

High Energy Physics data analysis in Python

Eduardo Rodrigues
University of Liverpool

Outline

Data analysis in High Energy Physics (HEP) has evolved considerably in recent years. In particular, the role of Python has been gaining much momentum, sharing at present the show with C++ as a language of choice.

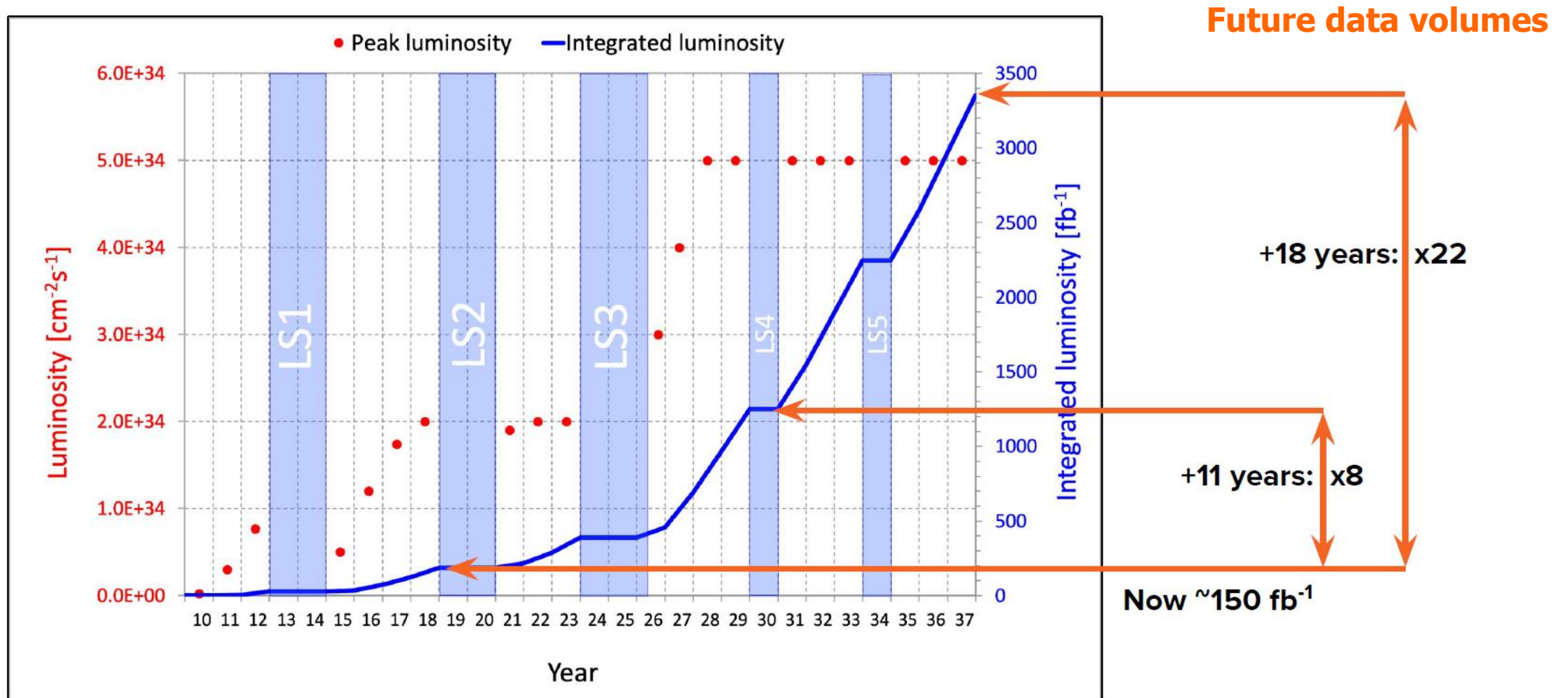
To support and enhance the usage of Python across the community, the HEP Software Foundation created a PyHEP - "Python in HEP" - working group and has been organising PyHEP workshops since 2018. Moreover, many projects and analysis packages have seen the light, which are now providing interesting, modern and alternative ways to perform analysis, in Python. In short, a global community effort is only getting stronger. I have been intimately involved in all these endeavours, and will provide an overview of the landscape. I will also detail the Scikit-HEP project I started in late 2016 with a few colleagues from various backgrounds and domains of expertise. Scikit-HEP is a community-driven and community-oriented project with the aim of providing Particle Physics at large with an ecosystem for data analysis in Python. It has developed considerably in the past year and is now part of the official software stack of experiments such as Belle II and KM3NeT.

- ❑ **Challenges in data analysis in High Energy Physics (HEP)**
- ❑ **The reign of Python**
- ❑ **Community efforts – HSF, PyHEP**
- ❑ **The Scikit-HEP project**
- ❑ **Other community software projects**
- ❑ **Final remarks**

Challenges in data analysis in High Energy Physics

- ❑ “Big Data” projects
- ❑ Computing & software challenges

A "Big Data project" – HL-LHC (High Luminosity LHC)

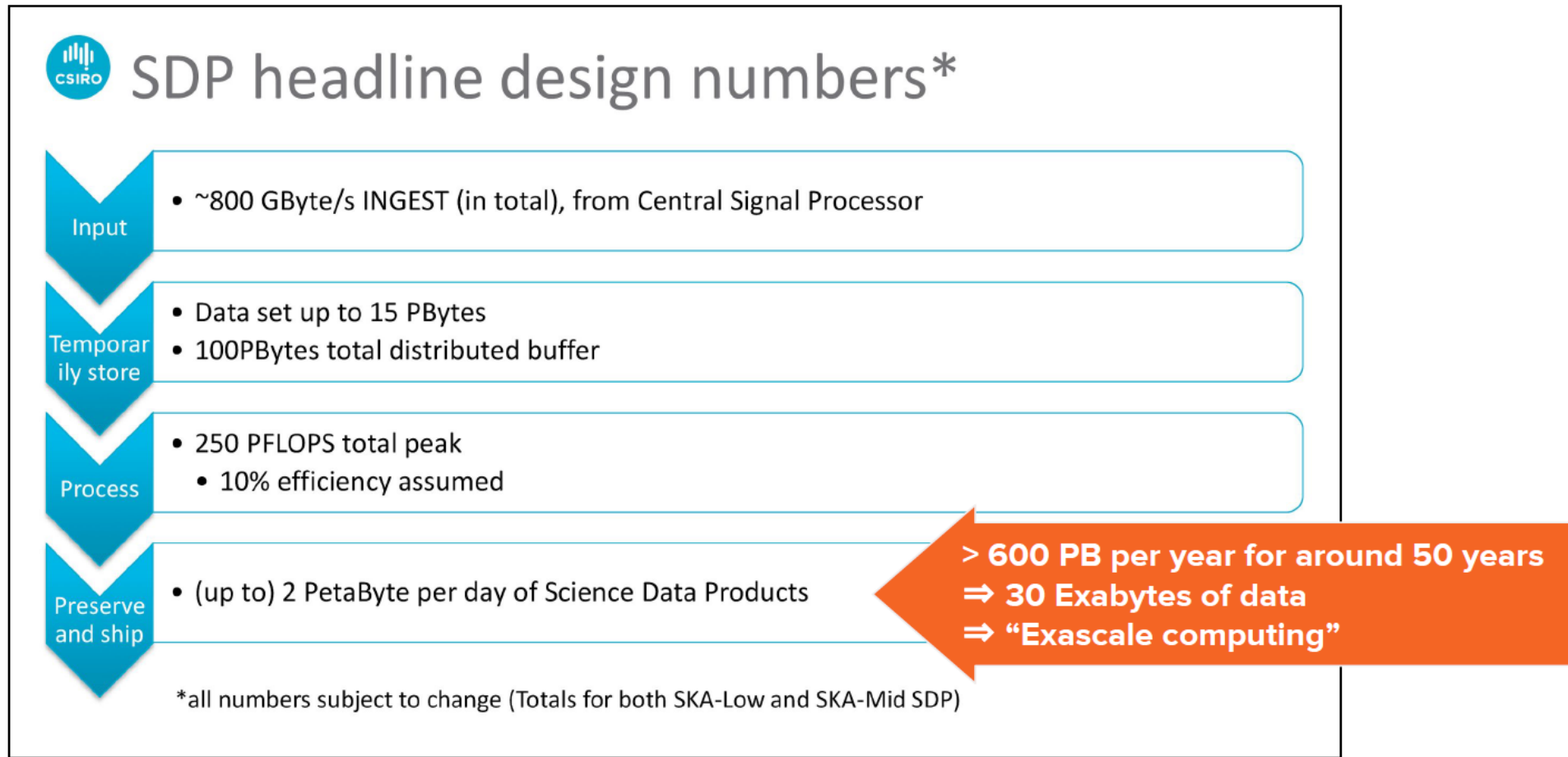


Beautification of <https://lhc-commissioning.web.cern.ch/lhc-commissioning/schedule/images/optimistic-nominal-19.png> taken from Ben Krikler

Square Kilometer Array

Minh Huynh, CHEP 2019

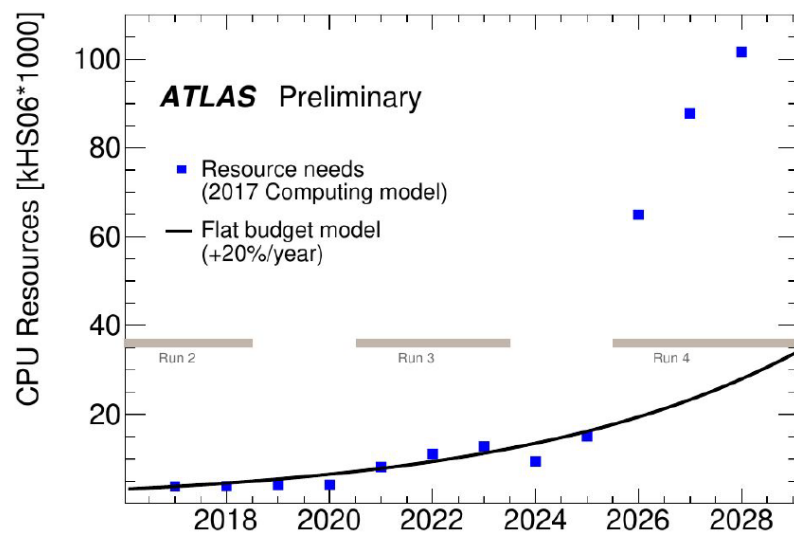
[The Square Kilometre Array Computing](#)



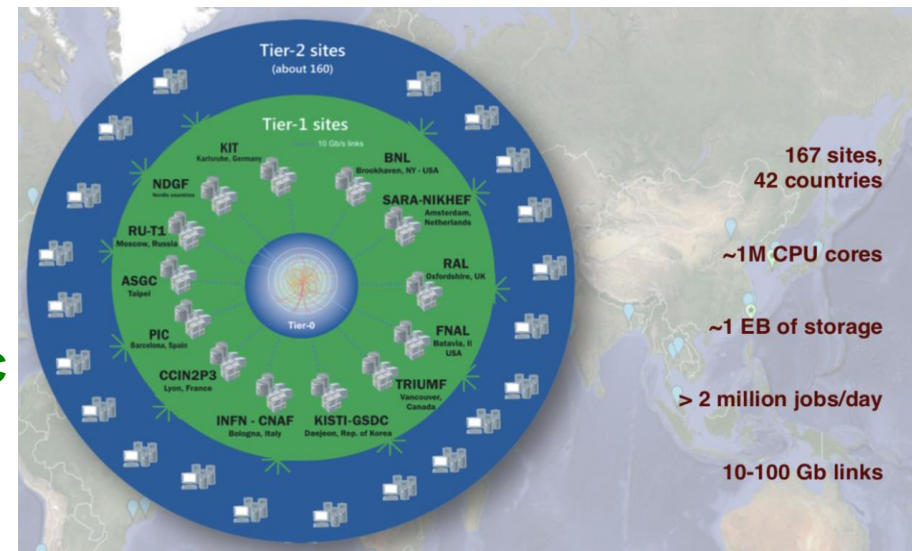
Slide taken from Ben Krikler

Computing & software challenges

- ❑ **HEP has made a vast investment in software**
 - Estimated to be around 50M lines of C++
 - It would cost \$500M to develop commercially
- ❑ **This software is a critical part of our physics production pipeline**
- ❑ **LHC experiments use about 1M CPU cores per hour per day:**
about 1 TB of data with 100 PB of data transfers per year
 - We are in the Exabyte era already
- ❑ **Future physics programmes pose significant challenges**
 - Factor of 10-100 more computing resources needed in the HL-LHC

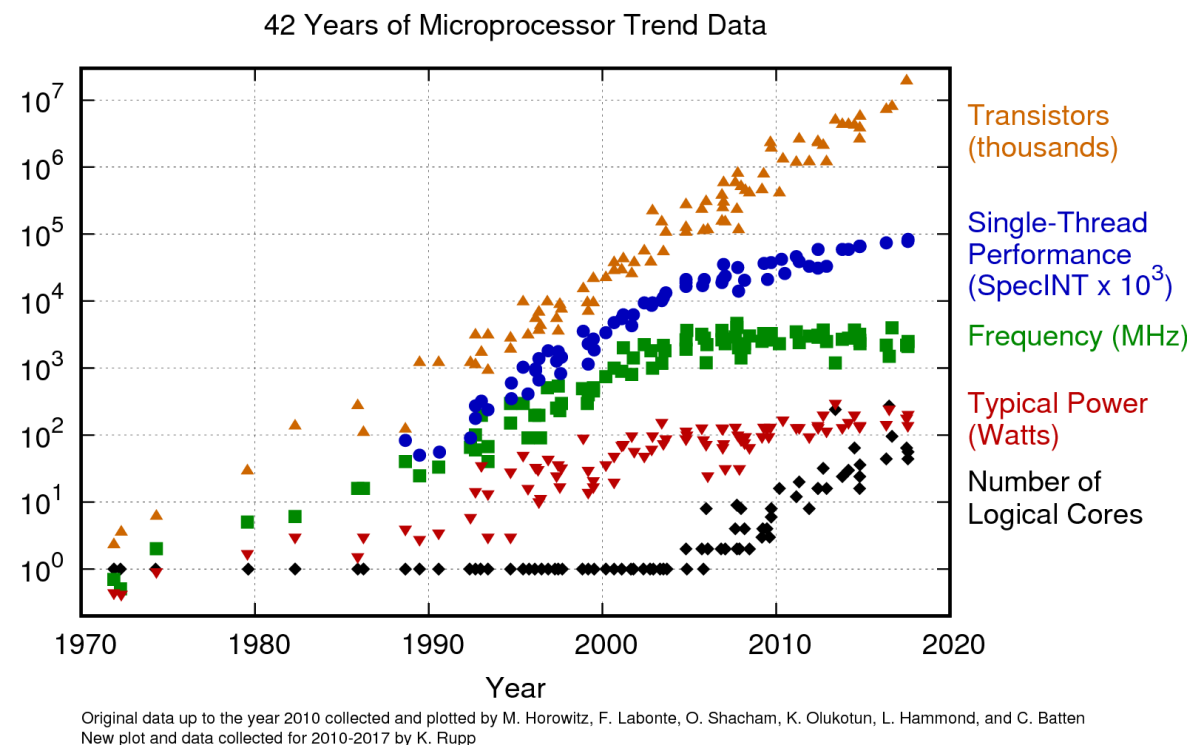


WLCG: the international collaboration to distribute and analyse LHC data



Technology evolution

- ❑ Moore's law continues but doubling time is lengthening
(is the observation that the number of transistors in a dense integrated circuit doubles about every two year)
- ❑ Clock speed scaling failed around 2006
- ❑ Memory access times are now ~100s of clock cycles
Poor data layouts are catastrophic for software performance
- ❑ HEP also needs to deal with non-CPU architectures
- GPUs, FPGAs, TPUs
- ❑ This makes it more complex to write (efficient) code



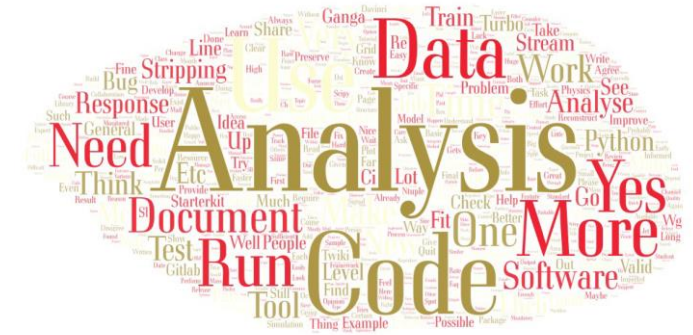
<https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/>

Tackling the challenges for (offline) data analysis – possible routes

Words from an LHCb 2018 Analysis Survey Report

- [illegible]

⇒ *This talk will focus on offline data analysis tools, hence post trigger processing*
(it will not discuss ROOT either)



The reign of Python

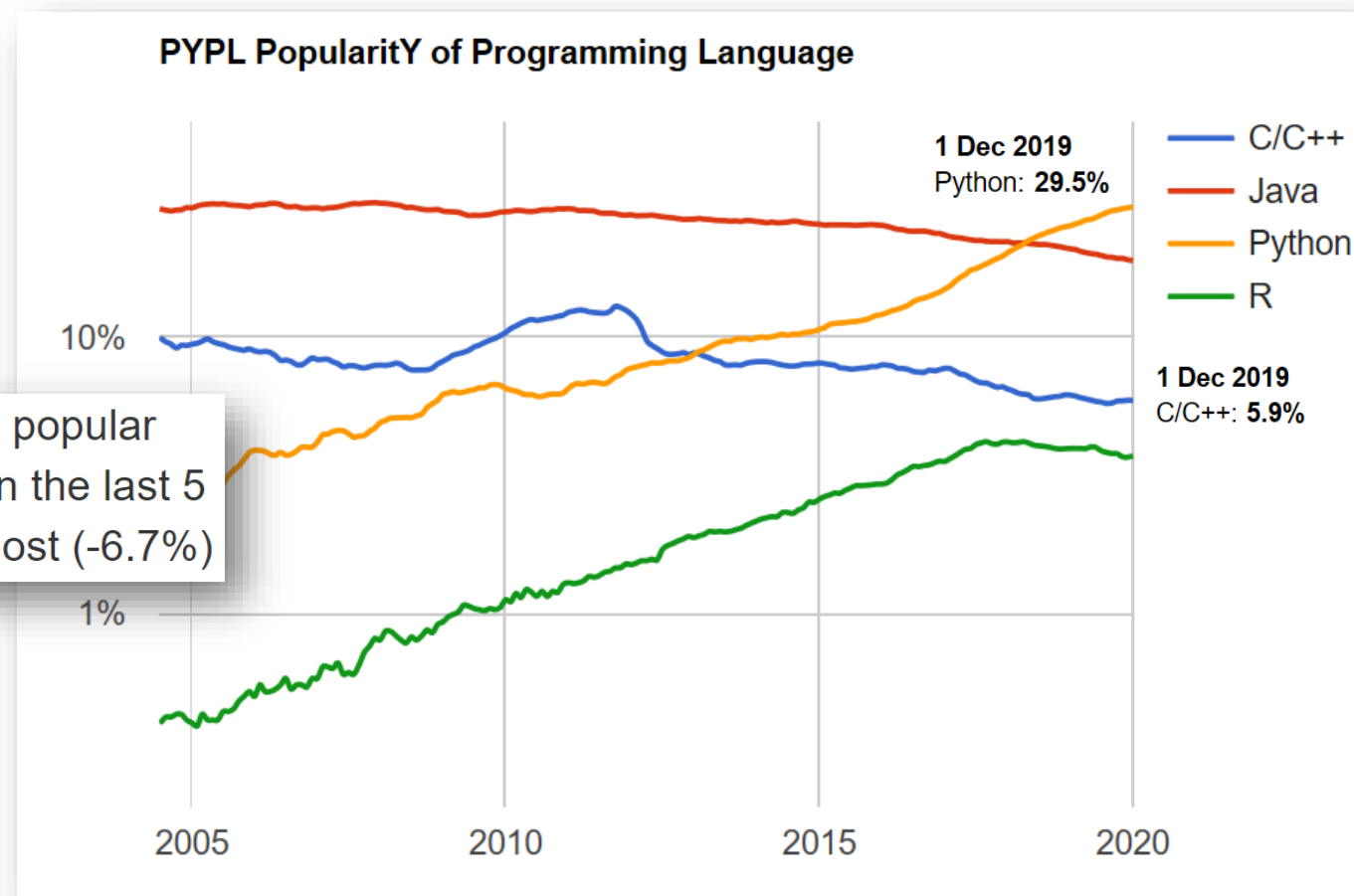
- ❑ **Popularity has never been so high**
 - in Data Science
 - in Particle Physics

PopularitY of Programming Languages (PyPL)

□ Popularity based on how often language tutorials are searched for – **Python is the big winner!**

- Data from Google Trends
- Log scale!

Worldwide, Python is the most popular language, Python grew the most in the last 5 years (19.0%) and Java lost the most (-6.7%)

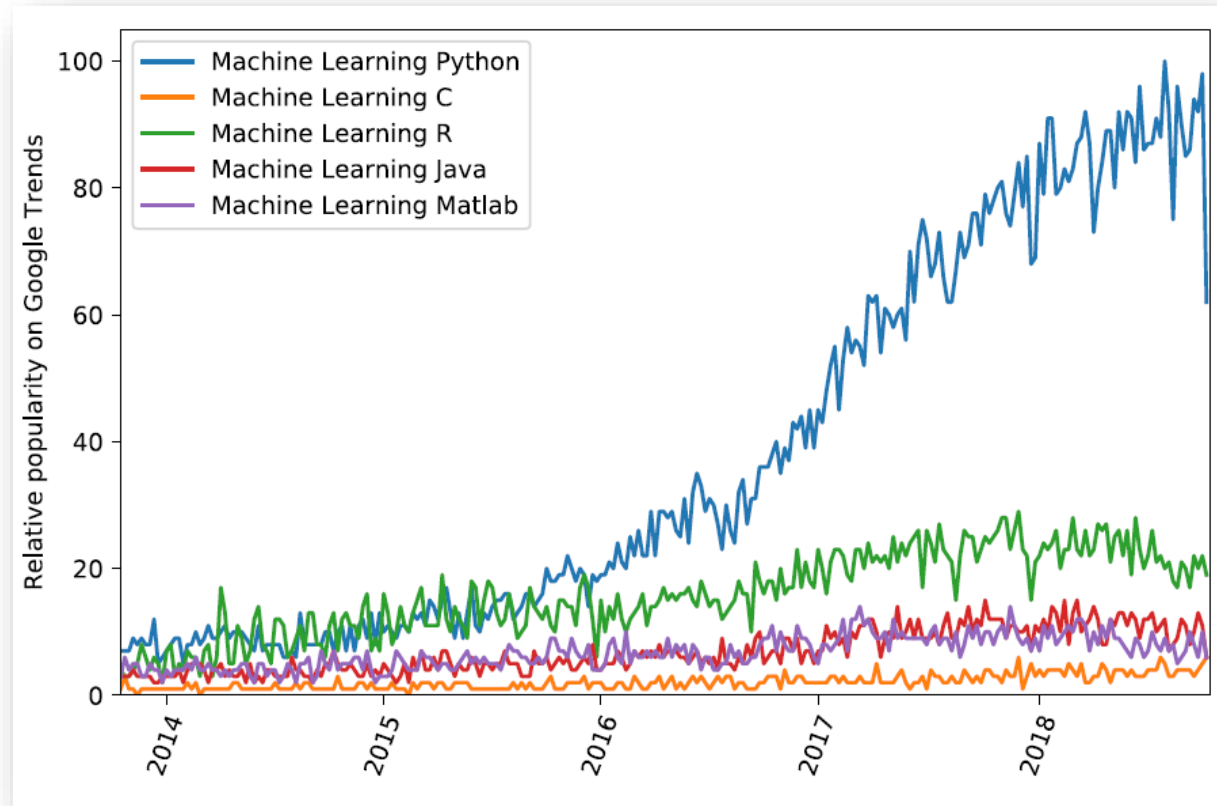


See

<http://pypl.github.io/PYPL.html>

Popularity of Programming Languages for Machine Learning

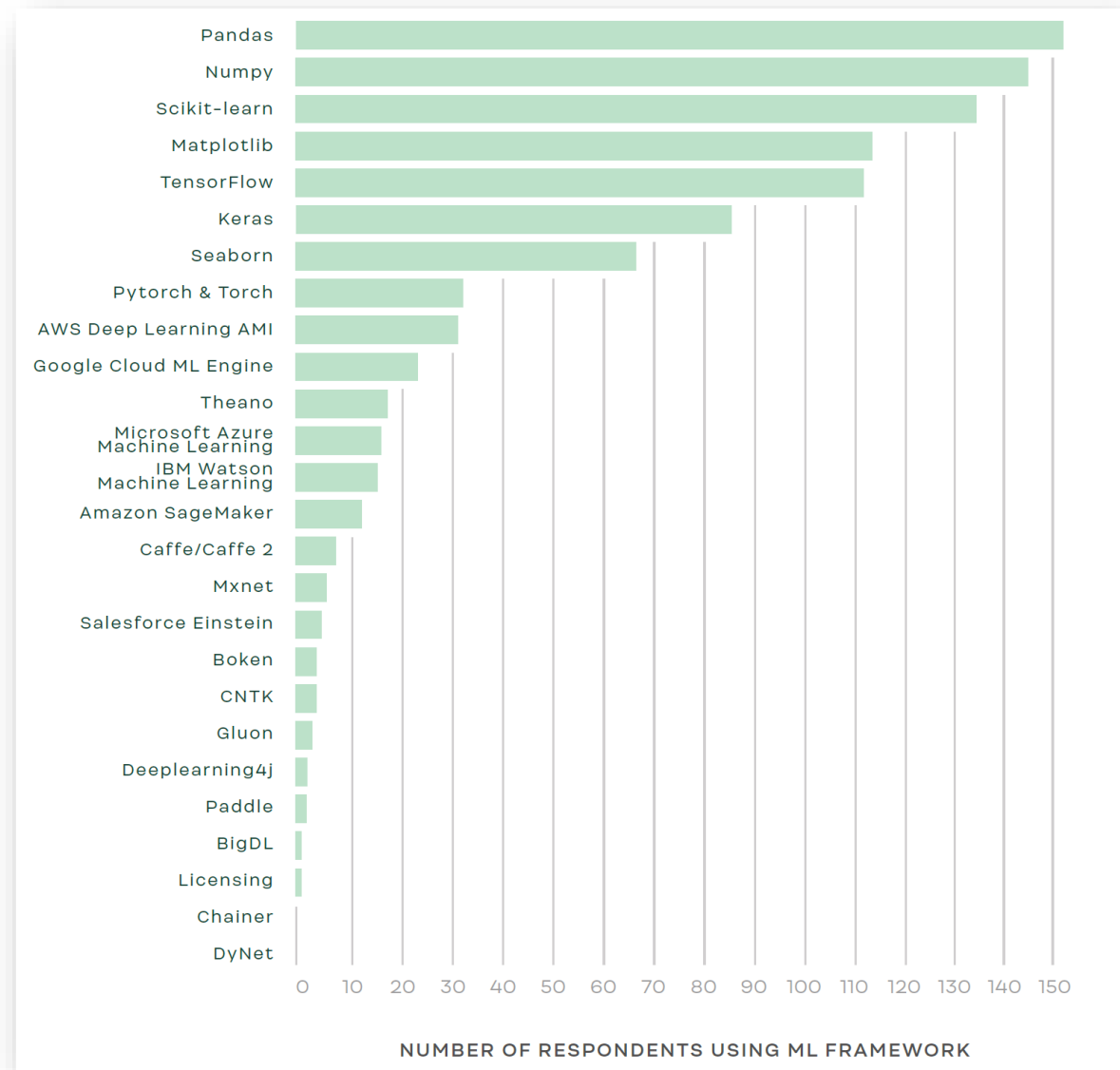
□ Popularity again from Google Trends data



Machine Learning "frameworks" usage among data scientists

- ❑ The Python scientific stack really is important
- ❑ It clearly stands out – far more ML libraries in Python than in C++
 - See also <https://github.com/josephmisiti/awesome-machine-learning>
- ❑ Side note:
 - since we talk a lot on how to help young people, **training** on (at least some of) these tools should be seen as **very relevant**

Taken from:
figure eight,
Data Scientist Report 2018
([full report](#))



Why Python for scientific research?

Adapted from Jake Vander Plas'
The unexpected effectiveness
of Python in Scientific Research

- Interoperability with other languages
 - Bindings to C++, fortran, etc
 - We can continue using existing tools (if wanted)
- Perfect for exploratory work
 - No compiling
 - Little boilerplate code
 - E.g. Jupyter notebooks (though this is no longer python-only)
- Package ecosystem
 - “Batteries included” so standard library provides many functions: argparse, globbing, regular expressions, URL requests, math
 - Package manager gives access to huge community-driven ecosystem
 - “Open-source” by default

HEP data analysis ... in C++ or Python ?

Surveys from the LHCb experiment

Python and C++ equally used among analysts

- Trend seen in our [LHCb survey](#) for the ROOT User's Workshop in 2018
- And in the [LHCb 2018 Analysis Survey Report](#)

Conclusion clearly even stronger if discussing analysis tools independent of ROOT

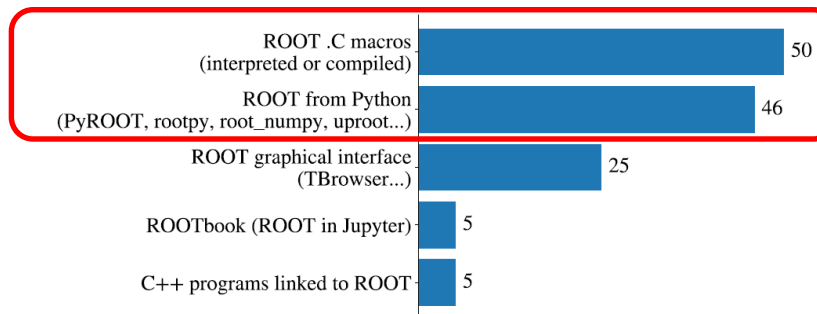
CMS study

- Most users code outside of CMSSW(*) is now Python
- Python has been eating the share of C/C++

(*) CMS software framework

Which ROOT interface are you using mostly?

multiple answers were possible



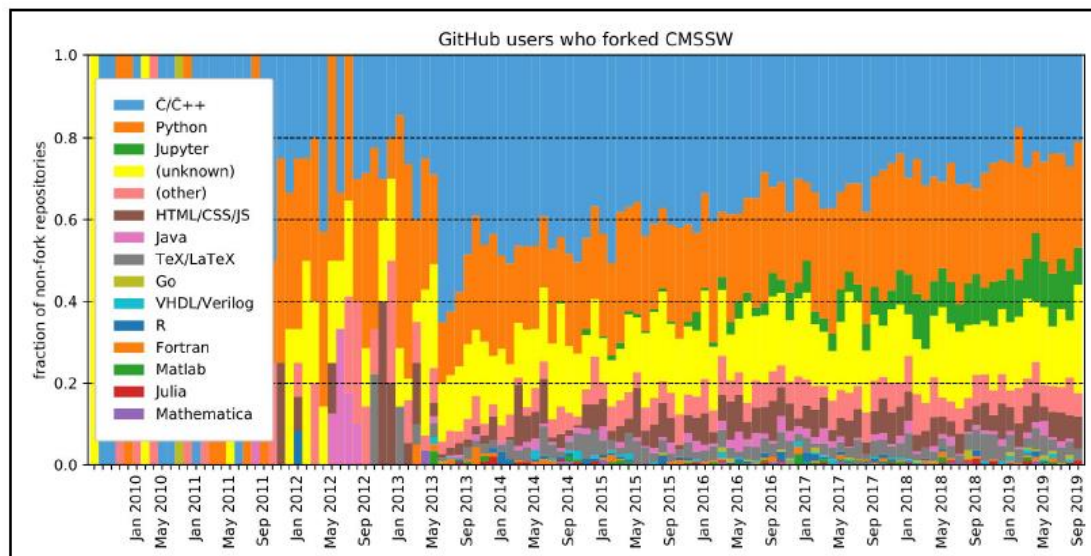
- Python scripts close second to ROOT .C macros
 - ROOT .C macros can be compiled
- Few people use ROOT in Jupyter (but those who do seem to like it a lot)
- Graphical interfaces are frequently used



Hans Dembinski | MPIK Heidelberg

5

Taken from
Hans Dembinski, User Feedback from LHCb,
ROOT Users' workshop, Sarajevo, Sep. 2018

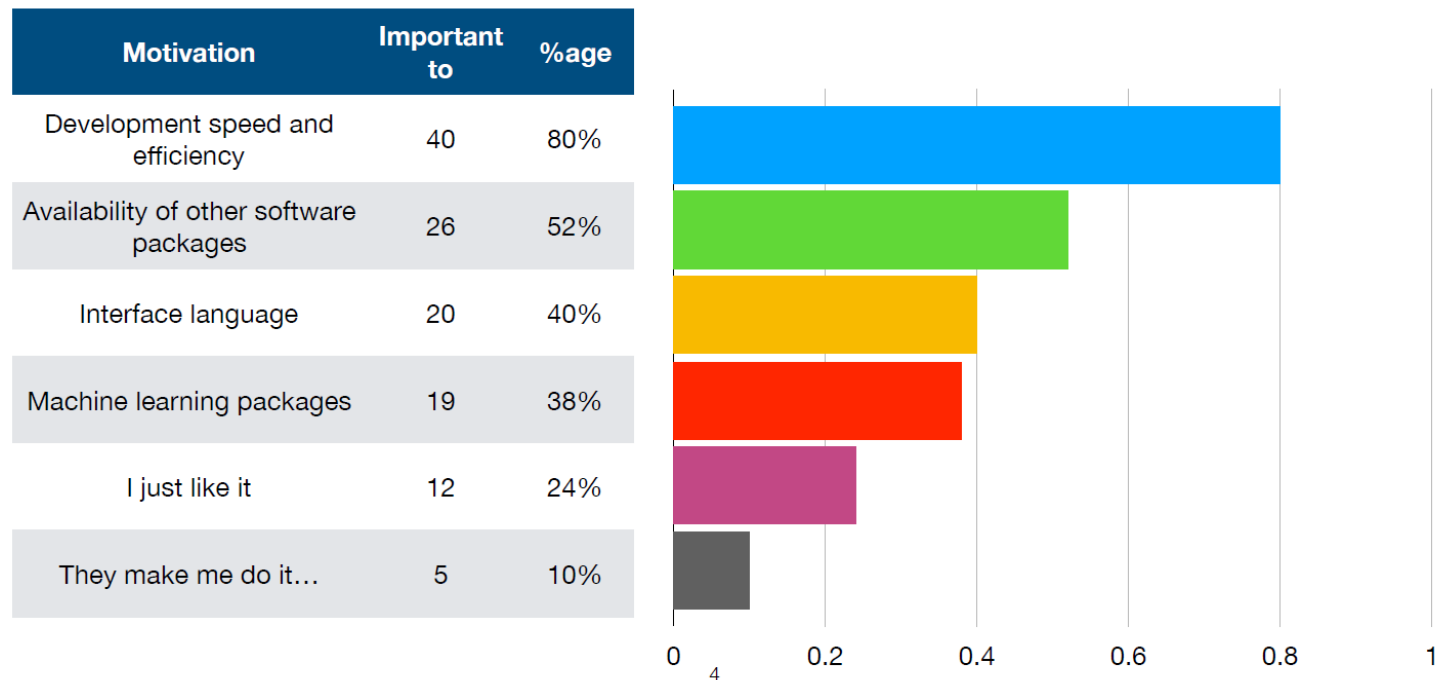


Why do particle physicists use Python ?

□ Result of PyHEP 2018 pre-workshop questionnaire

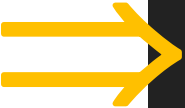
Stewart, Graeme. (2018, July). PyHEP - Questionnaire and Discussion. Zenodo. <http://doi.org/10.5281/zenodo.1419157>

Motivation to use Python



PHIALA SHANAHAN (MIT) - TRACK 6 SUMMARY

TRACK 6 SUMMARY: PHYSICS ANALYSIS

- 
- ▶ **Scalable, flexible, HEP-specific analysis ecosystems**
 - ▶ Address storage and bandwidth crunch
 - ▶ Comprehensive data and analysis preservation
 - ▶ Domain-aware machine learning
 - ▶ Visualization toolkits
 - ▶ **Diverse physics programs across HEP and NP are represented!**



Taken from
Phiala Shanahan, "Track 6 highlights - Physics Analysis",
CHEP 2019 Conference, Adelaide, Australia, 4-8 Nov. 2019

Focus of this talk – (offline) analysis software in Python

PHIALA SHANAHAN (MIT) - TRACK 6 SUMMARY

ANALYSIS TOOLS AND METHODS

- ▶ **Analysis ecosystems focused on HEP**
 - ▶ HEP-specific but flexible across experiments/analyses
 - ▶ Abstraction
➔ shareability, usability, reproducibility
- ▶ **Speed is important**
➔ interactivity, collaboration
 - ▶ Parallelization
 - ▶ Web-based interfaces



Taken from
Phiala Shanahan, "Track 6 highlights - Physics Analysis",
CHEP 2019 Conference, Adelaide, Australia, 4-8 Nov. 2019

Focus of this talk – (offline) analysis software in Python

PHIALA SHANAHAN (MIT) - TRACK 6 SUMMARY

ANALYSIS TOOLS AND METHODS

How can the analysis description be:

Concise

Shareable

Flexible

Complete

Quick



Ben Krikler (Uni of Bristol)



Collaboration



Reproducibility



Interoperability



Sustainability

Eduardo Rodrigues (Uni of Cincinnati)

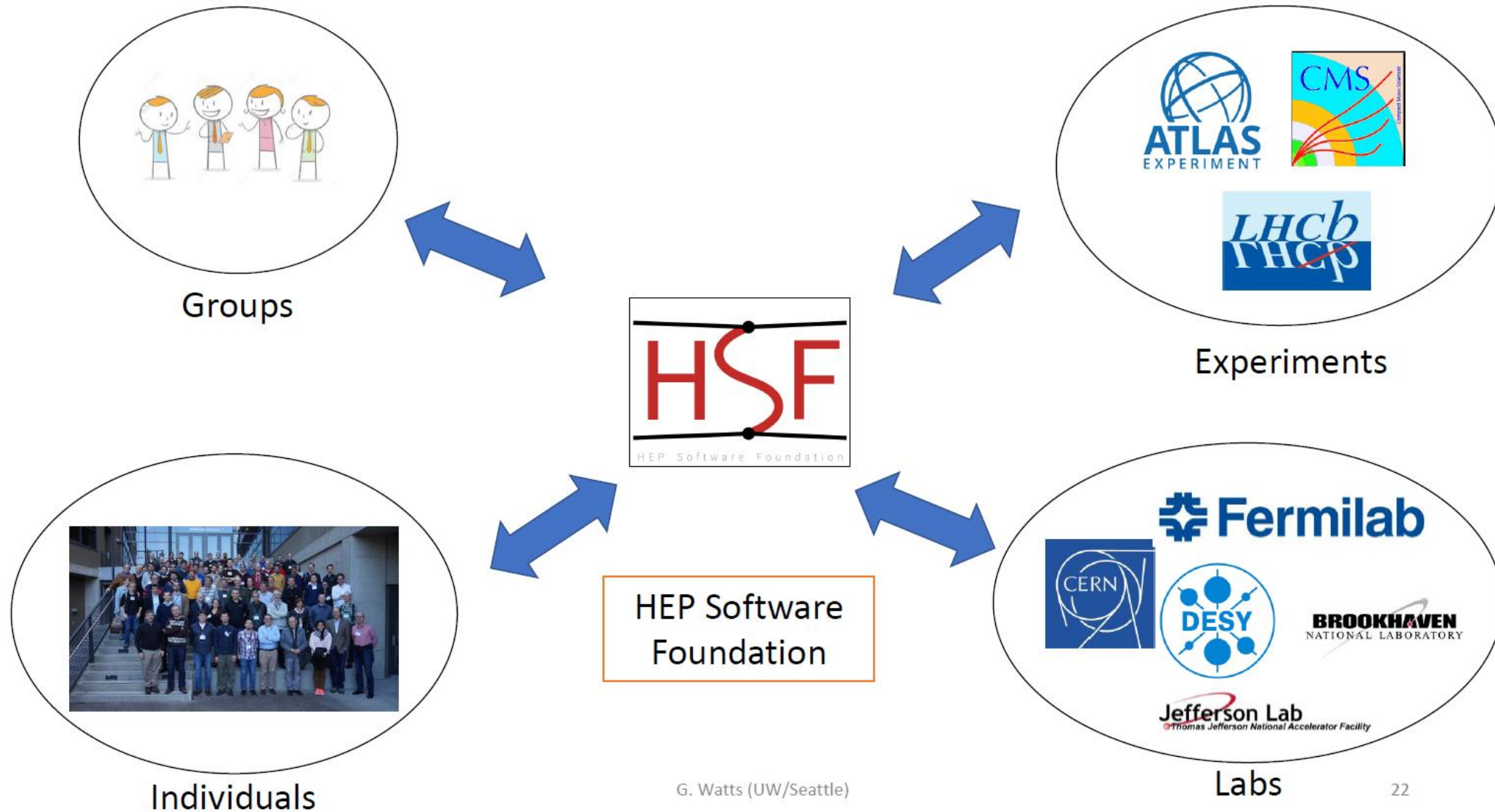
Taken from
Phiala Shanahan, "Track 6 highlights - Physics Analysis",
CHEP 2019 Conference, Adelaide, Australia, 4-8 Nov. 2019

Community efforts

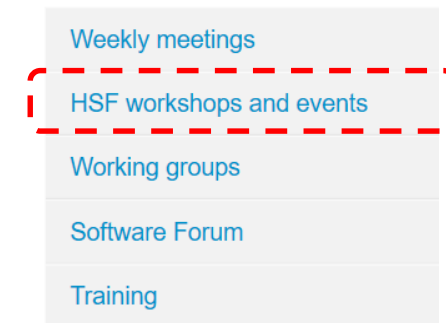
- ❑ The HEP Software Foundation (HSF)
- ❑ HSF PyHEP – “Python in HEP” Working Group
- ❑ PyHEP series of workshops
- ❑ Community projects towards a HEP Python ecosystem

The HEP Software Foundation (HSF)

- The goal of the HEP Software Foundation (HSF) is to facilitate coordination and common efforts in software and computing across HEP in general
 - ❑ Our philosophy is bottom up, a.k.a. *Do-ocracy*
 - ❑ Also work in common with like-minded organisations in other science disciplines
- Founded in 2014, explicitly to address current and future computing & software challenges in common
- Finalised in Dec. 2017 a Community White Paper (CWP)
“A Roadmap for HEP Software and Computing R&D for the 2020s”
 - ❑ Almost all major domains of HEP Software and Computing covered
 - ❑ Large support for the document from the community (> 300 authors from >120 institutions)
 - ❑ Comput Softw Big Sci (2019) 3, 7; arXiv:1712.06982
- The CWP was a major accomplishment made by the community, with HSF “coordination”
- But it was a milestone, not a final step
- HSF activities post-CWP are very diverse ...



Activities ▾	Working Groups ▾
Season of Docs	Data Analysis
Google Summer of Code	Detector Simulation
Licensing	Frameworks
Quantum Computing	Physics Generators
Reviews	PyHEP - Python in HEP
Software Forum	Reconstruction and Software Triggers
Visualisation	Software Developer Tools and Packaging
	Training



❑ The HSF also acts as an umbrella organisation for participation in the Google Summer of Code programme

HSF – PyHEP ("Python in HEP") Working Group

The PyHEP working group brings together a community of developers and users of Python in Particle Physics, with the aim of improving the sharing of knowledge and expertise. It embraces the broad community, from HEP to the Astroparticle and Intensity Frontier communities.

The group is currently coordinated by Ben Krikler (CMS, LZ), Eduardo Rodrigues (LHCb) and Jim Pivarski (CMS). All coordinators can be reached via hsf-pyhep-organisation@googlegroups.com.

Getting Involved

Everyone is welcome to join the community and participate by means of the following:

- [Gitter channel PyHEP](#) for any informal exchanges.
- [GitHub repository of resources](#), e.g., Python libraries of interest to Particle Physics.
- Twitter Handle: #PyHEP

Extra Gitter channels have been created by and for the benefit of the community:

- [PyHEP-newcomers](#) for newcomers support (very low entry threshold).
- [PyHEP-histogramming](#) for discussions around histogramming.
- [mpl-hep](#) for Matplotlib proposals related to Particle Physics.

PyHEP Series of Workshops

If you still need a motivation ;-):

☐ Python has become a first-class language in HEP

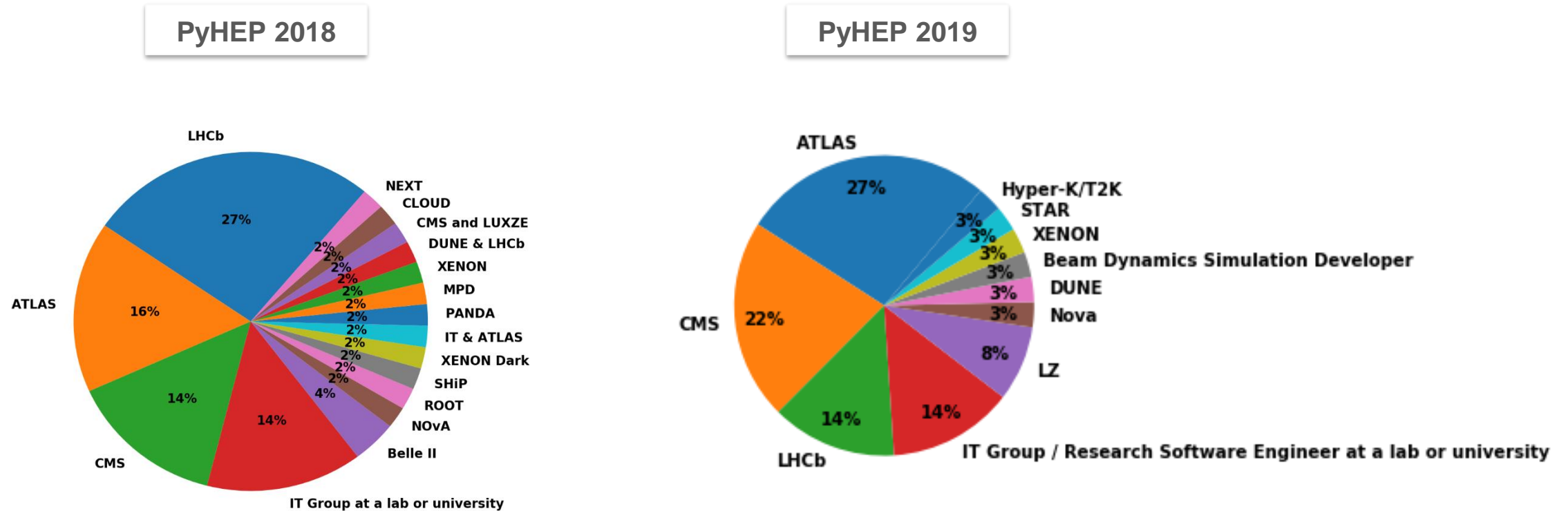
☐ And is the lingua franca for data science and machine learning

Workshop raison d'être and goals, in brief

- ❑ Step back and **review evolution of Python in the HEP community at large**
 - There are certainly HEP conferences & workshops discussing computing & software but none really devoted to this critical language in analysis
- ❑ **Python clearly identified as first-class language during the [Community White Paper](#) process**
- ❑ Need to consolidate this consensus and **plan the future** directions
 - Where we are going, want to go, need to improve
 - Tools usage, needs and developments, training and education, which Python, etc.
- ❑ **Bring together users and developers** from a wide audience
- ❑ Educative, not just informative, workshop,
with **lively discussions** in the many free and dedicated time slots we foresaw

PyHEP 2018 & 2019 workshops – community diversity

□ Very diverse set of participants – great!



(Both pie charts taken from the pre-workshop questionnaires)

PyHEP series of workshops

PyHEP 2018

Sofia, Bulgaria



PyHEP 2019

Abingdon, U.K.



PyHEP 2020

- To be held in Austin (Texas), U.S.A., in July 11-13
- Next to SciPy 2020 conference, to enhance cross-community exchange
- Stay tuned for an official announcement – see also [Indico agenda](#)

Community projects towards a HEP Python ecosystem for data analysis

❑ Citing Gordon Watts (ACAT 2019) – how can we tackle these issues?

- Increased LHC dataset sizes and CPU requirements
- Flat budgets & stable or decreasing staffing
- New software tools and communities inside and outside HEP
- High turn-over inside HEP
- Educational responsibility

Tackle them as a community !

(Note that much of this is not HEP specific ;-))

Various projects have seen the light:

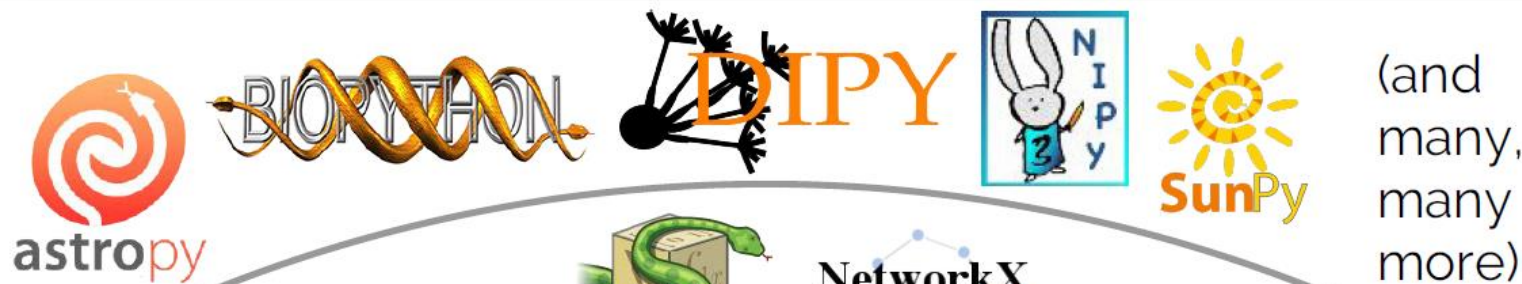
- ❑ Coffea
- ❑ FAST-HEP
- ❑ Scikit-HEP (1st one of the gang)
- ❑ zfit

The Scikit-HEP project

- ❑ Motivation for such a community project
- ❑ Whirlwind tour of packages

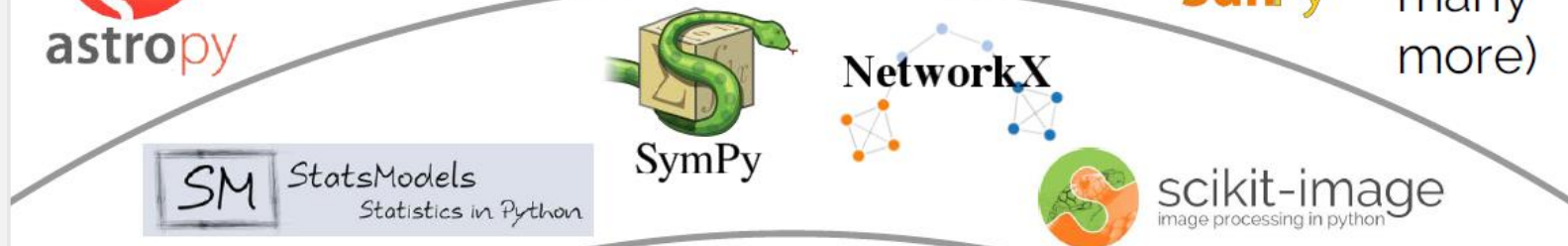
How's the Python scientific ecosystem like, outside HEP?

Domain-specific



**What about
HEP ...?**

Python's



Scientific



stack



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

Scikit-HEP project – the grand picture



- ❑ Create an ecosystem for particle physics data analysis in Python
- ❑ Initiative to improve the interoperability between HEP tools and the scientific ecosystem in Python
 - Expand the typical ~~toolkit~~ toolset for particle physicists
 - Set common APIs and definitions to ease “cross-talk”
- ❑ Promote high-standards, well documented and easily installable packages
- ❑ Initiative to build a community of developers and users
 - Community-driven and community-oriented project
- ❑ Effort to improve discoverability of (domain-specific) relevant tools



Collaboration



Reproducibility




Interoperability



Sustainability

Scikit-HEP project – 5 grand "pillars" embracing all major topics

Scikit-HEP



Home
[Getting in touch](#)
[Documentation](#)
[Who uses Scikit-HEP?](#)
[Affiliated packages](#)
[Miscellaneous resources](#)
[FAQ](#)
[Funding](#)
[Supported Python Versions](#)

This site uses [Just the Docs](#), a documentation theme for Jekyll.

Scikit-HEP on GitHub

Scikit-HEP project - welcome!

The Scikit-HEP project is a community-driven and community-oriented project with the aim of providing Particle Physics at large with an ecosystem for data analysis in Python. The project started in Autumn 2016 and is in full swing.

It is not just about providing core and common tools for the community. It is also about improving the interoperability between HEP tools and the scientific ecosystem in Python, and about improving on discoverability of utility packages and projects.

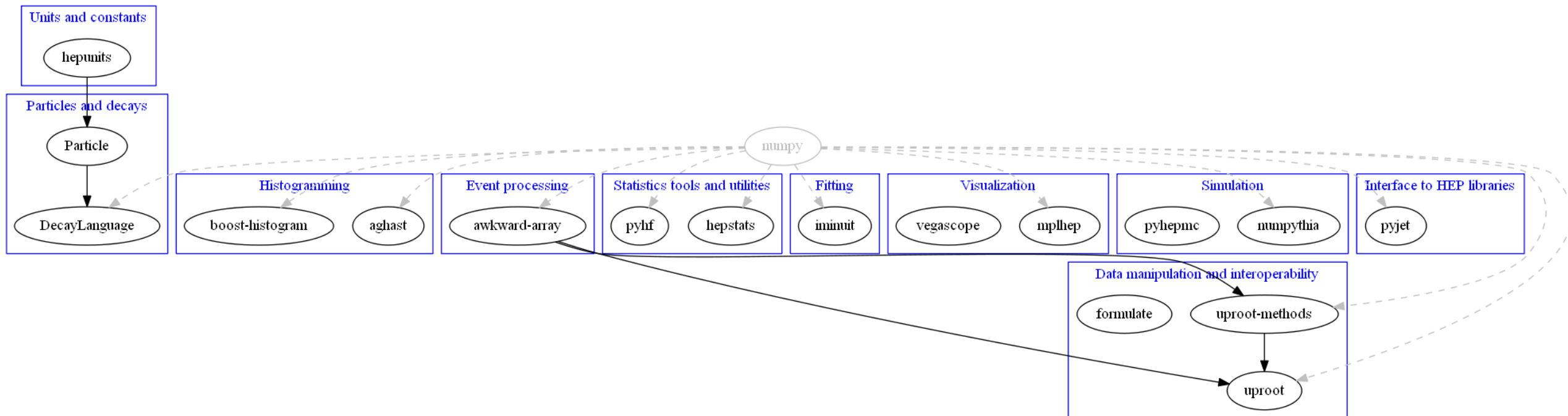
For what concerns the project grand structure, it should be seen as a *toolset* rather than a *toolkit*. The project defines a set of five *pillars*, which are seen to embrace all major topics involved in a physicist's work. These are:

- **Datasets:** data in various sources, such as ROOT, Numpy/Pandas, databases, wrapped in a common interface.
- **Aggregations:** e.g. histograms that summarize or project a dataset.
- **Modeling:** data models and fitting utilities.
- **Simulation:** wrappers for Monte Carlo engines and other generators of simulated data.
- **Visualization:** interface to graphics engines, from ROOT and Matplotlib to even beyond.

Toolset packages

Scikit-HEP project – overview of (most of the) packages

<https://scikit-hep.org/>

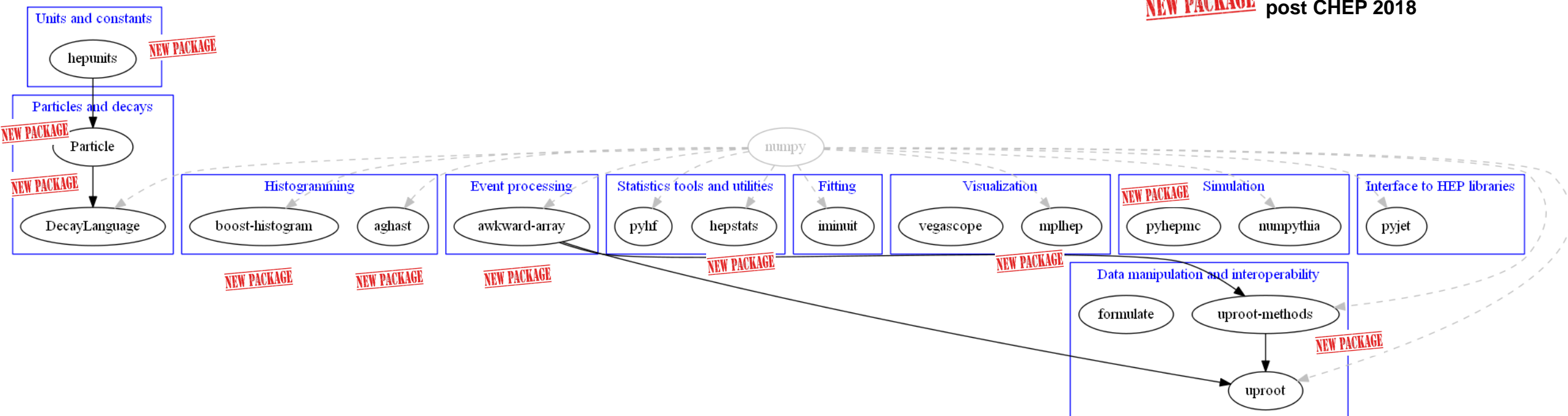


**There are other packages: test data, tutorials, org stats, etc.
(and some which tend to now be superseded, hence deprecated ...)**

Scikit-HEP project – overview of (most of the) packages

<https://scikit-hep.org/>

NEW PACKAGE = 1st release
post CHEP 2018



**There are other packages: test data, tutorials, org stats, etc.
(and some which tend to now be superseded, hence deprecated ...)**

Who uses (some of) Scikit-HEP ?

- ❑ Groups, other projects, HEP experiments
- ❑ Links are important, especially if they strengthen the overall ecosystem
- ❑ Community adoption going up \Leftrightarrow we're on the right path ;-)
- ❑ Rewarding to collaborate / work with / interact with many communities
 - Responsibility and importance of sustainability ...

Software projects



[Coffea](#) - a prototype [Analysis System](#) incorporating Scikit-HEP packages to provide a lightweight, scalable, portable, and user-friendly interface for columnar analysis of HEP data. Some of the sub-packages of Coffea may become Scikit-HEP packages as development continues.



The [zfit](#) project - it provides a model fitting library based on TensorFlow and optimised for simple and direct manipulation of probability density functions.

Experiment collaborations



[BelleII](#) - the Belle II experiment at KEK, Japan.



[CMS](#) - the Compact Muon Solenoid experiment at CERN, Switzerland.



[KM3NeT](#) - the Kilometre Cube Neutrino Telescope, an Astroparticle Physics European research infrastructure located in the Mediterranean Sea.

Phenomenology projects



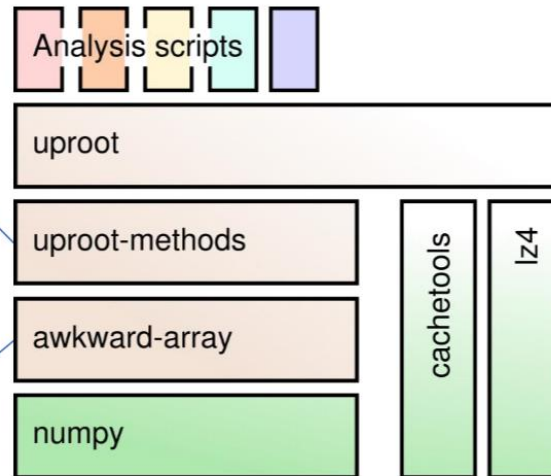
[flavio](#) - flavour physics phenomenology in the Standard Model and beyond.

Data manipulation and interoperability – uproot "suite of packages"

- ❑ (Does it still need an intro ;-)?
- ❑ Trivially and Python-ically read ROOT files
- ❑ Need only NumPy, no ROOT, using this pure I/O library!
- ❑ Design and dependencies:

uproot-methods is home to physics methods read from ROOT files.

awkward-array for array manipulation beyond numpy with Jagged and Lazy Arrays.



uproot is the layer most users interact with.

uproot

ROOT I/O
in pure Python and Numpy

uproot-methods

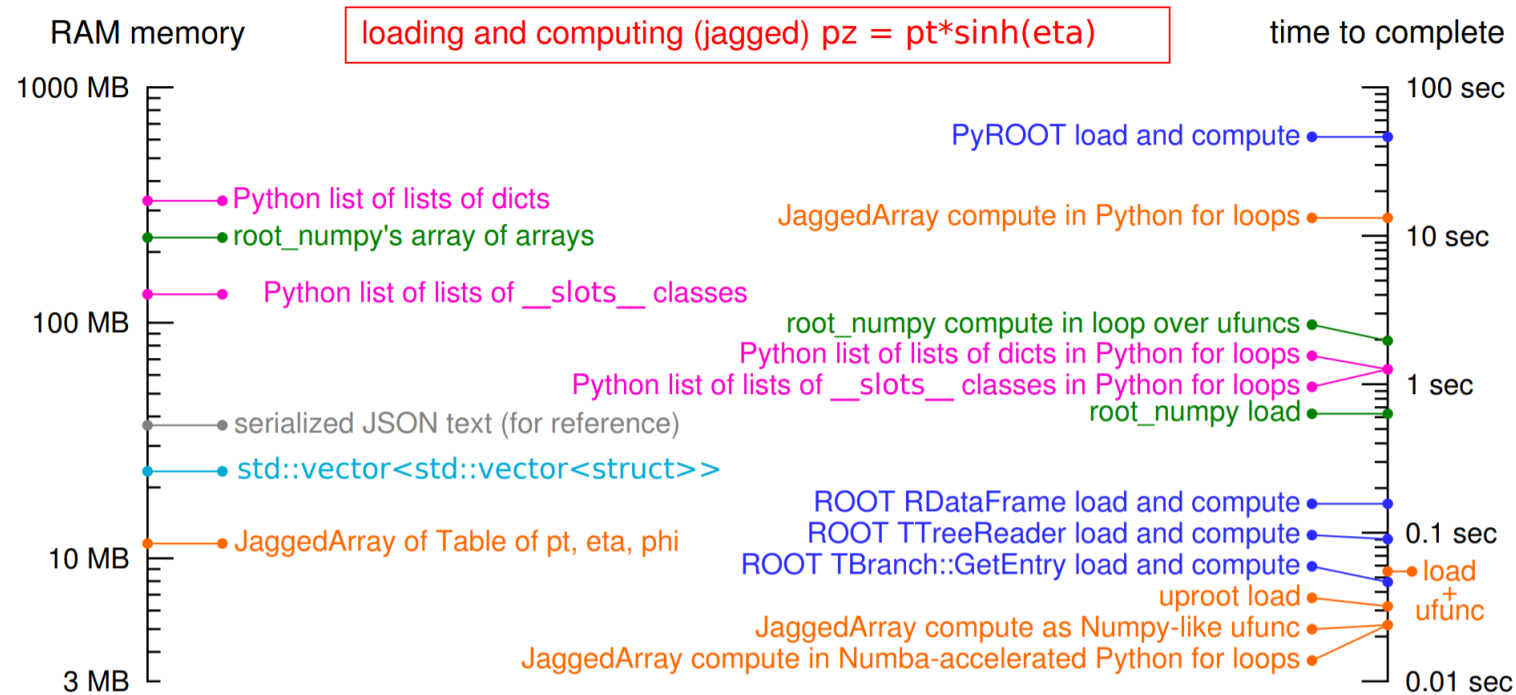
Pythonic mix-ins
for non-I/O ROOT classes

- ❑ **Write ROOT files:** newest development, limited scope = write Ttree, histograms and a couple more classes only
- See [talk at PyHEP 2019 workshop](#)

Intermezzo – wait, it's Python, it must be slow!

❑ NOPE !


“The lack of per-event processing is why reading in uproot and processing data with awkward-array can be fast, despite being written in Python.”




See <https://github.com/scikit-hep/uproot#jagged-array-performance>

Intermezzo – wait, it's Python, it must be slow!

- ❑ Much is thanks to building atop NumPy:



- A high level interface to express what you want to do
- Encourages you to work with entire arrays simultaneously



```
1 import numpy
2
3 px = numpy.random.normal(0, 100, size=1_000_000)
4 py = numpy.random.normal(0, 100, size=1_000_000)
5
```

```
6 pt = []
7 for i in range(len(px)):
8     pt.append(numpy.sqrt(px[i]**2 + py[i]**2))
```

- O(N) python instructions

```
6 pt = numpy.sqrt(px**2 + py**2)
```

- O(1) python instructions
- O(N) heavily optimised instructions

- Single (python) Instruction Multiple Data

```
selected = mass[(pt > 1000) & (2 < eta) & (eta < 5)]
```

- Simplifies code and encourages the use of vectorisation
- This is essential to efficiently use modern CPUs and GPUs

christopher.burr@cern.ch • The Python ecosystem in HEP @ LHCb UK Student Meeting

23

Event processing – awkward-array package



Manipulate arrays
of complex data structures
as easily as NumPy

- ❑ Provide a way to analyse variable-length and tree-like data in Python,
by extending NumPy's idioms from flat arrays to arrays of data structures
- ❑ Pure Python+NumPy library for manipulating complex data structures even if they
 - Contain variable-length lists (jagged/ragged)
 - Are deeply nested (record structure)
 - Have different data types in the same list (heterogeneous)
 - Are not contiguous in memory
 - Etc.
- ❑ This is all very relevant and important for HEP applications !

```
pip install awkward          # maybe with sudo or --user, or in virtualenv
pip install awkward-numba    # optional: integration with and optimization by Numba
```

- ❑ Package being re-implemented in C++, with a simpler interface and less limitations
 - Major endeavour
- ❑ Work-in-progress, see <https://github.com/scikit-hep/awkward-1.0> and dedicated talk ...

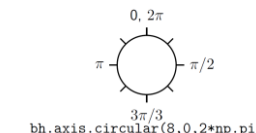
Histogramming – boost-histogram package



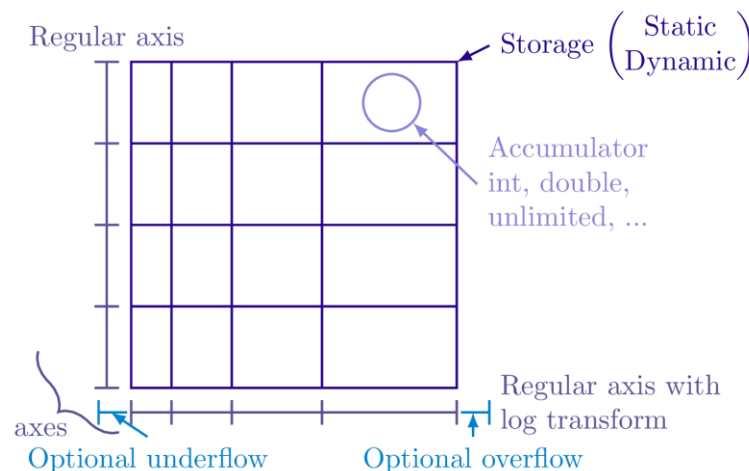
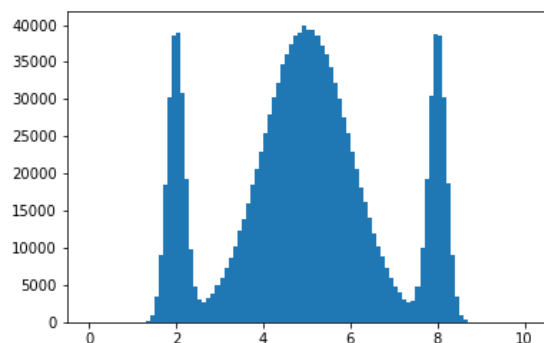
❑ (pybind11) Python bindings for the C++14 Boost.Histogram library

(multi-dimensional templated header-only)

❑ A histogram is seen as **collection of Axis objects and a storage** (several types available, e.g. circular axis)



❑ Example usage:



```
import boost_histogram as bh
h = bh.Histogram(bh.axis.Regular(100, 0, 10))
h.fill(values)
plt.bar(*h.axes.centers, h.view(), *h.axes.widths)
```

❑ See blog at <https://iscinumpy.gitlab.io/post/boost-histogram-06/> ...

Design

- Close to B.H
- Pythonic
- Numpy ready

Flexibility

- Composable
- 0-copy conversion

Speed

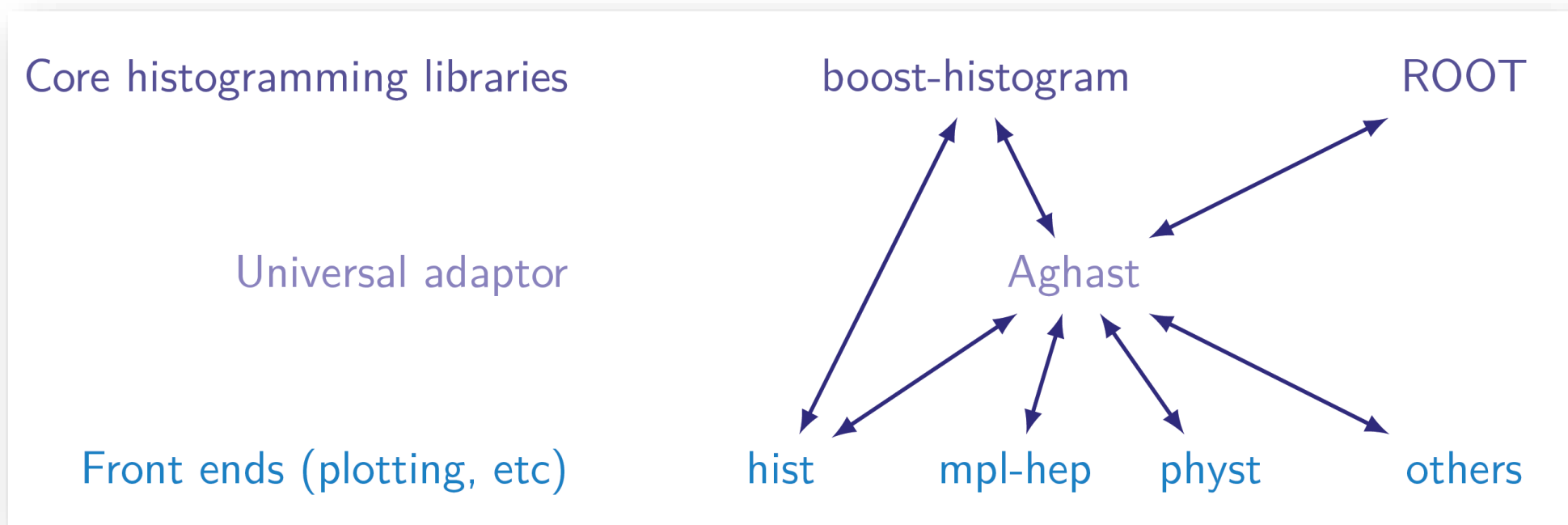
- 2-10x faster than Numpy
- Thread ready

Distribution

- Pip wheels
- Conda-forge
- C++14 only

Histogramming – looking ahead

- ❑ A fair amount of interest in the (HEP) community to develop a histogramming sub-ecosystem that meets our requirements
- ❑ Involves packages for core functionality such as filling, plotting, serialisation, and interoperability
- ❑ Interaction with popular fitting packages is also paramount



Taken from Henry Schreiner,
IRIS-HEP talks

Fitting – iminuit package

iminuit

Python interface to the
MINUIT2 C++ package

- ❑ Provides Python interface to the MINUIT2 C++ package (built on Cython)
- ❑ Minimises arbitrary functions and computes standard errors
 - Uses HESSE (inverse of Hesse matrix) or MINOS (profile likelihood method)
- ❑ Used as backend in many other HEP (e.g. zfit) and non-HEP (e.g. astroparticle) packages
- ❑ Binary wheels for all major platforms, supports for all Python versions; availability via conda-forge
- ❑ Used interactively (Jupyter-friendly displays) to do advanced fits or for learning

❑ Example usage:

```
from iminuit import Minuit

def f(x, y, z):
    return (x - 2) ** 2 + (y - 3) ** 2 + (z - 4) ** 2

m = Minuit(f)

m.migrad() # run optimiser
print(m.values) # {'x': 2, 'y': 3, 'z': 4}

m.hesse() # run covariance estimator
print(m.errors) # {'x': 1, 'y': 1, 'z': 1}
```

FCN = 1.624E-22		Ncalls = 36 (36 total)		
EDM = 1.62E-22 (Goal: 1E-05)		up = 1.0		
Valid Min.	Valid Param.	Above EDM	Reached call limit	
True	True	False	False	
Hesse failed	Has cov.	Accurate	Pos. def.	Forced
False	True	True	True	False

	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed
0	x	2.0	1.0					
1	y	3.0	1.0					
2	z	4.0	1.0					

- ❑ Pythonic interface to the Particle Data Group (PDG) particle data table and MC particle identification codes
- ❑ With many extra goodies
- ❑ Simple and natural APIs
- ❑ Main classes for queries and look-ups:
 - Particle
 - PDGID
 - Command-line queries also available
- ❑ Powerful and flexible searches as 1-liners, e.g.



```
from particle import Particle, PDGID
```

```
pid = PDGID(211)  
pid
```

```
<PDGID: 211>
```

```
pid.is_meson
```

```
True
```

```
Particle.from_pdgid(415)
```

```
 $D_2^*(2460)^+$ 
```

```
In [7]: from particle import Particle, SpinType
```

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

```
Out[7]: [<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=3637.6 ± 1.2 MeV>]
```

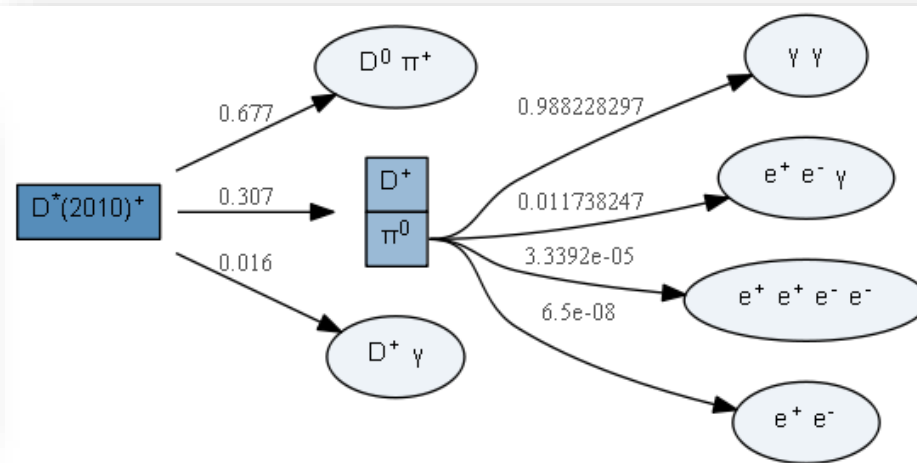
- Tools to parse decay files (aka .dec files) and programmatically manipulate them, query, display information
- Universal representation of particle decay chains
- Tools to translate decay amplitude models from AmpGen to GooFit, and manipulate them

□ Parse, extract information and visualise a decay chain:

```
from decaylanguage import DecFileParser, DecayChainViewer

dfp = DecFileParser('Dst.dec')
dfp.parse()

chain = dfp.build_decay_chains('D*+', stable_particles=['D+', 'D0'])
DecayChainViewer(chain)
```



□ Represent a complex decay chain:

```
dm1 = DecayMode(0.0124, 'K_S0 pi0', model='PHSP')
dm2 = DecayMode(0.692, 'pi+ pi-')
dm3 = DecayMode(0.98823, 'gamma gamma')
dc = DecayChain('D0', {'D0':dm1, 'K_S0':dm2, 'pi0':dm3})
```

```
dc.print_as_tree()
```

```
D0
+--> K_S0
|   +--> pi+
|   +--> pi-
+--> pi0
      +--> gamma
      +--> gamma
```


Statistics tools and utilities – hepstats package

❑ Statistical tools and utilities in Python, under development

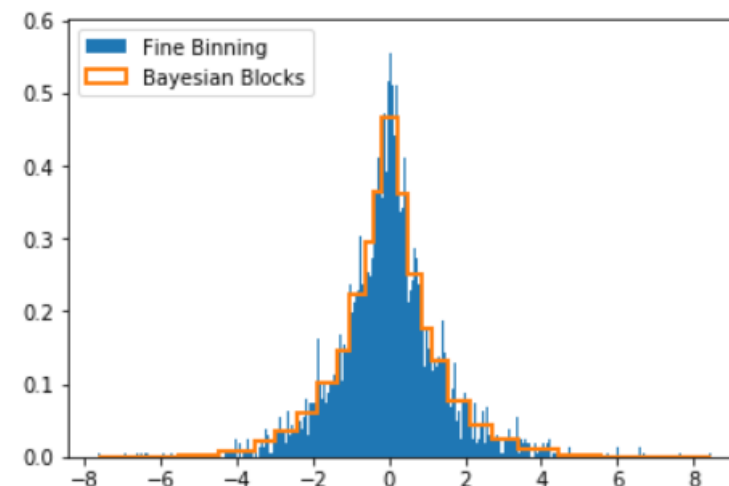
❑ Currently implements two submodules:

- Modeling with the Bayesian block algorithm – improved binning determination, robust to statistical fluctuations

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from hepstats.modeling import bayesian_blocks

>>> data = np.random.laplace(size=10000)
>>> blocks = bayesian_blocks(data)

>>> plt.hist(data, bins=1000, label='Fine Binning', density=True, alpha=0.6)
>>> plt.hist(data, bins=blocks, label='Bayesian Blocks', histtype='step', density=True, linewidth=2)
>>> plt.legend(loc=2)
```

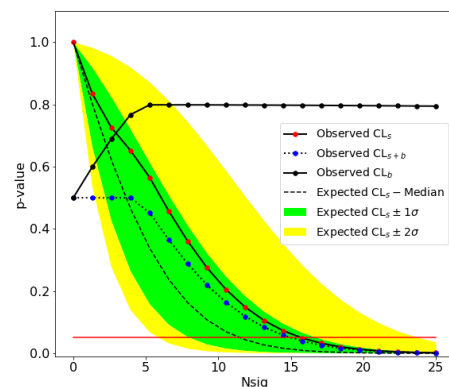


- Likelihood-based hypothesis tests, upper limit and confidence interval calculations

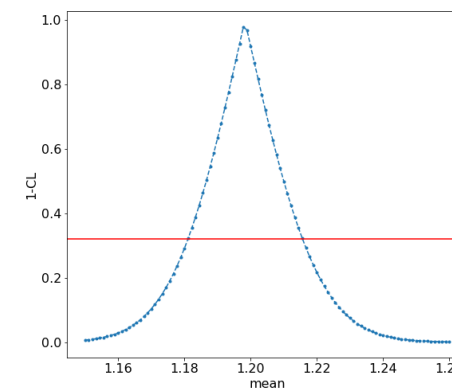
- Works with a fitting library providing models, likelihood, etc.

- Built on a common interface, used by zfit, and does not depend on a fitting backend

Upper limit on signal yield:

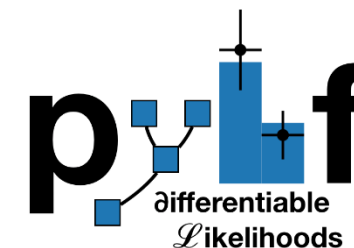


1-CL plot for the mean of a peak:



Statistics tools and utilities – pyhf package

- ❑ Pure Python implementation of ROOT's HistFactory, widely used for *binned* measurements and searches
- ❑ Benefit that can on CPUs and GPUs, transparently
- ❑ JSON specification that *fully* describes the HistFactory model
- ❑ Used for re-interpretation



Declarative binned likelihoods

$$f(\mathbf{n}, \mathbf{a} | \phi, \chi) = \underbrace{\prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}(n_{cb} | \nu_{cb}(\eta, \chi))}_{\text{Simultaneous measurement of multiple channels}} \underbrace{\prod_{\chi \in \chi} c_{\chi}(a_{\chi} | \chi)}_{\text{constraint terms for "auxiliary measurements"}}$$

Primary Measurement:

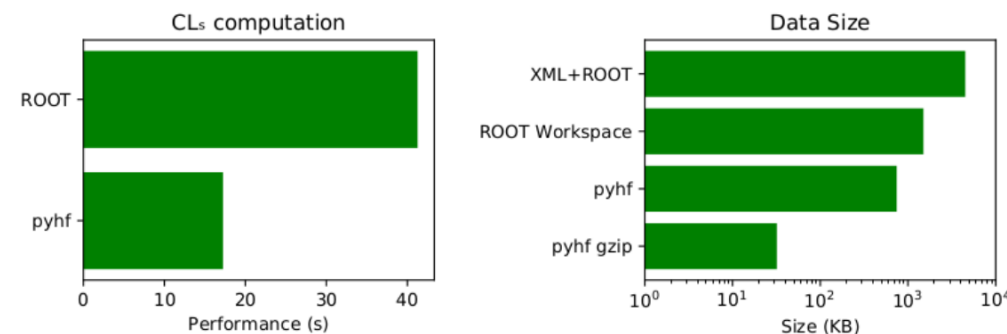
- Multiple disjoint “channels” (e.g. event observables) each with multiple bins of data
- Example parameter of interest: strength of physics signal, μ

Auxiliary Measurements:

- Nuisance parameters (e.g. in-situ measurements of background samples)
- Systematic uncertainties (e.g. normalization, shape, luminosity)

Performance

Efficient use of tensor computation makes pyhf fast



Competitive with traditional C++ implementation — often faster

(Taken from M. Feickert's CHEP 2019 poster)

Full analysis likelihoods published on HEPData

- ❑ Test theory against LHC data
- ❑ All that's needed captured in a convenient format
- ❑ **“Full likelihoods in all their glory” on HEPData**
 - “While ATLAS had published likelihood scans ... those did not expose the full complexity of the measurements”



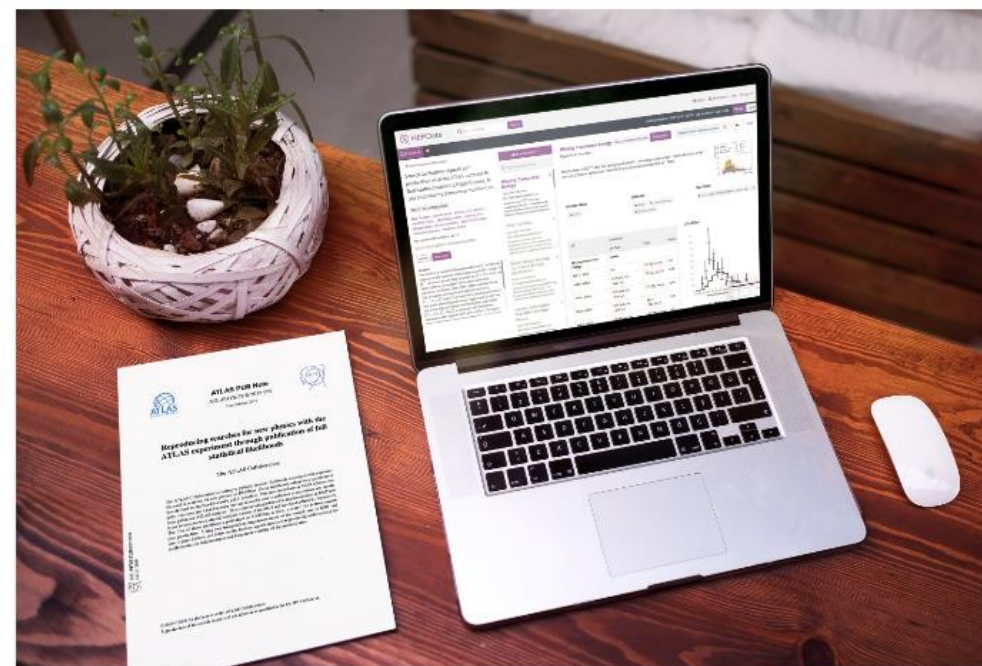
Work done with

- ❑ RooStats
- ❑ pyhf

New open release allows theorists to explore LHC data in a new way

The ATLAS collaboration releases full analysis likelihoods, a first for an LHC experiment

9 JANUARY, 2020 | By Katarina Anthony



Explore ATLAS open likelihoods on the HEPData platform (Image: CERN)

What if you could test a new theory against LHC data? Better yet, what if the expert knowledge needed to do this was captured in a convenient format? This tall order is now on its way from the ATLAS collaboration, with the first open release of full analysis likelihoods from an LHC experiment.

Taken from <https://home.cern/news/news/knowledge-sharing/new-open-release-allows-theorists-explore-lhc-data-new-way>

Simulation – pyhepmc packages

- ❑ HepMC3: a new rewrite of the C++ HepMC event record for MC generators

- ❑ pyhepmc: Python wrapper for the HepMC3 C++ library

- ❑ Bindings built on pybind11

- ❑ Supports all Python versions

- ❑ On PyPI as source distribution

- ❑ Development done with exchanges with the HepMC3 team

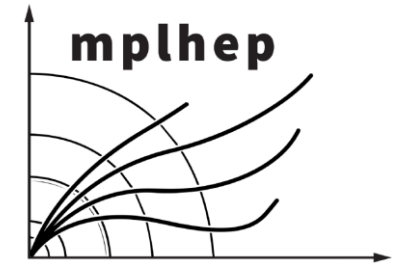
 - Idea is to provide pyhepmc as the official bindings, included in the HepMC3 distribution

pyhepmc

Python wrapper for the
HepMC3 C++ library

mplhep package – helper visualisation tool for HEP atop Matplotlib

- ❑ Matplotlib is a key tool for visualisation in the data science domain
- ❑ But it not provide all that HEP wants
 - Requires a lot of tinkering
- ❑ mplhep idea:
 - Keep matplotlib as a versatile and well-tested backend
 - Provide a new domain-specific API

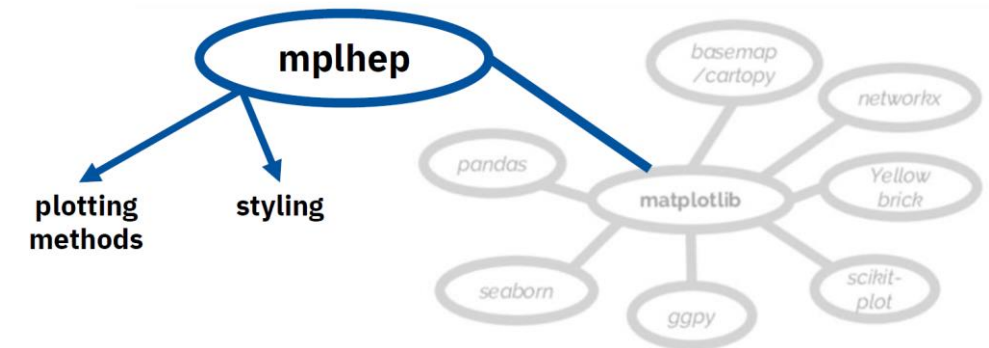
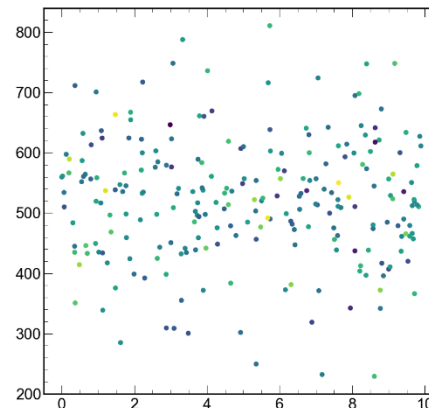


Minimal Example

```
import numpy as np
import matplotlib.pyplot as plt
+ import mplhep as hep

x = np.random.uniform(0, 10, 240)
y = np.random.normal(512, 112, 240)
z = np.random.normal(0.5, 0.1, 240)

+ plt.style.use(hep.style.ROOT)
f, ax = plt.subplots()
ax.scatter(x,y, c=z);
```



Visualisation – VegaScope package



❑ Minimal viewer of Vega & Vega-Lite graphics on the browser from local or remote Python processes

- Vega = declarative “visualisation grammar”, see [GitHub org](#)
- The Python process generating the graphics does not need to be on the same machine as the web browser viewing them

❑ 0 dependencies - can be installed as single file, used as a Python library or as a shell command, watching a file or stdin

❑ Example:

```
import vegascope
canvas = vegascope.LocalCanvas()
canvas("https://vega.github.io/vega/examples/stacked-bar-chart.vg.json")
```

❑ Altair can use VegaScope as a renderer:



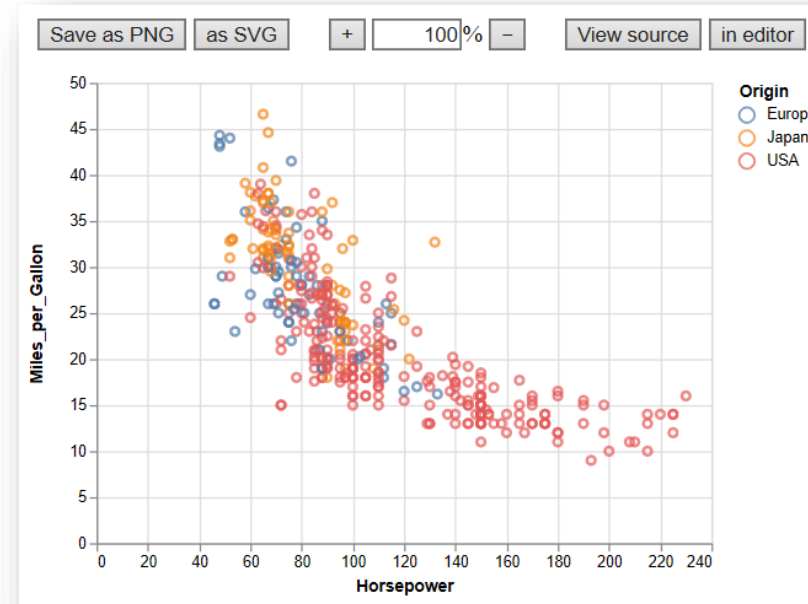
```
import vegascope
canvas = vegascope.LocalCanvas()
canvas("https://vega.github.io/vega/examples/stacked-bar-chart.vg.json")
```

```
import altair as alt
alt.renderers.enable('vegascope')

RendererRegistry.enable('vegascope')
```

```
from vega_datasets import data
cars = data.cars()
alt.Chart(cars).mark_point().encode(x='Horsepower',
                                     y='Miles_per_Gallon',
                                     color='Origin')
.interactive()
```

Rendered at <http://localhost:56574>



Simulation & jet clustering – numpythia and pyjet packages

- Generate events with Pythia and pipe them into NumPy arrays

```
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]
```

numpythia

Interface between
PYTHIA and NumPy

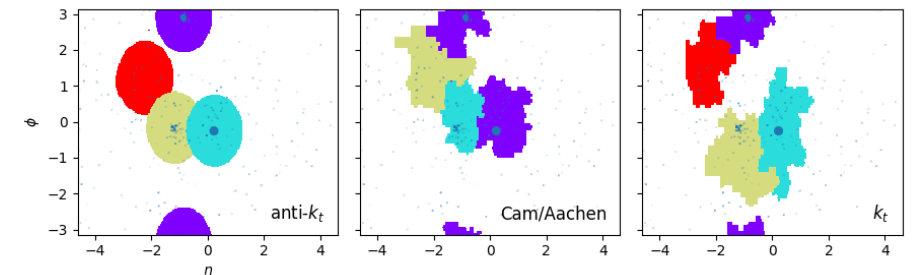
pyjet

Interface between
FastJet and NumPy

- Possible to feed those events into FastJet using pyjet

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

vectors = get_event()
sequence = cluster(vectors, R=1.0, p=-1)
jets = sequence.inclusive_jets() # list of PseudoJets
```



Units and constants in the HEP system of units – hepunits package

❑ Units and constants in the HEP system of units

- Not the same as the SI system of units

❑ Trivial package, but handy

❑ Typical usage:

```
from hepunits.constants import c_light
from hepunits.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
```

```
0.44968868700000003
```

```
print(ctau_Bs / micrometer) # result in micrometers
```

```
449.688687
```

❑ More “advanced”:

```
from hepunits import c_light, GeV, meter, ps
from math import sqrt

def ToF(m, p, l):
    """Time-of-Flight = particle path length l / (c * beta)"""
    one_over_beta = sqrt(1 + m*m/(p*p))
    return (l * one_over_beta / c_light)
```

```
from particle.particle.literals import pi_plus, K_plus # particle name literals
```

```
delta = ( ToF(K_plus.mass, 10*GeV, 10*meter) - ToF(pi_plus.mass, 10*GeV, 10*meter) ) / ps
print("At 10 GeV, Delta-TOF(K-pi) over 10 meters = {:.5} ps".format(delta))
```

```
At 10 GeV, Delta-TOF(K-pi) over 10 meters = 37.374 ps
```

Quantity	Name	Unit
Length	millimeter	mm
Time	nanosecond	ns
Energy	Mega electron Volt	MeV
Positron charge	eplus	
Temperature	kelvin	K
Amount of substance	mole	mol
Luminous intensity	candela	cd
Plane angle	radian	rad
Solid angle	steradian	sr

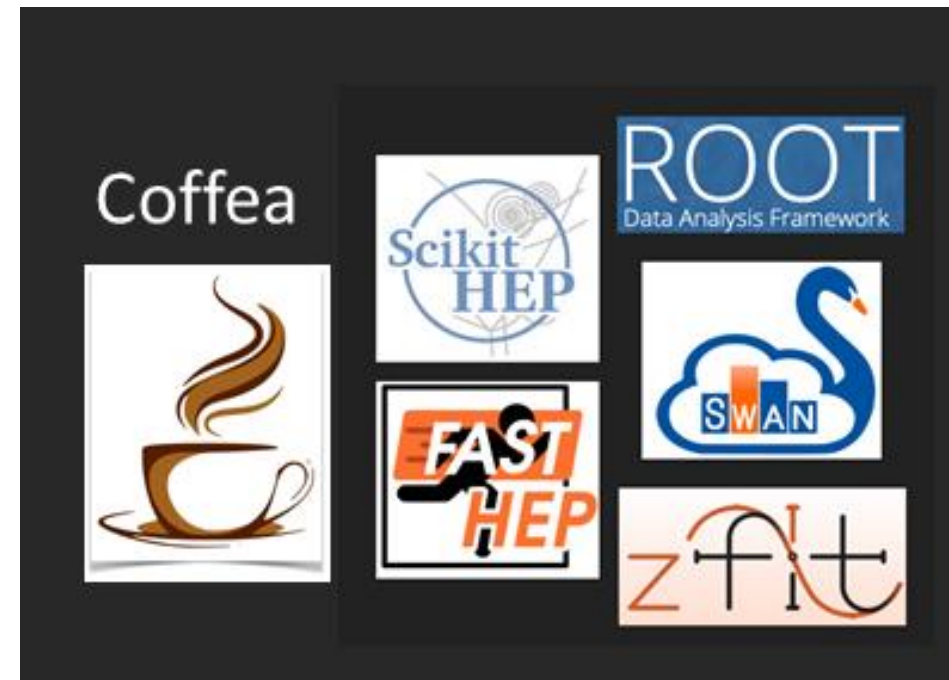
Other community projects

- ❑ **zfit - fitting**
- ❑ **Coffea - Columnar Object Framework For Effective Analysis**
- ❑ **FAST-HEP – Analysis Description Language oriented toolkit**
- ❑ **Package availability via conda-forge, not just pip**

Other community projects

- ❑ Other groups are working toward the same goal,
i.e. a Python(ic) ecosystem for data analysis in Particle Physics,
which is community-driven and community-oriented
- ❑ Interested? Get involved, become a user *and* a developer !

- ❑ <https://github.com/CoffeaTeam>
- ❑ <https://github.com/FAST-HEP>
- ❑ <https://github.com/root-project/>
- ❑ <https://scikit-hep.org/>
- ❑ <https://github.com/zfit>



The zfit project and package

❑ Project: provide a stable fitting ecosystem, in close collaboration with the community

❑ zfit package:

- Scalable, Pythonic, HEP specific features
- Pure Python, no ROOT dependency, performant (TensorFlow as main backend)
- Highly customisable and extendable
- Depends on iminuit



❑ Simple example:

```
obs = zfit.Space("x", limits=(-2, 3))
```

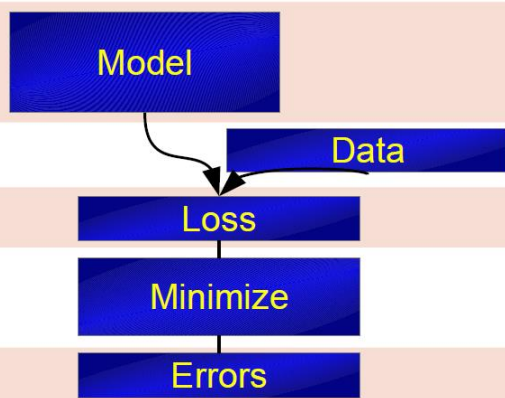
```
mu = zfit.Parameter("mu", 1.2, -4, 6)
sigma = zfit.Parameter("sigma", 1.3, 0.1, 10)
gauss = zfit.pdf.Gauss(mu=mu, sigma=sigma, obs=obs)
```

```
data = zfit.Data.from_numpy(obs=obs, array=normal_np)
```

```
nll = zfit.loss.UnbinnedNLL(model=gauss, data=data)
```

```
minimizer = zfit.minimize.Minuit()
result = minimizer.minimize(nll)
```

```
param_errors = result.error()
```



implement custom function

```
from zfit import ztf
```

```
class CustomPDF(zfit.pdf.ZPDF):
    _PARAMS = ['alpha']
```

```
def _unnormalized_pdf(self, x):
    data = x.unstack_x()
    alpha = self.params['alpha']

    return ztf.exp(alpha * data)
```

```
custom_pdf = CustomPDF(obs=obs, alpha=0.2)
```

```
integral = custom_pdf.integrate(limits=(-1, 2))
sample = custom_pdf.sample(n=1000)
prob = custom_pdf.pdf(sample)
```



Coffea - Column Object Framework for Effective Analysis



Fermilab project to build an analysis framework on top of awkward array and uproot

Separation of “user code” and “executors”

- User writes a Processor to do the analysis
- Executor runs this on different distributed job systems, e.g.:
 - Local multiprocessing, Parsl or Dask (batch systems), Spark cluster

Coffea *achieved 1 to 3 MHz* event processing rates

- Using Spark cluster on same site as data at Fermilab

The FAST-HEP project



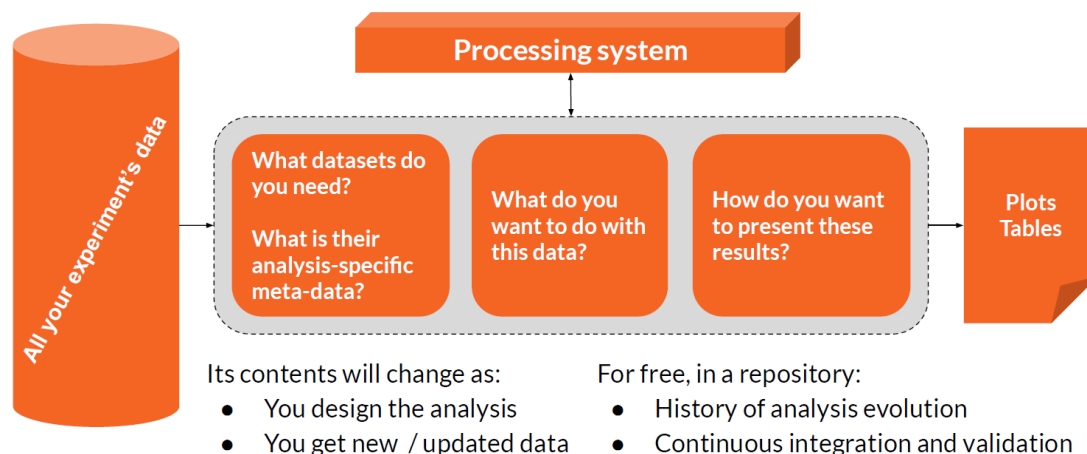
FAST-HEP

Toolkit to help high-level analyses, in particular, within particle physics

<http://fast-hep.web.cern.ch> fast-hep@cern.ch

- ❑ The main product should be the repository
 - Talking about contents – publication is another matter ;-)

Your analysis repository is your analysis



55

- ❑ Use a declarative programming approach:
User sys WHAT, interpretation decides HOW

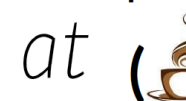
- ❑ Project towards an Analysis Description Language ...

The FAST implementation

For tools:
use Python



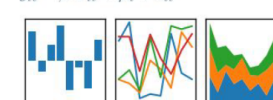
NumExpr



For data:
use Pandas
Demoed at CHEP 2018

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



For descriptions:
use YAML...

63

Material taken from Ben Krikler

Conda-forge – making it easy for users



conda-forge

A community led collection of recipes, build infrastructure and distributions for the conda package manager.

<https://conda-forge.org> conda-forge@googlegroups.com

❑ Easy / trivial installation in many environments is a must !

❑ Much work has been done in 2019 to provide
binary “wheels” on PyPI, and conda-forge packages
for many of these new packages

❑ Example of uproot:

Conda

Files

📄 License: BSD-3-Clause

🏠 Home: <https://github.com/scikit-hep/uproot>

🔗 Development: <https://github.com/scikit-hep/uproot>

📖 Documentation: <https://uproot.readthedocs.io/en/latest/>

📥 161999 total downloads

📅 Last upload: 8 days and 16 hours ago

Installers

conda install ?

linux-64 v3.11.1

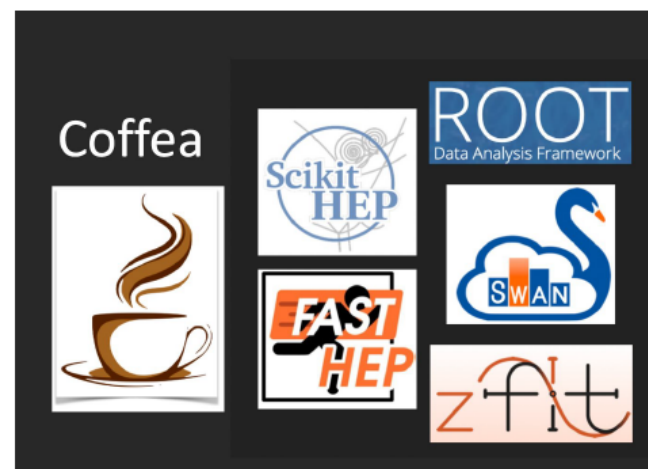
osx-64 v3.11.1

win-64 v3.11.1

To install this package with conda run:
`conda install -c conda-forge uproot`

PyHEP (“Python in HEP”) and New Approaches

- Python is ever more popular in Particle Physics
- Impressive developments of a Python scientific ecosystem for HEP in the last 2 years
- With strong links to the general scientific ecosystem
 - Interest in *data science* tools and *machine learning* is significant for this growing community
- Inspiring new approaches for data analysis
 - Exploiting modern approaches - declarative programming, heterogeneous resources, etc.
 - This is an ecosystem into which HEP can, and does, contribute
 - Knowledge transfer goes both ways
 - Various projects under development, inter-communicating
- Yearly PyHEP workshops have been a success
 - Next year hoping to co-locate with SciPy 2020



Thank you for listening

- ❑ HEP Software Foundation (HSF)
 - HSF general forum hsf-forum@googlegroups.com
- ❑ HSF PyHEP Working Group
 - [Gitter channel](#)
 - GitHub repository [“Python in HEP” resources](#)
- ❑ Scikit-HEP project
 - [Get in touch](#)