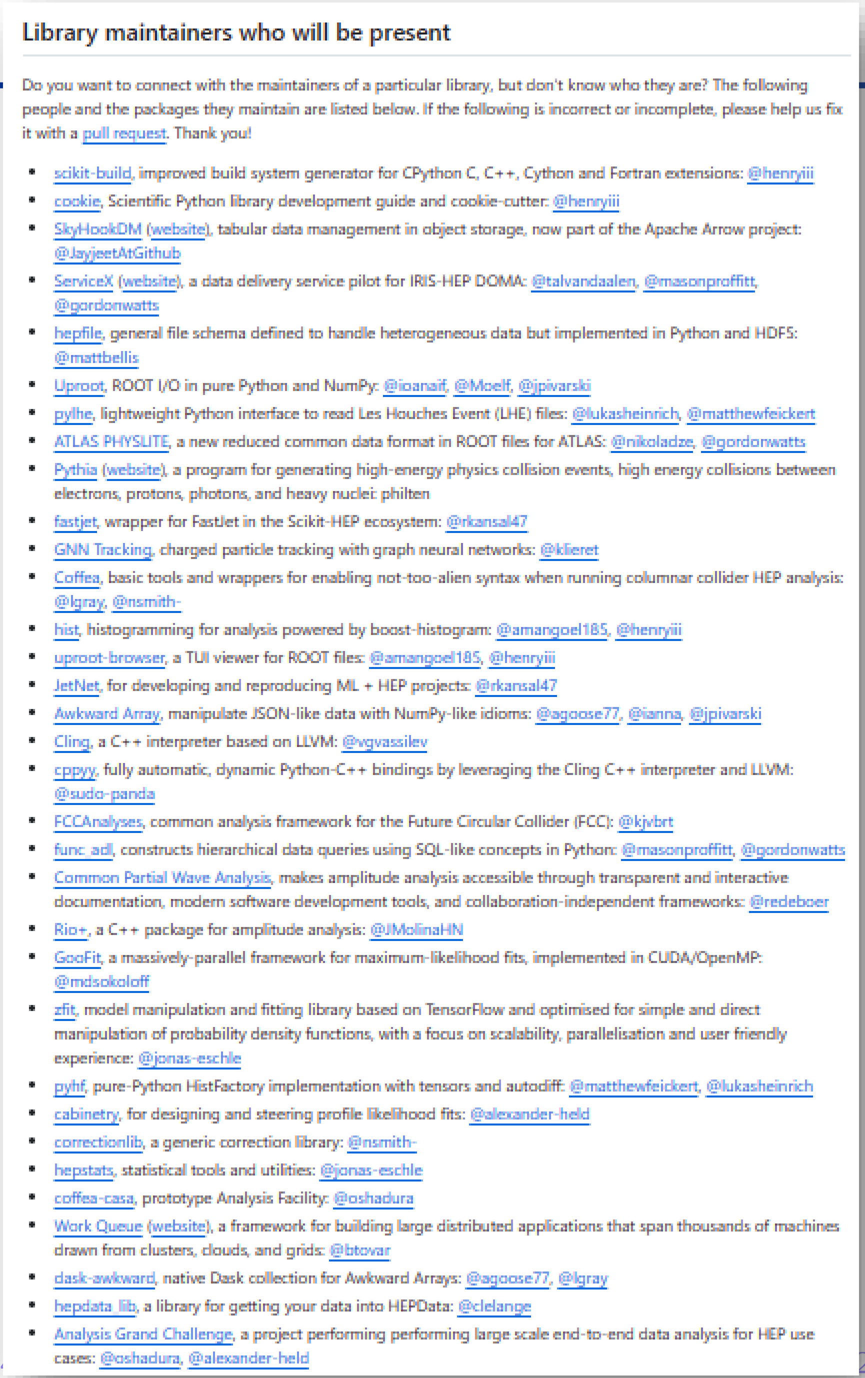
General events

- ❑ Scikit-HEP gets represented regularly
  - We engage proactively
  - Also in non-HEP events such as the SciPy Conferences

PyHEP workshops

- ❑ Crucial series of workshops to exchange on "Python in HEP"

- ❑ In 2023 a first PyHEP.dev "developers workshop" took place
- ❑ Preparations have been done via a dedicated repository https://github.com/HSF/PyHEP.dev-workshops
- ❑ List of community libraries / projects presented is very large!



Library maintainers who will be present

Do you want to connect with the maintainers of a particular library, but don't know who they are? The following people and the packages they maintain are listed below. If the following is incorrect or incomplete, please help us fix it with a pull request. Thank you!

- scikit-build, improved build system generator for CPython C, C++, Cython and Fortran extensions: @henryiii
- cookie, Scientific Python library development guide and cookie-cutter: @henryiii
- SkyHookDM (website), tabular data management in object storage, now part of the Apache Arrow project: @JayjeetAtGithub
- ServiceX (website), a data delivery service pilot for IRIS-HEP DOMA: @talvandaalen, @masonproffitt, @gordonwatts
- hepfile, general file schema defined to handle heterogeneous data but implemented in Python and HDF5: @mattbellis
- Uproot, ROOT I/O in pure Python and NumPy: @ioanaif, @Moelf, @jpivarski
- pylhe, lightweight Python interface to read Les Houches Event (LHE) files: @lukasheinrich, @matthewfeickert
- ATLAS PHYSLITE, a new reduced common data format in ROOT files for ATLAS: @nikoladze, @gordonwatts
- Pythia (website), a program for generating high-energy physics collision events, high energy collisions between electrons, protons, photons, and heavy nuclei: philten
- fastjet, wrapper for FastJet in the Scikit-HEP ecosystem: @rkansal47
- GNN Tracking, charged particle tracking with graph neural networks: @klieret
- Coffea, basic tools and wrappers for enabling not-too-alien syntax when running columnar collider HEP analysis: @lgray, @nsmith-
- hist, histogramming for analysis powered by boost-histogram: @amangoel185, @henryiii
- uproot-browser, a TUI viewer for ROOT files: @amangoel185, @henryiii
- JetNet, for developing and reproducing ML + HEP projects: @rkansal47
- Awkward Array, manipulate JSON-like data with NumPy-like idioms: @agoose77, @ianna, @jpivarski
- Cling, a C++ interpreter based on LLVM: @vgvassilev
- cppyy, fully automatic, dynamic Python-C++ bindings by leveraging the Cling C++ interpreter and LLVM: @sudo-panda
- FCCAnalyses, common analysis framework for the Future Circular Collider (FCC): @kjvbrt
- func_adl, constructs hierarchical data queries using SQL-like concepts in Python: @masonproffitt, @gordonwatts
- Common Partial Wave Analysis, makes amplitude analysis accessible through transparent and interactive documentation, modern software development tools, and collaboration-independent frameworks: @redeboer
- Rio+, a C++ package for amplitude analysis: @JMolinaHN
- GooFit, a massively-parallel framework for maximum-likelihood fits, implemented in CUDA/OpenMP: @mdsokoloff
- zfit, model manipulation and fitting library based on TensorFlow and optimised for simple and direct manipulation of probability density functions, with a focus on scalability, parallelisation and user friendly experience: @jonas-eschle
- pyhf, pure-Python HistFactory implementation with tensors and autodiff: @matthewfeickert, @lukasheinrich
- cabinetry, for designing and steering profile likelihood fits: @alexander-held
- correctionlib, a generic correction library: @nsmith-
- hepstats, statistical tools and utilities: @jonas-eschle
- coffea-casa, prototype Analysis Facility: @oshadura
- Work Queue (website), a framework for building large distributed applications that span thousands of machines drawn from clusters, clouds, and grids: @btovar
- dask-awkward, native Dask collection for Awkward Arrays: @agoose77, @lgray
- hepdata_lib, a library for getting your data into HEPData: @clelange
- Analysis Grand Challenge, a project performing performing large scale end-to-end data analysis for HEP use cases: @oshadura, @alexander-held

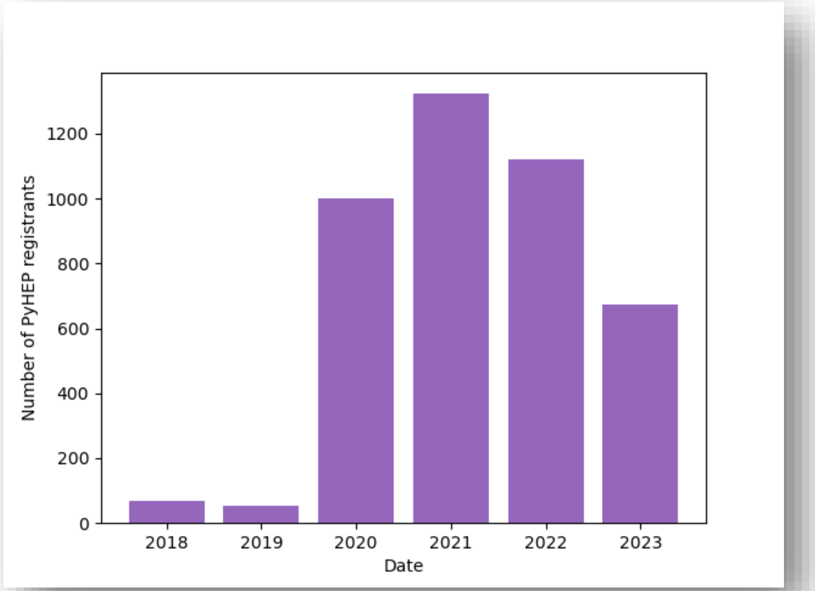
PyHEP workshops – aside advert ☺

- ❑ Crucial series of workshops to exchange on "Python in HEP"

- ❑ 2023 was the first year with 2 workshops !
  - It became relevant to complement the "Users workshop" held online since 2020 with an in-person, informal workshop for developers of Python software in HEP to plan a coherent roadmap and make priorities for the upcoming year

- ❑ Upcoming events:
  PyHEP 2024 "Users workshop", online, July 1-4
  PyHEP.dev 2024 "Developers workshop", Aachen, August 26-30
  - Registration is open, as well as the call for abstracts !

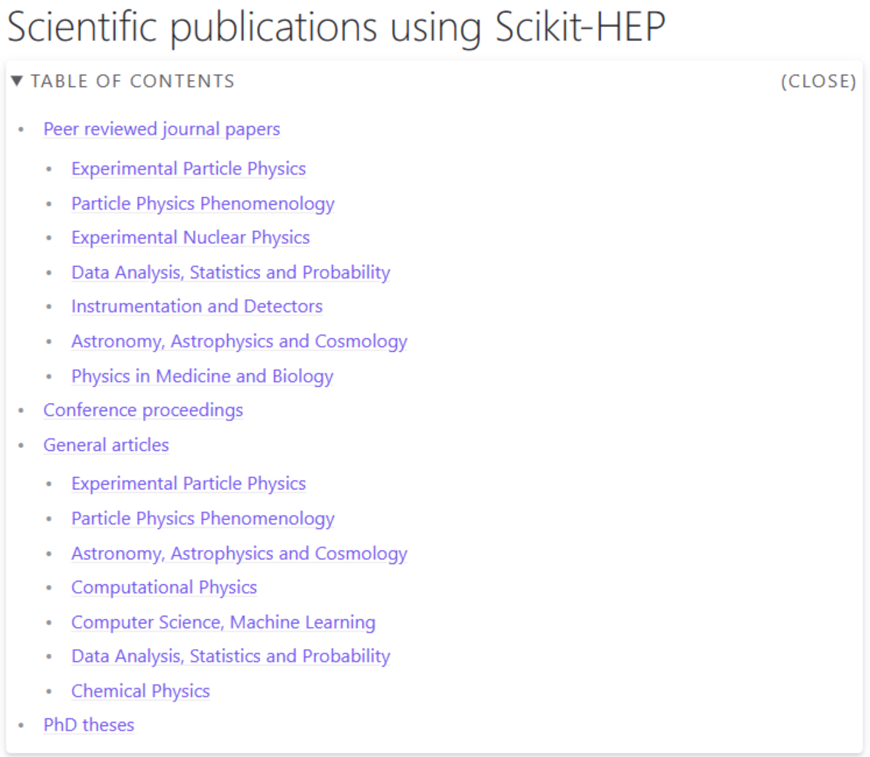
Number of PyHEP registrations by year, showing the huge jump when we went online.



Snapshots from PyHEP.dev 2023 sessions

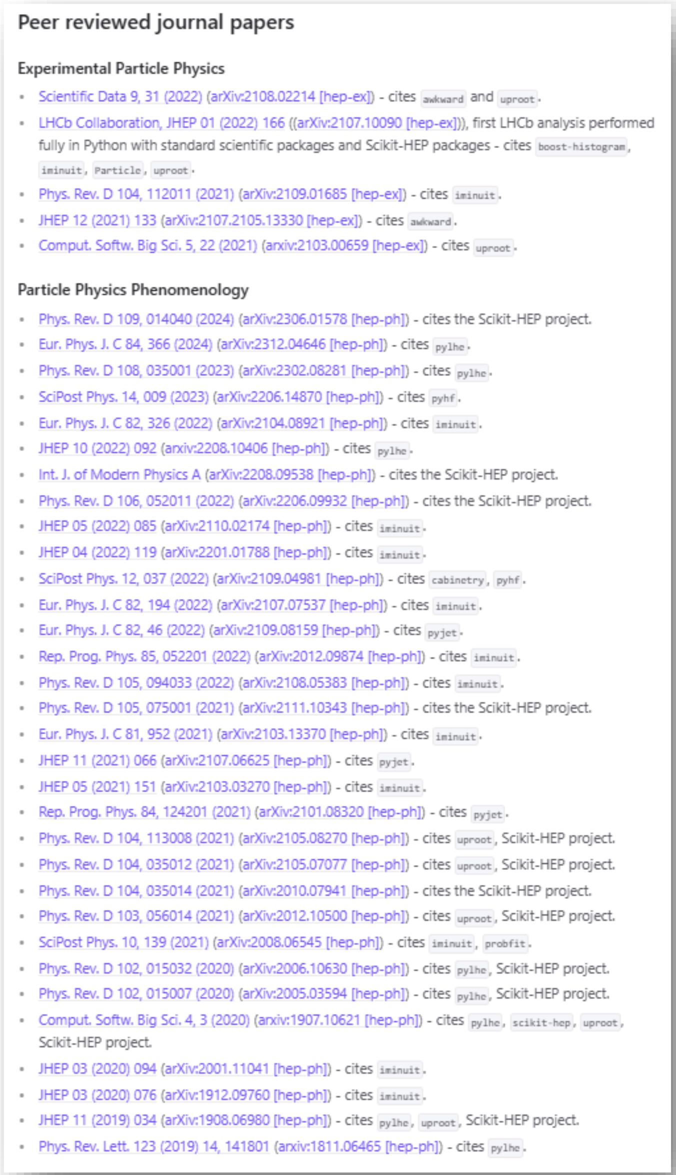
Measuring relevance & impact – scientific publications using Scikit-HEP

- ❑ **Scikit-HEP packages cited in a wide range of publications, within HEP and far beyond !**
- ❑ **Interestingly enough, HEP phenomenologists seem to be doing a much better job at citing the software they use !**



Scientific publications using Scikit-HEP

▼ TABLE OF CONTENTS (CLOSE)



Peer reviewed journal papers

Experimental Particle Physics

- Scientific Data 9, 31 (2022) (arXiv:2108.02214 [hep-ex]) - cites `awkward` and `uproot`.
- LHCb Collaboration, JHEP 01 (2022) 166 ((arXiv:2107.10090 [hep-ex])), first LHCb analysis performed fully in Python with standard scientific packages and Scikit-HEP packages - cites `boost-histogram`, `iminuit`, `Particle`, `uproot`.
- Phys. Rev. D 104, 112011 (2021) (arXiv:2109.01685 [hep-ex]) - cites `iminuit`.
- JHEP 12 (2021) 133 (arXiv:2107.2105.13330 [hep-ex]) - cites `awkward`.
- Comput. Softw. Big Sci. 5, 22 (2021) (arxiv:2103.00659 [hep-ex]) - cites `uproot`.

Particle Physics Phenomenology

- Phys. Rev. D 109, 014040 (2024) (arXiv:2306.01578 [hep-ph]) - cites the Scikit-HEP project.
- Eur. Phys. J. C 84, 366 (2024) (arXiv:2312.04646 [hep-ph]) - cites `pylhe`.
- Phys. Rev. D 108, 035001 (2023) (arXiv:2302.08281 [hep-ph]) - cites `pylhe`.
- SciPost Phys. 14, 009 (2023) (arXiv:2206.14870 [hep-ph]) - cites `pyhf`.
- Eur. Phys. J. C 82, 326 (2022) (arXiv:2104.08921 [hep-ph]) - cites `iminuit`.
- JHEP 10 (2022) 092 (arxiv:2208.10406 [hep-ph]) - cites `pylhe`.
- Int. J. of Modern Physics A (arXiv:2208.09538 [hep-ph]) - cites the Scikit-HEP project.
- Phys. Rev. D 106, 052011 (2022) (arXiv:2206.09932 [hep-ph]) - cites the Scikit-HEP project.
- JHEP 05 (2022) 085 (arXiv:2110.02174 [hep-ph]) - cites `iminuit`.
- JHEP 04 (2022) 119 (arXiv:2201.01788 [hep-ph]) - cites `iminuit`.
- SciPost Phys. 12, 037 (2022) (arXiv:2109.04981 [hep-ph]) - cites `cabinetry`, `pyhf`.
- Eur. Phys. J. C 82, 194 (2022) (arXiv:2107.07537 [hep-ph]) - cites `iminuit`.
- Eur. Phys. J. C 82, 46 (2022) (arXiv:2109.08159 [hep-ph]) - cites `pyjet`.
- Rep. Prog. Phys. 85, 052201 (2022) (arXiv:2012.09874 [hep-ph]) - cites `iminuit`.
- Phys. Rev. D 105, 094033 (2022) (arXiv:2108.05383 [hep-ph]) - cites `iminuit`.
- Phys. Rev. D 105, 075001 (2021) (arXiv:2111.10343 [hep-ph]) - cites the Scikit-HEP project.
- Eur. Phys. J. C 81, 952 (2021) (arXiv:2103.13370 [hep-ph]) - cites `iminuit`.
- JHEP 11 (2021) 066 (arXiv:2107.06625 [hep-ph]) - cites `pyjet`.
- JHEP 05 (2021) 151 (arXiv:2103.03270 [hep-ph]) - cites `iminuit`.
- Rep. Prog. Phys. 84, 124201 (2021) (arXiv:2101.08320 [hep-ph]) - cites `pyjet`.
- Phys. Rev. D 104, 113008 (2021) (arXiv:2105.08270 [hep-ph]) - cites `uproot`, Scikit-HEP project.
- Phys. Rev. D 104, 035012 (2021) (arXiv:2105.07077 [hep-ph]) - cites `uproot`, Scikit-HEP project.
- Phys. Rev. D 104, 035014 (2021) (arXiv:2010.07941 [hep-ph]) - cites the Scikit-HEP project.
- Phys. Rev. D 103, 056014 (2021) (arXiv:2012.10500 [hep-ph]) - cites `uproot`, Scikit-HEP project.
- SciPost Phys. 10, 139 (2021) (arXiv:2008.06545 [hep-ph]) - cites `iminuit`, `probfit`.
- Phys. Rev. D 102, 015032 (2020) (arXiv:2006.10630 [hep-ph]) - cites `pylhe`, Scikit-HEP project.
- Phys. Rev. D 102, 015007 (2020) (arXiv:2005.03594 [hep-ph]) - cites `pylhe`, Scikit-HEP project.
- Comput. Softw. Big Sci. 4, 3 (2020) (arxiv:1907.10621 [hep-ph]) - cites `pylhe`, `scikit-hep`, `uproot`, Scikit-HEP project.
- JHEP 03 (2020) 094 (arXiv:2001.11041 [hep-ph]) - cites `iminuit`.
- JHEP 03 (2020) 076 (arXiv:1912.09760 [hep-ph]) - cites `iminuit`.
- JHEP 11 (2019) 034 (arXiv:1908.06980 [hep-ph]) - cites `pylhe`, `uproot`, Scikit-HEP project.
- Phys. Rev. Lett. 123 (2019) 14, 141801 (arxiv:1811.06465 [hep-ph]) - cites `pylhe`.

Measuring relevance & impact – «projects» (in HEP) using Scikit-HEP

ATLAS EXPERIMENT

CMS

KM3NeT

Belle II

LHCb

Flavio

Coffea

zfit

Engagement with broader scientific Python community

- Contributions “upstream” to Python scientific ecosystem
- Engagement and impact

Engagement with the broader scientific Python community

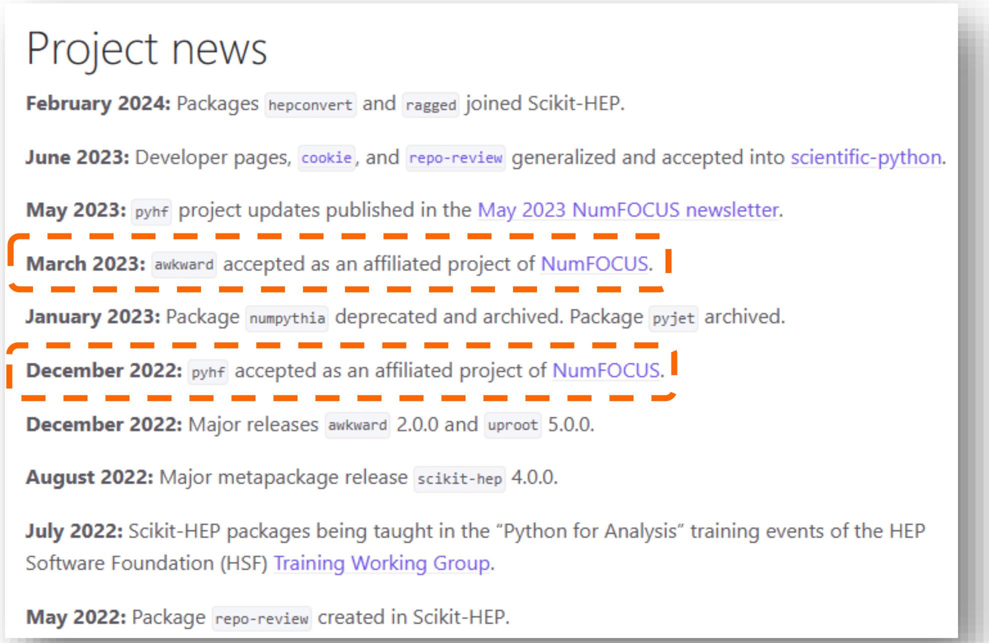
Python ecosystem in general

- We made contributions “upstream” to matplotlib and scipy
- Long ago we built our own wheel building tooling (for Azure DevOps), then merged the features we needed into cibuildwheel, and have been a major contributor ever since
- We contributed to pybind11 and scikit-build and support scikit-build-core, the modern CMake-based build backend
- HEP specific – we also did a fair share of the work to make ROOT available on Conda

NUMFOCUS
OPEN CODE = BETTER SCIENCE

NumFOCUS

- 2 packages (pyhf, then awkward) are NumFOCUS Affiliated Projects
- Affiliation brings visibility and various other benefits
- Working with the broader Scientific Python ecosystem to make sure that we adopt better sustainability practices



Project news

February 2024: Packages `hepconvert` and `ragged` joined Scikit-HEP.

June 2023: Developer pages, `cookie`, and `repo-review` generalized and accepted into scientific-python.

May 2023: `pyhf` project updates published in the May 2023 NumFOCUS newsletter.

March 2023: `awkward` accepted as an affiliated project of NumFOCUS.

January 2023: Package `numpythia` deprecated and archived. Package `pyjet` archived.

December 2022: `pyhf` accepted as an affiliated project of NumFOCUS.

December 2022: Major releases `awkward` 2.0.0 and `uproot` 5.0.0.

August 2022: Major metapackage release `scikit-hep` 4.0.0.

July 2022: Scikit-HEP packages being taught in the “Python for Analysis” training events of the HEP Software Foundation (HSF) Training Working Group.

May 2022: Package `repo-review` created in Scikit-HEP.

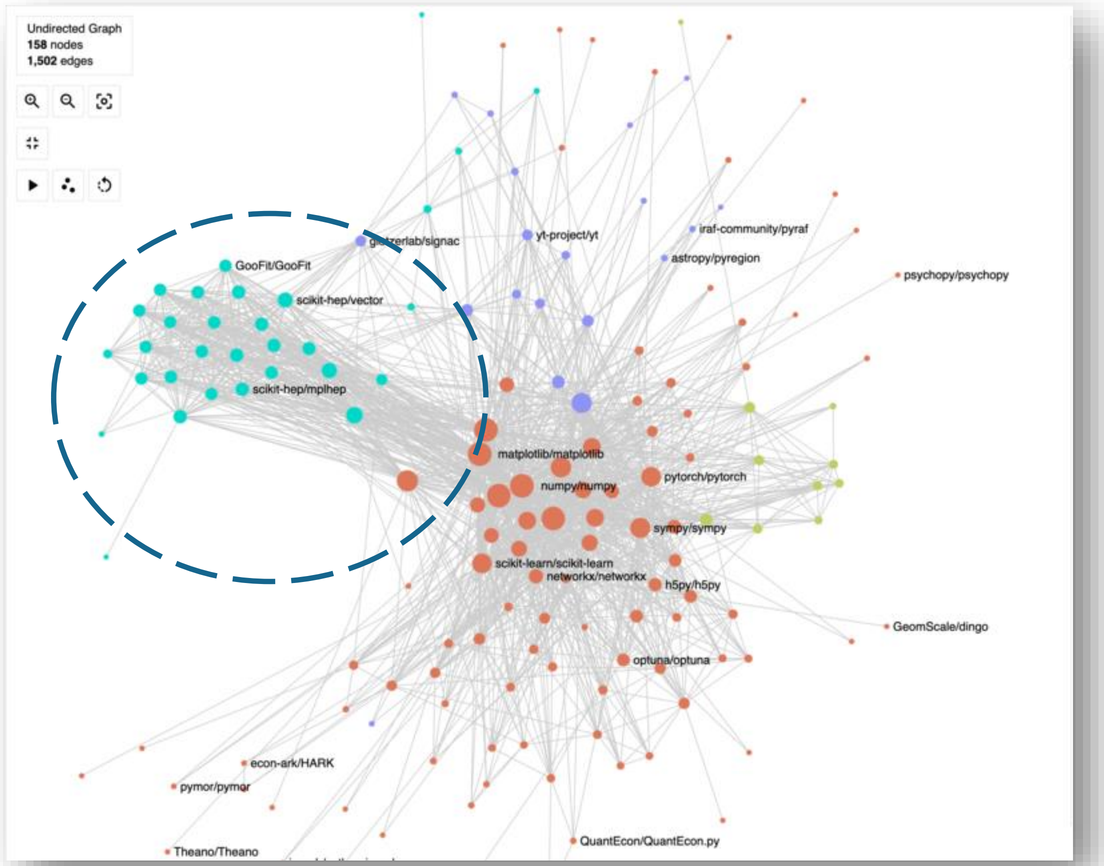
Engagement with the broader scientific Python community

Scientific Python

scientific-python.org

- **Such engagement is paramount and a win-win !**
- **Our developer guide, cookiecutter, and repo-review, were adopted into scientific-python, and are now used by other projects, including AstroPy (see next page …)**

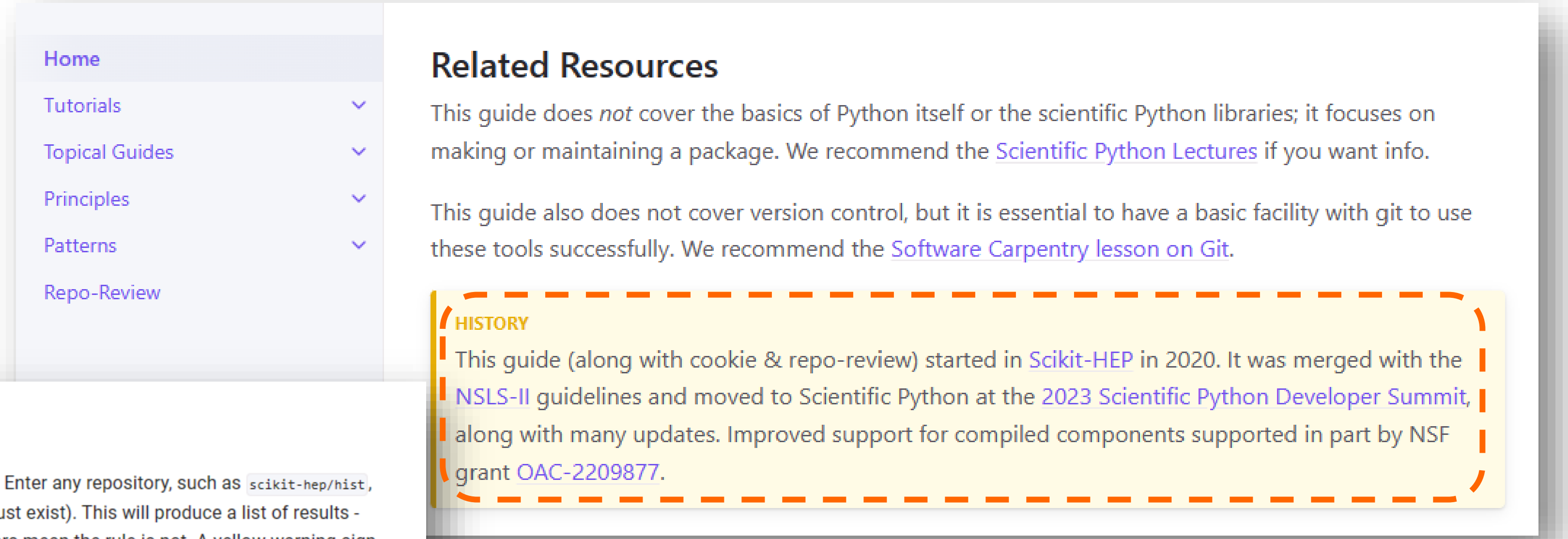
Graph shows how people who contribute to Scikit-HEP repos
are connected to people who contribute to core Scientific Python repos
- We're still a bit farther than many others …



Taken from Blog - Developer Summit 1 (scientific-python.org)

Engagement with the broader scientific Python community

- ❑ Check out https://learn.scientific-python.org/



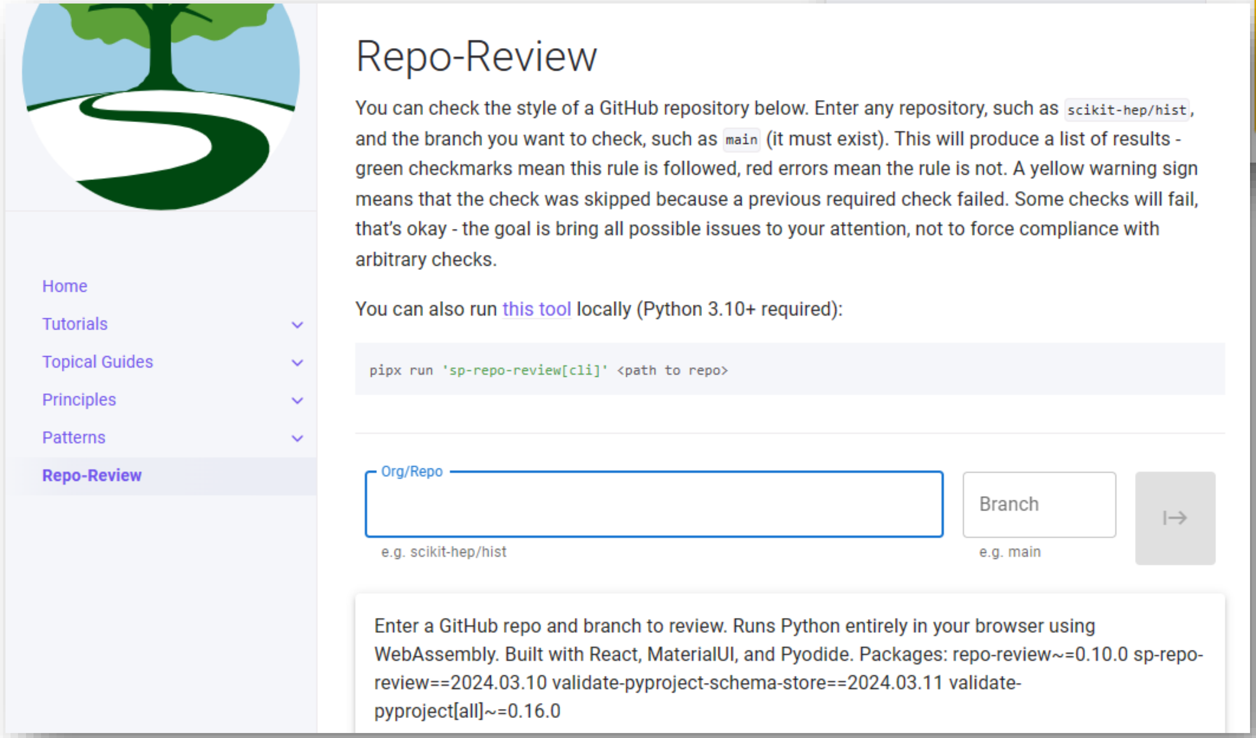
Related Resources

This guide does *not* cover the basics of Python itself or the scientific Python libraries; it focuses on making or maintaining a package. We recommend the Scientific Python Lectures if you want info.

This guide also does not cover version control, but it is essential to have a basic facility with git to use these tools successfully. We recommend the Software Carpentry lesson on Git.

HISTORY

This guide (along with cookie & repo-review) started in Scikit-HEP in 2020. It was merged with the NSLS-II guidelines and moved to Scientific Python at the 2023 Scientific Python Developer Summit, along with many updates. Improved support for compiled components supported in part by NSF grant OAC-2209877.



Repo-Review

You can check the style of a GitHub repository below. Enter any repository, such as `scikit-hep/hist`, and the branch you want to check, such as `main` (it must exist). This will produce a list of results - green checkmarks mean this rule is followed, red errors mean the rule is not. A yellow warning sign means that the check was skipped because a previous required check failed. Some checks will fail, that's okay - the goal is bring all possible issues to your attention, not to force compliance with arbitrary checks.

You can also run this tool locally (Python 3.10+ required):

```
pipx run 'sp-repo-review[cli]' <path to repo>
```

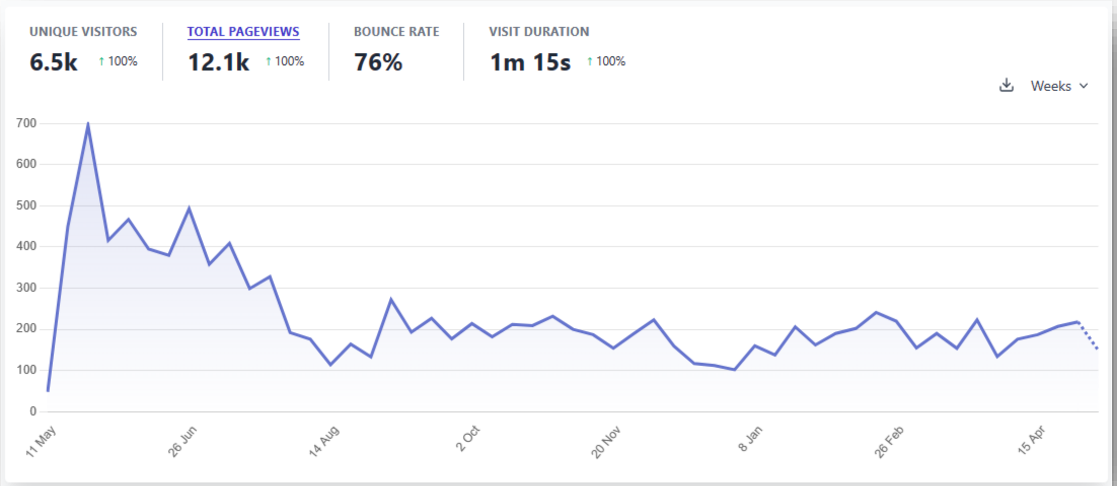
Enter a GitHub repo and branch to review. Runs Python entirely in your browser using WebAssembly. Built with React, MaterialUI, and Pyodide. Packages: repo-review~=0.10.0 sp-repo-review==2024.03.10 validate-pyproject-schema-store==2024.03.11 validate-pyproject[all]~=0.16.0

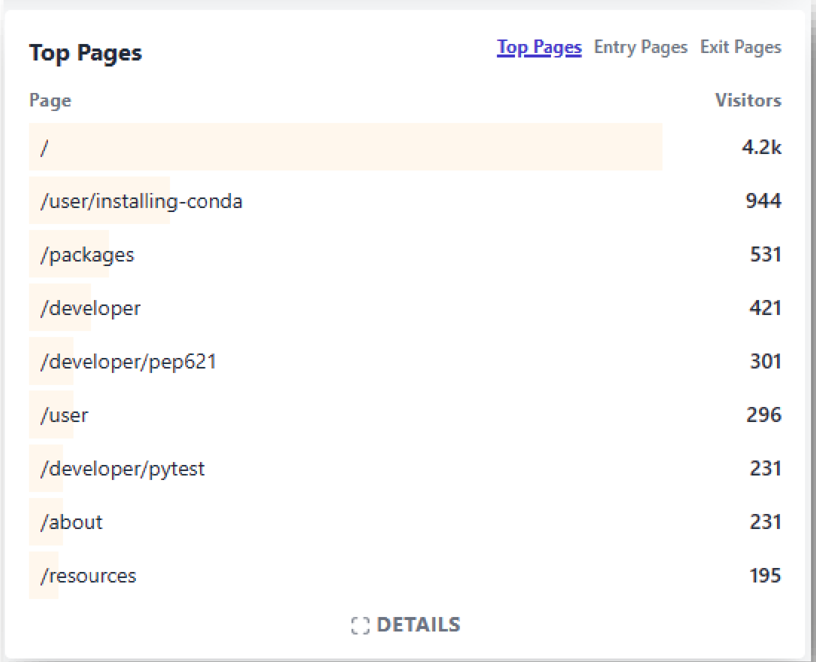
- ❑ Does your library follow the guidelines ?
- ❑ Repo-Review is your friend …

Engagement with the broader scientific Python community

- ❑ We gave/give but also get in return
- ❑ Concrete and useful tool we have access to – statistics from Plausible Analystics !

- ❑ Unique visitors of scikit-hep.org site and pages most viewed
  - Interesting to see that Top Pages are largely about info for developers ☺


UNIQUE VISITORS 6.5k ↑100% | TOTAL PAGEVIEWS 12.1k ↑100% | BOUNCE RATE 76% | VISIT DURATION 1m 15s ↑100%


Top Pages — Top Pages | Entry Pages | Exit Pages

Page	Visitors
/	4.2k
/user/installing-conda	944
/packages	531
/developer	421
/developer/pep621	301
/user	296
/developer/pytest	231
/about	231
/resources	195

DETAILS

Lessons learned and future

- Future directions and developments
- Matters of sustainability

Lessons learned and future

Early days and avoiding the "quantum fluctuation drama"

❑ A vision is not enough without a dedicated team *and* a community

❑ It is not necessary to find many people in early days to bring a project over threshold.
2-3 very motivated and skilful colleagues is enough

❑ Scikit-HEP seems to have come to the arena at the right moment and constant engagement with the community was key

Future and sustainability

❑ It is the ecosystem surrounding a piece of software, built not only by original authors but also by community members, that forms a long-term hence sustainable solution in a continuously evolving space

❑ Small package granularity – a set of "building blocks" and well-scoped generic or domain-specific libraries providing tools that address a well-defined class of problems at one level of abstraction has been a success story
- Increasing number of projects/libraries build on top of these

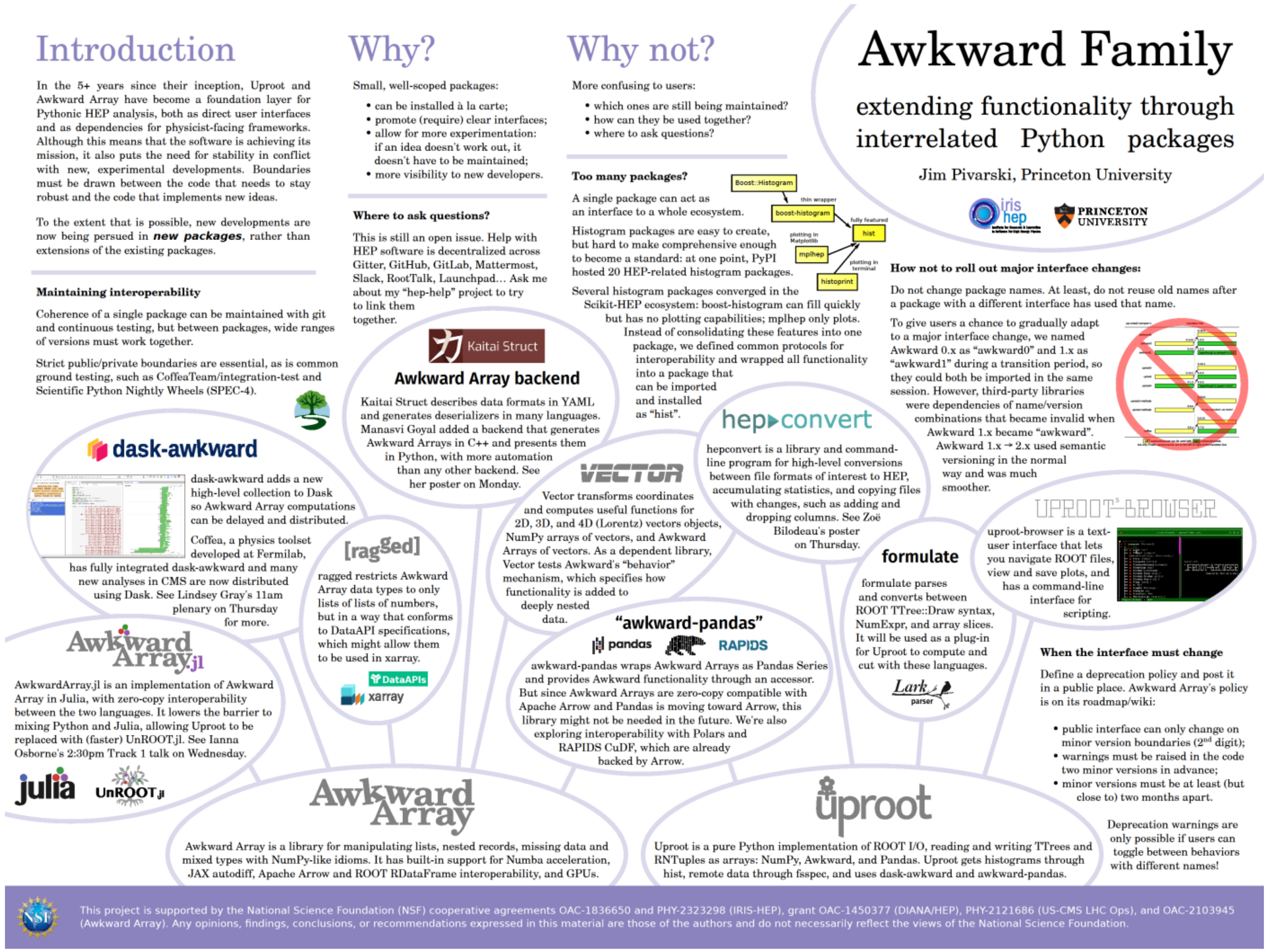
❑ Last but not least, without adequate support, career path and funding, nothing of this is sustainable !

Going forward – family of small and scoped packages

- ❑ **Packages that surround Awkward and Uproot: new developments are mostly going to be in new packages now, rather than turning the base packages into monoliths**

- ❑ **Tiny packages are great – the easiest to try something new. Did not work / was a success? No worries, try another approach (with no significant loss for the ecosystem)!**

- ❑ **Areas under active development / investigation:**
  - Autodifferentiation
  - Fitting with JAX as backend (zfit2, evermore)
  - Leverage Dask but not only
  - Even better interoperability
  - Etc.



Awkward Family

extending functionality through interrelated Python packages

Jim Pivarski, Princeton University

Introduction

In the 5+ years since their inception, Uproot and Awkward Array have become a foundation layer for Pythonic HEP analysis, both as direct user interfaces and as dependencies for physicist-facing frameworks. Although this means that the software is achieving its mission, it also puts the need for stability in conflict with new, experimental developments. Boundaries must be drawn between the code that needs to stay robust and the code that implements new ideas.

To the extent that is possible, new developments are now being persued in ***new packages***, rather than extensions of the existing packages.

Maintaining interoperability

Coherence of a single package can be maintained with git and continuous testing, but between packages, wide ranges of versions must work together.

Strict public/private boundaries are essential, as is common ground testing, such as CoffeaTeam/integration-test and Scientific Python Nightly Wheels (SPEC-4).

Why?

Small, well-scoped packages:

- can be installed à la carte;
- promote (require) clear interfaces;
- allow for more experimentation: if an idea doesn't work out, it doesn't have to be maintained;
- more visibility to new developers.

Where to ask questions?

This is still an open issue. Help with HEP software is decentralized across Gitter, GitHub, GitLab, Mattermost, Slack, RootTalk, Launchpad… Ask me about my "hep-help" project to try to link them together.

Why not?

More confusing to users:

- which ones are still being maintained?
- how can they be used together?
- where to ask questions?

Too many packages?

A single package can act as an interface to a whole ecosystem.

Histogram packages are easy to create, but hard to make comprehensive enough to become a standard: at one point, PyPI hosted 20 HEP-related histogram packages.

Several histogram packages converged in the Scikit-HEP ecosystem: boost-histogram can fill quickly but has no plotting capabilities; mplhep only plots. Instead of consolidating these features into one package, we defined common protocols for interoperability and wrapped all functionality into a package that can be imported and installed as "hist".

How not to roll out major interface changes:

Do not change package names. At least, do not reuse old names after a package with a different interface has used that name.

To give users a chance to gradually adapt to a major interface change, we named Awkward 0.x as "awkward0" and 1.x as "awkward1" during a transition period, so they could both be imported in the same session. However, third-party libraries were dependencies of name/version combinations that became invalid when Awkward 1.x became "awkward". Awkward 1.x → 2.x used semantic versioning in the normal way and was much smoother.

When the interface must change

Define a deprecation policy and post it in a public place. Awkward Array's policy is on its roadmap/wiki:

- public interface can only change on minor version boundaries (2nd digit);
- warnings must be raised in the code two minor versions in advance;
- minor versions must be at least (but close to) two months apart.

Deprecation warnings are only possible if users can toggle between behaviors with different names!

Awkward Array backend

Kaitai Struct describes data formats in YAML and generates deserializers in many languages. Manasvi Goyal added a backend that generates Awkward Arrays in C++ and presents them in Python, with more automation than any other backend. See her poster on Monday.

dask-awkward

dask-awkward adds a new high-level collection to Dask so Awkward Array computations can be delayed and distributed.

Coffea, a physics toolset developed at Fermilab, has fully integrated dask-awkward and many new analyses in CMS are now distributed using Dask. See Lindsey Gray's 11am plenary on Thursday for more.

[ragged]

ragged restricts Awkward Array data types to only lists of lists of numbers, but in a way that conforms to DataAPI specifications, which might allow them to be used in xarray.

VECTOR

Vector transforms coordinates and computes useful functions for 2D, 3D, and 4D (Lorentz) vectors objects, NumPy arrays of vectors, and Awkward Arrays of vectors. As a dependent library, Vector tests Awkward's "behavior" mechanism, which specifies how functionality is added to deeply nested data.

hep▸convert

hepconvert is a library and command-line program for high-level conversions between file formats of interest to HEP, accumulating statistics, and copying files with changes, such as adding and dropping columns. See Zoë Bilodeau's poster on Thursday.

UPROOT-BROWSER

uproot-browser is a text-user interface that lets you navigate ROOT files, view and save plots, and has a command-line interface for scripting.

formulate

formulate parses and converts between ROOT TTree::Draw syntax, NumExpr, and array slices. It will be used as a plug-in for Uproot to compute and cut with these languages.

"awkward-pandas"

awkward-pandas wraps Awkward Arrays as Pandas Series and provides Awkward functionality through an accessor. But since Awkward Arrays are zero-copy compatible with Apache Arrow and Pandas is moving toward Arrow, this library might not be needed in the future. We're also exploring interoperability with Polars and RAPIDS CuDF, which are already backed by Arrow.

AwkwardArray.jl

AwkwardArray.jl is an implementation of Awkward Array in Julia, with zero-copy interoperability between the two languages. It lowers the barrier to mixing Python and Julia, allowing Uproot to be replaced with (faster) UnROOT.jl. See Ianna Osborne's 2:30pm Track 1 talk on Wednesday.

Awkward Array

Awkward Array is a library for manipulating lists, nested records, missing data and mixed types with NumPy-like idioms. It has built-in support for Numba acceleration, JAX autodiff, Apache Arrow and ROOT RDataFrame interoperability, and GPUs.

uproot

Uproot is a pure Python implementation of ROOT I/O, reading and writing TTrees and RNTuples as arrays: NumPy, Awkward, and Pandas. Uproot gets histograms through hist, remote data through fsspec, and uses dask-awkward and awkward-pandas.

This project is supported by the National Science Foundation (NSF) cooperative agreements OAC-1836650 and PHY-2323298 (IRIS-HEP), grant OAC-1450377 (DIANA/HEP), PHY-2121686 (US-CMS LHC Ops), and OAC-2103945 (Awkward Array). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Jim Pivarski, ACAT 2024

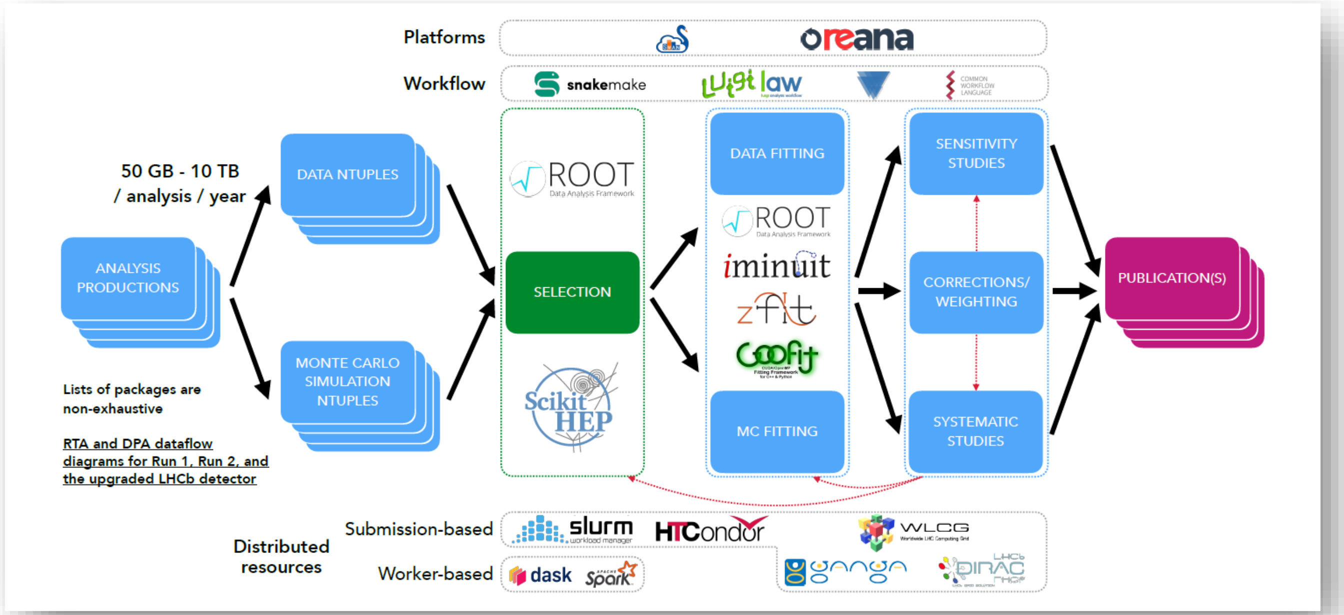
Side-comment wrapper: Heterogeneous hardware *and* software, as it were

*diverse in character or content

- ❑ **The recent decade has been very rich as far as Computing & Software is concerned**

- ❑ **Not only the hardware we use is ever more heterogeneous*, the same is true for the ecosystem of software libraries !**

- Biased overview example from LHCb for offline analysis:



LHCb-FIGURE-2024-002 (CERN CDS link)

Thank you for listening

And thank you for using/contributing to/… Scikit-HEP ☺ !