HEP data analysis in Python community efforts and ecosystem

Eduardo Rodrigues
University of Liverpool

RAL Particle Physics Seminar, All@Home, 23rd June 2021

Outline

Data analysis in High Energy Physics (HEP) has evolved considerably in recent years. In particular, the role of Python has gained 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 Sofware 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 introduce 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 a Big Data ecosystem for analysis in Python. It has developed considerably in the past year or so, and is now part of the official software stack of the experiments ATLAS, Belle II, CMS and KM3NeT.

- ❑ **Particle Physics and Big Data**
- ❑ **The reign of Python**
- ❑ **Community efforts – HSF, PyHEP**
- ❑ **The PyHEP 2020 & 2021 workshops**
- ❑ **The Scikit-HEP project**
- ❑ **Community software projects**
- ❑ **Final remarks**

Outline

Data analysis in High Energy Physics (HEP) has evolved considerably in recent years. In particular, the role of Python has gained 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 Sofware Foundation created a PyHEP - "Pythor in HEP" - working group and has been organising P workshops since 2018. Moreover, many analysis packages have seen the lic providing interesting, modern an perform analysis, in Python. In shor effort is only getting stronger.

I have been intimately involved in al and will provide an overview of the lar introduce the Scikit-HEP project I startec a few colleagues from various backgroun of expertise. Scikit-HEP is a commur community-oriented project with the aim uding Particle Physics at large with a Big Data e system for analysis in Python. It has developed considerably in the past year or so, and is now part of the official software stack of the experiments ATLAS, Belle II, CMS and KM3NeT.

Not a typical seminar:
Very little personal work.
Lots of community work.

- and Big Data
- gh of Python
- □ Community efforts – HSF, PyHEP
- □ The PyHEP 2020 workshop
- □ The Scikit-HEP project
- □ Community software projects
- □ Final remarks

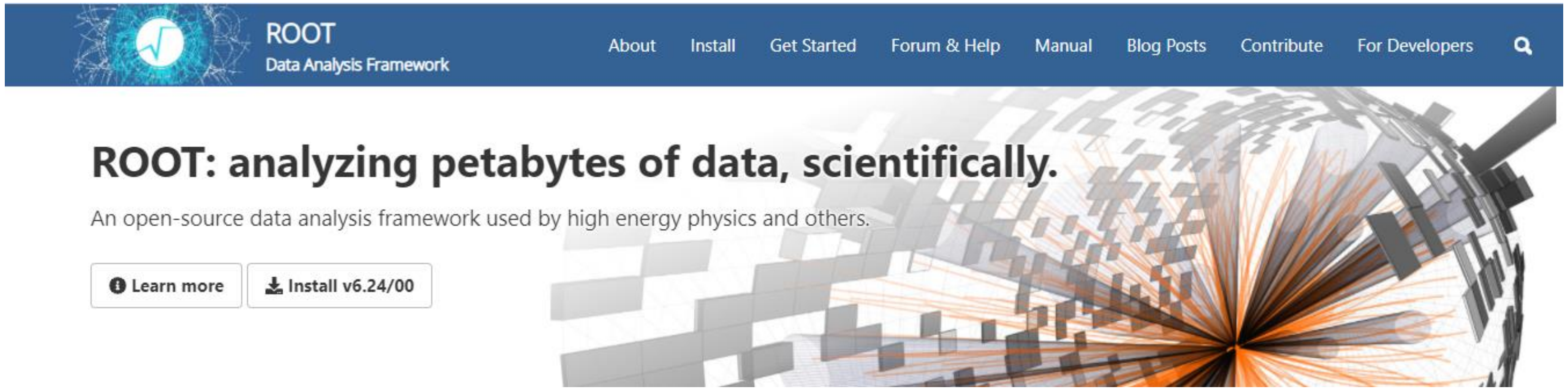
Particle Physics and Big Data
Some “random” thoughts

❑ “Big Data” projects

❑ Setting the scene

Particle Physics and Big Data

- ❑ A lot of what has happened in the HEP Python ecosystem (recent years) can be thought of as trying to bridge the Particle Physics & Big Data worlds and profit from what the Data Science scientific software stack has to offer
- ❑ We will come back to software, but what about the data itself? Is that "Big Data"?
- ❑ The CERN ROOT team advertises that of the order of 1 EB of data exists right now in the *.root* format:



ROOT: analyzing petabytes of data, scientifically.

An open-source data analysis framework used by high energy physics and others.

- ❑ Impressive. We are already in the exascale era!

[Many figures have active links.]

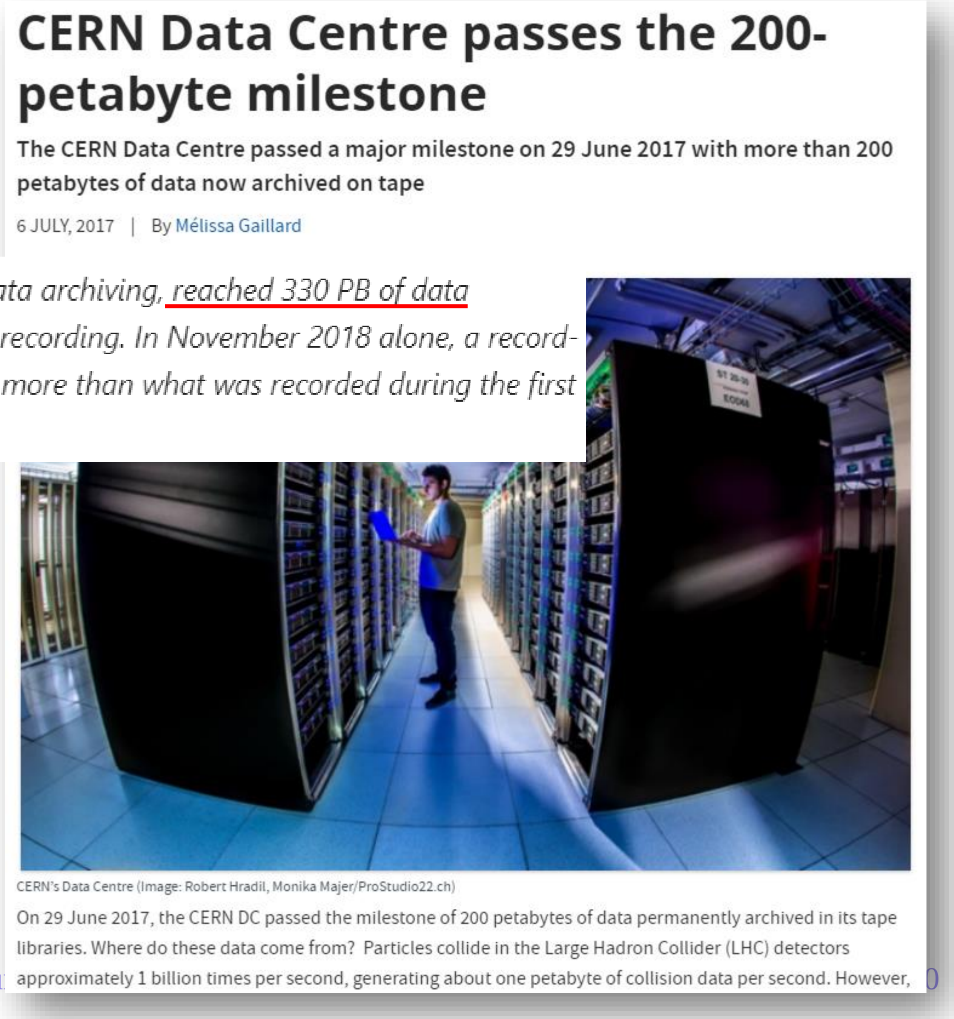
Particle Physics and Big Data – on the CERN Data Centre storage

- ❑ **For the record, the CERN Data Centre had accumulated more than 200 PB of data back in 2017 already!**
  - CERN news, July 6, 2017

- ❑ **And in just an extra 1.5 years,**
  50% more data got saved in CERN's Data Centre

- ❑ **Citing the CERN news, March 1, 2019:**

> *"The CERN Advanced Storage system (CASTOR), which relies on a tape-based backend for permanent data archiving, reached 330 PB of data (equivalent to 330 million gigabytes) stored on tape, an equivalent of over 2000 years of 24/7 HD video recording. In November 2018 alone, a record-breaking 15.8 PB of data were recorded on tape, a remarkable achievement given that it corresponds to more than what was recorded during the first year of the LHC's Run 1."*



CERN Data Centre passes the 200-petabyte milestone

The CERN Data Centre passed a major milestone on 29 June 2017 with more than 200 petabytes of data now archived on tape

6 JULY, 2017 | By Mélissa Gaillard

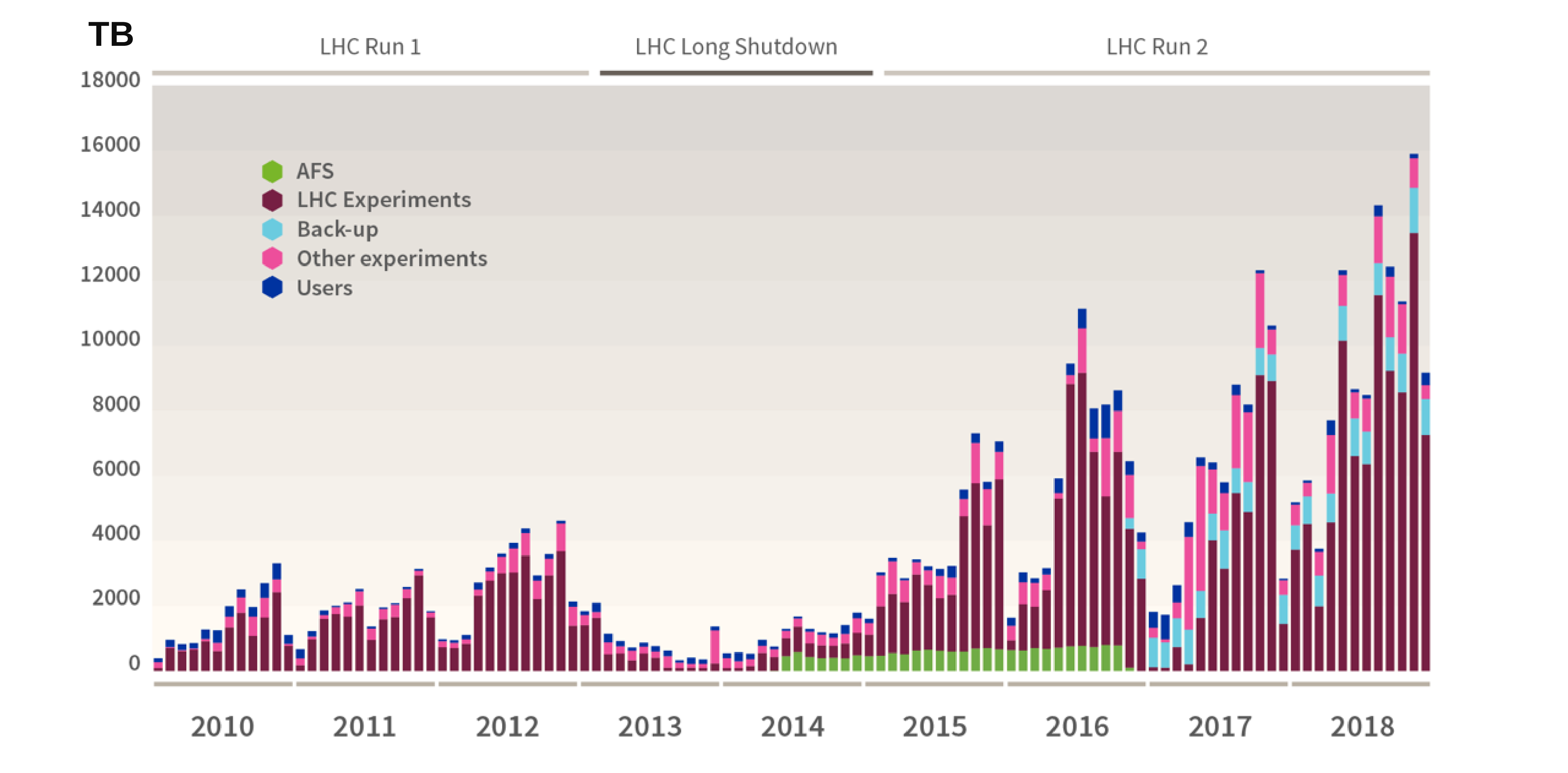
CERN's Data Centre (Image: Robert Hradil, Monika Majer/ProStudio22.ch)

On 29 June 2017, the CERN DC passed the milestone of 200 petabytes of data permanently archived in its tape libraries. Where do these data come from? Particles collide in the Large Hadron Collider (LHC) detectors approximately 1 billion times per second, generating about one petabyte of collision data per second. However,

- ❑ **In fact, in 2018, over 115 PB of data in total**
  (including about 88 PB of LHC data) were recorded on tape

Particle Physics and Big Data – on the CERN Data Centre storage

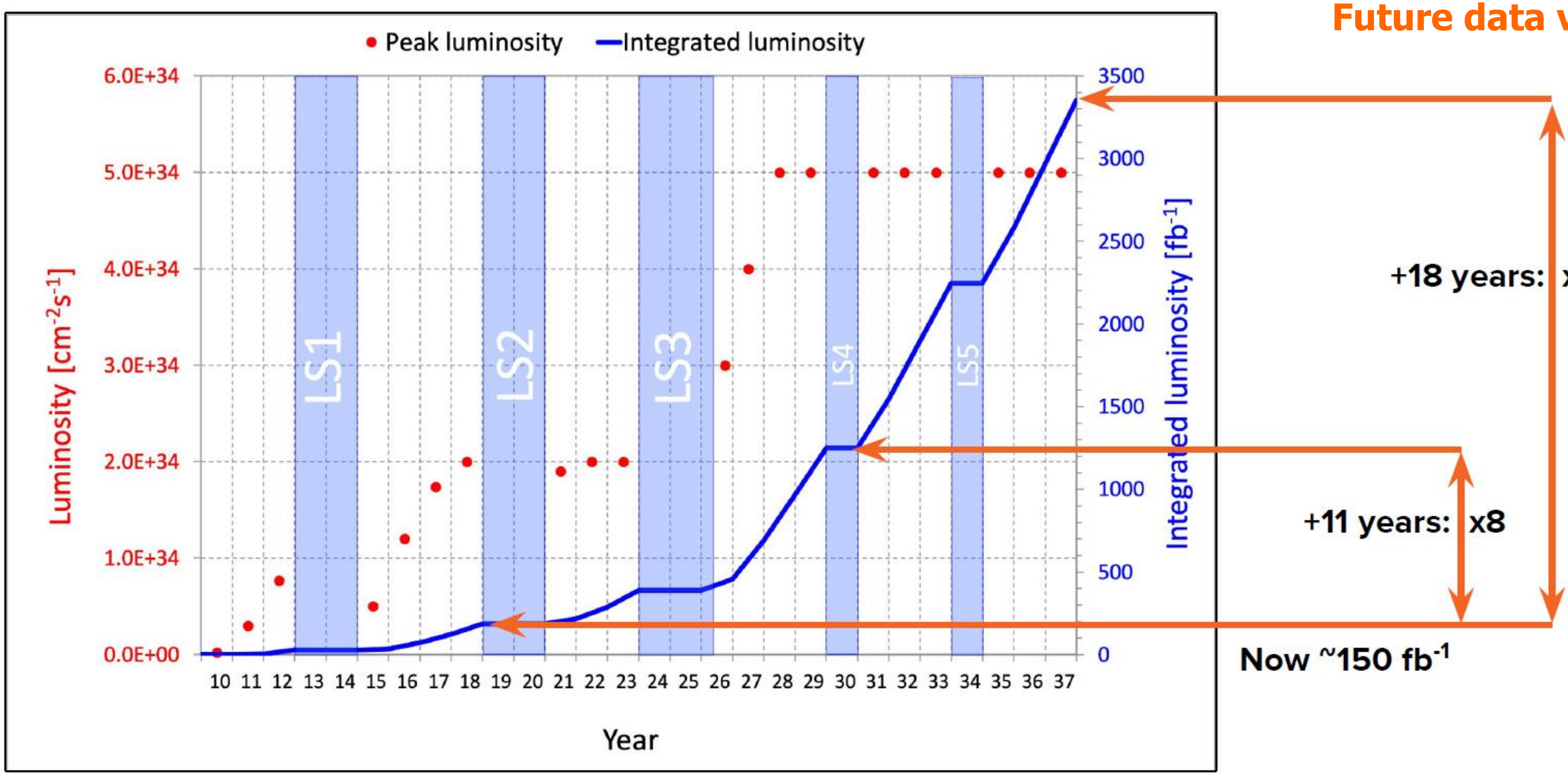
- ❑ The accumulation of data generated by the LHC experiments alone, over a decade-ish, speaks for itself, as seen by this graph on CERN computing:
  data (in terabytes) recorded on tapes at CERN month-by-month (2010–2018)



Taken from CERN CDS.

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

Future data volumes

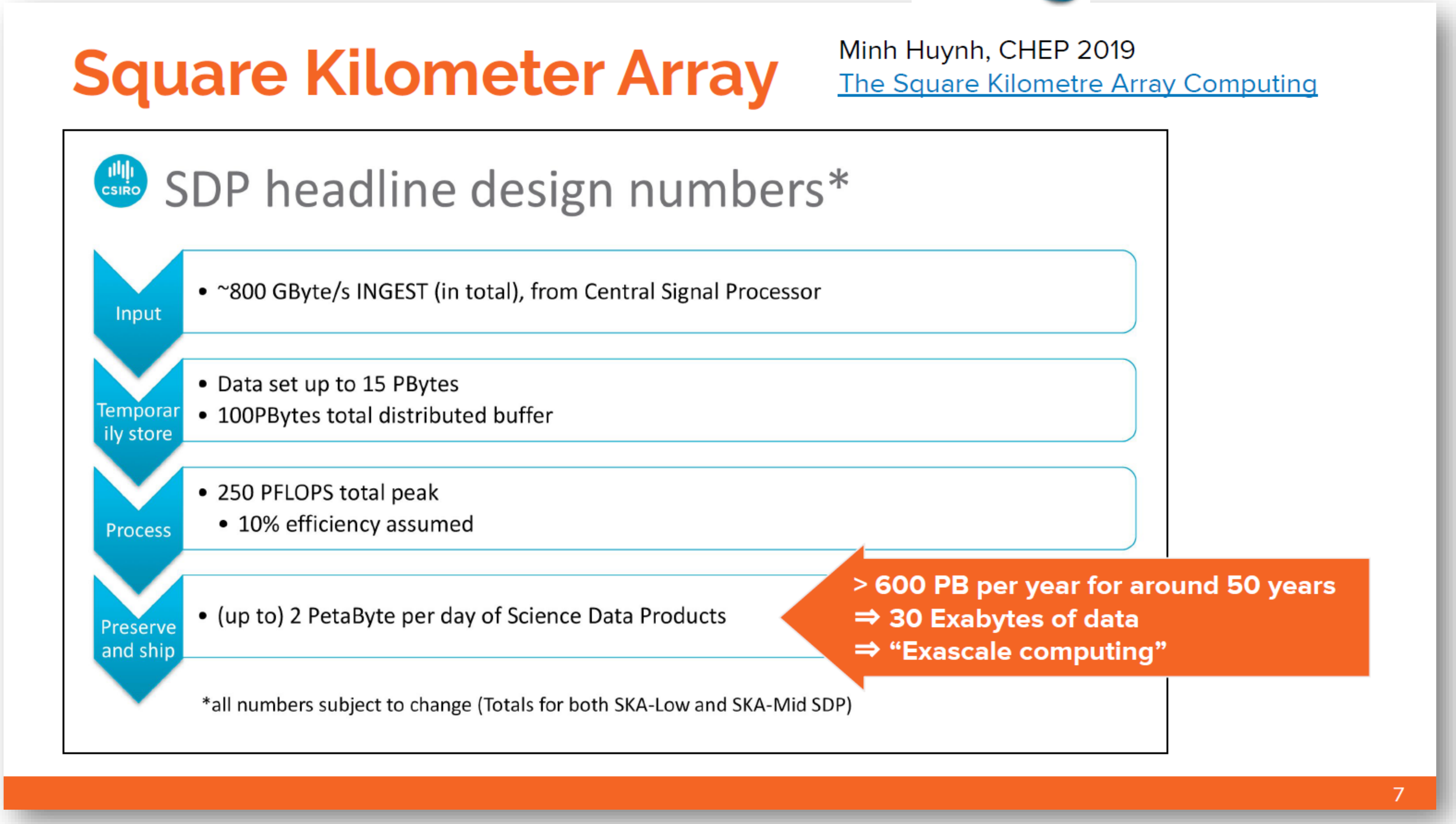


Now ~150 fb^{-1}

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

Another "Big Data project" – SKA

SQUARE KILOMETRE ARRAY
Exploring the Universe with the world's largest radio telescope



Square Kilometer Array

Minh Huynh, CHEP 2019
The Square Kilometre Array Computing

SDP headline design numbers*

Input
- ~800 GByte/s INGEST (in total), from Central Signal Processor

Temporarily store
- Data set up to 15 PBytes
- 100PBytes total distributed buffer

Process
- 250 PFLOPS total peak
  - 10% efficiency assumed

Preserve and ship
- (up to) 2 PetaByte per day of Science Data Products

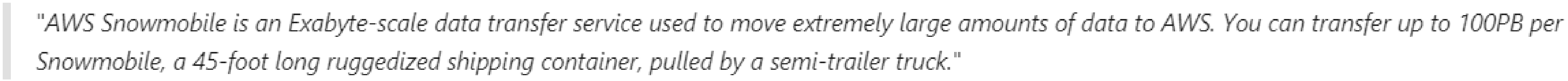
> 600 PB per year for around 50 years
⇒ 30 Exabytes of data
⇒ “Exascale computing”

*all numbers subject to change (Totals for both SKA-Low and SKA-Mid SDP)

Slide taken from Ben Krikler

Particle Physics and Big Data – and what about the outside world?

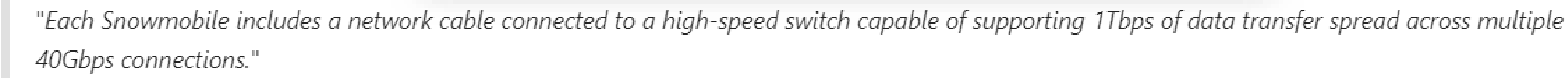
- ❑ **Let's look at Amazon for the sake of argument:**



> *"AWS Snowmobile is an Exabyte-scale data transfer service used to move extremely large amounts of data to AWS. You can transfer up to 100PB per Snowmobile, a 45-foot long ruggedized shipping container, pulled by a semi-trailer truck."*

NEW
AWS Snowmobile: 100PB Container
45-foot long rugged container & truck
Connect to your datacenter with fiber cable
Fill 'er Up!
Transports Data To AWS

Taken from Amazon page.

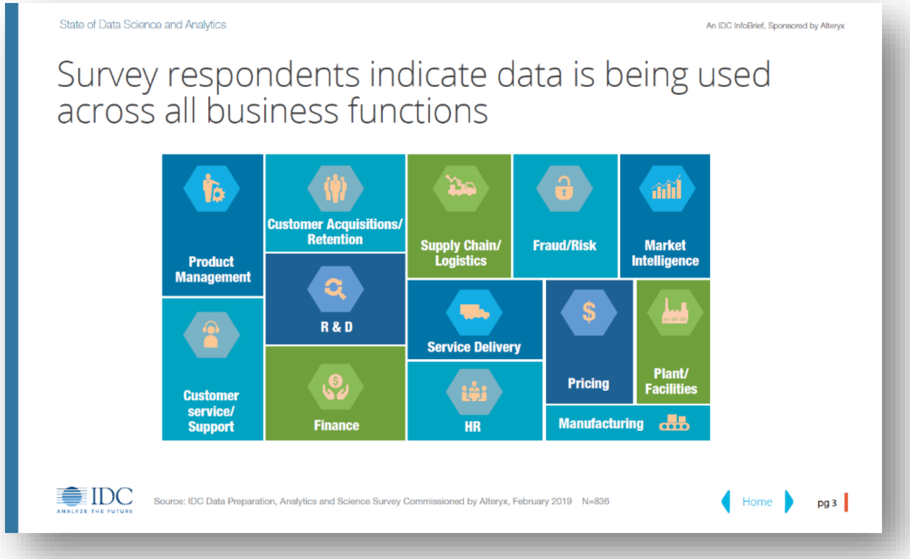


> *"Each Snowmobile includes a network cable connected to a high-speed switch capable of supporting 1Tbps of data transfer spread across multiple 40Gbps connections."*

- ❑ **To be compared with the throughput of 0.5-1.5 Tbps the LHCb experiment's first high-level trigger HLT1 (partial reconstruction on GPUs) will put to buffer while the real-time calibration and alignment is run, which is needed to digest the data in the HLT2 (full reconstruction) … next year !**

Is it relevant and useful to learn non-HEP tools? Mostly Python, BTW!

- ❑ We've just quickly recalled that data requirements for Particle Physics match those of the Big Data world
- ❑ Huge amounts of data are in fact used by companies worldwide for just about any business, see for example the report "State of Data Science and Analytics, IDC InfoBrief, April 2019":



State of Data Science and Analytics

An IDC InfoBrief, Sponsored by Alteryx

Survey respondents indicate data is being used across all business functions

IDC ANALYZE THE FUTURE

Source: IDC Data Preparation, Analytics and Science Survey Commissioned by Alteryx, February 2019 N=836

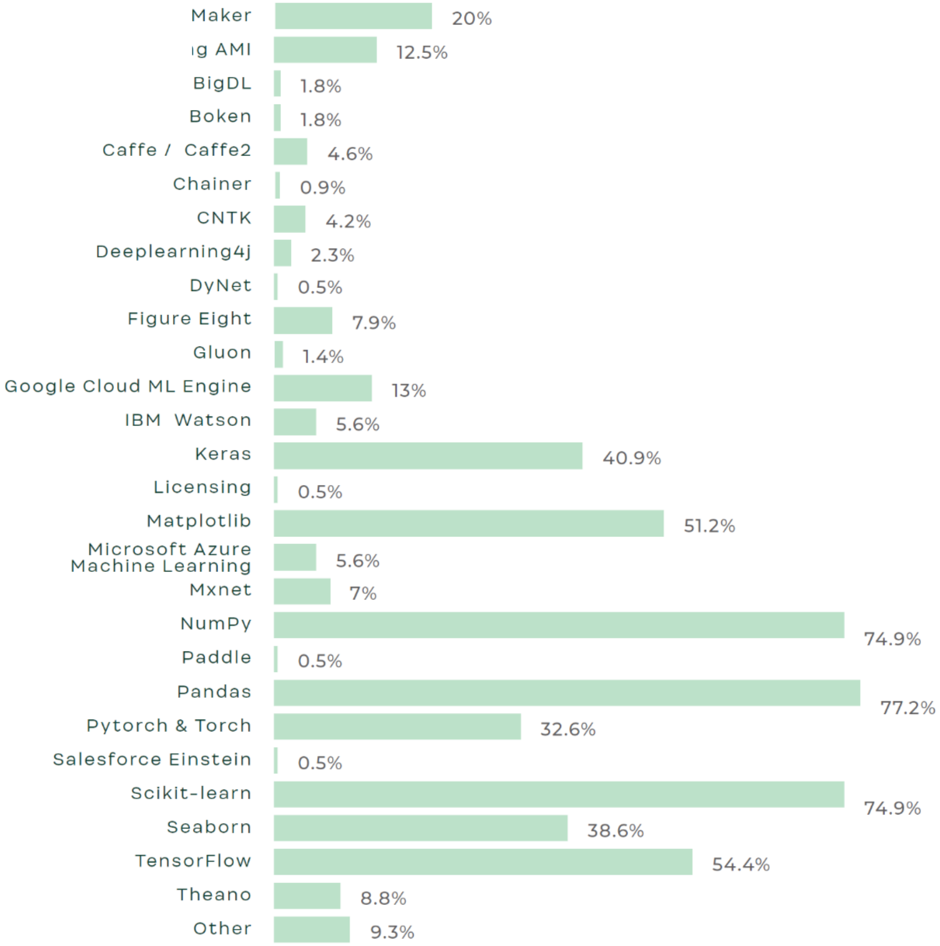
Home pg 3

- ❑ International surveys indicate over 50M data workers worldwide!
- ❑ Can we really compete in terms of tools for data analysis?... Or should we rather try and profit from, and even contribute to, the huge ecosystem available to do Data Science?
- ❑ Anyway, what are data scientists, data engineers, etc., using for their daily work? That largely involves to some (larger and larger) extent Machine Learning, statistics, and even AI. International surveys give a hint ... the tools are dominated by Python tools ...!

Is it relevant and useful to learn non-HEP tools? Mostly Python, BTW!

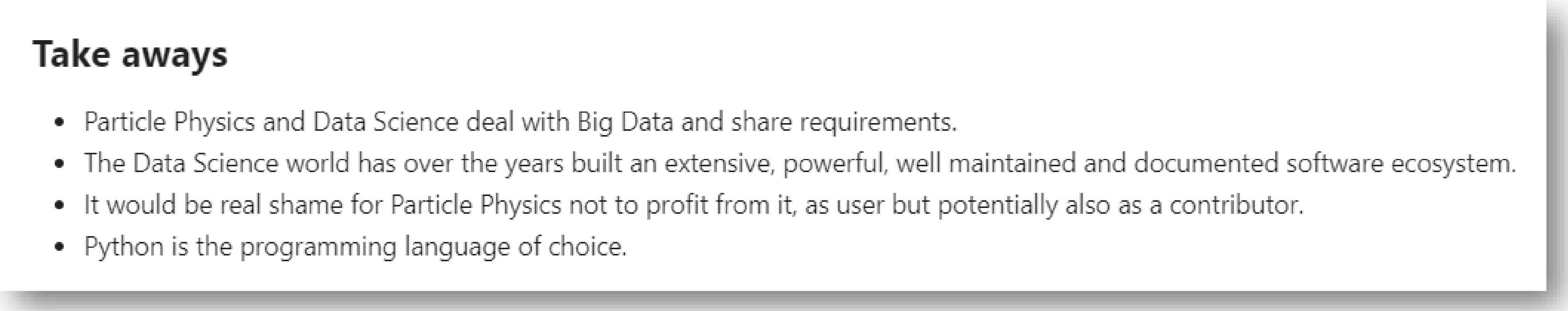
- ❑ **From the 2019 Figure Eight report "The State of AI and Machine Learning“:**

> *"Some popular frameworks and tools technical practitioners prefer in different stages of the ML pipeline are:*
>
> *Numpy and Pandas for loading data; Matplotlib for visualization;*
>
> *Scikit-learn and TensorFlow (including Keras) for ML models.":*



(Figure 17: Machine learning frameworks used by AI technical practitioners)

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



Take aways

- Particle Physics and Data Science deal with Big Data and share requirements.
- The Data Science world has over the years built an extensive, powerful, well maintained and documented software ecosystem.
- It would be real shame for Particle Physics not to profit from it, as user but potentially also as a contributor.
- Python is the programming language of choice.

□ **Lots of data?**

⇒ **Look at what the Big Data community is doing**

□ **Evolution of computing resources won’t be enough to digest all data**

⇒ **Use resources as efficiently as possible**

□ **Physicists want to minimise the “time to insight”. But coding takes a fair share of one’s time, and is error-prone.**

⇒ **Adopt open-source best practices, popular and easy languages**

⇒ ***This talk will focus on offline data analysis tools, hence post trigger processing***

(it will not discuss ROOT either)

The reign of Python

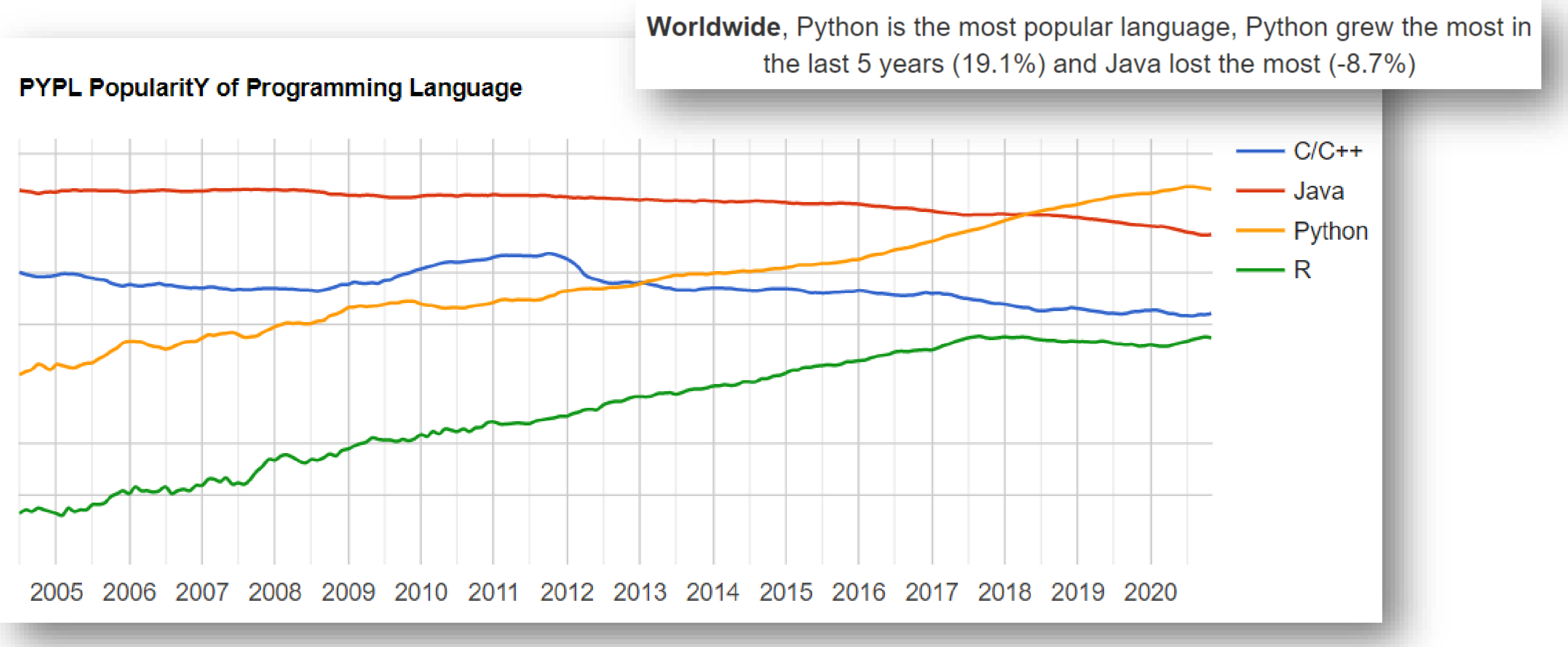
- ❑ **Popularity has never been so high**

  - in Data Science

  - in Particle Physics

Python (in HEP), you say?

- ❑ PopularitY of Programming Languages (PYPL) – Python is the big winner!
- ❑ Popularity based on how often language tutorials are searched for
  - - Data from Google Trends
  - - Log scale!
- ❑ Same conclusion for popularity of languages for ML



Worldwide, Python is the most popular language, Python grew the most in the last 5 years (19.1%) and Java lost the most (-8.7%)

[Taken from http://pypl.github.io/PYPL.html]

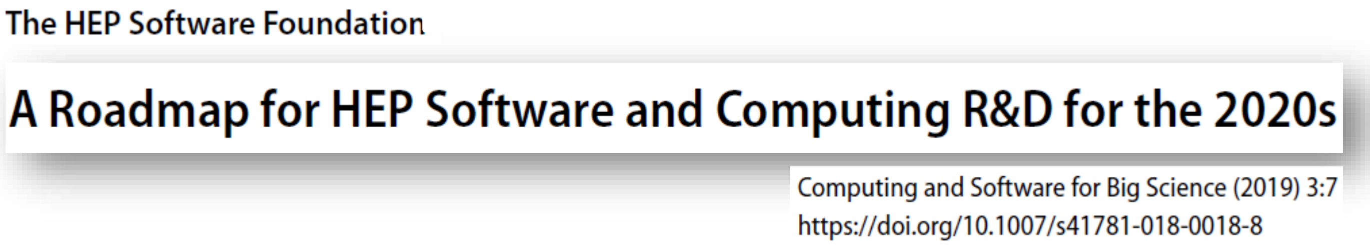
Why?

- ❑ Very large software ecosystem built atop NumPy and SciPy
- ❑ With very large and active community
- ❑ In general, excellent documentation and community support

(All Open Source – FOSS has proven its worth!)

- ❑ …

Let’s roll back a few years … at the time of the Community White Paper



The HEP Software Foundation

A Roadmap for HEP Software and Computing R&D for the 2020s

Computing and Software for Big Science (2019) 3:7
https://doi.org/10.1007/s41781-018-0018-8

- ❑ **Python had already been identified as a first-class language for Particle Physics back in 2016-17:**

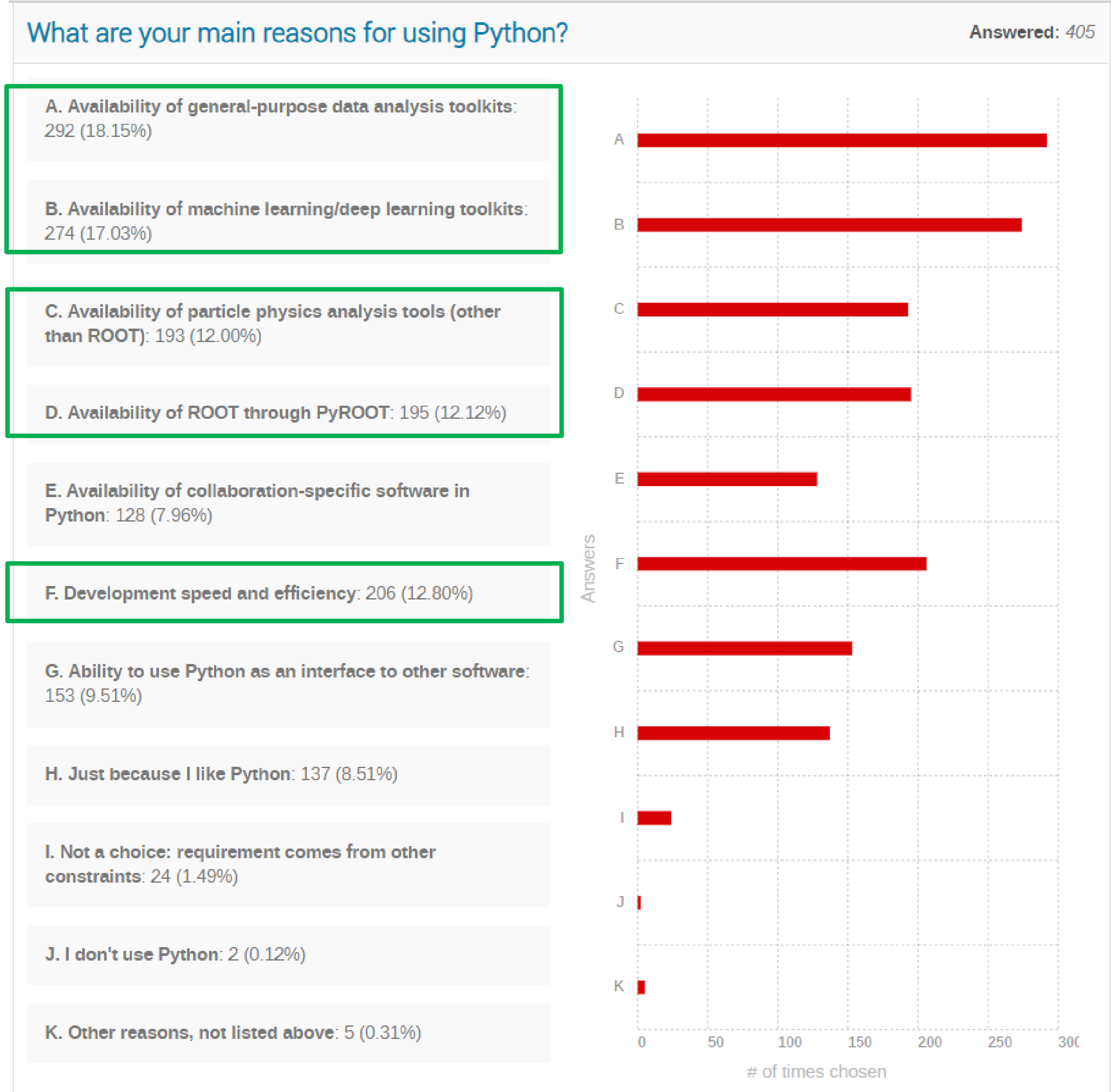
> “Python has emerged as the language of choice in the data science community, and its use continues to grow within HEP. … Python could reduce the complexity of analysis code, and therefore contribute to decreasing the “time to insight” for HEP analyses, as well as increasing their sustainability. Increased HEP investment is needed to allow Python to become a first-class supported language.”

- ❑ **A lot happened in the meantime**
  - **- Evolution of (Py)ROOT**
  - **- Community-wise**

> “– Expand support of Python in our ecosystem with a strategy for ensuring long-term maintenance and sustainability. In particular in ROOT, the current Python bindings should evolve to reach the ease of use of native Python modules..”

- ❑ **What follows exemplifies the evolution and what helped shape that evolution …**

Why do particle physicists use Python ?



What are your main reasons for using Python?

Answered: *405*

A. Availability of general-purpose data analysis toolkits: 292 (18.15%)

B. Availability of machine learning/deep learning toolkits: 274 (17.03%)

C. Availability of particle physics analysis tools (other than ROOT): 193 (12.00%)

D. Availability of ROOT through PyROOT: 195 (12.12%)

E. Availability of collaboration-specific software in Python: 128 (7.96%)

F. Development speed and efficiency: 206 (12.80%)

G. Ability to use Python as an interface to other software: 153 (9.51%)

H. Just because I like Python: 137 (8.51%)

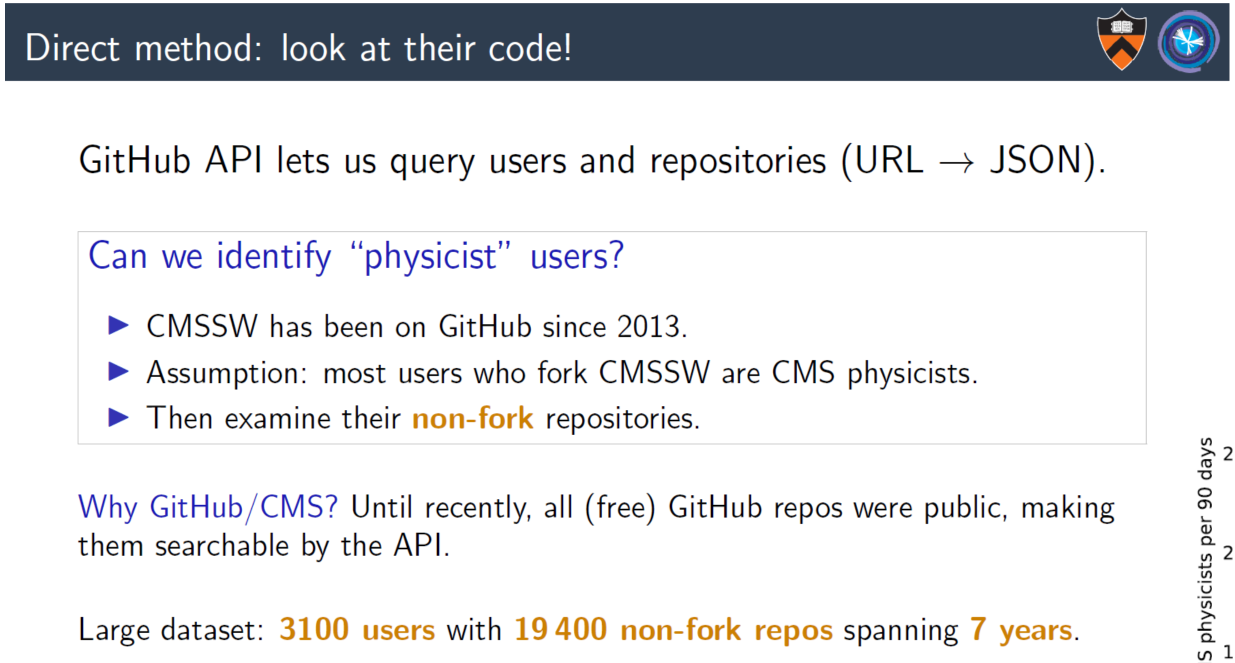
I. Not a choice: requirement comes from other constraints: 24 (1.49%)

J. I don't use Python: 2 (0.12%)

K. Other reasons, not listed above: 5 (0.31%)

Taken from the PyHEP 2020 pre-workshop survey (408 respondents)

Python adoption in HEP – CMS study



Direct method: look at their code!

GitHub API lets us query users and repositories (URL → JSON).

Can we identify “physicist” users?

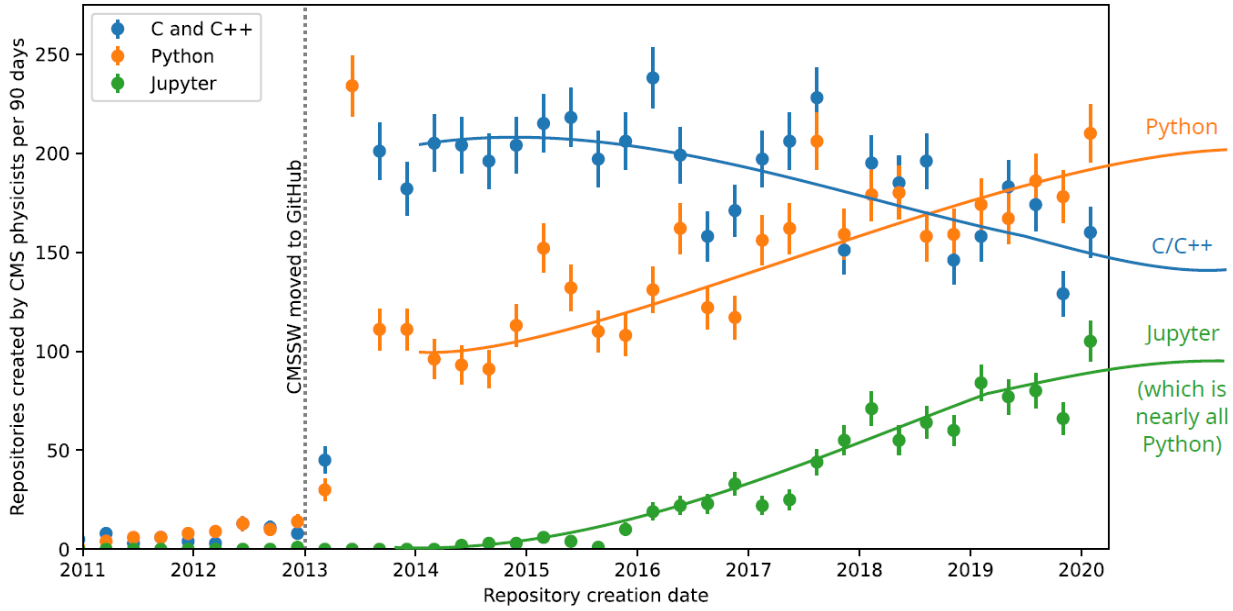
- CMSSW has been on GitHub since 2013.
- Assumption: most users who fork CMSSW are CMS physicists.
- Then examine their **non-fork** repositories.

Why GitHub/CMS? Until recently, all (free) GitHub repos were public, making them searchable by the API.

Large dataset: **3100 users** with **19 400 non-fork repos** spanning **7 years**.

- ❑ **Study by Jim Pivarski**
  [presentation @ Snowmass 2020, Aug. 11th]
- ❑ **Not from survey but rather directly using GitHub API to measure software adoption**

Language of repos created by CMS physicists

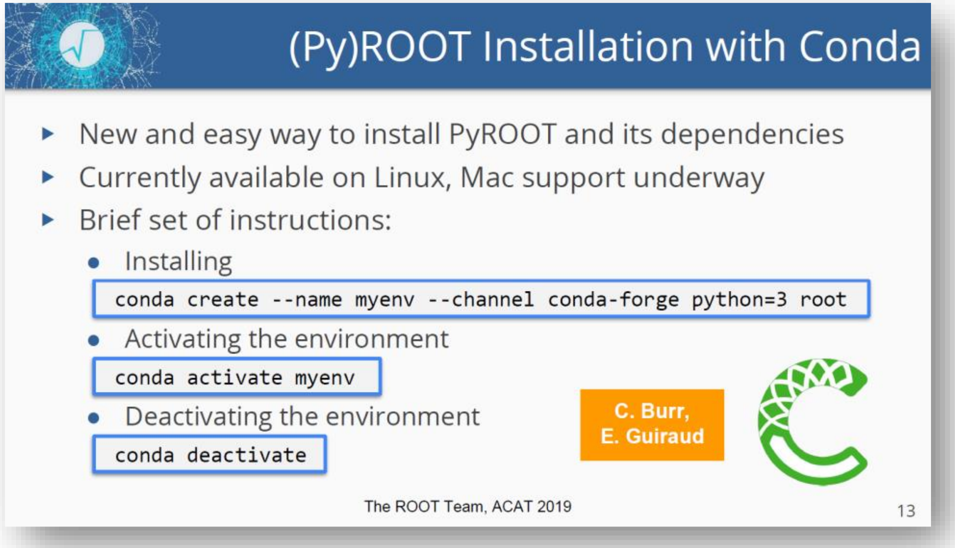


ROOT & Python

❑ (Py)ROOT has evolved enormously over the last few years !

❑ Some sources of material on latest goodies:
- ICHEP 2020 talk on "Hello RNTuple and friends: what the new ROOT means for your analysis"
- CHEP 2019 talk on "A New PyROOT: Modern, Interoperable and more Pythonic" :

❑ A game-changer in my opinion –installation via Conda !
- Came largely from the community and not the ROOT team!



(Py)ROOT Installation with Conda

- New and easy way to install PyROOT and its dependencies
- Currently available on Linux, Mac support underway
- Brief set of instructions:
  - Installing

```
conda create --name myenv --channel conda-forge python=3 root
```

  - Activating the environment

```
conda activate myenv
```

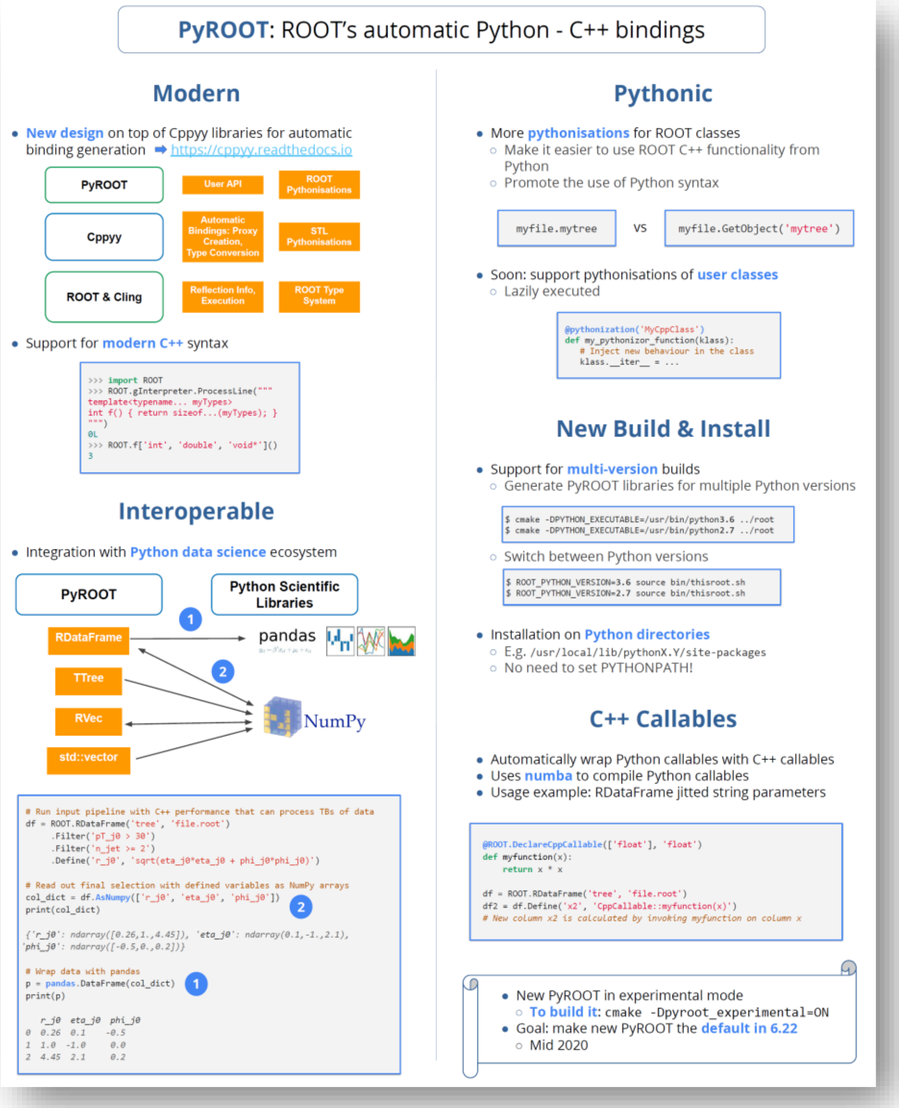
  - Deactivating the environment

```
conda deactivate
```

C. Burr,
E. Guiraud

The ROOT Team, ACAT 2019

13



PyROOT: ROOT's automatic Python - C++ bindings

Modern

- New design on top of Cppyy libraries for automatic binding generation ➡ https://cppyy.readthedocs.io

- Support for modern C++ syntax

```
>>> import ROOT
>>> ROOT.gInterpreter.ProcessLine("""
template<typename... myTypes>
int f() { return sizeof...(myTypes); }
""")
0L
>>> ROOT.f['int', 'double', 'void*']()
3
```

Interoperable

- Integration with Python data science ecosystem

```
# Run input pipeline with C++ performance that can process TBs of data
df = ROOT.RDataFrame('tree', 'file.root')
      .Filter('pT_j0 > 30')
      .Filter('n_jet >= 2')
      .Define('r_j0', 'sqrt(eta_j0*eta_j0 + phi_j0*phi_j0)')

# Read out final selection with defined variables as NumPy arrays
col_dict = df.AsNumpy(['r_j0', 'eta_j0', 'phi_j0'])       2
print(col_dict)

{'r_j0': ndarray([0.26,1.,4.45]), 'eta_j0': ndarray(0.1,-1.,2.1),
'phi_j0': ndarray([-0.5,0.,0.2])}

# Wrap data with pandas
p = pandas.DataFrame(col_dict)     1
print(p)

   r_j0  eta_j0  phi_j0
0  0.26  0.1    -0.5
1  1.0  -1.0     0.0
2  4.45  2.1     0.2
```

Pythonic

- More pythonisations for ROOT classes
  - Make it easier to use ROOT C++ functionality from Python
  - Promote the use of Python syntax

`myfile.mytree` VS `myfile.GetObject('mytree')`

- Soon: support pythonisations of user classes
  - Lazily executed

```
@pythonization('MyCppClass')
def my_pythonizor_function(klass):
   # Inject new behaviour in the class
   klass.__iter__ = ...
```

New Build & Install

- Support for multi-version builds
  - Generate PyROOT libraries for multiple Python versions

```
$ cmake -DPYTHON_EXECUTABLE=/usr/bin/python3.6 ../root
$ cmake -DPYTHON_EXECUTABLE=/usr/bin/python2.7 ../root
```

  - Switch between Python versions

```
$ ROOT_PYTHON_VERSION=3.6 source bin/thisroot.sh
$ ROOT_PYTHON_VERSION=2.7 source bin/thisroot.sh
```

- Installation on Python directories
  - E.g. /usr/local/lib/pythonX.Y/site-packages
  - No need to set PYTHONPATH!

C++ Callables

- Automatically wrap Python callables with C++ callables
- Uses numba to compile Python callables
- Usage example: RDataFrame jitted string parameters

```
@ROOT.DeclareCppCallable(['float'], 'float')
def myfunction(x):
    return x * x

df = ROOT.RDataFrame('tree', 'file.root')
df2 = df.Define('x2', 'CppCallable::myfunction(x)')
# New column x2 is calculated by invoking myfunction on column x
```

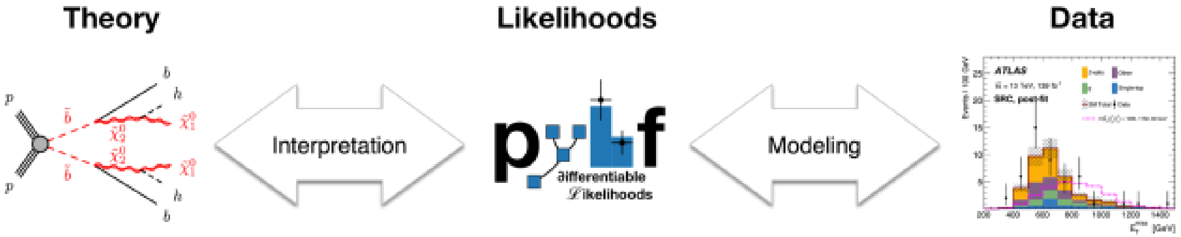
- New PyROOT in experimental mode
  - To build it: cmake -Dpyroot_experimental=ON
- Goal: make new PyROOT the default in 6.22
  - Mid 2020

Python increasingly present in analysis tools used in publications

Full analysis likelihoods published on HEPData

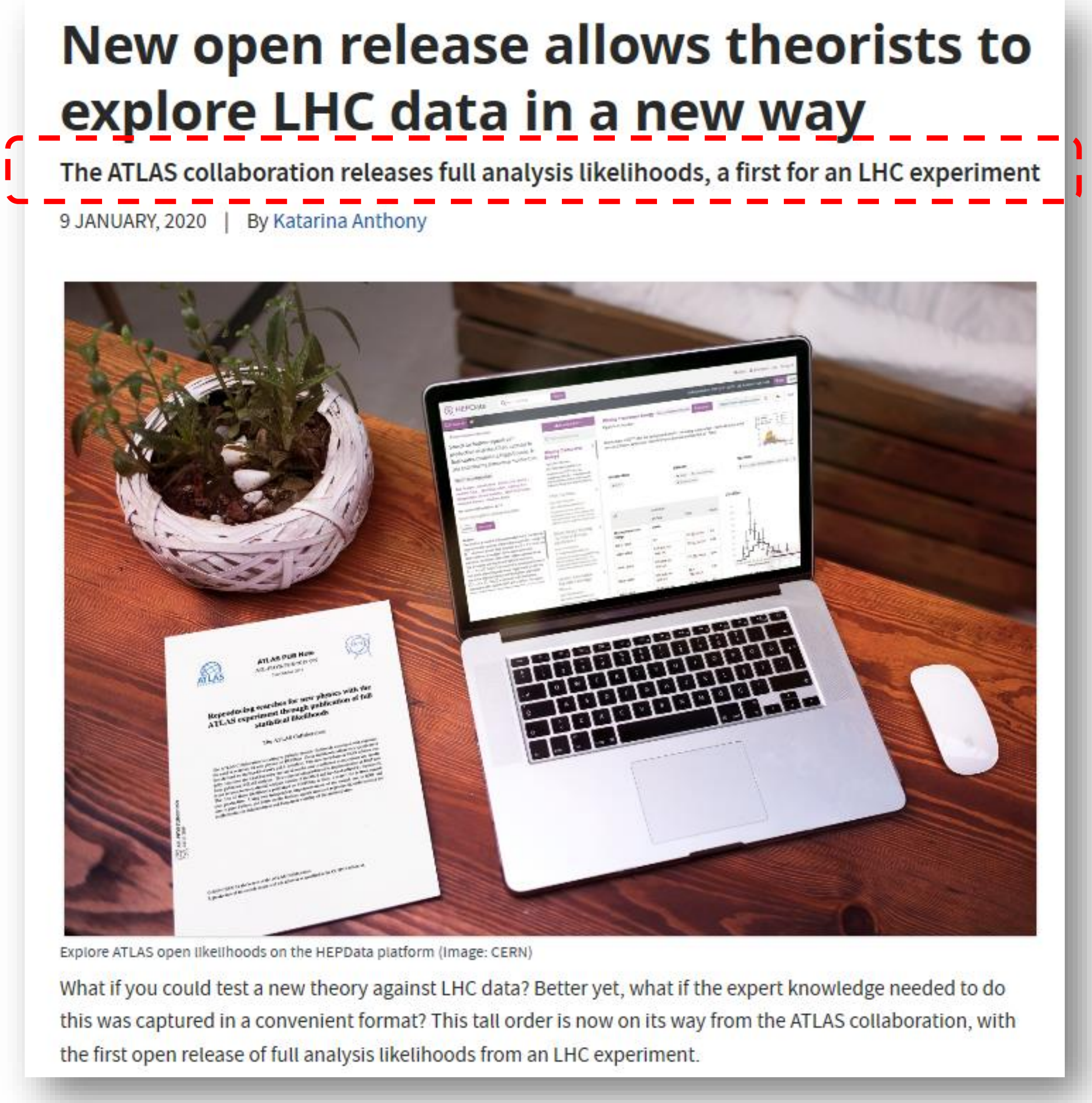
- □ Test theory against LHC data
- □ All that's needed captured in a convenient format

- □ "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 (C++)
- □ pyhf (Python)



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.

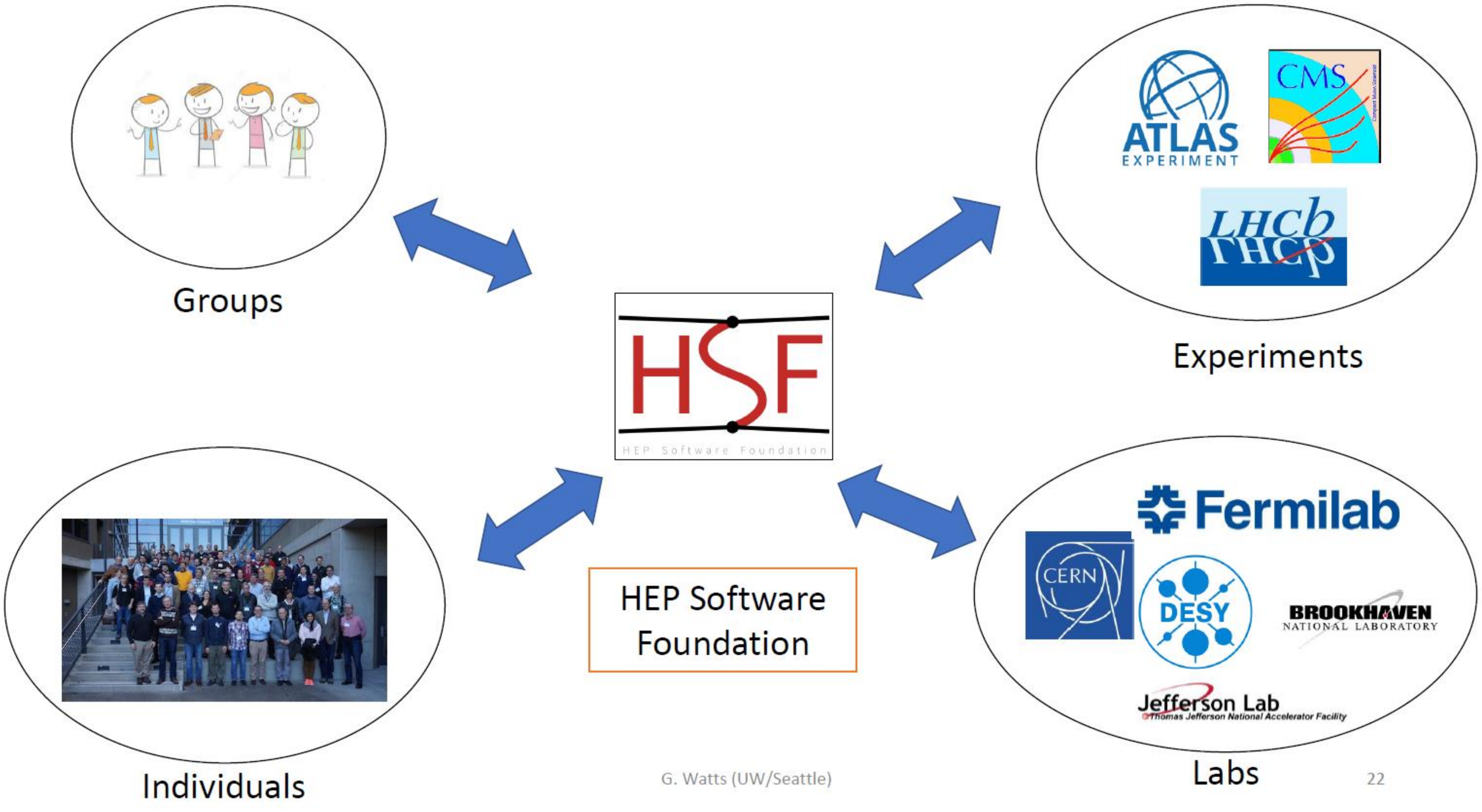
Taken from https://home.cern/news/news/knowledge-sharing/new-open-release-allows-theorists-explore-lhc-data-new-way

Community efforts
HSF & PyHEP

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

HSF - "facilitate coordination and common efforts"

Gordon Watts, ACAT 2019



Groups

Experiments

HEP Software Foundation

Individuals

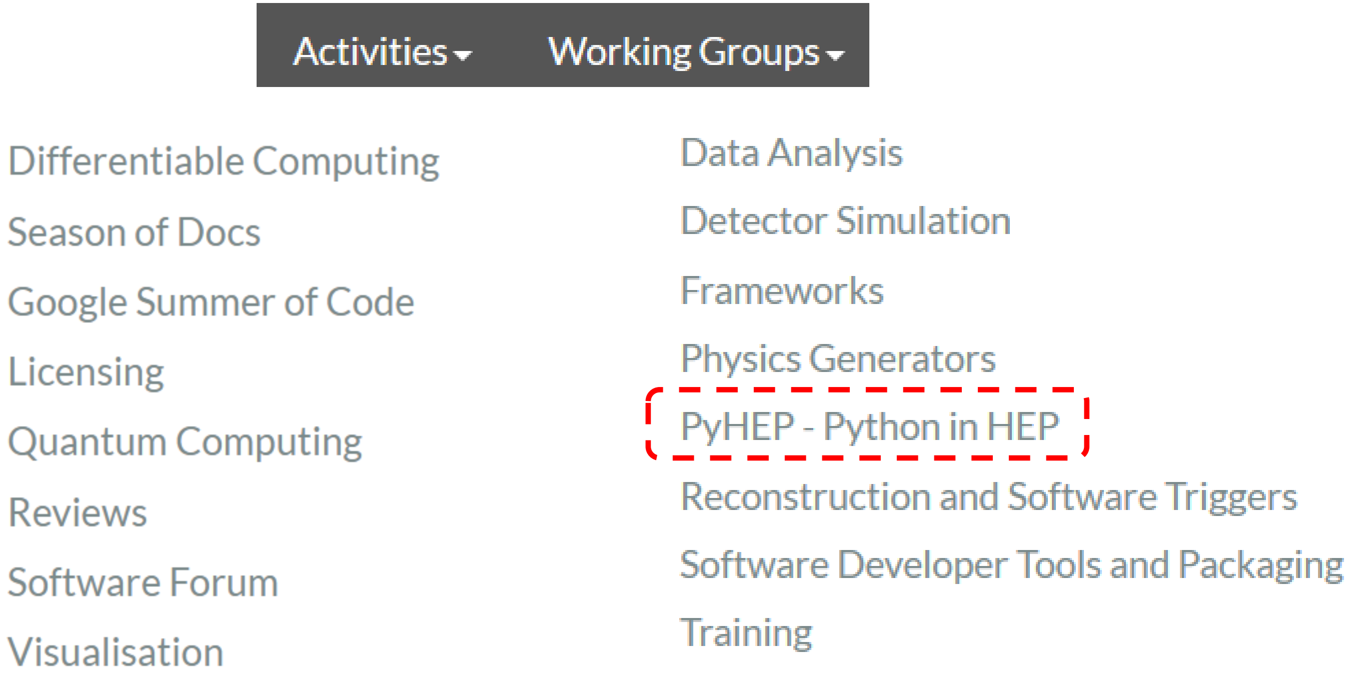
Labs

G. Watts (UW/Seattle)

22

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

- ❑ The “Python in HEP” WG effectively started in early 2018 as an activity group
  - I put it forward with the proposal of the 1st workshop, held as a pre-CHEP 2018 event
- ❑ It became “formally” a WG in 2020 ☺



Activities ▾

- Differentiable Computing
- Season of Docs
- Google Summer of Code
- Licensing
- Quantum Computing
- Reviews
- Software Forum
- Visualisation

Working Groups ▾

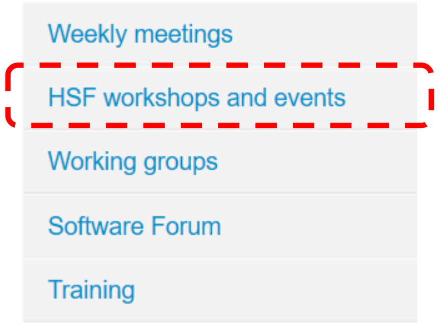
- Data Analysis
- Detector Simulation
- Frameworks
- Physics Generators
- PyHEP - Python in HEP
- Reconstruction and Software Triggers
- Software Developer Tools and Packaging
- Training

indico

Home | Create event ▾ | Room booking

Home » Projects » HEP Software Foundation

HEP Software Foundation

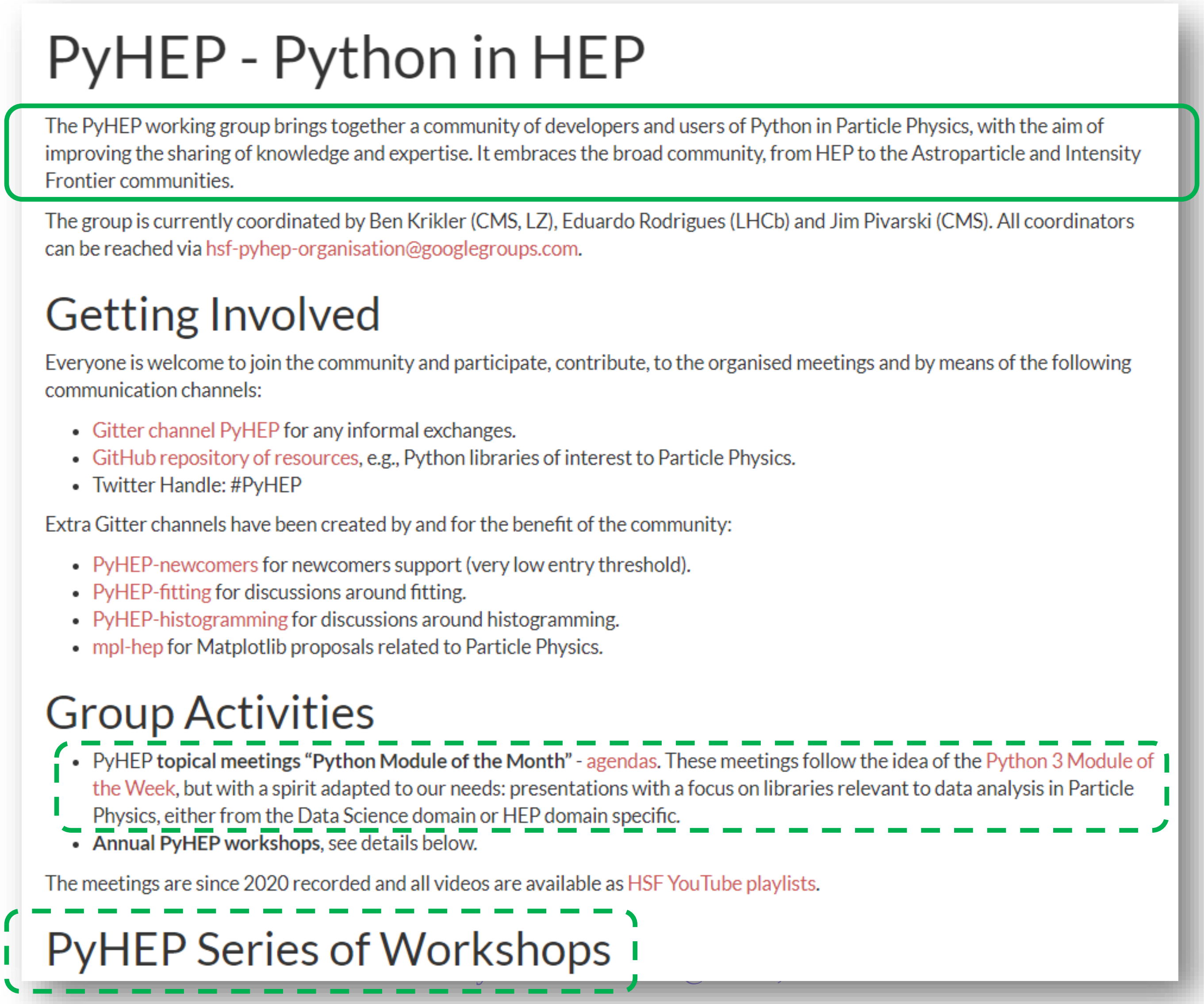


- Weekly meetings
- HSF workshops and events
- Working groups
- Software Forum
- Training

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

❑ **Lots of ways to communicate !**

- The main (Gitter) channel reached 200 people registered just a few days ago



PyHEP - Python in HEP

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, contribute, to the organised meetings and by means of the following communication channels:

- 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-fitting for discussions around fitting.
- PyHEP-histogramming for discussions around histogramming.
- mpl-hep for Matplotlib proposals related to Particle Physics.

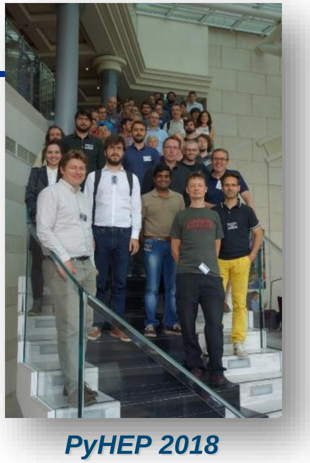
Group Activities

- PyHEP **topical meetings "Python Module of the Month"** - agendas. These meetings follow the idea of the Python 3 Module of the Week, but with a spirit adapted to our needs: presentations with a focus on libraries relevant to data analysis in Particle Physics, either from the Data Science domain or HEP domain specific.
- **Annual PyHEP workshops**, see details below.

The meetings are since 2020 recorded and all videos are available as HSF YouTube playlists.

PyHEP Series of Workshops

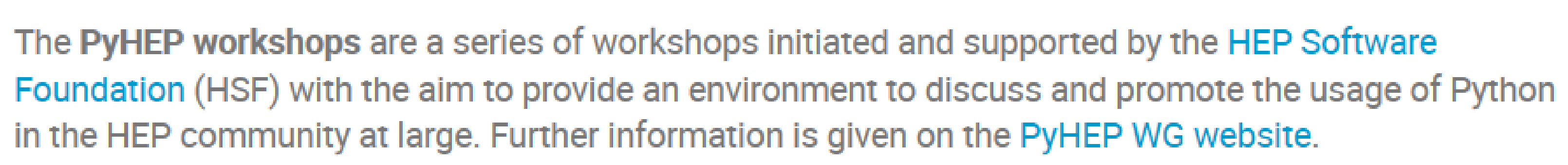
PyHEP (not so new) series of workshops



PyHEP 2018

Sofia, Bulgaria

- ❑ **Started in 2018, recognising the increasing importance of Python in Particle Physics**
  - **- There are several conferences & workshops on C&SW but nothing existed with Python as first-class**

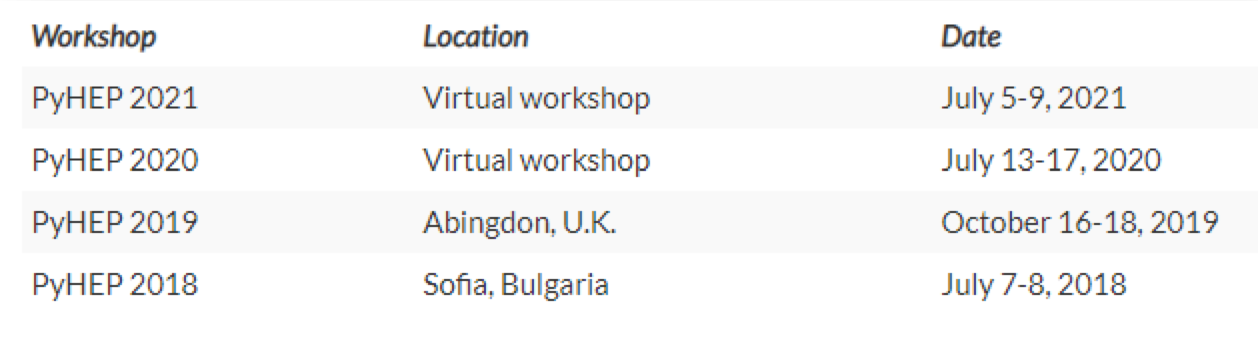


The **PyHEP workshops** are a series of workshops initiated and supported by the HEP Software Foundation (HSF) with the aim to provide an environment to discuss and promote the usage of Python in the HEP community at large. Further information is given on the PyHEP WG website.

- ❑ **Workshop format – unchanged for in-person and virtual events:**
  - **- Only plenary sessions**
  - **- Very informal, lots of time for (lively) discussions**
  - **- Bring together users and developers**
  - **- Educative, not just informative**

PyHEP - Python in HEP (hepsoftwarefoundation.org)

- ❑ **In-person seemed the adequate format but we had to run online in 2020 …**
  … and we learned from this experience …



Workshop	*Location*	*Date*
PyHEP 2021	Virtual workshop	July 5-9, 2021
PyHEP 2020	Virtual workshop	July 13-17, 2020
PyHEP 2019	Abingdon, U.K.	October 16-18, 2019
PyHEP 2018	Sofia, Bulgaria	July 7-8, 2018

PyHEP 2020

Virtual, online

PyHEP workshops – topics & content type

- ❑ Timely topics following trends – what is hot or new “at the moment”
- ❑ 2020 example:

- ○ Analysis fundamentals
- ○ Analysis platforms & systems
- ○ Automatic differentiation
- ○ Performance
- ○ Fitting & statistics
- ○ HEP analysis ecosystem

+

2 keynote presentations (astronomy & pheno.)

- ❑ Several types of presentations:
  - Tutorials, typically 1h
  - Standard talks of 20 + 10 minutes
  - Keynote presentations

Word cloud of PyHEP 2020 abstracts

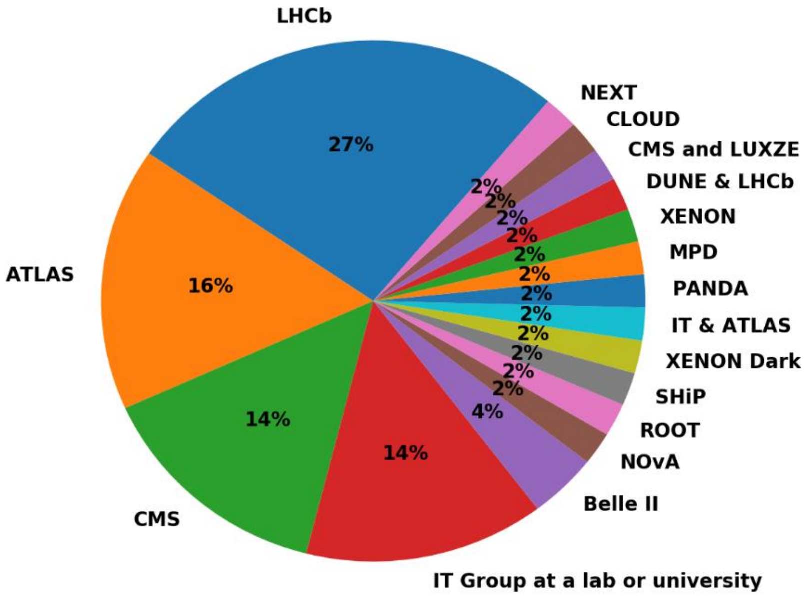
analysis
data
will
Notebook
ROOT
talk
physics
coffea
CMS
CERN
can
PyROOT
TensorFlow
training
library
libraries
using
new
detector
statistical
HL-LHC
workflow
resources
fit
iminuit

(Made with https://www.wordclouds.com/
removing author names, institutes and some other trivial words.)

PyHEP workshops – diversity & inclusion

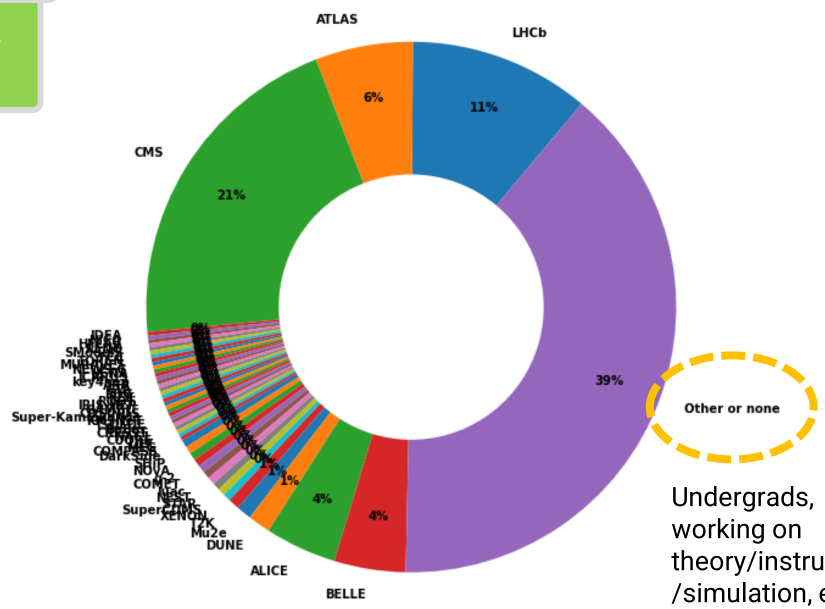
- ❑ **Always a goal to foster diversity & inclusion!**
  - We had a code of conduct from the onset (expanded & improved significantly for PyHEP 2021)
- ❑ **Two aspects:**
  - Communities participating – Energy & intensity frontiers, neutrinos, astroparticles, theorists, etc.
  - Cultural backgrounds, gender, ethnicity, disability, sexual orientation, etc.

PyHEP 2018
Sofia, Bulgaria



Very many more students for an online event!

PyHEP 2020
Virtual event



Undergrads, working on theory/instrumentation /simulation, etc.

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

Community projects towards a HEP Python ecosystem for data analysis

❑ Citing Gordon Watts (ACAT 2019) – how can we tackle the following 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 ;-))

❑ PyHEP WG serves as a forum for discussion, means to exchange experiences and material

❑ Our workshops present many of these packages and provide educative material

⇒ *strong link with Training WG ☺*

Various projects have seen the light:

❑ Coffea

❑ FAST-HEP

❑ Scikit-HEP (1st one of the gang)

❑ zfit

❑ https://github.com/CoffeaTeam

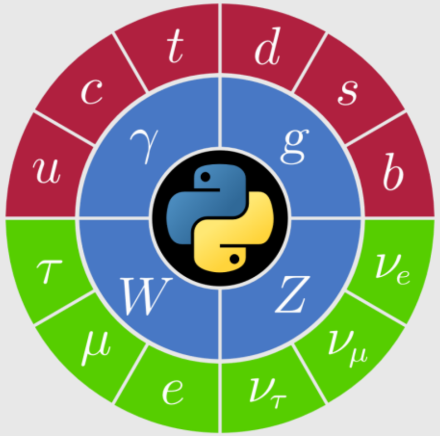
❑ https://github.com/FAST-HEP

❑ https://github.com/root-project/

❑ https://scikit-hep.org/

❑ https://github.com/zfit

PyHEP 2020 Workshop



We now even have a logo ☺ !

❑ A special cuvée – first online PyHEP

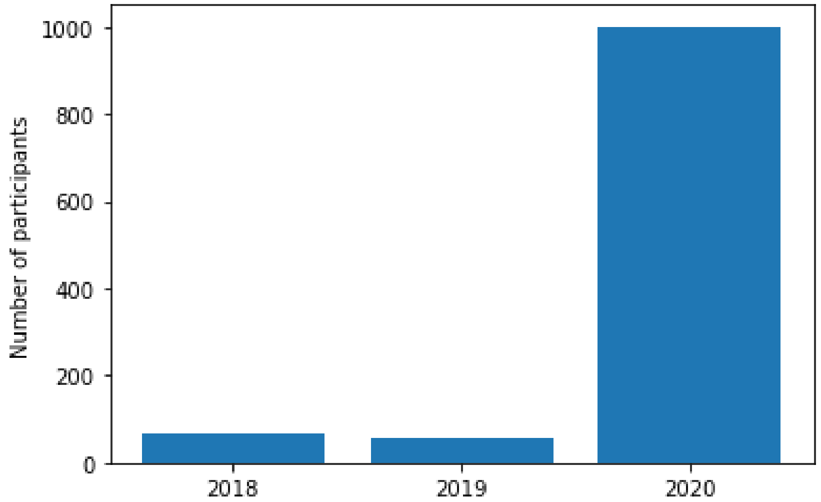
❑ Quick word on organisational aspects

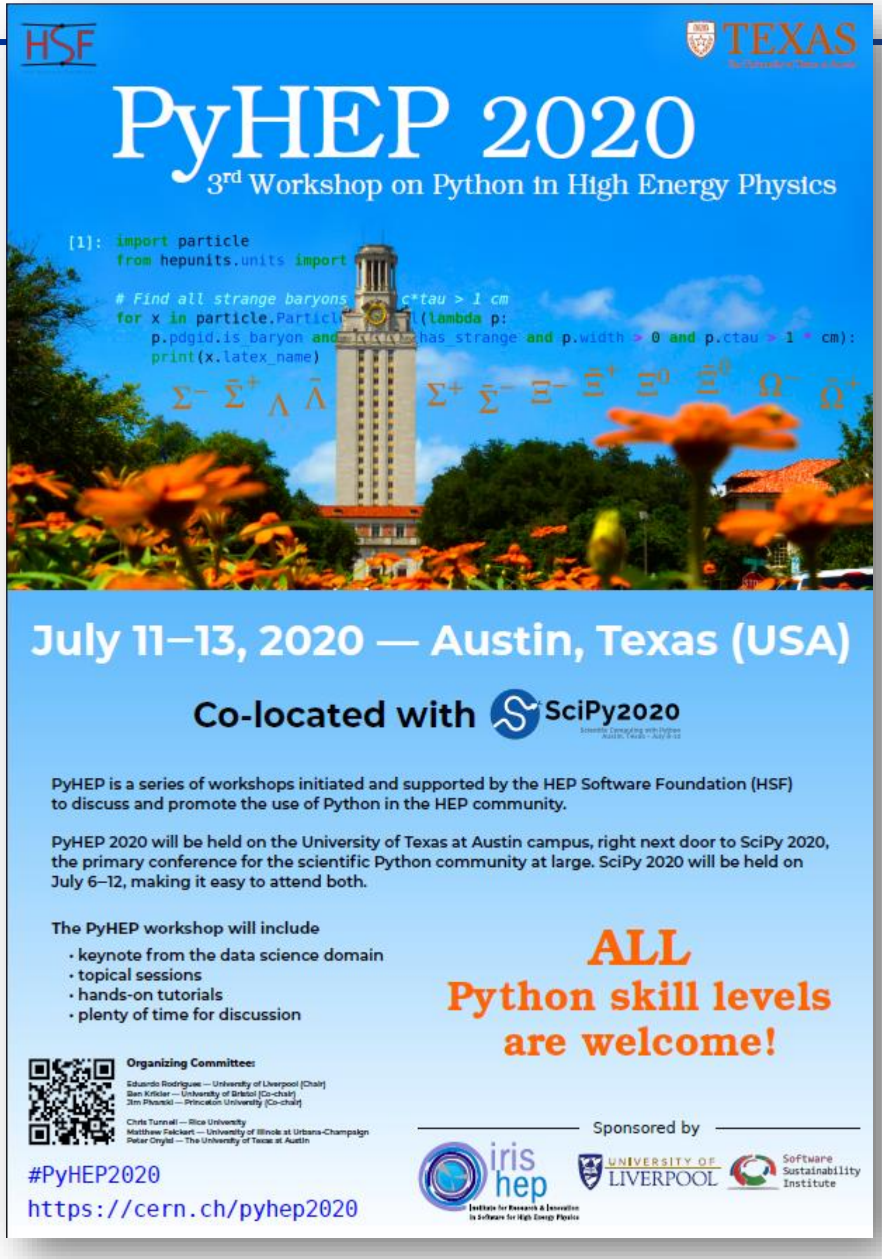
❑ Looking forward towards PyHEP 2021

❑ BTW, *a lot more information* in the back-up slides …

PyHEP 2020, a special cuvée

- ❑ 3rd edition was meant to be in the US for the first time, co-locating with the important SciPy 2020 conference
  - We even had a nice poster ;-)!
- ❑ We engaged with this very large scientific community
  - Had several talks from HEP colleagues @ SciPy 2020
- ❑ But we both had to materialise as a virtual event given the worldwide situation with COVID-19
- ❑ Truly global event with participants from all over the world (benefit from running virtual)
  - Impressive level of interest with 1000 registrations (limited to) (72, 55 in previous years)





PyHEP 2020

3rd Workshop on Python in High Energy Physics

```
[1]: import particle
     from hepunits.units import
     
     # Find all strange baryons            c*tau > 1 cm
     for x in particle.Particl          l(lambda p:
         p.pdgid.is_baryon and          has_strange and p.width > 0 and p.ctau > 1 * cm):
         print(x.latex_name)
```

$\Sigma^- \quad \bar{\Sigma}^+ \quad \Lambda \quad \bar{\Lambda} \quad \Sigma^+ \quad \bar{\Sigma}^- \quad \Xi^- \quad \bar{\Xi}^+ \quad \Xi^0 \quad \bar{\Xi}^0 \quad \Omega^- \quad \bar{\Omega}^+$

July 11–13, 2020 — Austin, Texas (USA)

Co-located with SciPy2020

PyHEP is a series of workshops initiated and supported by the HEP Software Foundation (HSF) to discuss and promote the use of Python in the HEP community.

PyHEP 2020 will be held on the University of Texas at Austin campus, right next door to SciPy 2020, the primary conference for the scientific Python community at large. SciPy 2020 will be held on July 6–12, making it easy to attend both.

The PyHEP workshop will include

- keynote from the data science domain
- topical sessions
- hands-on tutorials
- plenty of time for discussion

ALL Python skill levels are welcome!

Organizing Committee:

Eduardo Rodrigues — University of Liverpool (Chair)
Ben Krikler — University of Bristol (Co-chair)
Jim Pivarski — Princeton University (Co-chair)

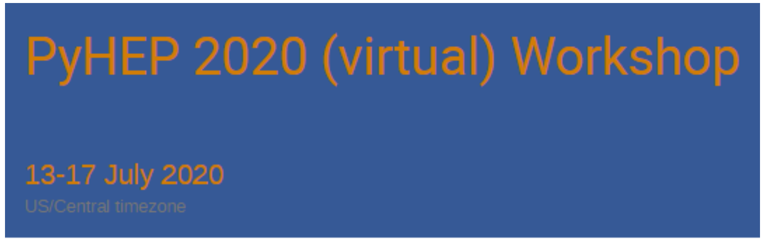
Chris Tunnell — Rice University
Matthew Feickert — University of Illinois at Urbana-Champaign
Peter Onyisi — The University of Texas at Austin

Sponsored by

iris hep
UNIVERSITY OF LIVERPOOL
Software Sustainability Institute

#PyHEP2020
https://cern.ch/pyhep2020

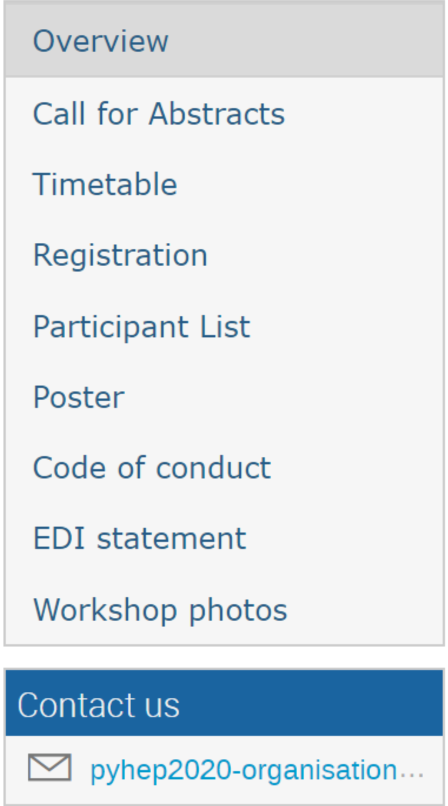
PyHEP 2020 – Indico page, organising team, sponsors



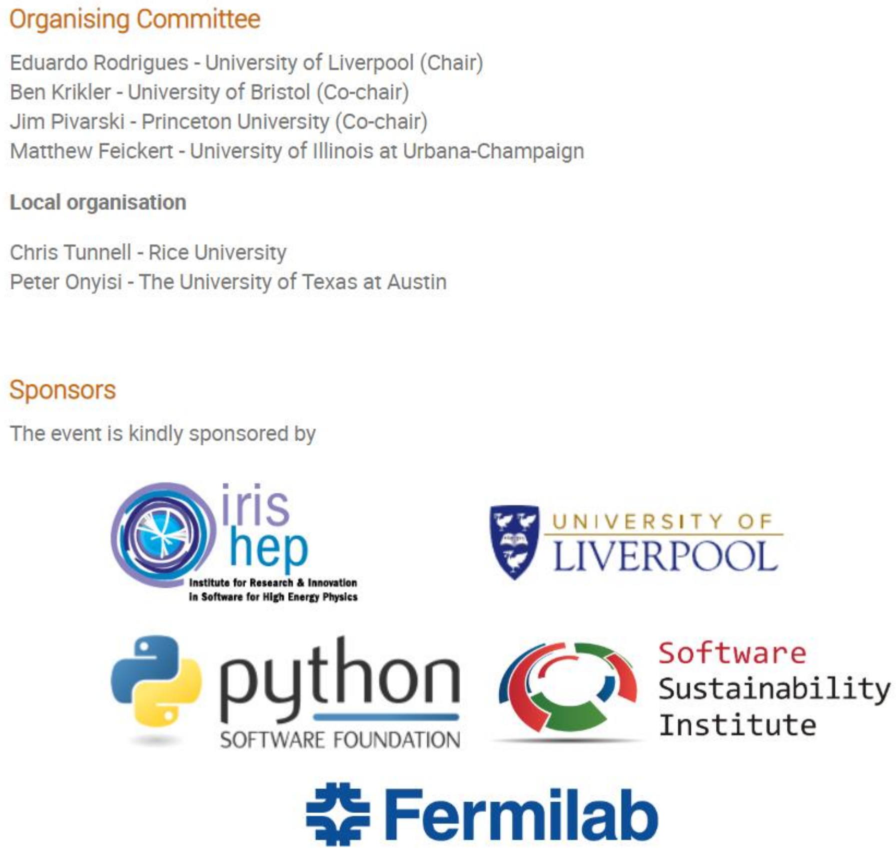
PyHEP 2020 (virtual) Workshop

13-17 July 2020

US/Central timezone



https://indico.cern.ch/e/PyHEP2020



Organising Committee

Eduardo Rodrigues - University of Liverpool (Chair)
Ben Krikler - University of Bristol (Co-chair)
Jim Pivarski - Princeton University (Co-chair)
Matthew Feickert - University of Illinois at Urbana-Champaign

Local organisation

Chris Tunnell - Rice University
Peter Onyisi - The University of Texas at Austin

Sponsors

The event is kindly sponsored by

iris hep
Institute for Research & Innovation in Software for High Energy Physics
UNIVERSITY OF LIVERPOOL
python SOFTWARE FOUNDATION
Software Sustainability Institute
Fermilab

- Great list of kind sponsors is a proof of workshops being relevant and attracting attention – my personal opinion ;-)

PyHEP 2020 – organisational aspects overview

- ❑ Sessions & presentations
  - Spread in sessions for “Atlantic”- and “Pacific”-friendly time zones
  - We strongly encouraged notebook presentations, available in public Github repositories with a Binder launch button
  - All presentational material posted on workshop agenda and later given a DOI with Zenodo, in a dedicated “pyhep2020 community” – formal citation, replaces proceedings
  - All talks got recorded, captioned and later uploaded to the HSF YouTube channel – dedicated playlist “PyHEP 2020 Workshop”

- ❑ Zoom video conferencing system
  - With capacity for 1000 participants
  - Public room but PIN provided via email

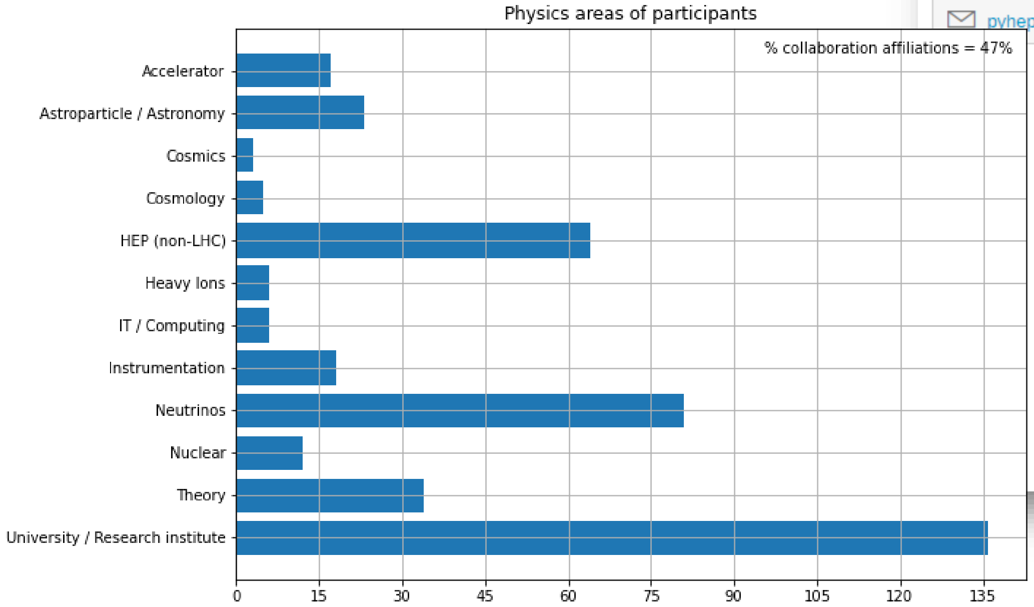
- ❑ Slack channels
  - Various channels:
    - By topic, mapping to sessions, discussions encouraged here
    - Announcements, for actual announcements
    - Random, used to encourage community spirit and add social context

- ❑ Questions & answers with slido
  - Used *slido* to crowd-source questions, to prioritise the most popular ones upvoted by participants
  - Session chair shares link to questions at end of presentation
  - Most popular ones get answered/discussed
  - At end of Q&A all questions are copied to Slack in the appropriate topical channel
    ⇒ participants can continue to discuss and exchange
  - A few polls also run via slido

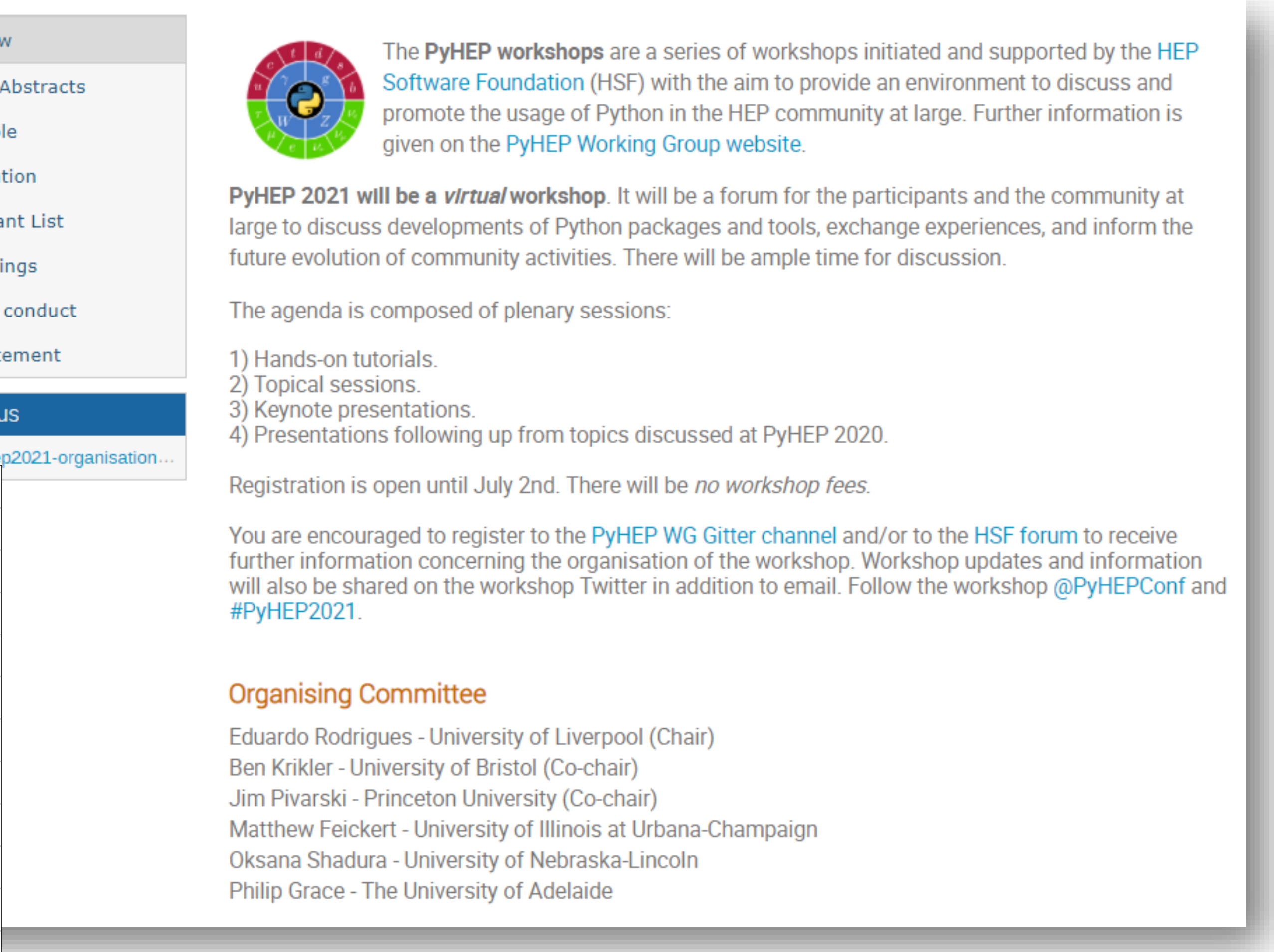
- ❑ Communication also on

PyHEP 2021 … in less than 2 months ☺ !

- ❑ Taking place online on July 5-9
- ❑ Format largely unchanged since 2020
  - But no sessions in 2 time zones
  - Plan to live stream to YouTube (atop Zoom) to avoid any limit
- ❑ Again seeing much interest (>900 reg.)
- ❑ Seeing new communities joining, in particular the neutrino one!
- ❑ ~63% participants so far are from LHC:



https://indico.cern.ch/e/PyHEP2021



The **PyHEP workshops** are a series of workshops initiated and supported by the HEP Software Foundation (HSF) with the aim to provide an environment to discuss and promote the usage of Python in the HEP community at large. Further information is given on the PyHEP Working Group website.

PyHEP 2021 will be a *virtual* workshop. It will be a forum for the participants and the community at large to discuss developments of Python packages and tools, exchange experiences, and inform the future evolution of community activities. There will be ample time for discussion.

The agenda is composed of plenary sessions:

1) Hands-on tutorials.
2) Topical sessions.
3) Keynote presentations.
4) Presentations following up from topics discussed at PyHEP 2020.

Registration is open until July 2nd. There will be *no workshop fees*.

You are encouraged to register to the PyHEP WG Gitter channel and/or to the HSF forum to receive further information concerning the organisation of the workshop. Workshop updates and information will also be shared on the workshop Twitter in addition to email. Follow the workshop @PyHEPConf and #PyHEP2021.

Organising Committee

Eduardo Rodrigues - University of Liverpool (Chair)
Ben Krikler - University of Bristol (Co-chair)
Jim Pivarski - Princeton University (Co-chair)
Matthew Feickert - University of Illinois at Urbana-Champaign
Oksana Shadura - University of Nebraska-Lincoln
Philip Grace - The University of Adelaide

The Scikit-HEP project

❑ **Motivation for such a community project**

❑ **Whirlwind tour of packages**

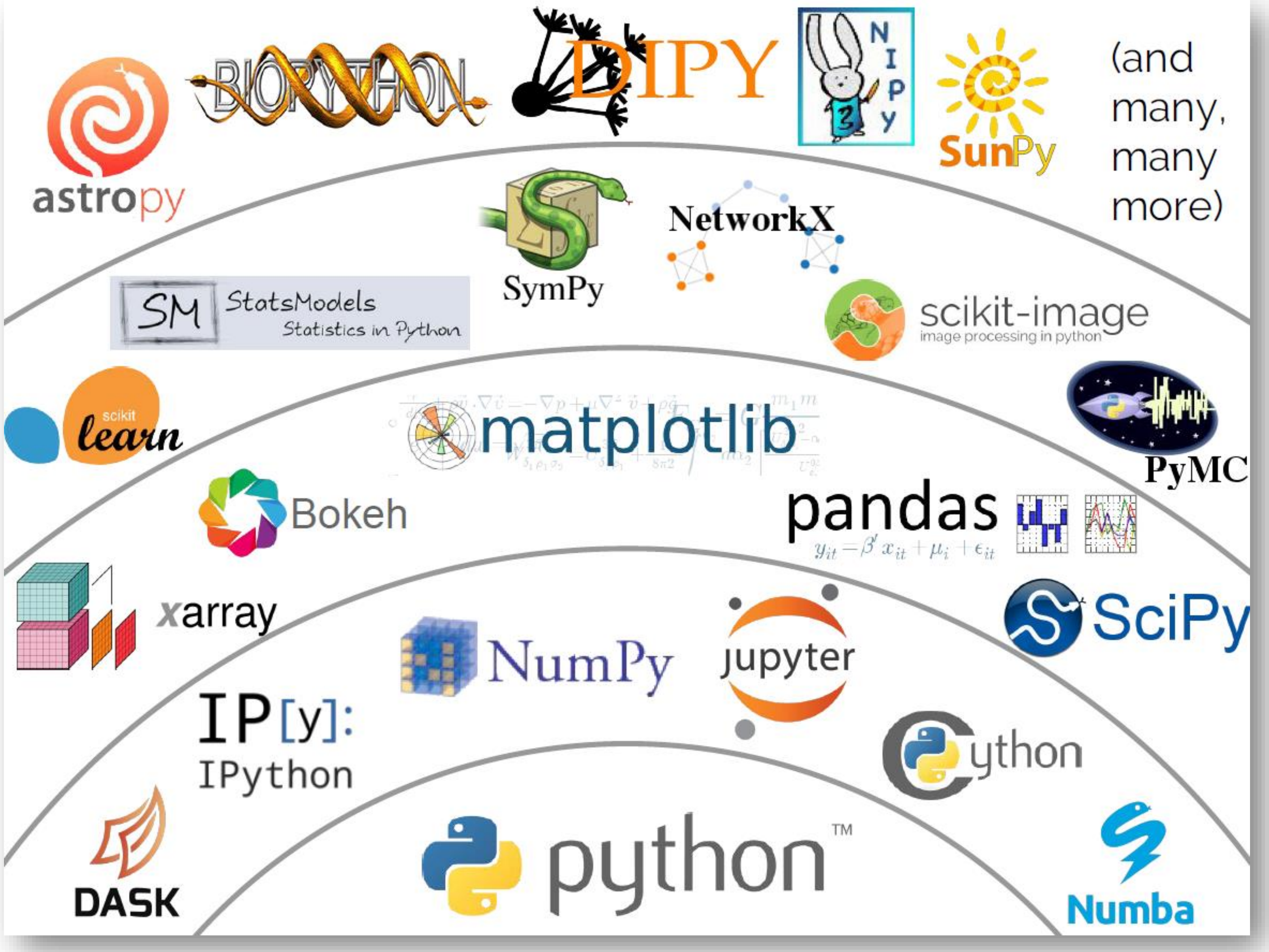
How’s the Python scientific ecosystem like, outside HEP?

Domain-specific

Python’s

Scientific

stack



What about HEP …?

Community projects towards HEP domain-specific Python tools ⇒ ecosystem

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
- Open forum to discuss

□ Effort to improve discoverability of (domain-specific) relevant tools

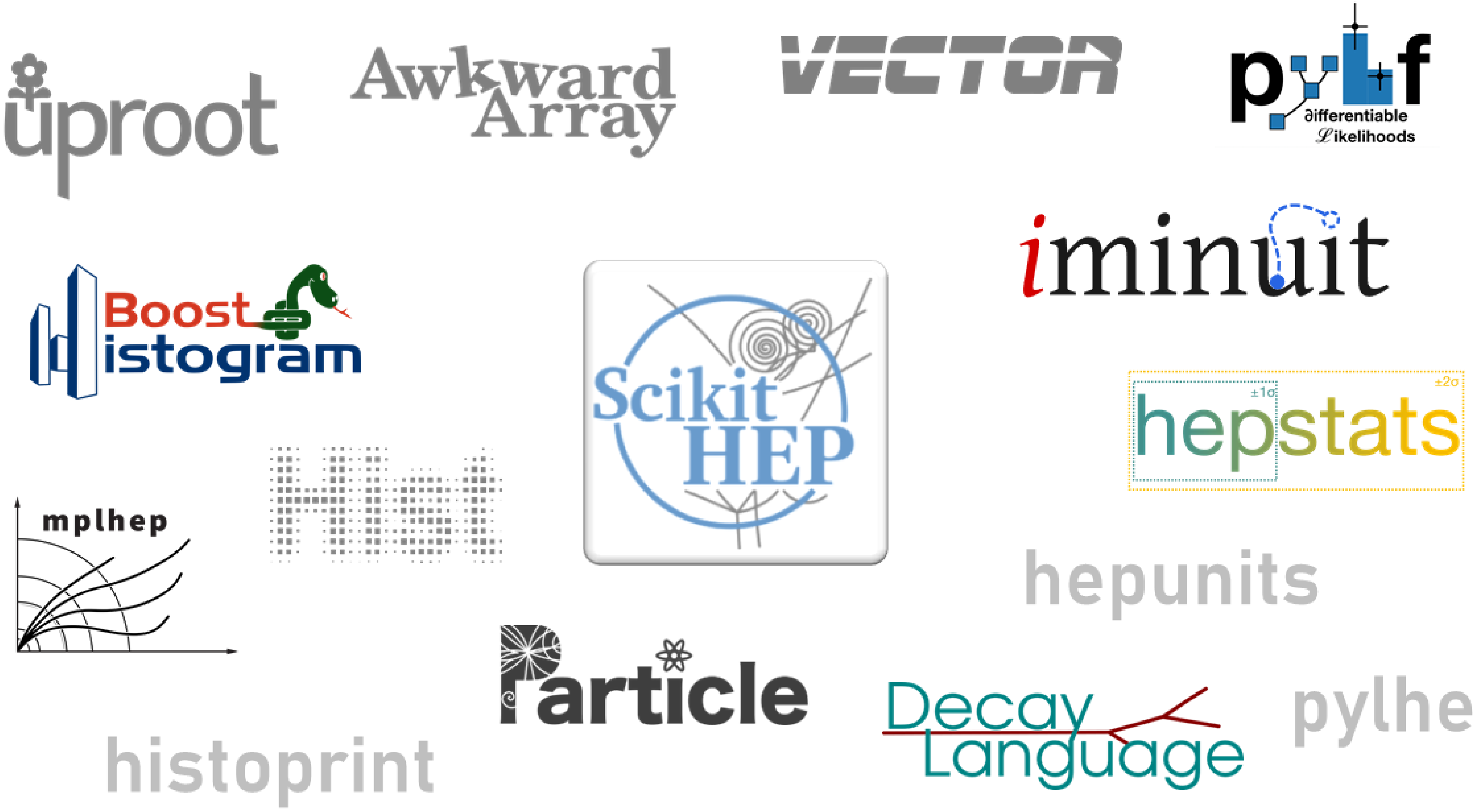
Collaboration

Reproducibility

Interoperability

Sustainability

Scikit-HEP project – overview of most popular and/or used packages

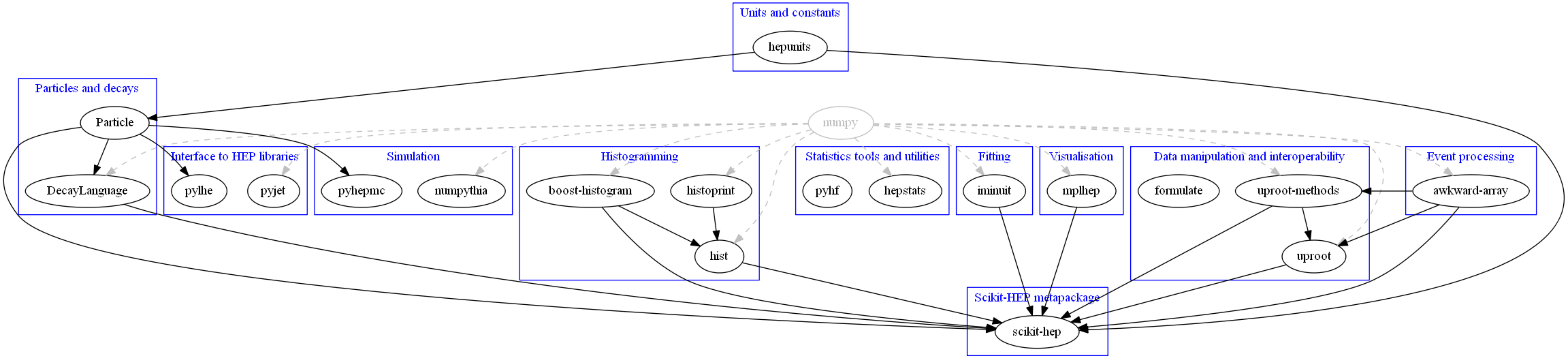


[Not the full set of Scikit-HEP packages.]

Scikit-HEP project – packages and dependencies

- Pattern of inter-package dependencies nicely "explains" why the project is a *toolset* and not a toolkit !

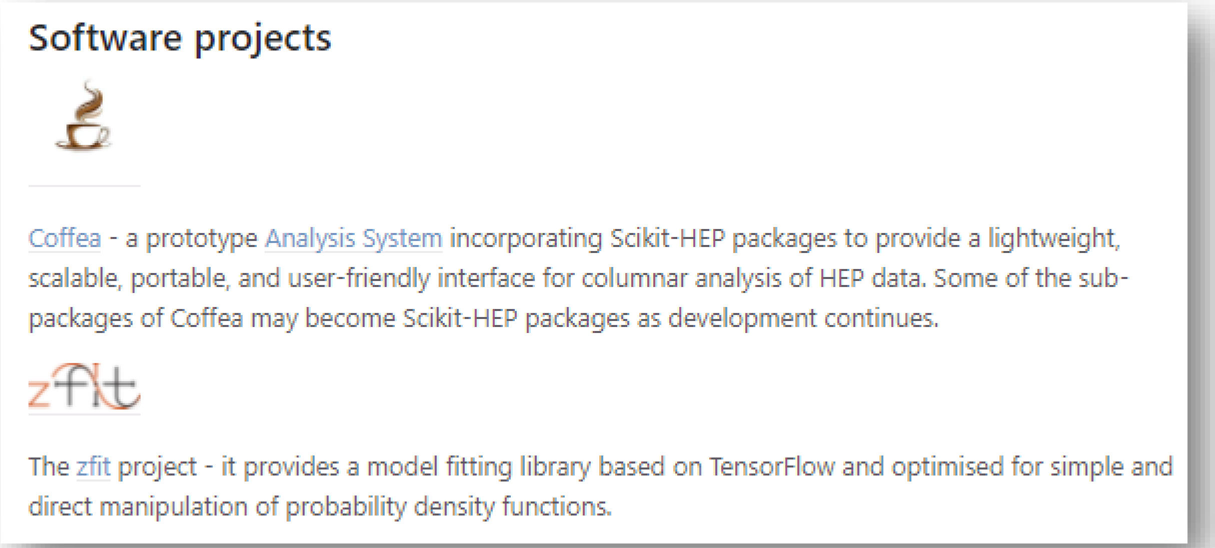
https://scikit-hep.org/



Not a comprehensive list. Needs updating. 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 ?

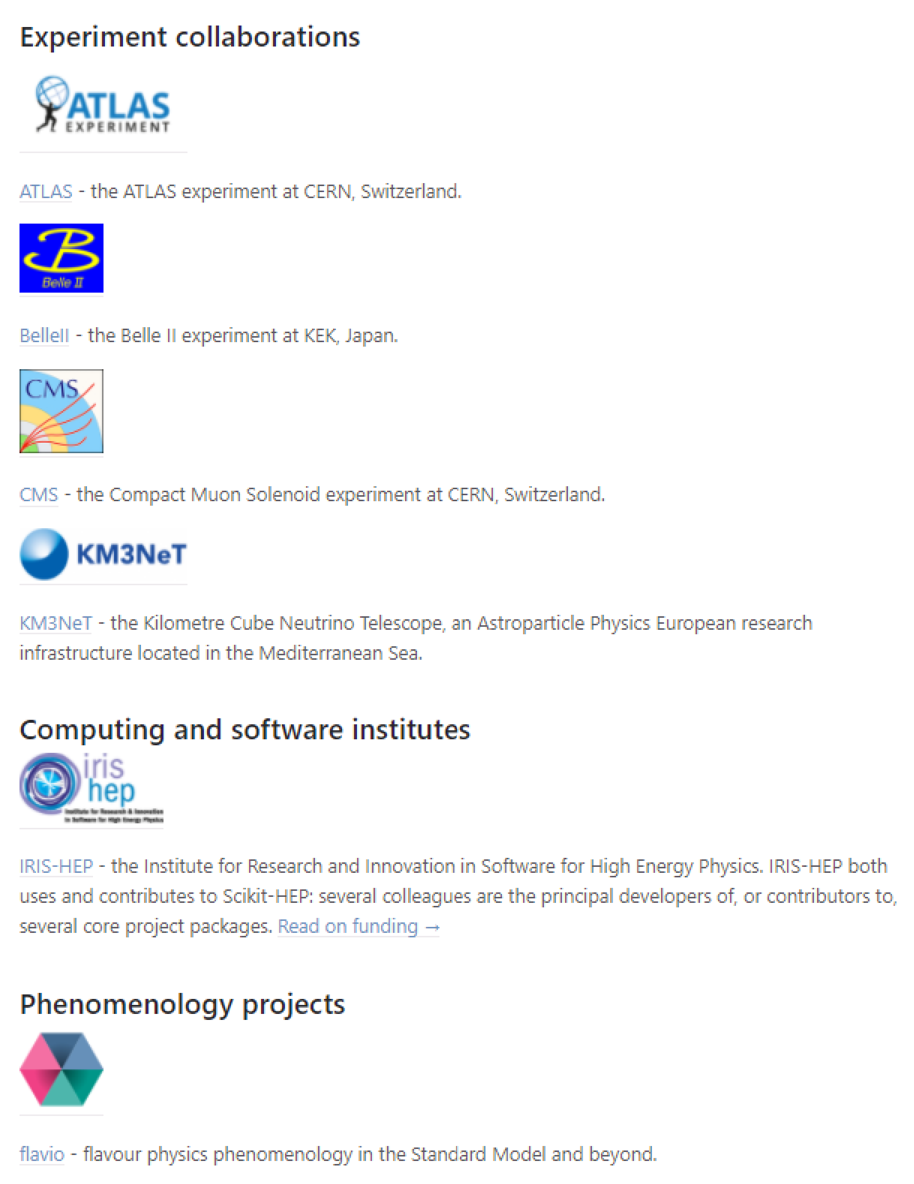
- ❑ **HEP experiments, other projects, groups**
- ❑ **Links are important, especially if they strengthen the overall ecosystem**
- ❑ **Good community adoption ⇔ we're on the right path ;-)**
- ❑ **Rewarding to collaborate / work with / interact with many communities**
  - **- Responsibility and importance 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

ATLAS - the ATLAS experiment at CERN, Switzerland.

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.

Computing and software institutes

IRIS-HEP - the Institute for Research and Innovation in Software for High Energy Physics. IRIS-HEP both uses and contributes to Scikit-HEP: several colleagues are the principal developers of, or contributors to, several core project packages. Read on funding →

Phenomenology projects

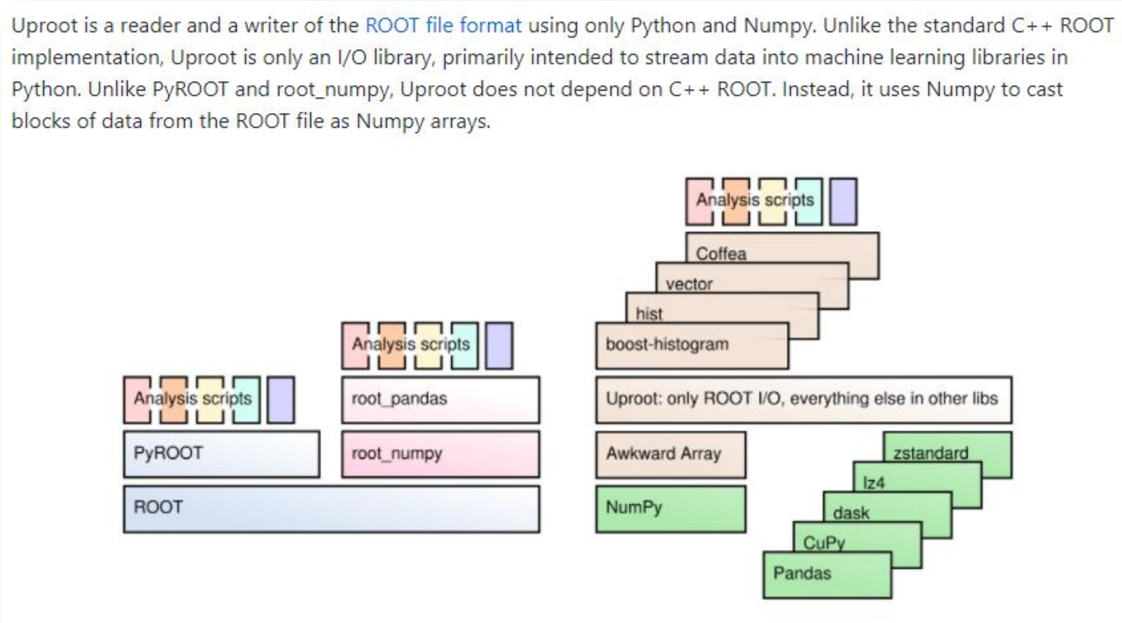
flavio - flavour physics phenomenology in the Standard Model and beyond.

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 and minimalistic I/O library!
- ❑ Primarily intended to stream data into machine learning libraries in Python

uproot

ROOT I/O
in pure Python and Numpy



Uproot is a reader and a writer of the ROOT file format using only Python and Numpy. Unlike the standard C++ ROOT implementation, Uproot is only an I/O library, primarily intended to stream data into machine learning libraries in Python. Unlike PyROOT and root_numpy, Uproot does not depend on C++ ROOT. Instead, it uses Numpy to cast blocks of data from the ROOT file as Numpy arrays.

- ❑ Note that uproot3 and uproot4 still coexist. The former provides some writing functionality but the latter is a major rework, together with awkward-array

Event processing – `awkward-array` package

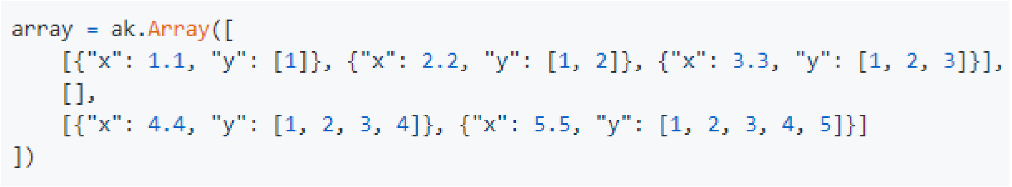
- ❑ Provide a way to analyse nested, variable-sized data in Python, by extending NumPy's idioms from flat arrays to arrays of data structures

- ❑ 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.

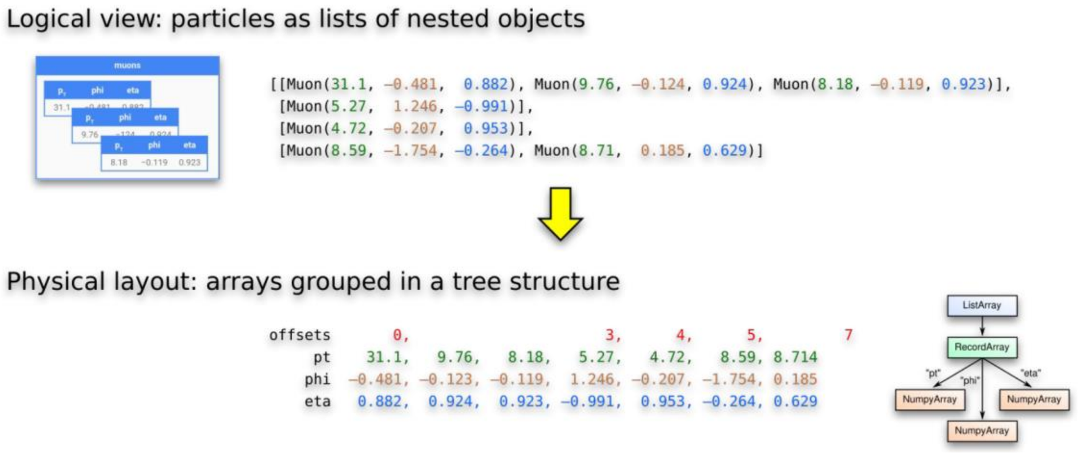
- ❑ This is all very relevant and important for HEP applications !

Awkward Array

Manipulate arrays of complex data structures as easily as NumPy



```
array = ak.Array([
    [{"x": 1.1, "y": [1]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2, 3]}],
    [],
    [{"x": 4.4, "y": [1, 2, 3, 4]}, {"x": 5.5, "y": [1, 2, 3, 4, 5]}]
])
```



Logical view: particles as lists of nested objects

```
[[Muon(31.1, –0.481,  0.882), Muon(9.76, –0.124, 0.924), Muon(8.18, –0.119, 0.923)],
 [Muon(5.27,  1.246, –0.991)],
 [Muon(4.72, –0.207,  0.953)],
 [Muon(8.59, –1.754, –0.264), Muon(8.71,  0.185, 0.629)]
```

Physical layout: arrays grouped in a tree structure

```
offsets      0,                         3,      4,      5,         7
     pt   31.1,   9.76,   8.18,   5.27,   4.72,   8.59, 8.714
    phi –0.481, –0.123, –0.119,  1.246, –0.207, –1.754, 0.185
    eta  0.882,  0.924,  0.923, –0.991,  0.953, –0.264, 0.629
```

- ❑ See https://github.com/scikit-hep/awkward-1.0

P.S.: Uproot and awkward would need a talk on their own ! Go and explore … ☺

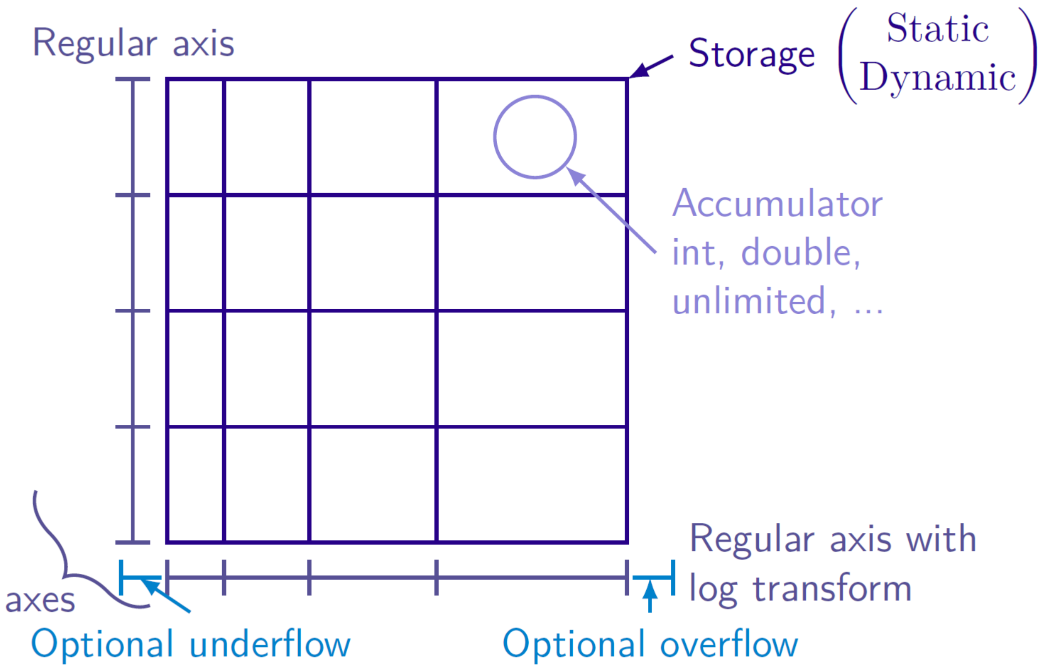
Histogramming – `boost-histogram` package

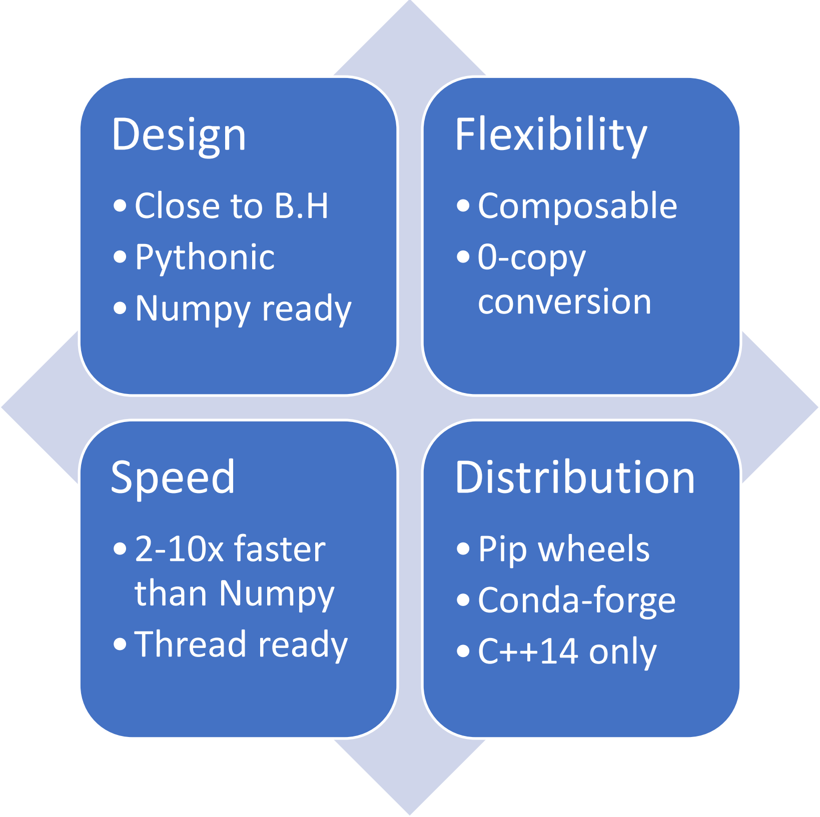
- ❑ (pybind11) Python bindings for the C++14 Boost.Histogram library

  (multi-dimensional templated header-only, designed by Hans Dembinski)

- ❑ A histogram is seen as collection of Axis objects and a storage

  - Several types available, e.g. regular, circular, category





Design
- Close to B.H
- Pythonic
- Numpy ready

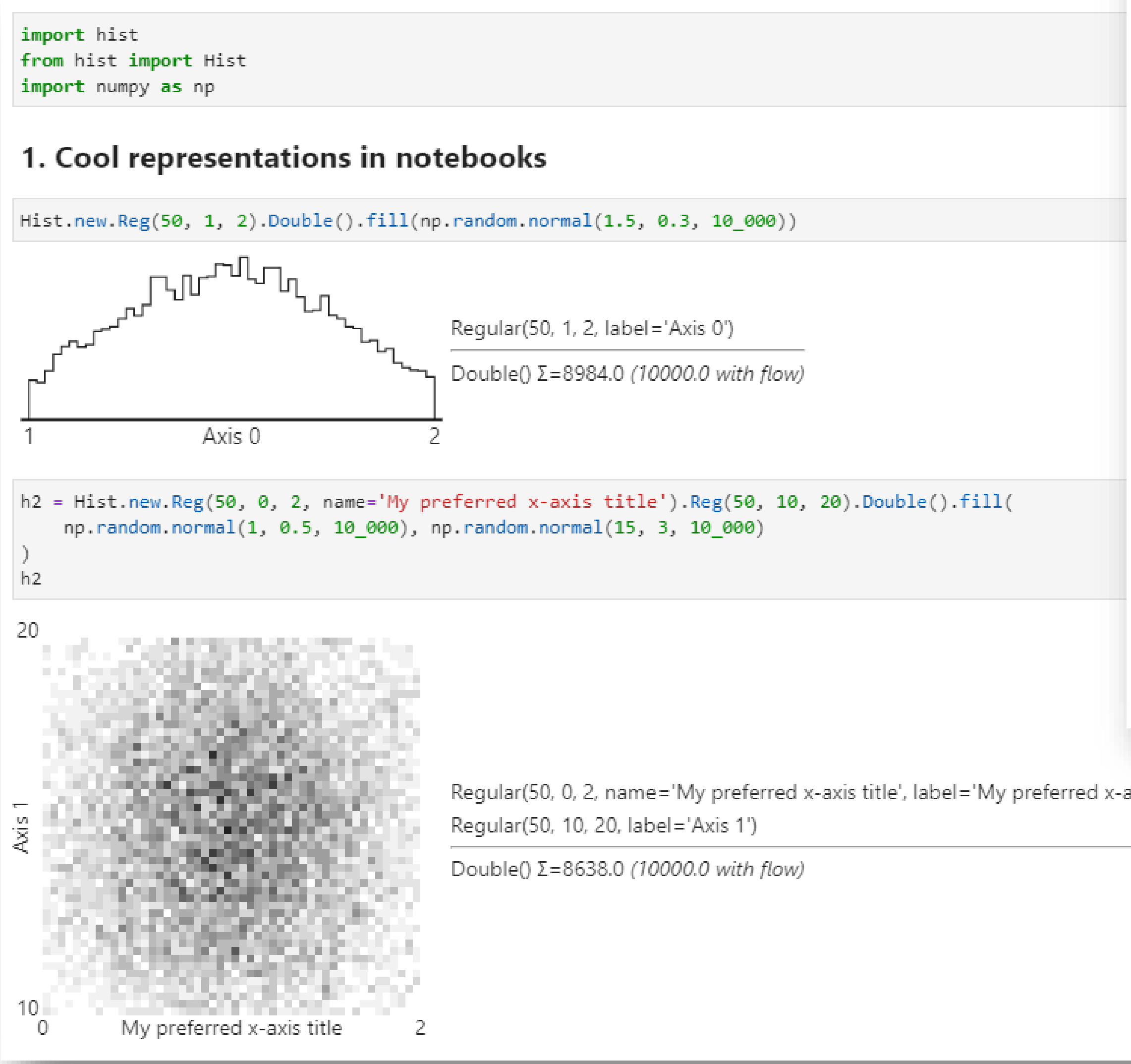
Flexibility
- Composable
- 0-copy conversion

Speed
- 2-10x faster than Numpy
- Thread ready

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

Histogramming & visualisation – build atop `boost-histogram`

❑ Trivial creation and display in notebooks !



```
import hist
from hist import Hist
import numpy as np
```

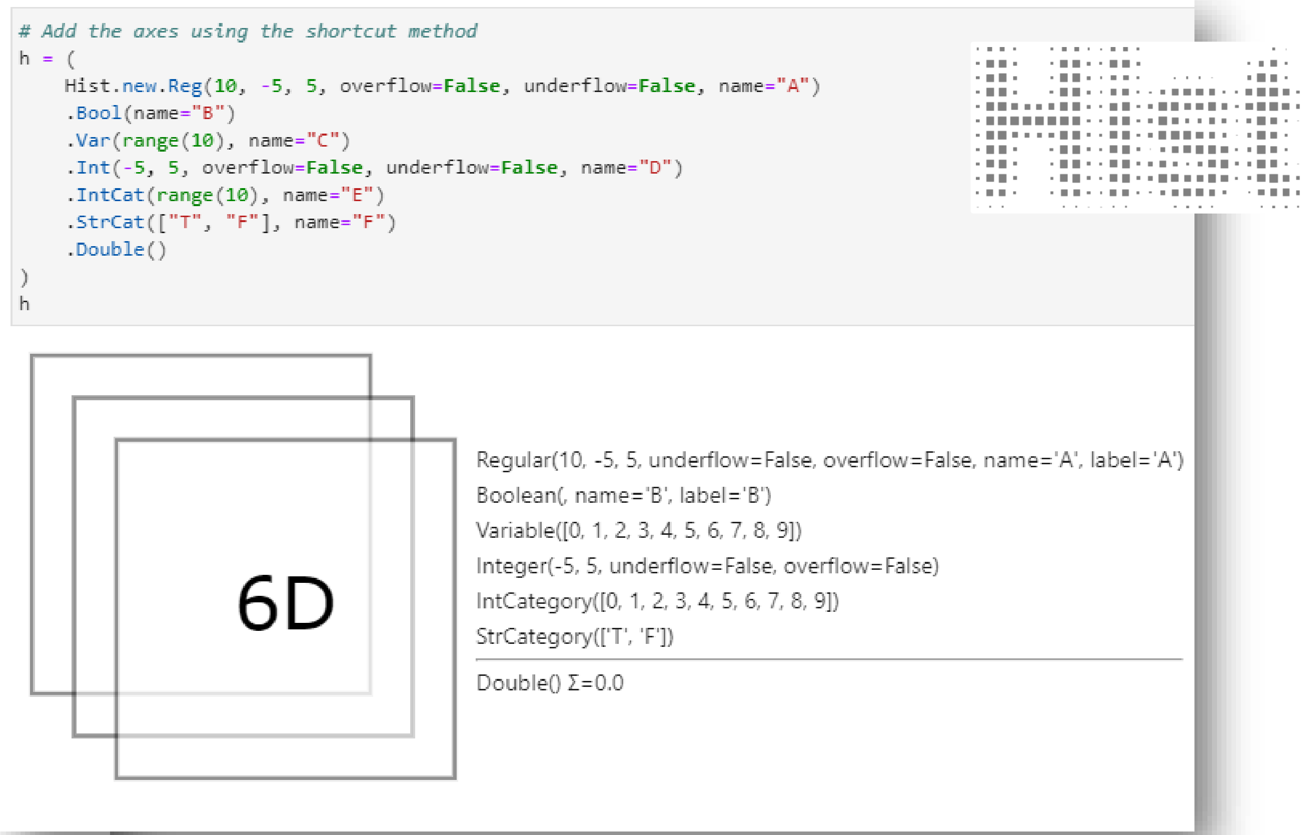
1. Cool representations in notebooks

```
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=8984.0 (10000.0 with flow)
```

```
h2 = Hist.new.Reg(50, 0, 2, name='My preferred x-axis title').Reg(50, 10, 20).Double().fill(
    np.random.normal(1, 0.5, 10_000), np.random.normal(15, 3, 10_000)
)
h2
```

```
Regular(50, 0, 2, name='My preferred x-axis title', label='My preferred x-axis title')
Regular(50, 10, 20, label='Axis 1')
Double() Σ=8638.0 (10000.0 with flow)
```



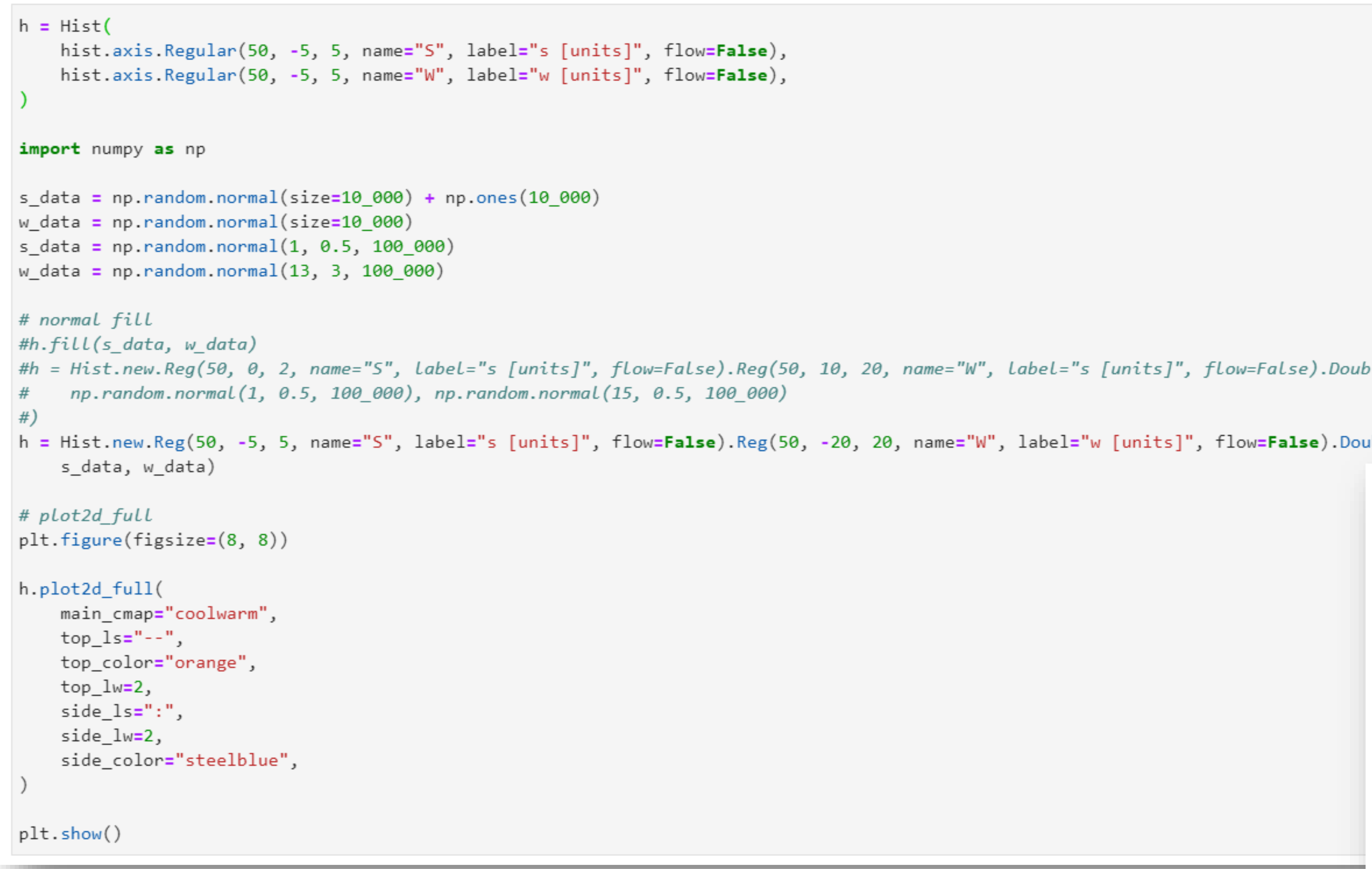
```
# Add the axes using the shortcut method
h = (
    Hist.new.Reg(10, -5, 5, overflow=False, underflow=False, name="A")
    .Bool(name="B")
    .Var(range(10), name="C")
    .Int(-5, 5, overflow=False, underflow=False, name="D")
    .IntCat(range(10), name="E")
    .StrCat(["T", "F"], name="F")
    .Double()
)
h
```

```
Regular(10, -5, 5, underflow=False, overflow=False, name='A', label='A')
Boolean(, name='B', label='B')
Variable([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
Integer(-5, 5, underflow=False, overflow=False)
IntCategory([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
StrCategory(['T', 'F'])
Double() Σ=0.0
```

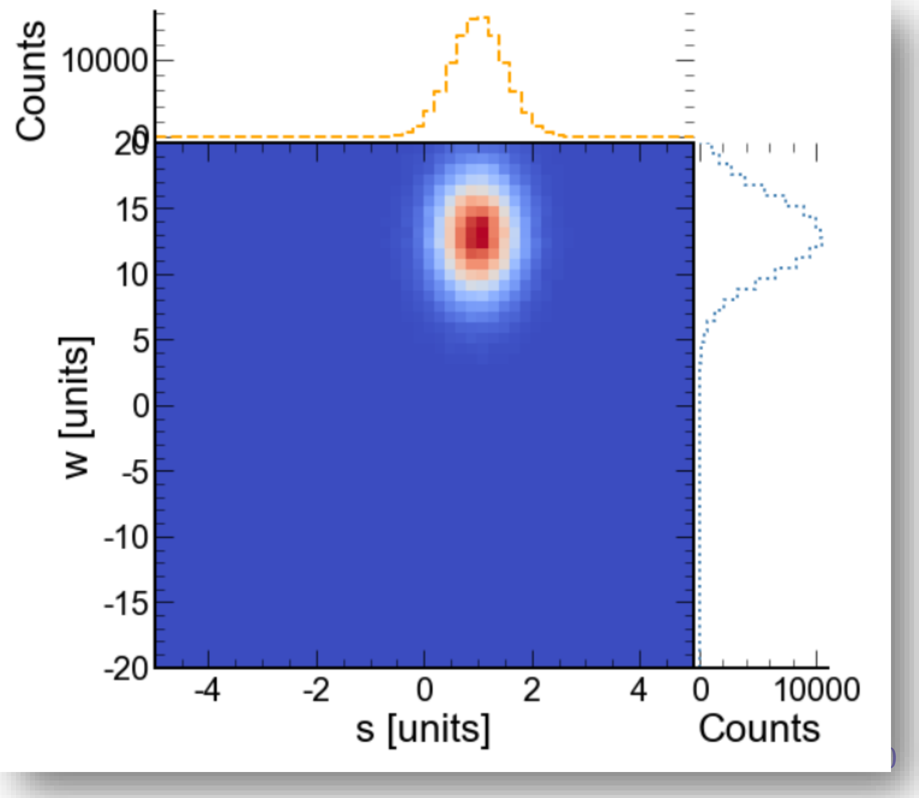
❑ Several types of axis available

❑ Lots of developments recently, in particular for better interoperability

Histogramming & visualisation – build atop `boost-histogram`



```python
h = Hist(
    hist.axis.Regular(50, -5, 5, name="S", label="s [units]", flow=False),
    hist.axis.Regular(50, -5, 5, name="W", label="w [units]", flow=False),
)

import numpy as np

s_data = np.random.normal(size=10_000) + np.ones(10_000)
w_data = np.random.normal(size=10_000)
s_data = np.random.normal(1, 0.5, 100_000)
w_data = np.random.normal(13, 3, 100_000)

# normal fill
#h.fill(s_data, w_data)
#h = Hist.new.Reg(50, 0, 2, name="S", label="s [units]", flow=False).Reg(50, 10, 20, name="W", label="s [units]", flow=False).Double().fill(
#    np.random.normal(1, 0.5, 100_000), np.random.normal(15, 0.5, 100_000)
#)
h = Hist.new.Reg(50, -5, 5, name="S", label="s [units]", flow=False).Reg(50, -20, 20, name="W", label="w [units]", flow=False).Double().fill(
    s_data, w_data)

# plot2d_full
plt.figure(figsize=(8, 8))

h.plot2d_full(
    main_cmap="coolwarm",
    top_ls="--",
    top_color="orange",
    top_lw=2,
    side_ls=":",
    side_lw=2,
    side_color="steelblue",
)

plt.show()
```

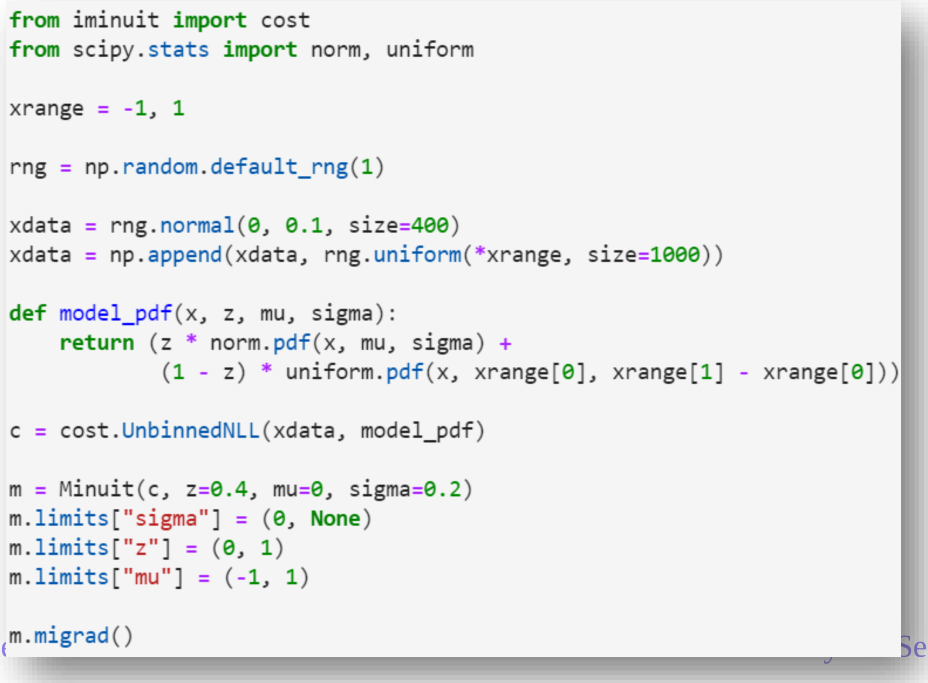


Fitting – `iminuit` package

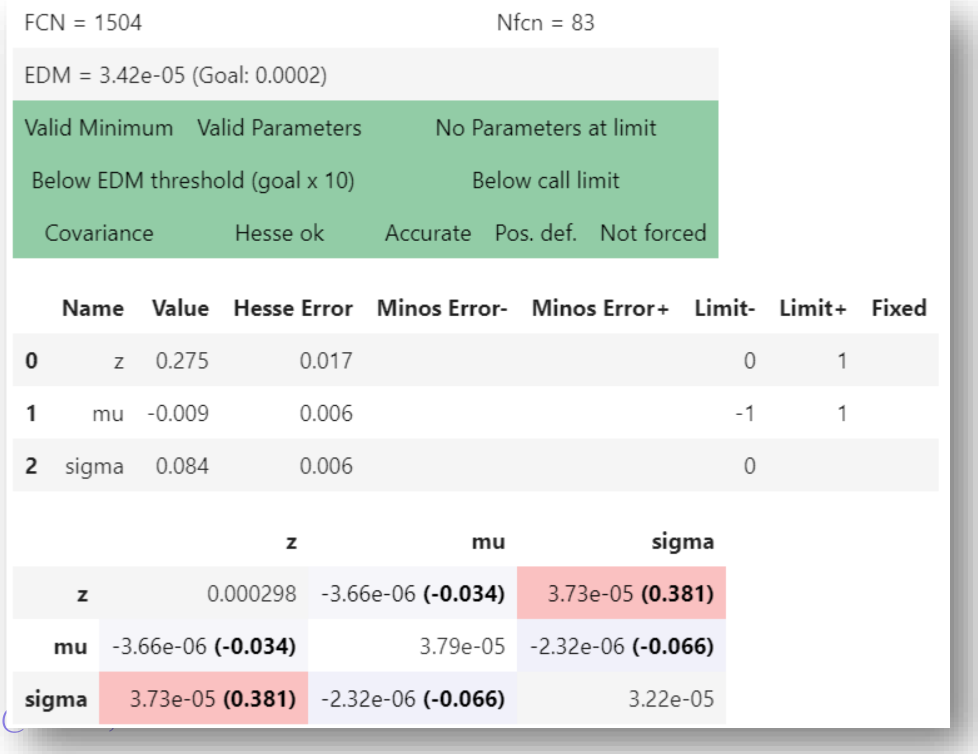
- ❑ **Provides Python interface to the MINUIT2 C++ package (built on Cython)**
  - **- Version 2.0 about to be out uses PyBind11 instead – much better**
- ❑ **Minimises arbitrary functions and computes standard errors**
  - **- Uses HESSE (inverse of Hesse matrix) or MINOS (profile likelihood method)**
- ❑ **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:**

iminuit

Python interface
to the Minuit2 C++ package



```python
from iminuit import cost
from scipy.stats import norm, uniform

xrange = -1, 1

rng = np.random.default_rng(1)

xdata = rng.normal(0, 0.1, size=400)
xdata = np.append(xdata, rng.uniform(*xrange, size=1000))

def model_pdf(x, z, mu, sigma):
    return (z * norm.pdf(x, mu, sigma) +
            (1 - z) * uniform.pdf(x, xrange[0], xrange[1] - xrange[0]))

c = cost.UnbinnedNLL(xdata, model_pdf)

m = Minuit(c, z=0.4, mu=0, sigma=0.2)
m.limits["sigma"] = (0, None)
m.limits["z"] = (0, 1)
m.limits["mu"] = (-1, 1)

m.migrad()
```



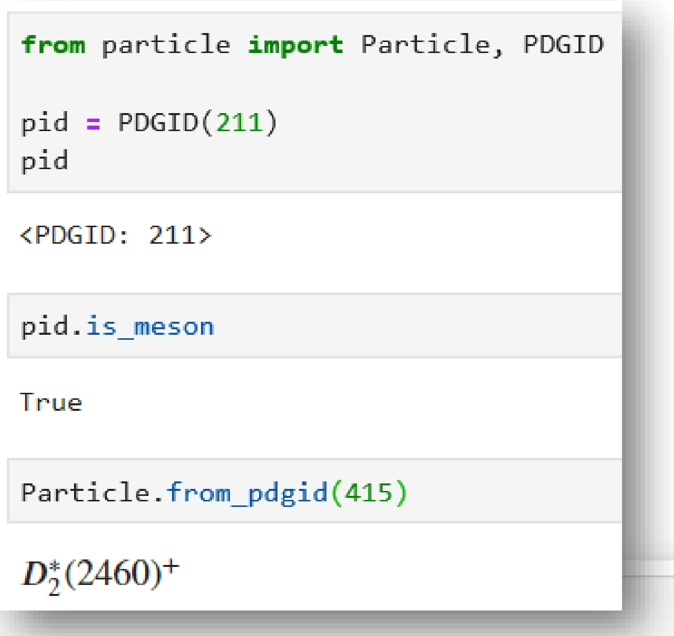
FCN = 1504		Nfcn = 83		
EDM = 3.42e-05 (Goal: 0.0002)				
Valid Minimum	Valid Parameters	No Parameters at limit		
Below EDM threshold (goal x 10)		Below call limit		
Covariance	Hesse ok	Accurate	Pos. def.	Not forced

	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed
0	z	0.275	0.017			0	1	
1	mu	-0.009	0.006			-1	1	
2	sigma	0.084	0.006			0		

	z	mu	sigma
z	0.000298	-3.66e-06 **(-0.034)**	3.73e-05 **(0.381)**
mu	-3.66e-06 **(-0.034)**	3.79e-05	-2.32e-06 **(-0.066)**
sigma	3.73e-05 **(0.381)**	-2.32e-06 **(-0.066)**	3.22e-05

Particles and decays – `Particle` package

- ❑ Pythonic interface to the Particle Data Group (PDG) particle data table and MC particle identification codes
- ❑ With many extra goodies
- ❑ Simple and natural APIs



```
from particle import Particle, PDGID

pid = PDGID(211)
pid
```

```
<PDGID: 211>
```

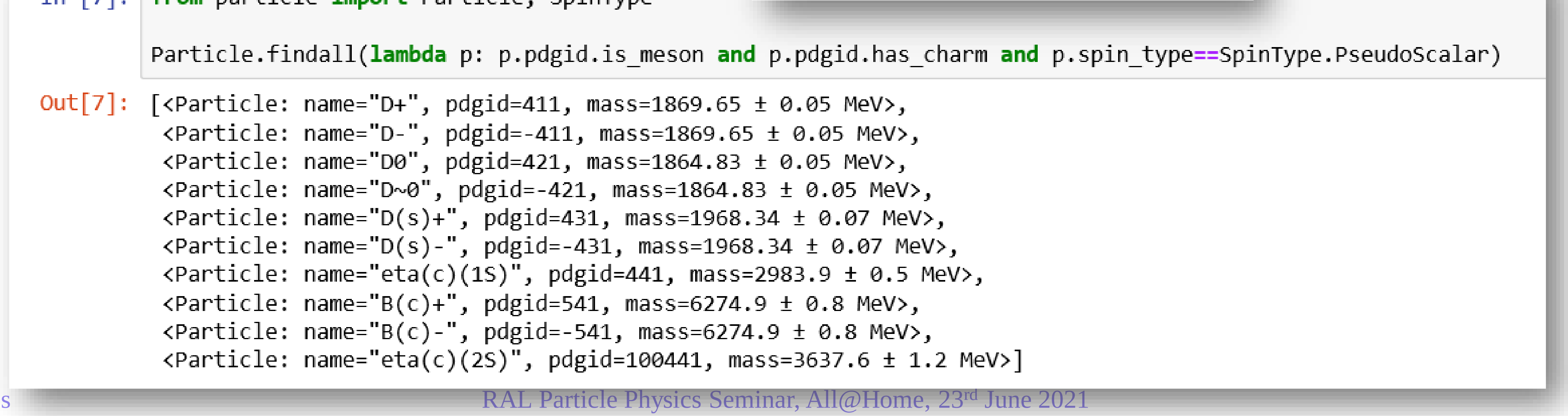
```
pid.is_meson
```

```
True
```

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

$D_2^*(2460)^+$

- ❑ Main classes for queries and look-ups:
  - Particle
  - `PDGID`
  - Command-line queries also available
- ❑ Powerful and flexible searches as 1-liners, e.g.

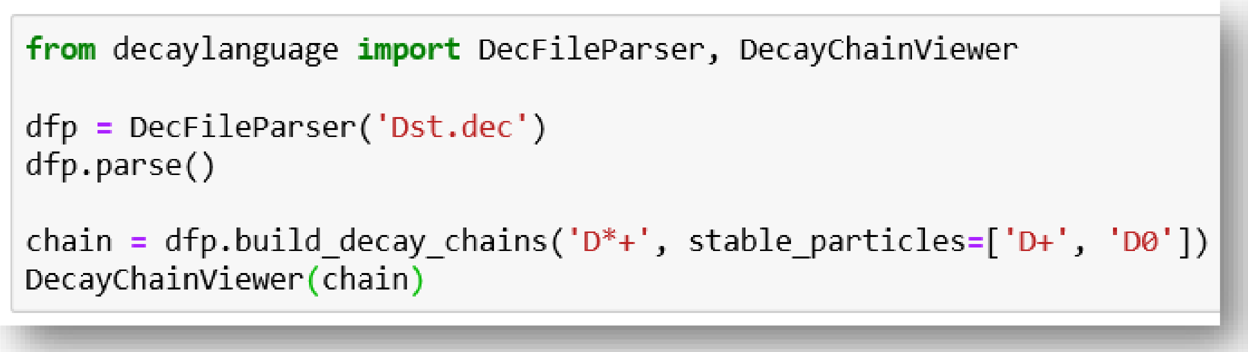


```
In [7]: from particle import Particle, SpinType

        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>]
```

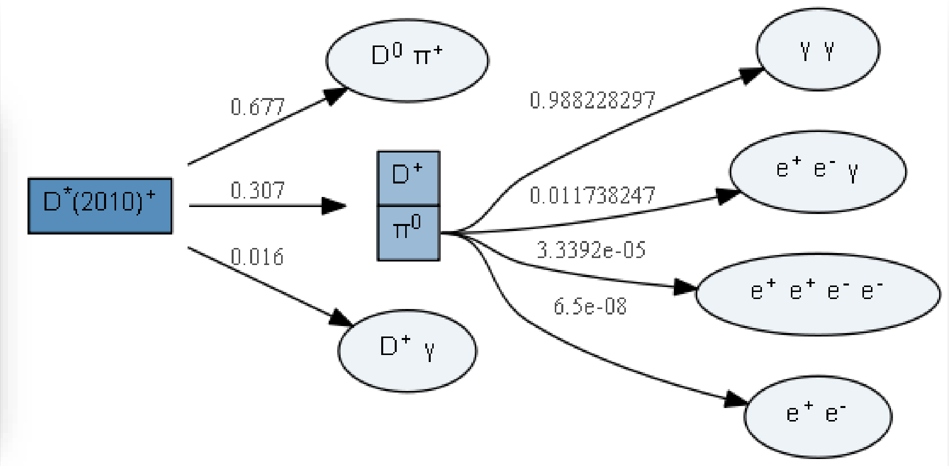
Particles and decays – DecayLanguage package

- ❑ 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

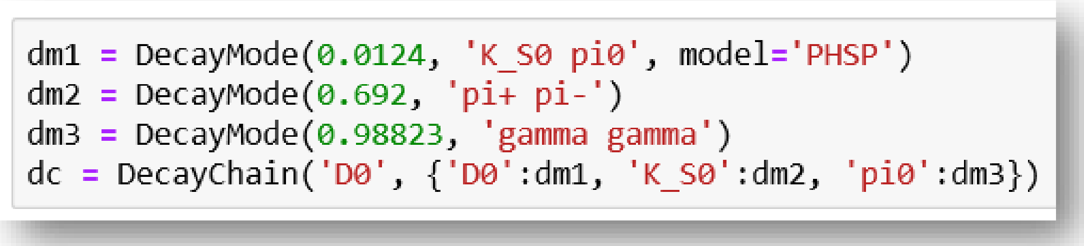
- ❑ Parse, extract information and visualise a decay chain:



```
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})
```

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

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

Primary Measurement:

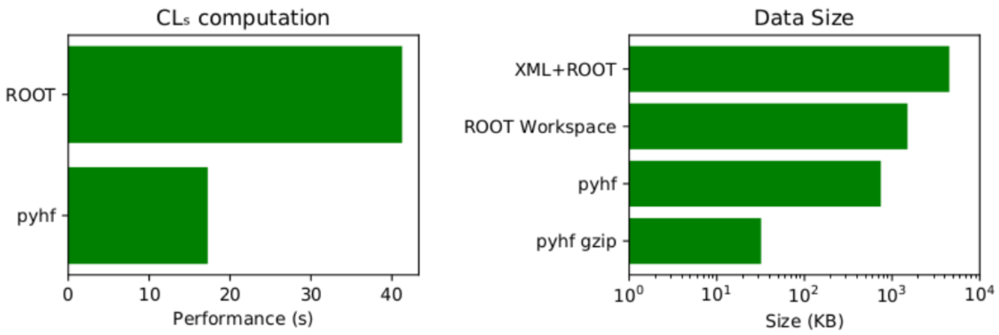
- Multiple disjoint "channels" (e.g. event observables) each with multiple bins of data
- Example parameter of interest: strength of physics signal, μ

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)

A metapackage for Scikit-HEP – `scikit-hep` package

❑ **The project now has a special package,**

scikit-hep

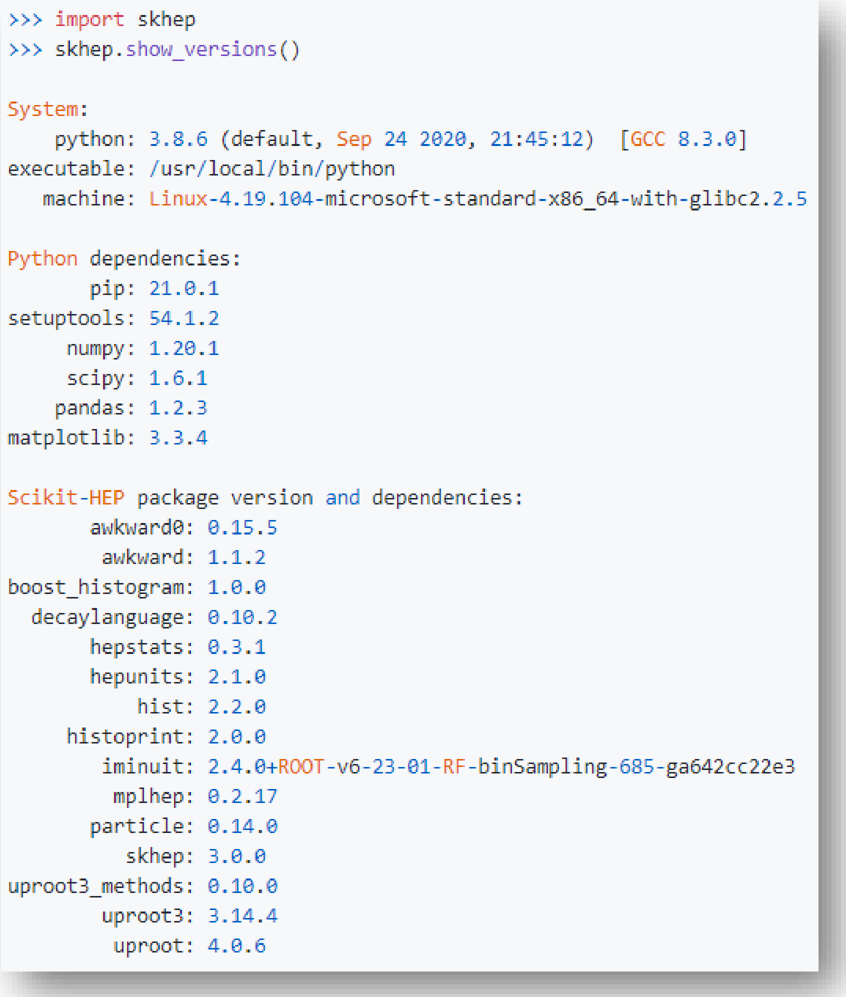
A metapackage

❑ **Unlike all others, which target specific topics, this metapackage simply provides an easy way to have a compatible set of project packages installed via a simple `pip install scikit-hep` (also available via Conda!)**

- **Benefit especially for stacks for experiments, since tags define compatible releases of the whole toolset**
- **Stable stacks installable in a simple way**
- **Helps in analysis preservation matters**

❑ **Trivial to check the versions available**

- Example of my laptop:



```
>>> import skhep
>>> skhep.show_versions()

System:
    python: 3.8.6 (default, Sep 24 2020, 21:45:12)  [GCC 8.3.0]
executable: /usr/local/bin/python
   machine: Linux-4.19.104-microsoft-standard-x86_64-with-glibc2.2.5

Python dependencies:
       pip: 21.0.1
setuptools: 54.1.2
     numpy: 1.20.1
     scipy: 1.6.1
    pandas: 1.2.3
matplotlib: 3.3.4

Scikit-HEP package version and dependencies:
       awkward0: 0.15.5
        awkward: 1.1.2
boost_histogram: 1.0.0
  decaylanguage: 0.10.2
       hepstats: 0.3.1
       hepunits: 2.1.0
           hist: 2.2.0
     histoprint: 2.0.0
        iminuit: 2.4.0+ROOT-v6-23-01-RF-binSampling-685-ga642cc22e3
         mplhep: 0.2.17
       particle: 0.14.0
          skhep: 3.0.0
uproot3_methods: 0.10.0
        uproot3: 3.14.4
         uproot: 4.0.6
```

Other community projects

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

- ❑ Also several libraries / projects in particular from IRIS-HEP (ServiceX, cabinetry, func-adl, etc.)

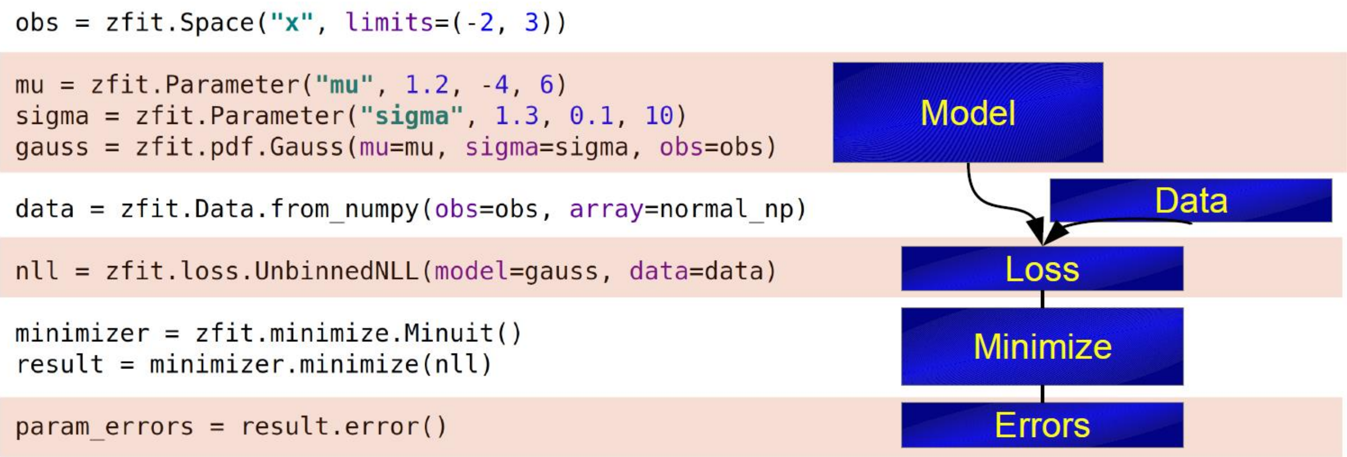
(Not a comprehensive list!)

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

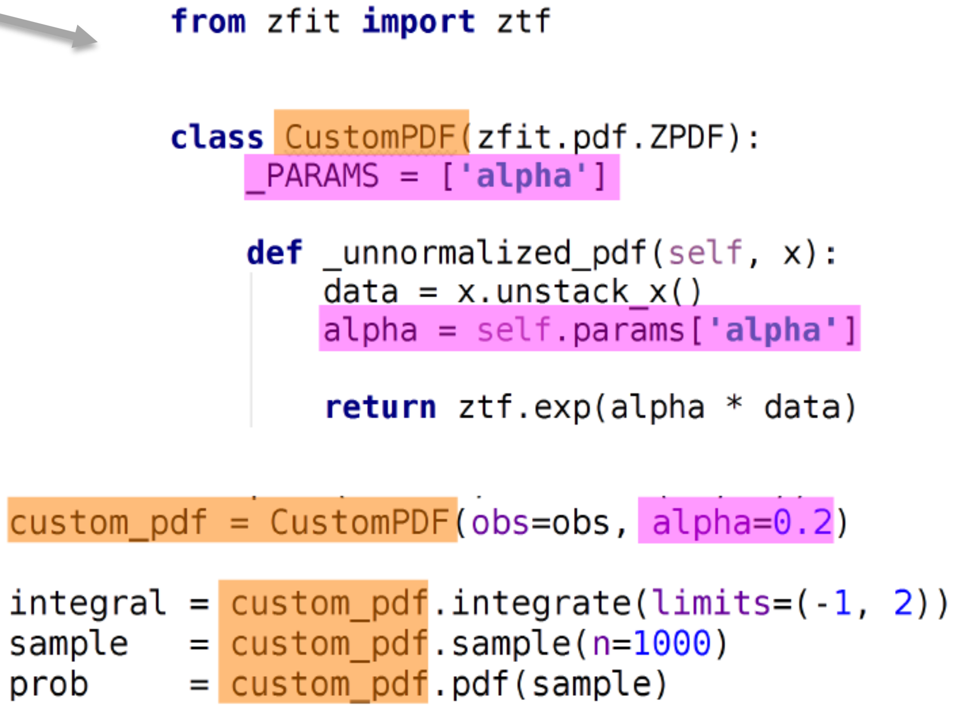
Scikit HEP affiliated

- ❑ Simple example:



```python
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()
```

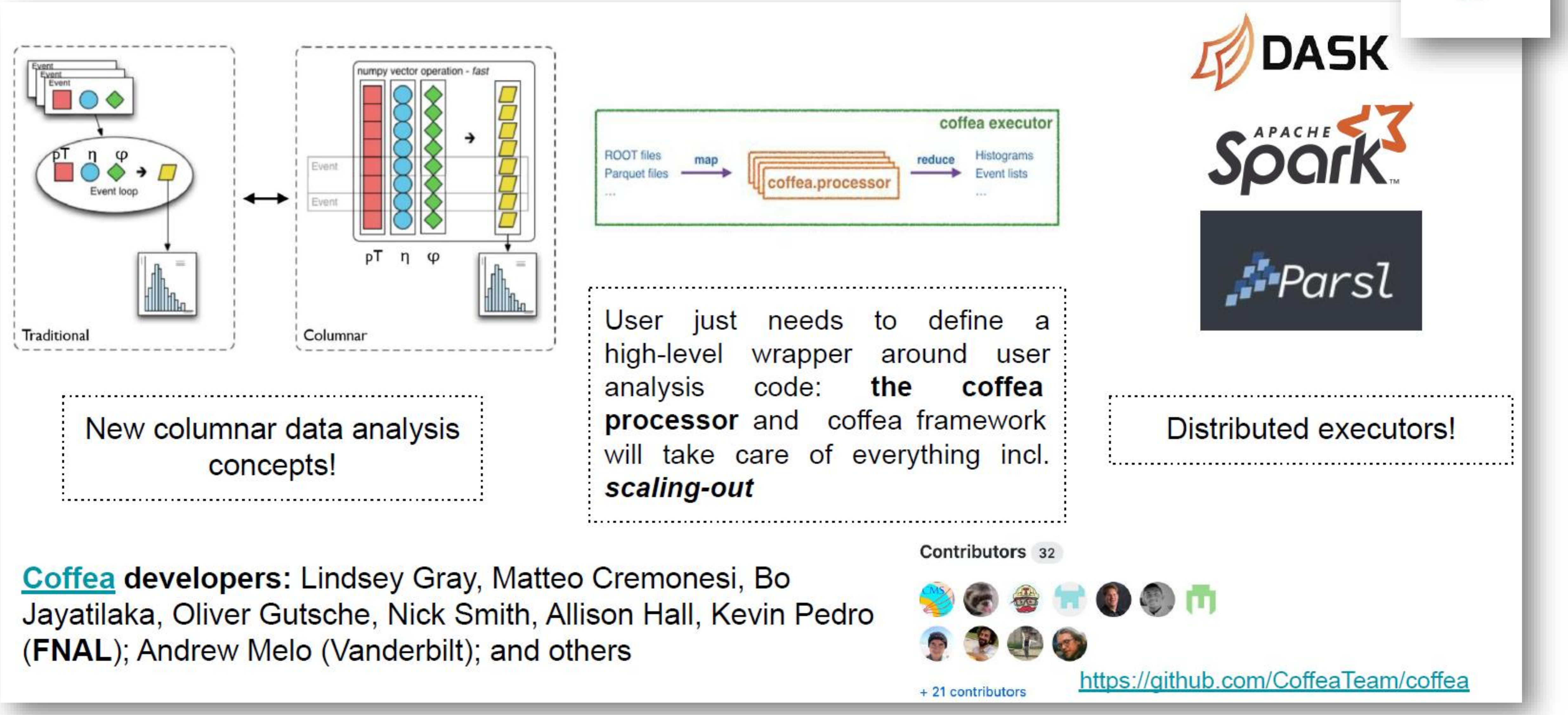
implement custom function



```python
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)
```

The coffea project & analysis framework

[coffea = Column Object Framework for Effective Analysis]



New columnar data analysis concepts!

User just needs to define a high-level wrapper around user analysis code: **the coffea processor** and coffea framework will take care of everything incl. ***scaling-out***

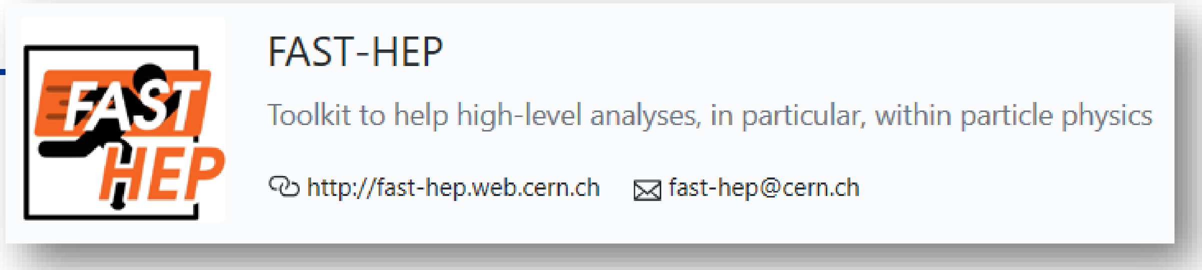
Distributed executors!

Coffea **developers:** Lindsey Gray, Matteo Cremonesi, Bo Jayatilaka, Oliver Gutsche, Nick Smith, Allison Hall, Kevin Pedro (**FNAL**); Andrew Melo (Vanderbilt); and others

https://github.com/CoffeaTeam/coffea

Slide taken from Oksana Shadura

The FAST-HEP project

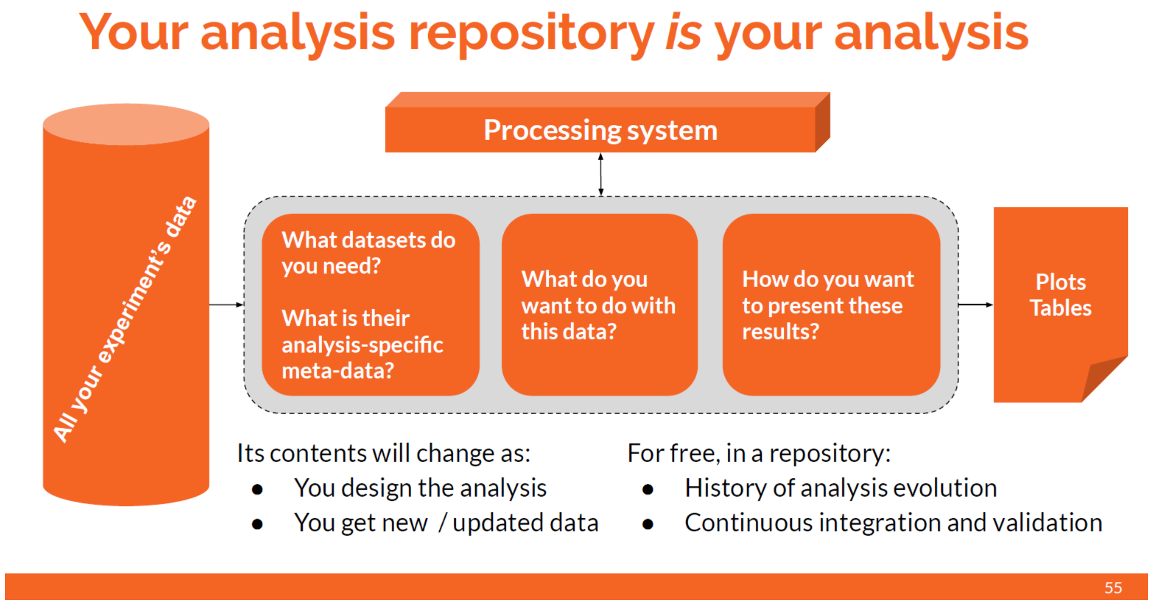


FAST-HEP

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

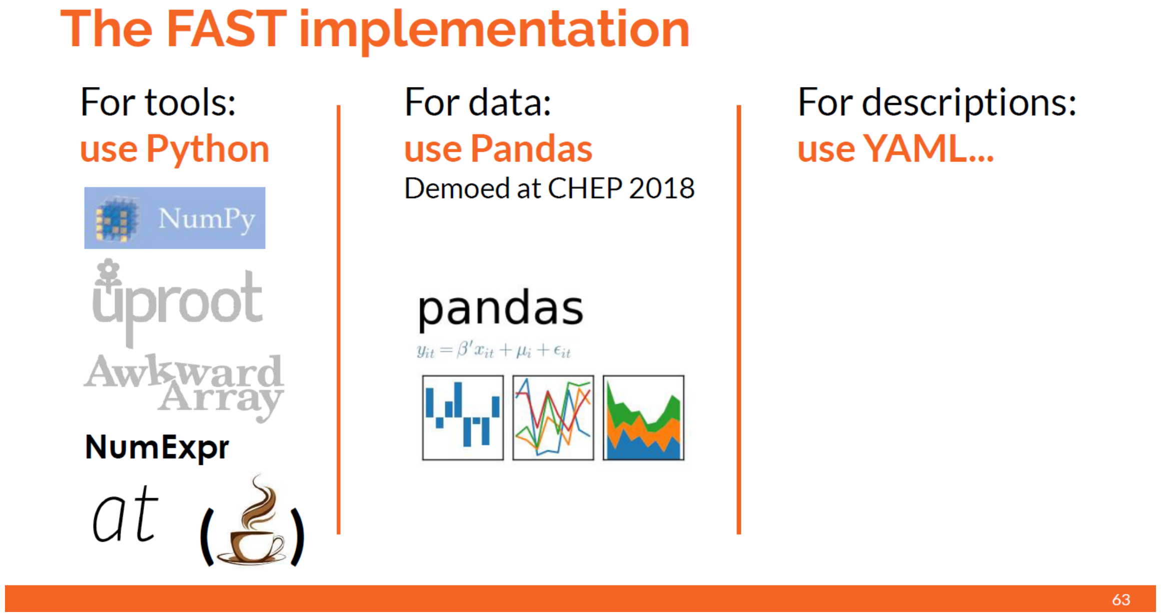
All your experiment's data
Processing system
What datasets do you need?
What is their analysis-specific meta-data?
What do you want to do with this data?
How do you want to present these results?
Plots
Tables

Its contents will change as:

- You design the analysis
- You get new / updated data

For free, in a repository:

- History of analysis evolution
- Continuous integration and validation



The FAST implementation

For tools:
use Python

For data:
use Pandas
Demoed at CHEP 2018

For descriptions:
use YAML...

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

- ❑ Project towards an Analysis Description Language …

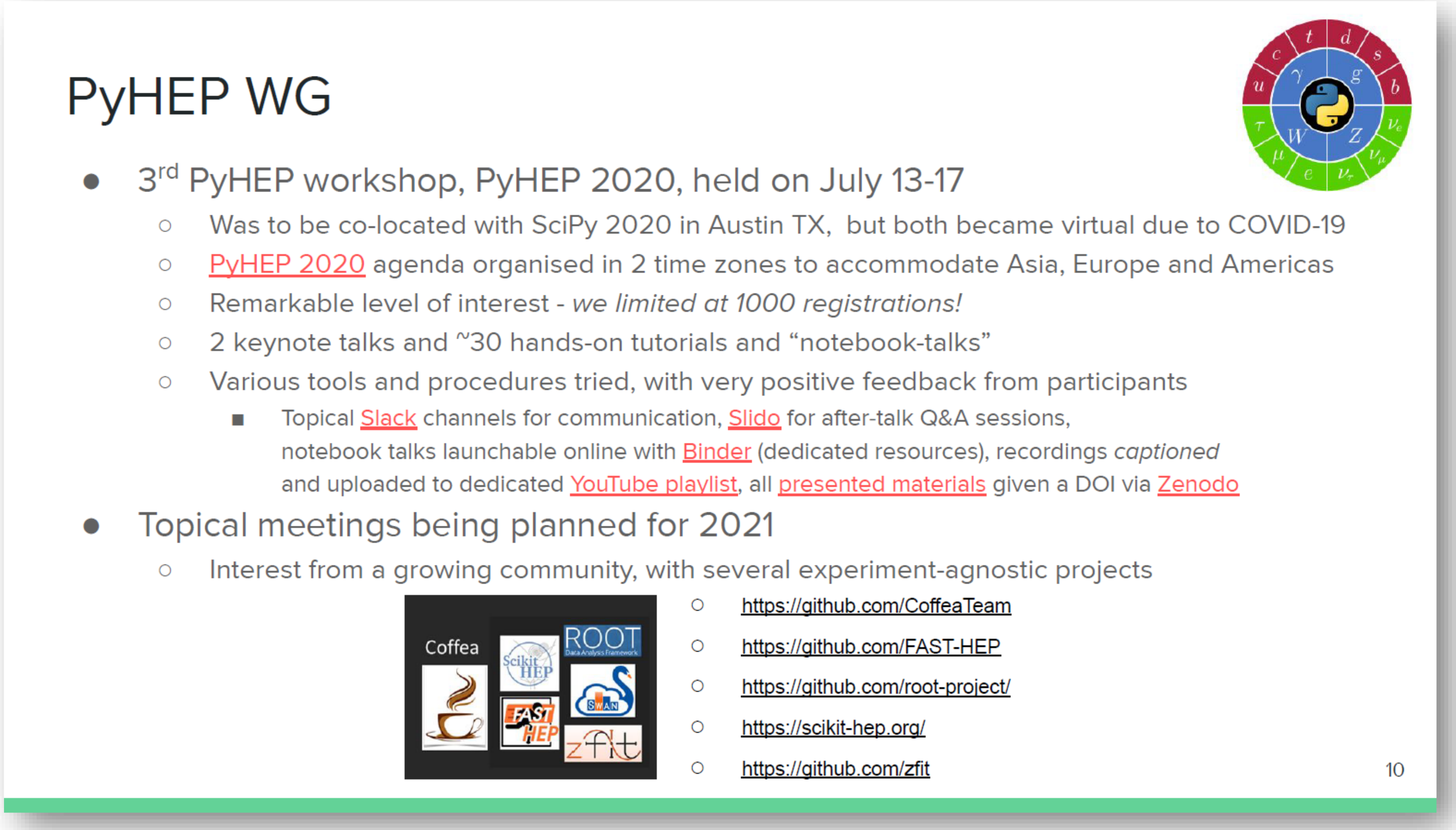
Material taken from Ben Krikler

Let’s step back a second
Some final remarks

- ❑ Is “Python in HEP” making an impact? Examples …
- ❑ Towards a
  Big Data Python ecosystem for HEP (analysis)!

LHCC referees now also get software reports in parallel to collab. reports

Snapshop of PyHEP WG report in Graeme Stewart's presentation on November 17th 2020:



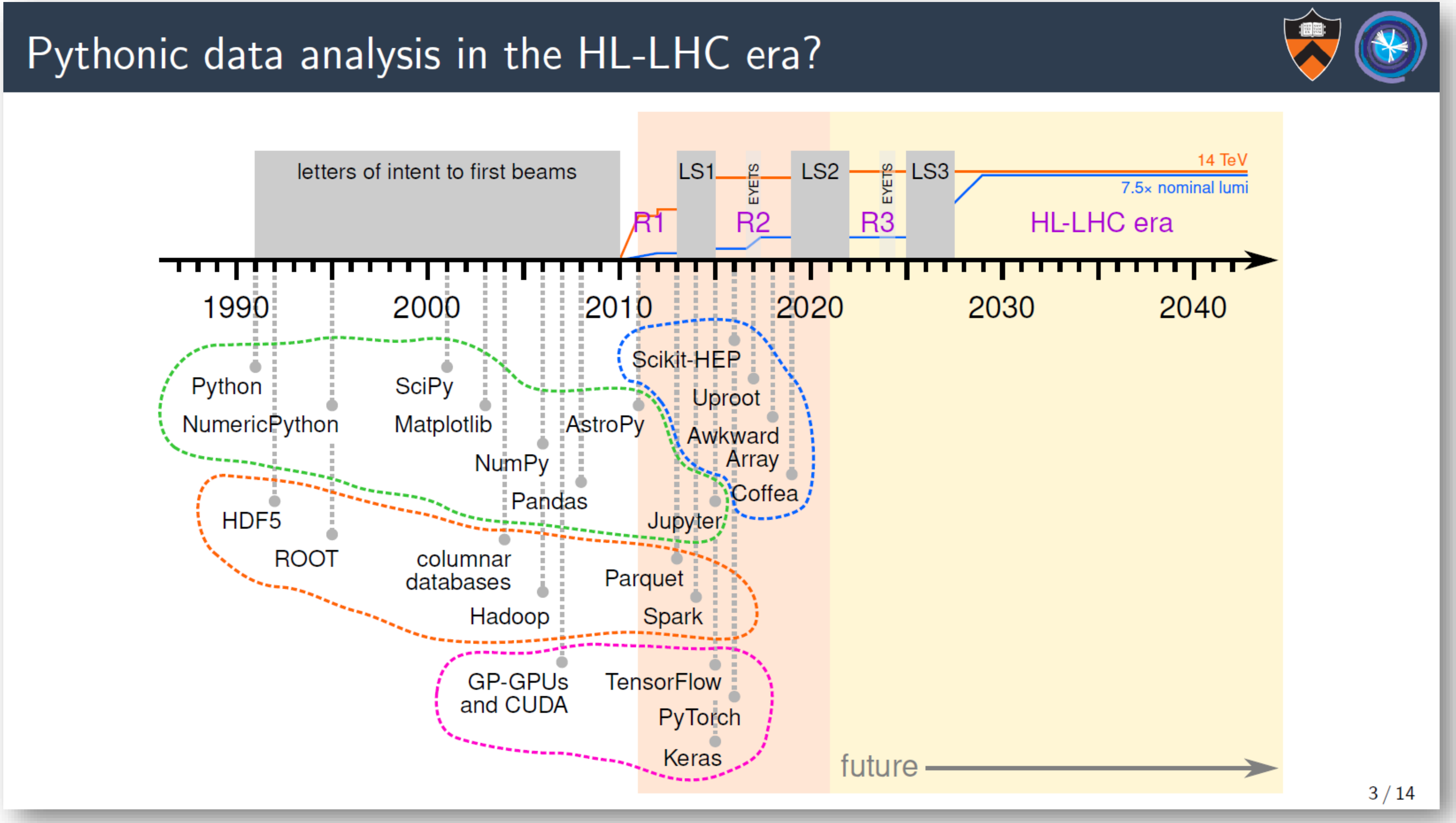
PyHEP WG

- 3rd PyHEP workshop, PyHEP 2020, held on July 13-17
  - Was to be co-located with SciPy 2020 in Austin TX, but both became virtual due to COVID-19
  - PyHEP 2020 agenda organised in 2 time zones to accommodate Asia, Europe and Americas
  - Remarkable level of interest - *we limited at 1000 registrations!*
  - 2 keynote talks and ~30 hands-on tutorials and "notebook-talks"
  - Various tools and procedures tried, with very positive feedback from participants
    - Topical Slack channels for communication, Slido for after-talk Q&A sessions, notebook talks launchable online with Binder (dedicated resources), recordings *captioned* and uploaded to dedicated YouTube playlist, all presented materials given a DOI via Zenodo
- Topical meetings being planned for 2021
  - Interest from a growing community, with several experiment-agnostic projects

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

10

Where are we heading ?

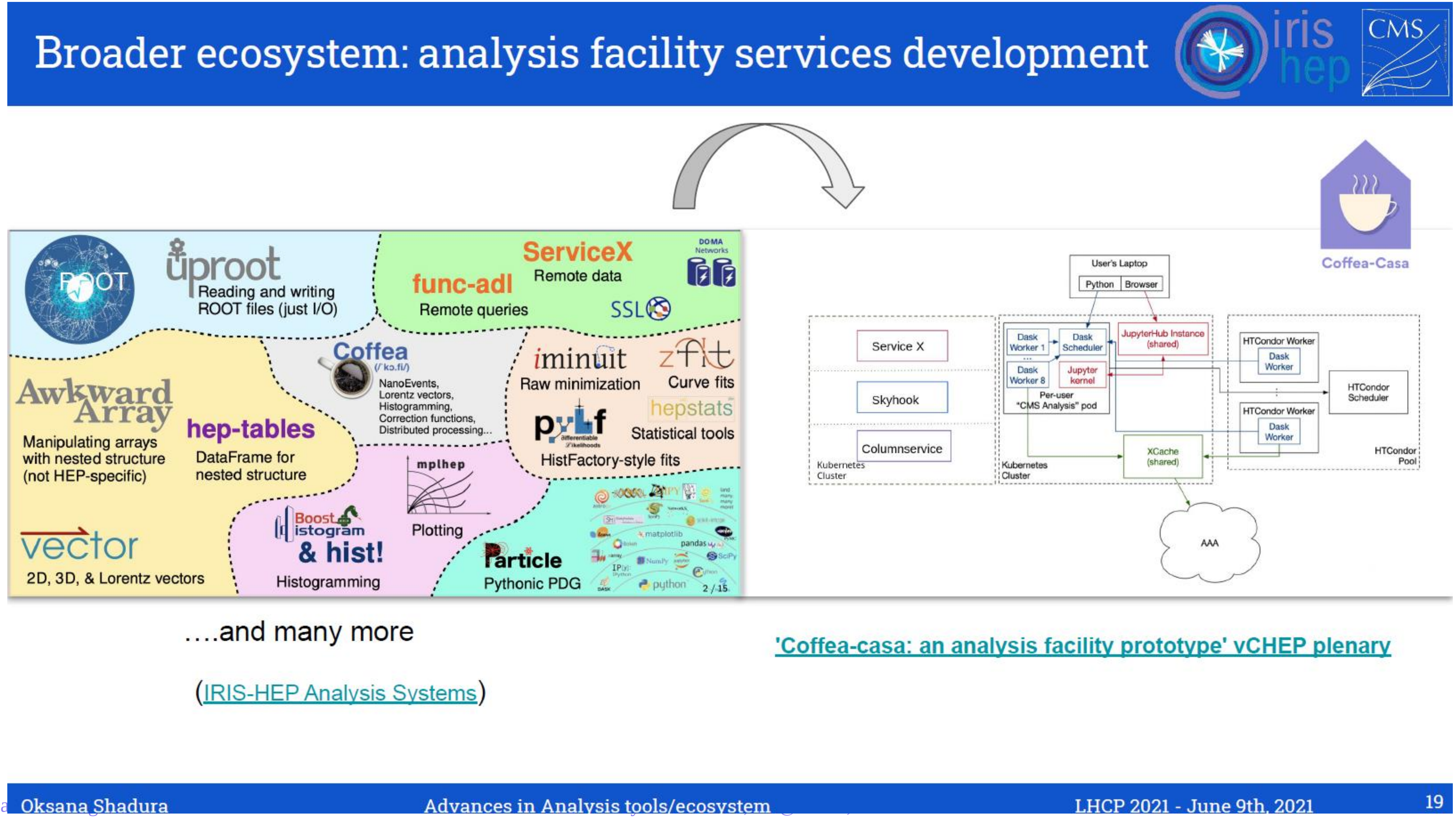


Pythonic data analysis in the HL-LHC era?

3 / 14

Slide by Jim Pivarski, May 2021

Where are we heading ?



Broader ecosystem: analysis facility services development

….and many more

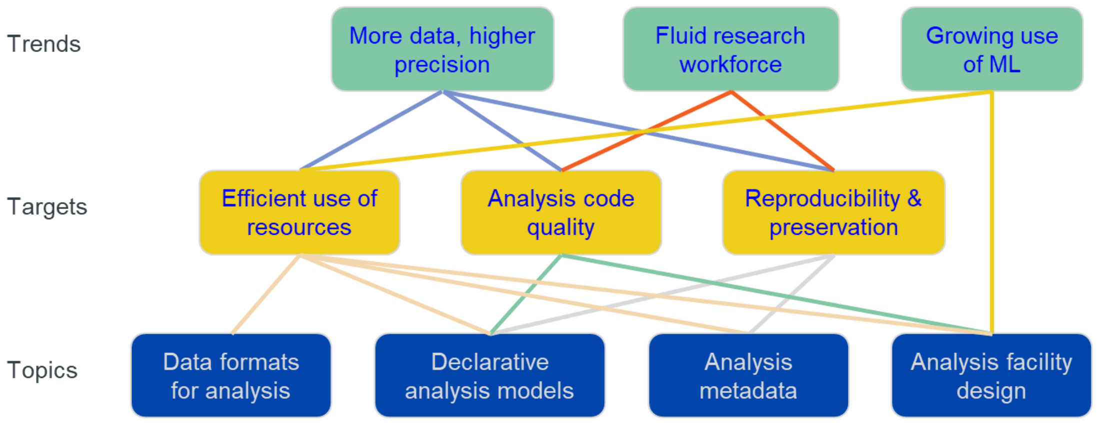
(IRIS-HEP Analysis Systems)

'Coffea-casa: an analysis facility prototype' vCHEP plenary

Oksana Shadura Advances in Analysis tools/ecosystem LHCP 2021 - June 9th, 2021 19

Where are we heading ?

- ❑ Previous slides were more on “where did we come from and how did we get where we are?”
- ❑ Far easier a question!



Trends
Targets
Topics
More data, higher precision
Fluid research workforce
Growing use of ML
Efficient use of resources
Analysis code quality
Reproducibility & preservation
Data formats for analysis
Declarative analysis models
Analysis metadata
Analysis facility design

Taken from HSF Data Analysis WG Conveners

- ❑ If I had to guess …
- ❑ Python is here to stay. At least for as long as it dominates Data Science software for ML and AI
- ❑ HEP will continue to “tag along”
- ❑ In fact, usage of Data Science tools in Python have not yet reached a plateau. Same for HEP
- ❑ And things are not going to change immediately with other players coming to join the fun, namely Quantum Computing with Quantum ML in particular

Thank you for listening

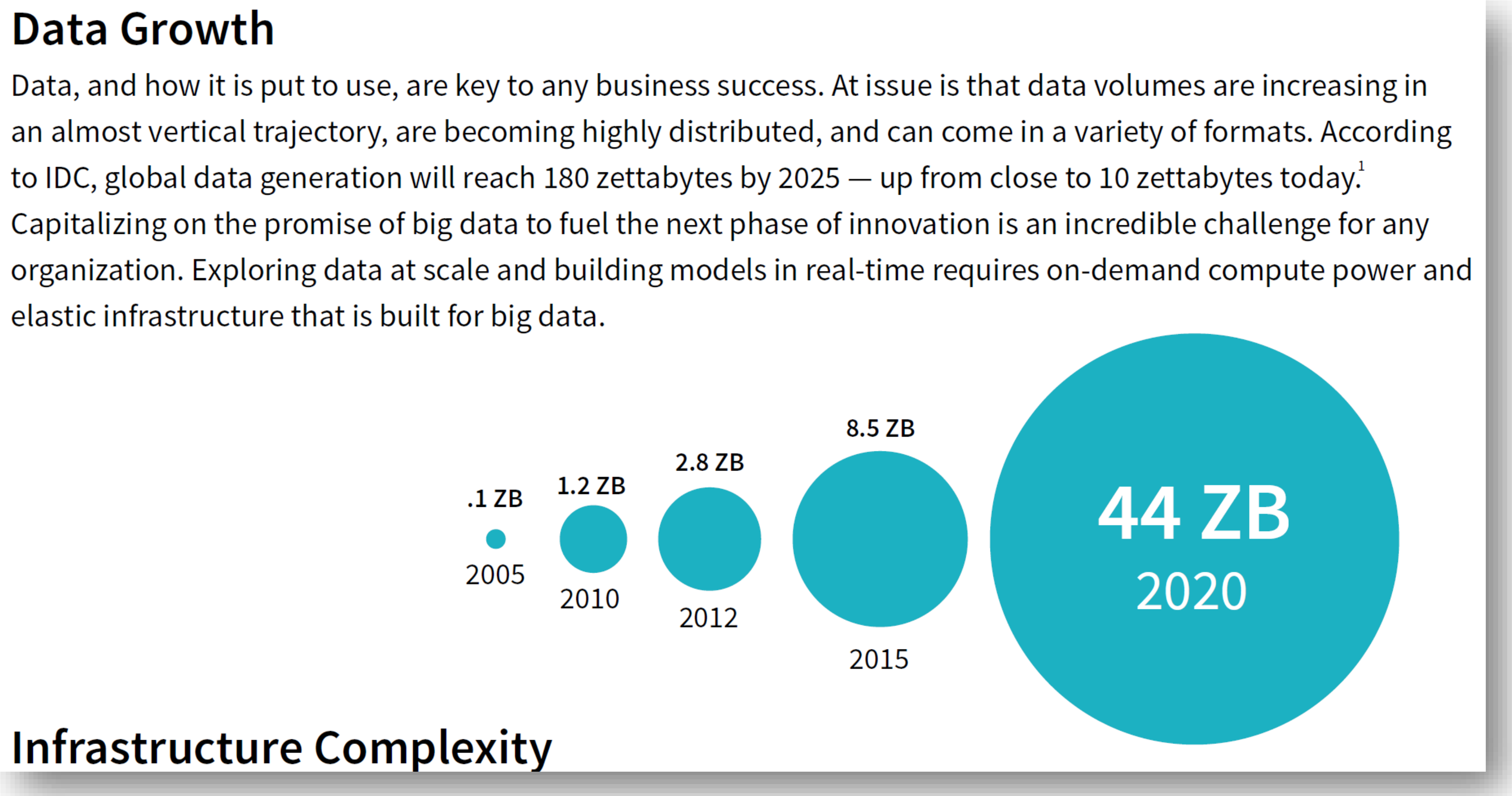
And see you at PyHEP 2021 ;-) !

https://indico.cern.ch/e/PyHEP2021

- ❑ HEP Software Foundation (HSF)
  - HSF general forum hsf-forum@googlegroups.com
- ❑ HSF PyHEP Working Group
  - (main) Gitter channel
  - GitHub repository "Python in HEP" resources
- ❑ PyHEP 2020 workshop
- ❑ Scikit-HEP project
  - Get in touch

Particle Physics and Big Data – data is growing big, and fast!

❑ **Solutions are necessary to massage the shear amounts of data being produced, and going to be produced, worldwide**



Data Growth

Data, and how it is put to use, are key to any business success. At issue is that data volumes are increasing in an almost vertical trajectory, are becoming highly distributed, and can come in a variety of formats. According to IDC, global data generation will reach 180 zettabytes by 2025 — up from close to 10 zettabytes today.[1] Capitalizing on the promise of big data to fuel the next phase of innovation is an incredible challenge for any organization. Exploring data at scale and building models in real-time requires on-demand compute power and elastic infrastructure that is built for big data.

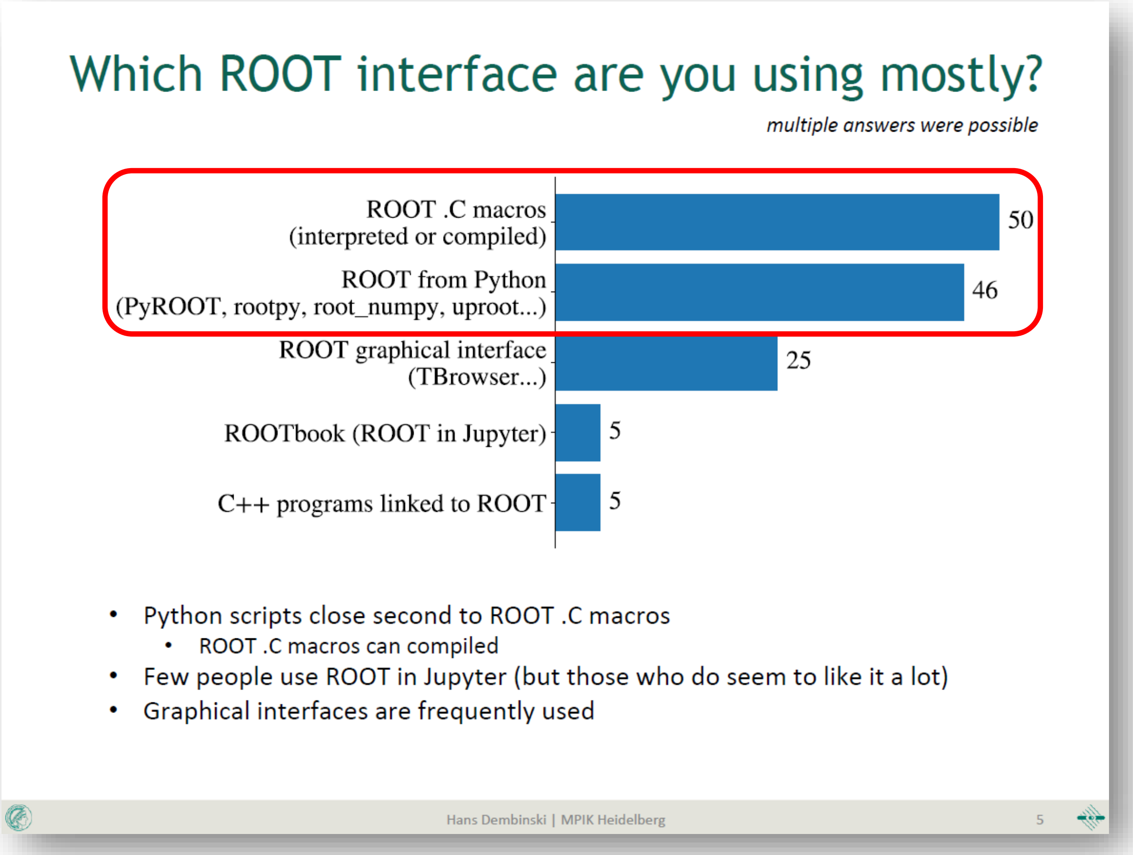
Infrastructure Complexity

Taken from this report.

Python adoption in HEP – ROOT from Python in LHCb

Surveys from the LHCb experiment

- ❑ **Python and C++ equally used among analysts**
  - Trend seen in our <u>LHCb survey</u> for the ROOT User's Workshop in 2018
  - And in the LHCb 2018 Analysis Survey Report (by Eduardo Rodrigues)
- ❑ **ROOT from Python is just as used as is from plain C++ !**
- ❑ **Conclusion even stronger if discussing analysis tools independent of ROOT**



Which ROOT interface are you using mostly?

multiple answers were possible

- Python scripts close second to ROOT .C macros
  - ROOT .C macros can 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*,
<u>ROOT Users' workshop</u>, Sarajevo, Sep. 2018

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

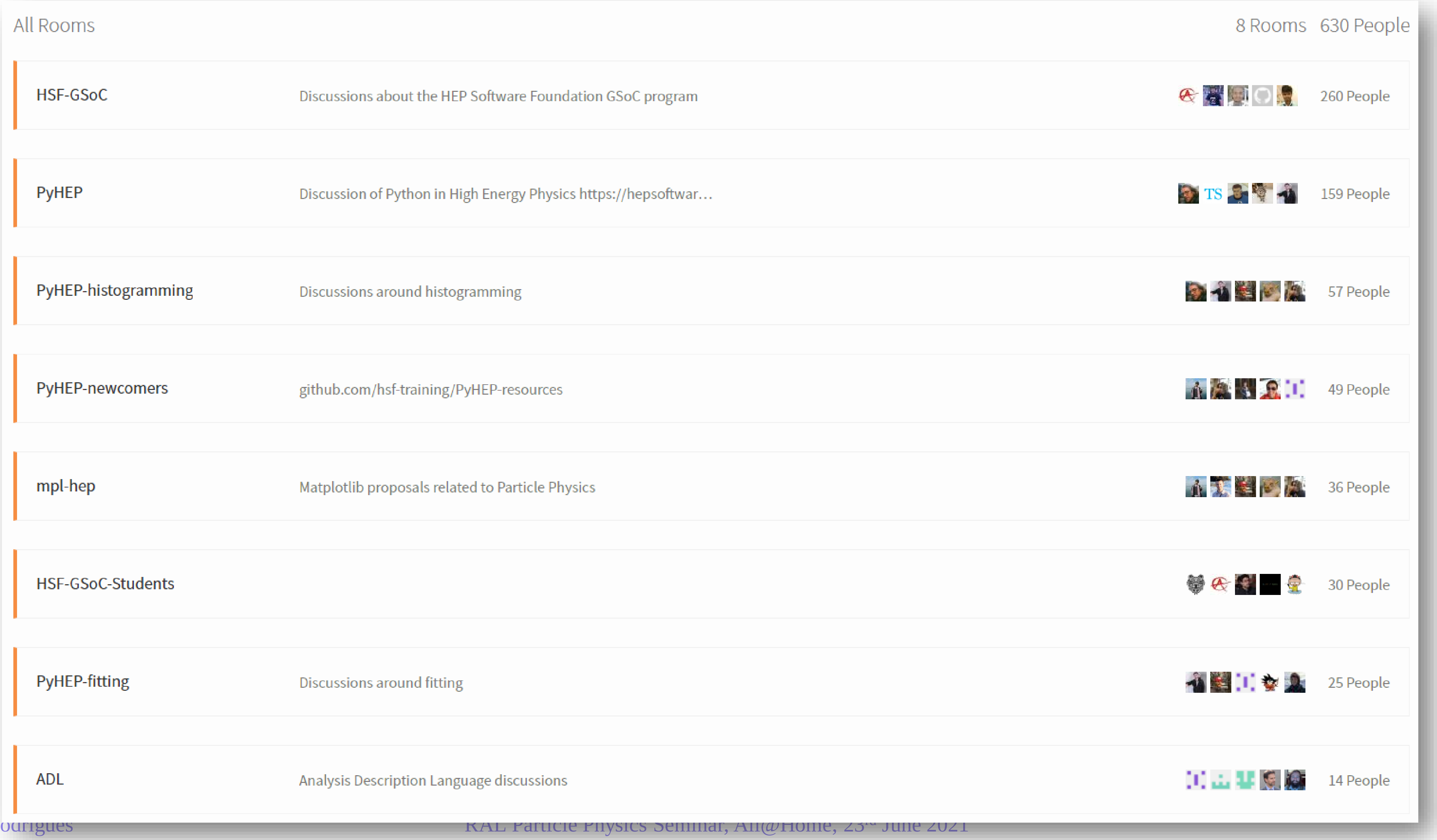
- Founded in 2014, explicitly to address current and future computing & software challenges in common

- 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 …

- 2020: new community document "HL-LHC Computing Review: Common Tools and Community Software",
  Stewart, Graeme Andrew *et al*. (2020, May 1). Zenodo. http://doi.org/10.5281/zenodo.3779250, HSF-DOC-2020-01

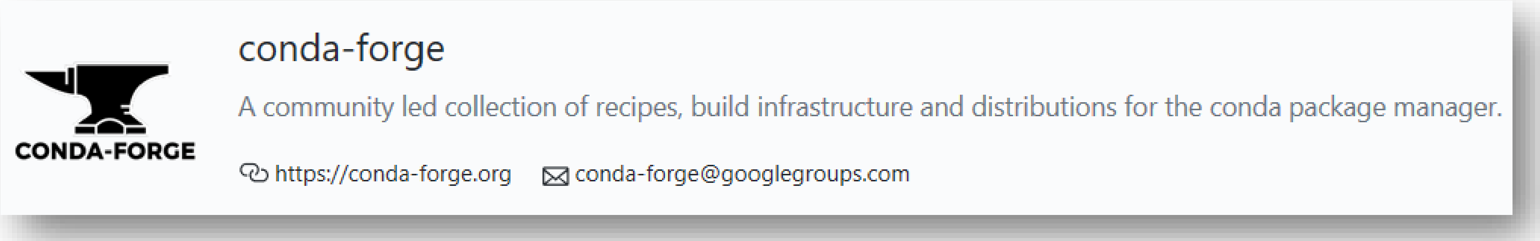
HSF – Gitter channels



All Rooms — 8 Rooms 630 People

HSF-GSoC	Discussions about the HEP Software Foundation GSoC program	260 People
PyHEP	Discussion of Python in High Energy Physics https://hepsoftwar…	159 People
PyHEP-histogramming	Discussions around histogramming	57 People
PyHEP-newcomers	github.com/hsf-training/PyHEP-resources	49 People
mpl-hep	Matplotlib proposals related to Particle Physics	36 People
HSF-GSoC-Students		30 People
PyHEP-fitting	Discussions around fitting	25 People
ADL	Analysis Description Language discussions	14 People

Conda-forge – making it easy for users

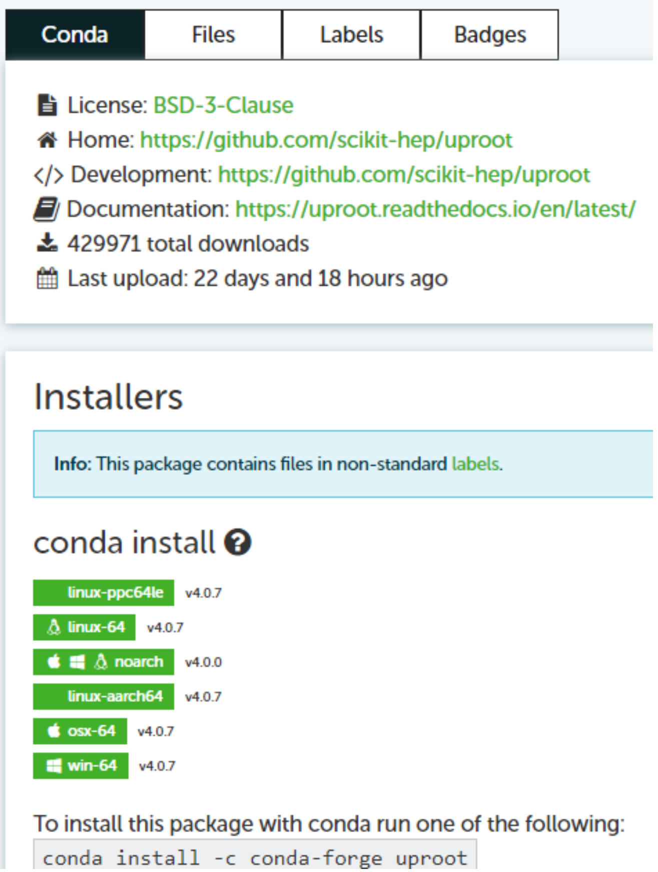


conda-forge

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`:



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/
429971 total downloads
Last upload: 22 days and 18 hours ago

Installers

Info: This package contains files in non-standard labels.

conda install

linux-ppc64le	v4.0.7
linux-64	v4.0.7
noarch	v4.0.0
linux-aarch64	v4.0.7
osx-64	v4.0.7
win-64	v4.0.7

To install this package with conda run one of the following:

```
conda install -c conda-forge uproot
```

PyHEP series of workshops

PyHEP 2018

Sofia, Bulgaria

PyHEP 2019

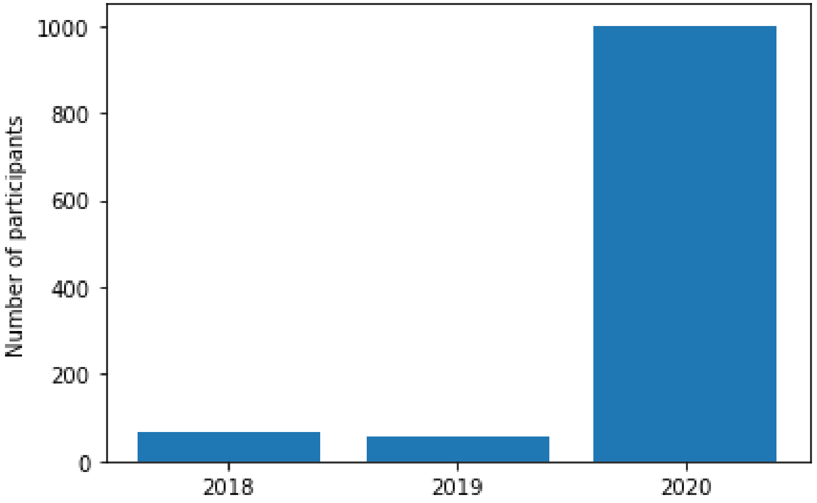
Abingdon, U.K.

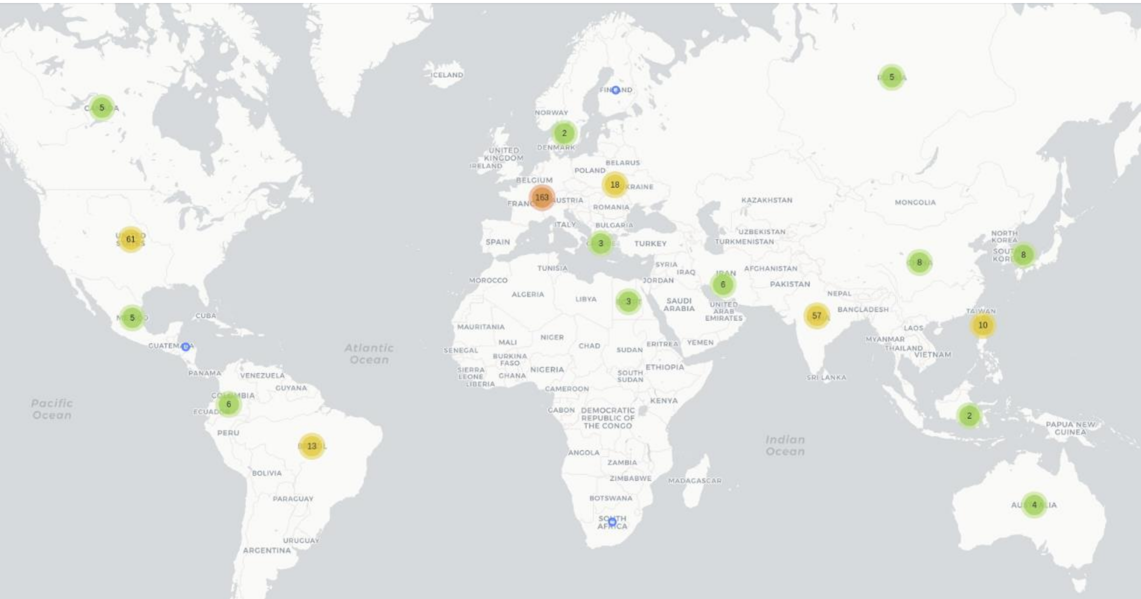
PyHEP 2020

- *Was meant to be held in Austin (Texas), U.S.A., in July 11-13*
- *Next to SciPy 2020 conference, to enhance cross-community exchange*
- *Run as a virtual event, as most conferences this year*

PyHEP workshops – what changed when moving to a virtual event in 2020

- ❑ Event spread over a week (5 days) rather than 3 (1.5) days in 2019 (2018)
- ❑ Shorter sessions per day, 3-hour long at most
- ❑ Sessions organised in 2 time zones!
  - “Pacific-friendly” and “Atlantic-friendly”
- ❑ Workshop became a truly global event with participants from all over the world
- ❑ No registration fees
- ❑ Impressive level of interest with 1000 registrations (limited to)
  (72, 55 in previous years)

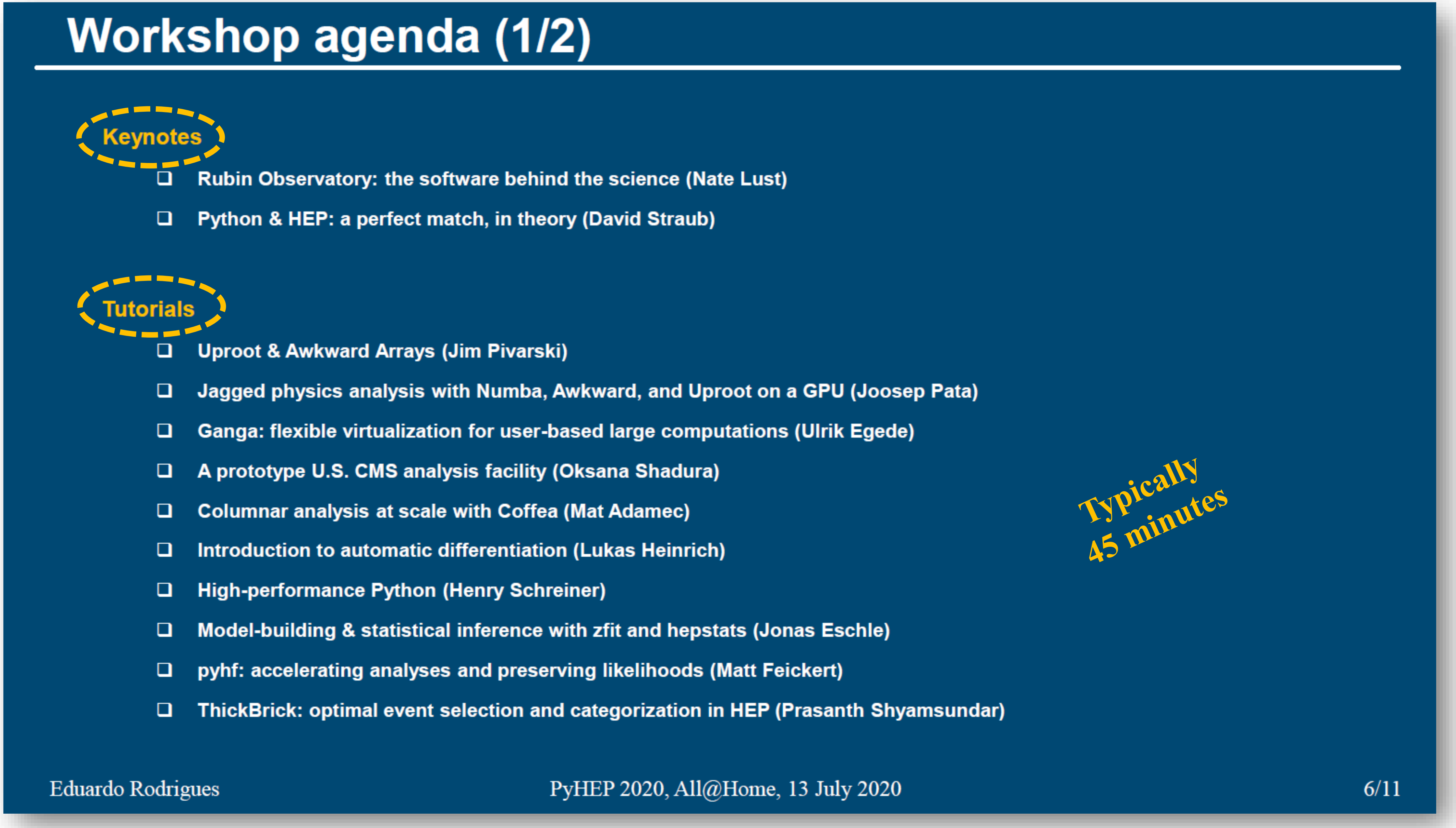




[Plot from Jim Pivarski]

[Information taken from the 408/1000 responses received from the pre-workshop survey.]

PyHEP 2020 – agenda (1/2)



Workshop agenda (1/2)

Keynotes

- Rubin Observatory: the software behind the science (Nate Lust)
- Python & HEP: a perfect match, in theory (David Straub)

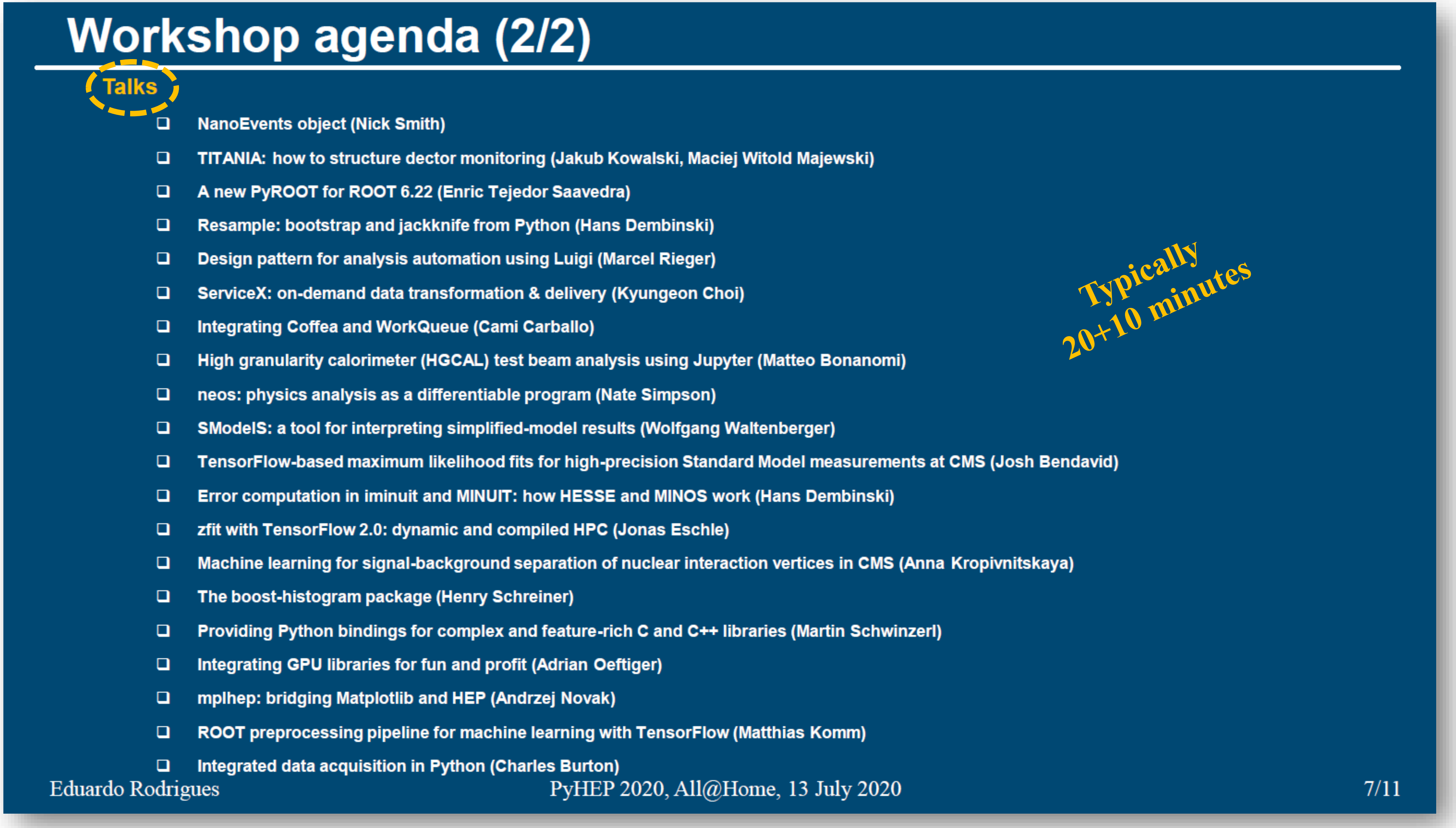
Tutorials

- Uproot & Awkward Arrays (Jim Pivarski)
- Jagged physics analysis with Numba, Awkward, and Uproot on a GPU (Joosep Pata)
- Ganga: flexible virtualization for user-based large computations (Ulrik Egede)
- A prototype U.S. CMS analysis facility (Oksana Shadura)
- Columnar analysis at scale with Coffea (Mat Adamec)
- Introduction to automatic differentiation (Lukas Heinrich)
- High-performance Python (Henry Schreiner)
- Model-building & statistical inference with zfit and hepstats (Jonas Eschle)
- pyhf: accelerating analyses and preserving likelihoods (Matt Feickert)
- ThickBrick: optimal event selection and categorization in HEP (Prasanth Shyamsundar)

Typically 45 minutes

Eduardo Rodrigues PyHEP 2020, All@Home, 13 July 2020 6/11

PyHEP 2020 – agenda (2/2)



Workshop agenda (2/2)

Talks

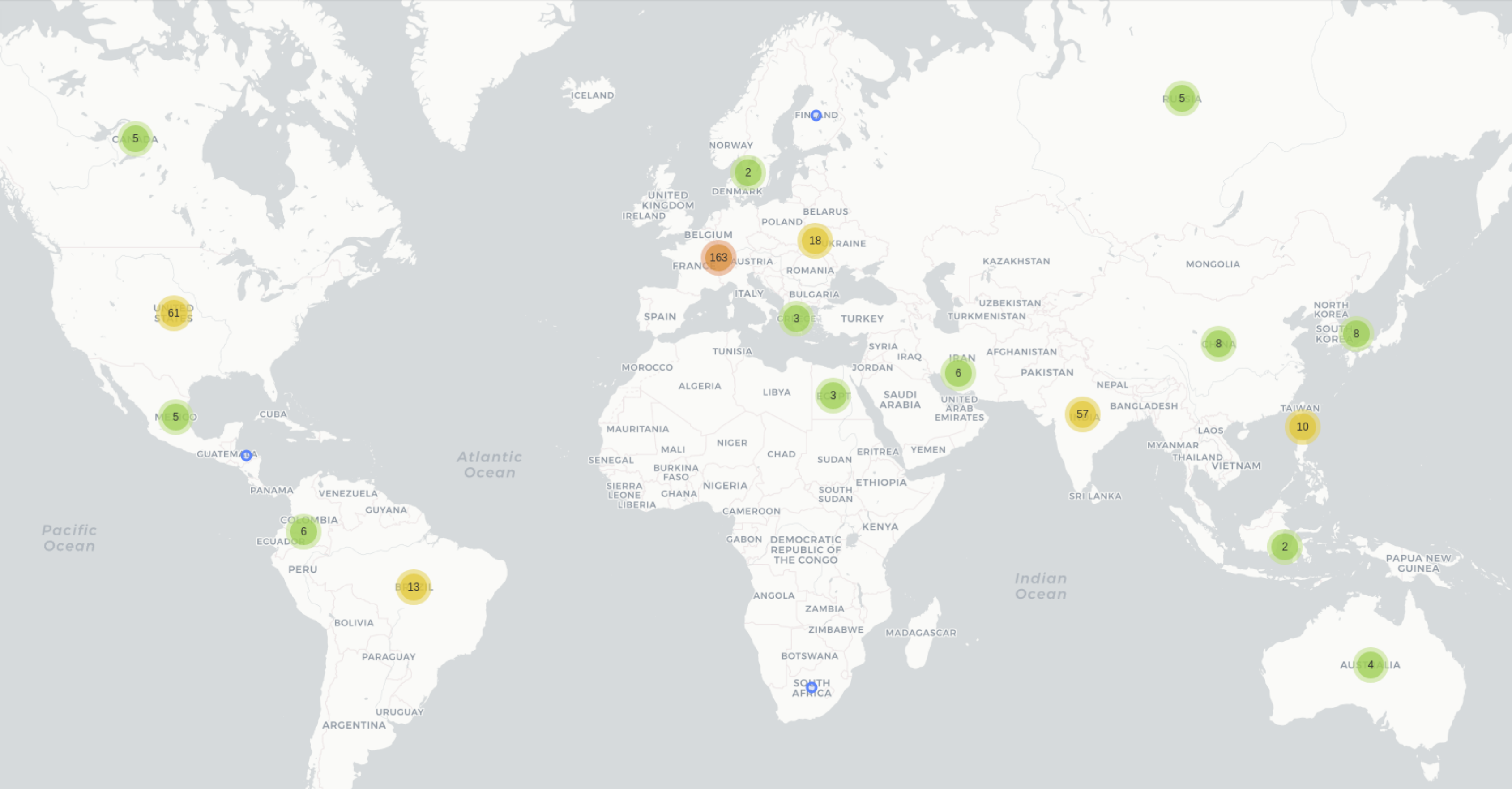
- □ NanoEvents object (Nick Smith)
- □ TITANIA: how to structure dector monitoring (Jakub Kowalski, Maciej Witold Majewski)
- □ A new PyROOT for ROOT 6.22 (Enric Tejedor Saavedra)
- □ Resample: bootstrap and jackknife from Python (Hans Dembinski)
- □ Design pattern for analysis automation using Luigi (Marcel Rieger)
- □ ServiceX: on-demand data transformation & delivery (Kyungeon Choi)
- □ Integrating Coffea and WorkQueue (Cami Carballo)
- □ High granularity calorimeter (HGCAL) test beam analysis using Jupyter (Matteo Bonanomi)
- □ neos: physics analysis as a differentiable program (Nate Simpson)
- □ SModelS: a tool for interpreting simplified-model results (Wolfgang Waltenberger)
- □ TensorFlow-based maximum likelihood fits for high-precision Standard Model measurements at CMS (Josh Bendavid)
- □ Error computation in iminuit and MINUIT: how HESSE and MINOS work (Hans Dembinski)
- □ zfit with TensorFlow 2.0: dynamic and compiled HPC (Jonas Eschle)
- □ Machine learning for signal-background separation of nuclear interaction vertices in CMS (Anna Kropivnitskaya)
- □ The boost-histogram package (Henry Schreiner)
- □ Providing Python bindings for complex and feature-rich C and C++ libraries (Martin Schwinzerl)
- □ Integrating GPU libraries for fun and profit (Adrian Oeftiger)
- □ mplhep: bridging Matplotlib and HEP (Andrzej Novak)
- □ ROOT preprocessing pipeline for machine learning with TensorFlow (Matthias Komm)
- □ Integrated data acquisition in Python (Charles Burton)

Typically 20+10 minutes

Eduardo Rodrigues PyHEP 2020, All@Home, 13 July 2020 7/11

PyHEP 2020 – diversity and inclusion

- ❑ **Diverse participation from all over the world !**

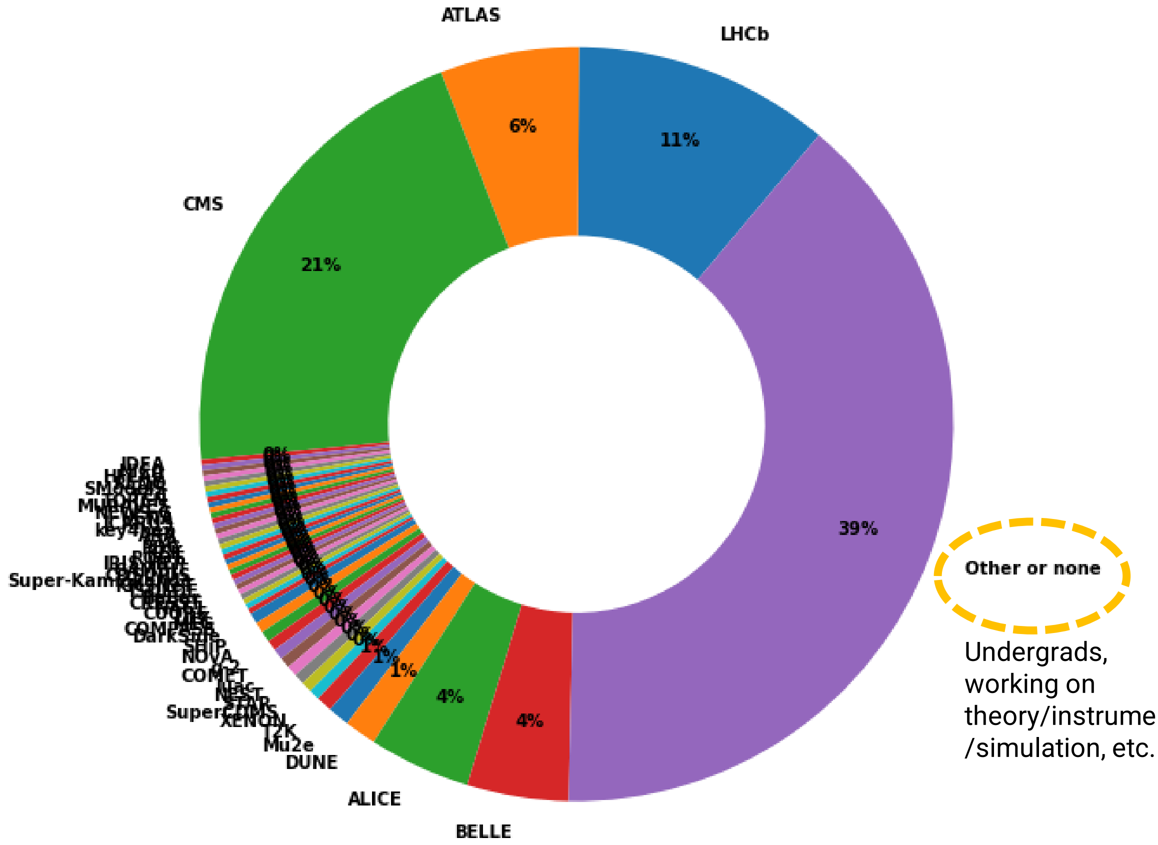


Plot by Jim Pivarski

- ❑ **Information taken from the 408/1000 responses received from the pre-workshop survey**

PyHEP 2020 – diversity and inclusion

PyHEP 2020
Virtual event

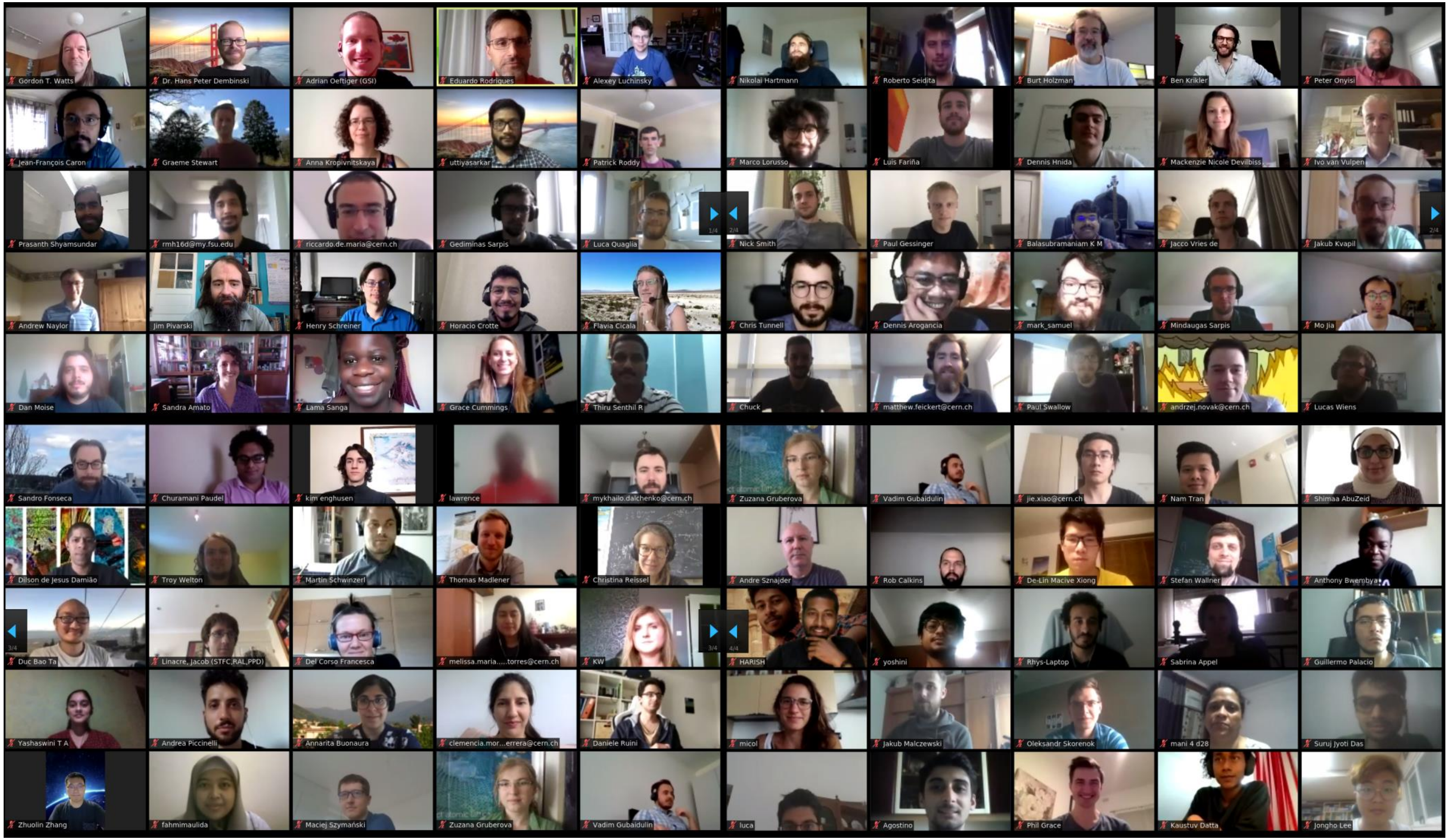


Undergrads, working on theory/instrumentation /simulation, etc.

- ❑ Great to see such a diverse set of participants !

(Pie chart and "logo art" with information taken from the pre-workshop questionnaire)

PyHEP 2020 – "workshop photo" @ end of last Atlantic session



PyHEP 2020 – "workshop photo" @ end of last Pacific session

PyHEP 2020 – sessions & presentations

- ❑ **Sessions spread in “Atlantic”- and “Pacific”-friendly time zones to accommodate Asia, Americas and Europe**

  - Atlantic: ~3h with 30-min breaks; Pacific: ~1h, no breaks. E.g.,

  Atlantic: 15h00 - 18h00 CET, 06h00 - 09h00 PDT, 18h30 - 21h30 IST, 21h00 - 24h00 CST, 22h00 - 01h00+1 JST
  Pacific: 15h00 - 16h00 PDT, 00h00 - 01h00+1 CET, 03h30+1 - 04h30+1 IST, 06h00+1 - 07h00+1 CST, 07h00+1 - 08h00+1 JST

  - Quite a bit more work and more demanding for the session chairs

  - The Pacific sessions ended up far less popular than the Atlantic sessions. Decision not to replicate this year

- ❑ **We strongly encouraged “notebook presentations” + Binder for an interactive experience**

- ❑ **Notebooks and related material made available in public GitHub repositories with a Binder launch button**
  (all presentational material posted on workshop agenda, including repo. links)

binder

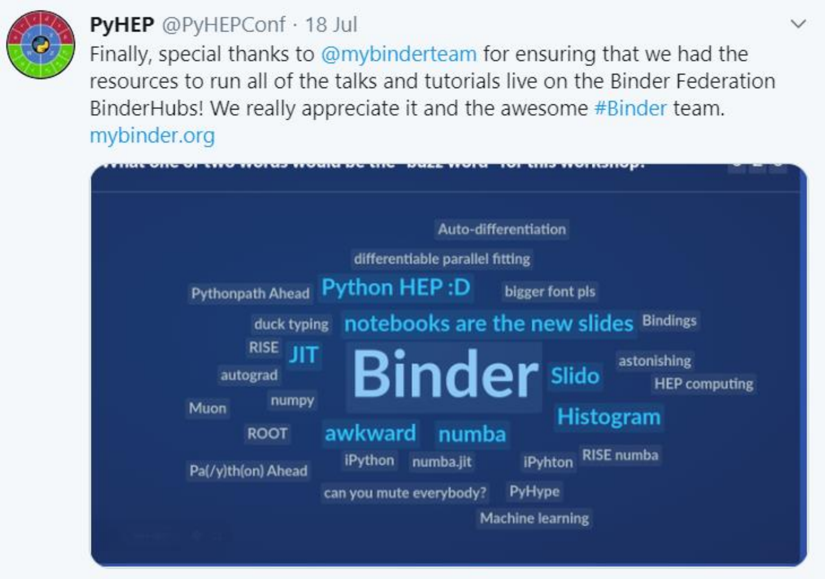
launch binder

- ❑ **We used both the**
  Binder Federation and the CERN Binder Hub resources
  (for those with CERN accounts)

- ❑ **Got in touch with the Binder team to have**
  resources allocated to talk repositories at the relevant time !

  - It worked very well – thank you MyBinderTeam

  - Binder was a leitmotif during the workshop:



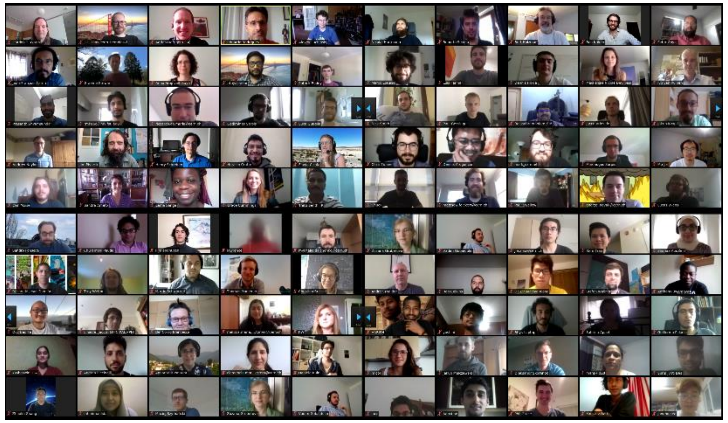
PyHEP @PyHEPConf · 18 Jul

Finally, special thanks to @mybinderteam for ensuring that we had the resources to run all of the talks and tutorials live on the Binder Federation BinderHubs! We really appreciate it and the awesome #Binder team.
mybinder.org

From the PyHEPConf Twitter account

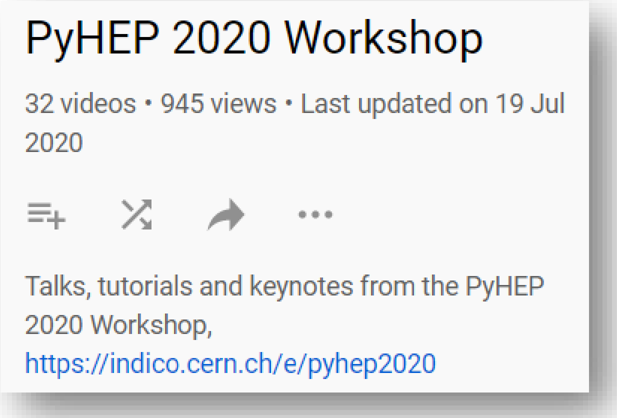
PyHEP 2020 – videoconferencing

- ❑ Zoom video conferencing system
  - With capacity for 1000 participants
  - Public room but PIN provided via email
- ❑ All recorded. No pre-recordings



"Workshop photo" from close-up Atlantic session

- ❑ (HSF has its own YouTube channel, with several playlists)
- ❑ All presentations got recorded and captioned
  (captioning thanks to sponsors, see later)
- ❑ Later uploaded to the HSF YouTube channel
  to a dedicated playlist "PyHEP 2020 Workshop"



PyHEP 2020 Workshop

32 videos • 945 views • Last updated on 19 Jul 2020

Talks, tutorials and keynotes from the PyHEP 2020 Workshop, https://indico.cern.ch/e/pyhep2020

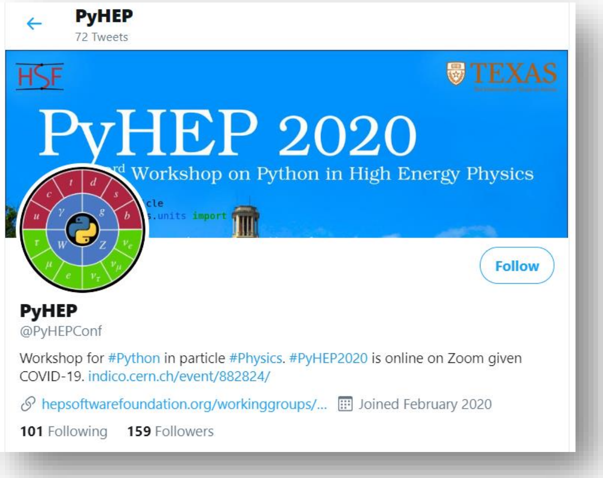
PyHEP 2020 – communication & interactions

- ❑ **Slack channels**
  - Various channels:
    - By topic, mapping to sessions, discussions encouraged here
    - Announcements, for actual announcements
    - Random, used to encourage community spirit and add social context

- ❑ **Communication also on**

@PyHEPConf

#PyHEP2020



PyHEP
72 Tweets

PyHEP
@PyHEPConf

Workshop for #Python in particle #Physics. #PyHEP2020 is online on Zoom given COVID-19. indico.cern.ch/event/882824/

hepsoftwarefoundation.org/workinggroups/... Joined February 2020

101 Following 159 Followers

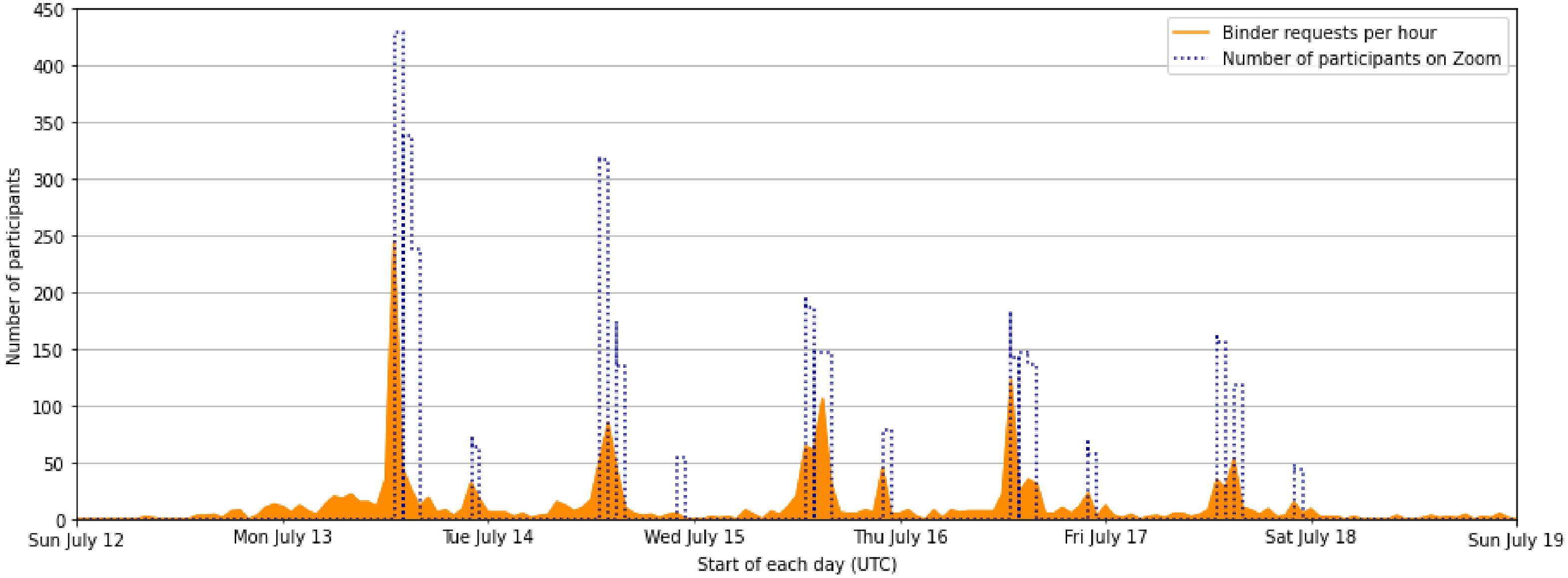
- ❑ **Questions & answers with slido**
  - (AFAIK we were among the very first to try Slido in HEP)
  - Used *slido* to crowd-source questions, to prioritise the most popular ones upvoted by participants
  - Session chair shares link to questions at end of presentation
  - Most popular ones get answered/discussed
  - At end of Q&A all questions are copied to Slack in the appropriate topical channel
    ⇒ participants can continue to discuss and exchange
  - A few polls also run via slido

- ❑ **Post-conference:** participants encouraged to join the PyHEP Gitter channel(s)

PyHEP 2020 – session attendance & Binder usage

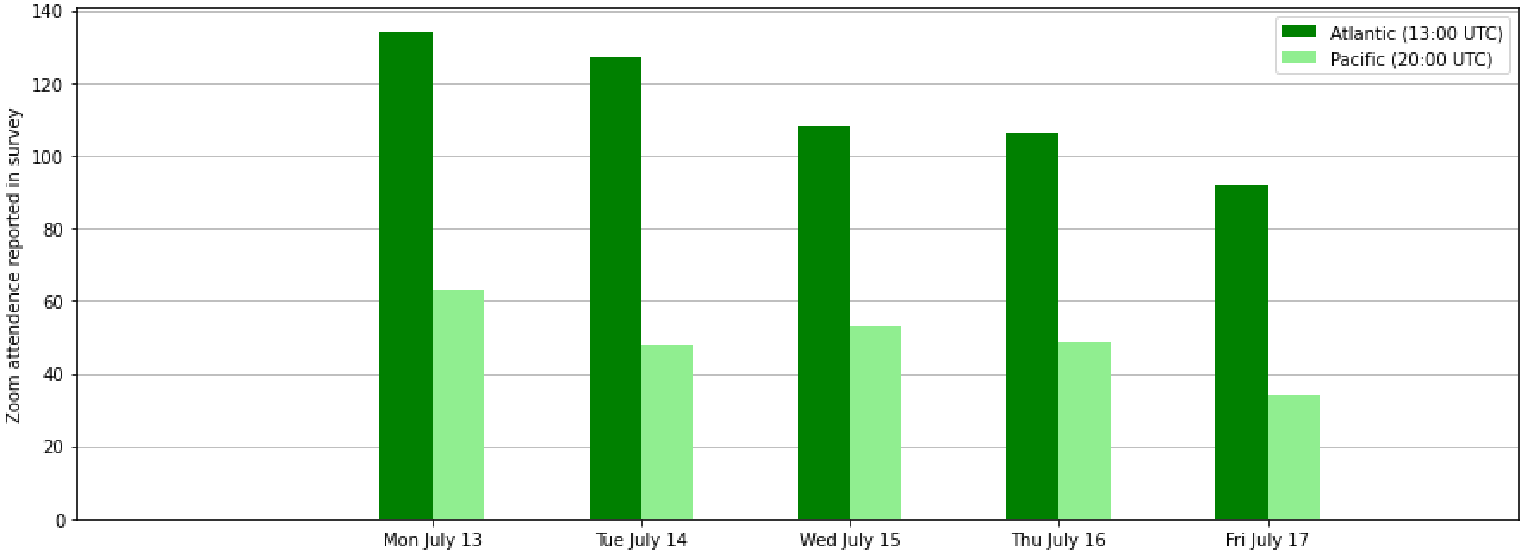
- ❑ **Session participants**
- ❑ **Binder requests during sessions**

⇒ **Clear correlation !**



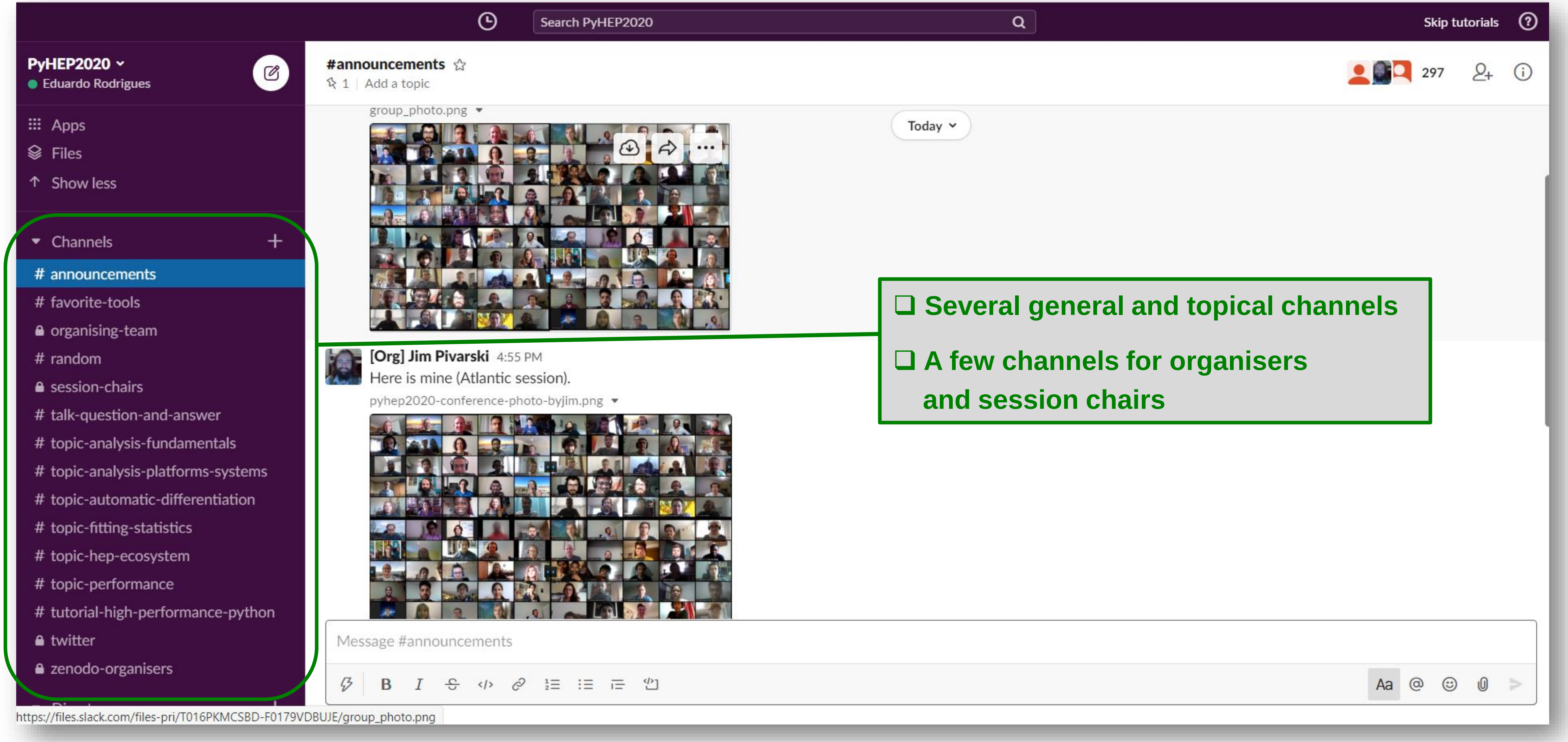
- ❑ **Number of participants per day & time zone,** ***as reported by those who filled in the post-workshop survey***

  - “Atlantic” time zone suited most



Study by Jim Pivarski

PyHEP 2020 – Slack for discussion during *and* after sessions



- # announcements
- # favorite-tools
- organising-team
- # random
- session-chairs
- # talk-question-and-answer
- # topic-analysis-fundamentals
- # topic-analysis-platforms-systems
- # topic-automatic-differentiation
- # topic-fitting-statistics
- # topic-hep-ecosystem
- # topic-performance
- # tutorial-high-performance-python
- twitter
- zenodo-organisers

#announcements

group_photo.png

[Org] Jim Pivarski 4:55 PM

Here is mine (Atlantic session).

pyhep2020-conference-photo-byjim.png

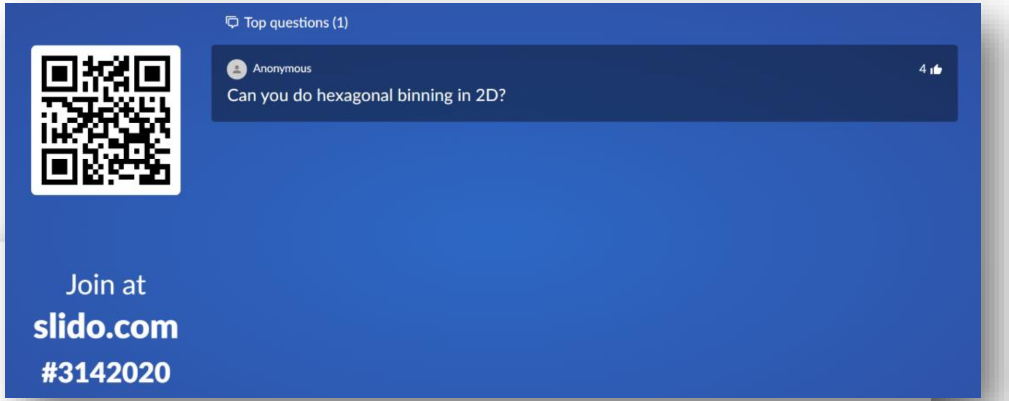
Message #announcements

https://files.slack.com/files-pri/T016PKMCSBD-F0179VDBUJE/group_photo.png

- ❑ Several general and topical channels
- ❑ A few channels for organisers and session chairs

PyHEP 2020 logistics – slido for Q&A post-talk sessions

As actually seen by participants:



Top questions (1)

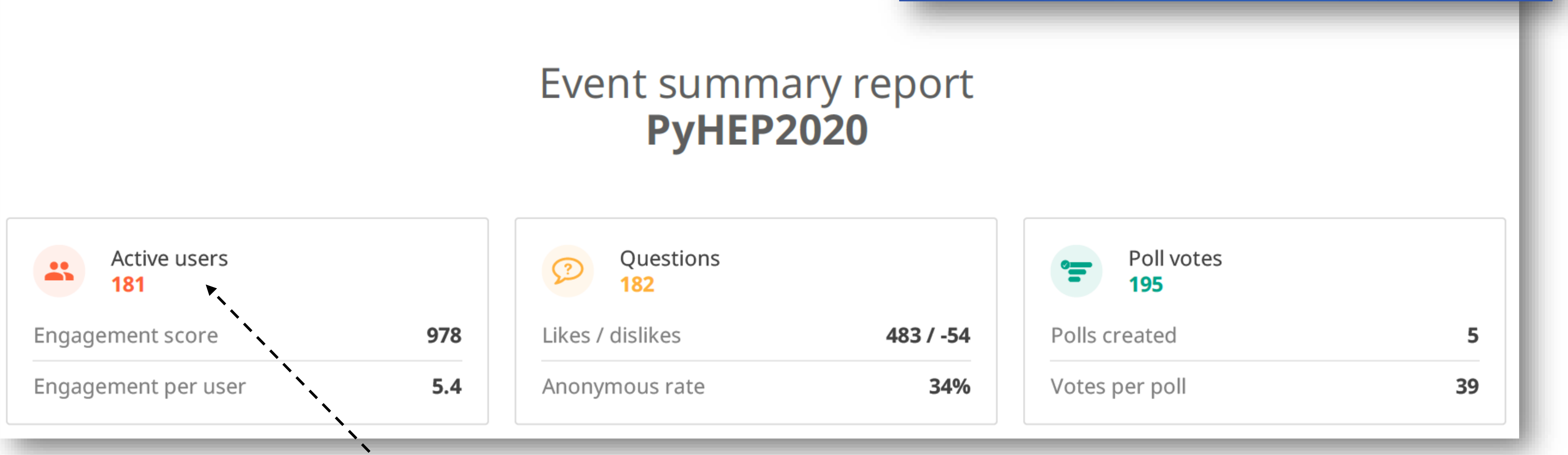
Anonymous 4

Can you do hexagonal binning in 2D?

Join at
slido.com
#3142020

Was slido a success? Yes !

slido



Event summary report
PyHEP2020

Active users	181
Engagement score	978
Engagement per user	5.4

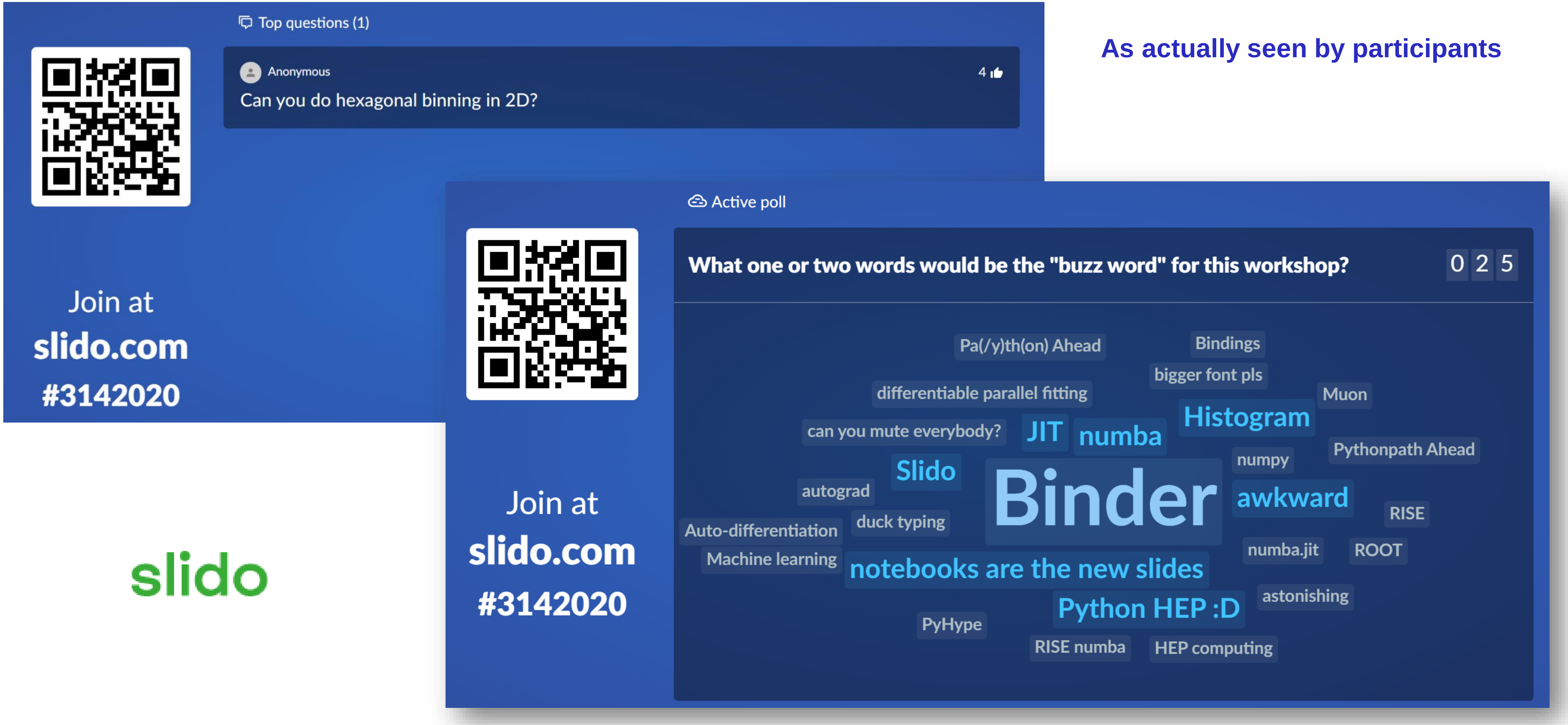
Questions	182
Likes / dislikes	483 / -54
Anonymous rate	34%

Poll votes	195
Polls created	5
Votes per poll	39

With 413 joined participants in total

PyHEP 2020 logistics – slido at work for Q&As and polls

As actually seen by participants



Top questions (1)

Anonymous

Can you do hexagonal binning in 2D?

4

Join at
slido.com
#3142020

Active poll

What one or two words would be the "buzz word" for this workshop?

0 2 5

Pa(/y)th(on) Ahead

Bindings

bigger font pls

differentiable parallel fitting

Muon

Histogram

can you mute everybody?

JIT

numba

Pythonpath Ahead

numpy

Slido

autograd

Binder

awkward

RISE

Auto-differentiation

duck typing

numba.jit

ROOT

Machine learning

notebooks are the new slides

astonishing

Python HEP :D

PyHype

RISE numba

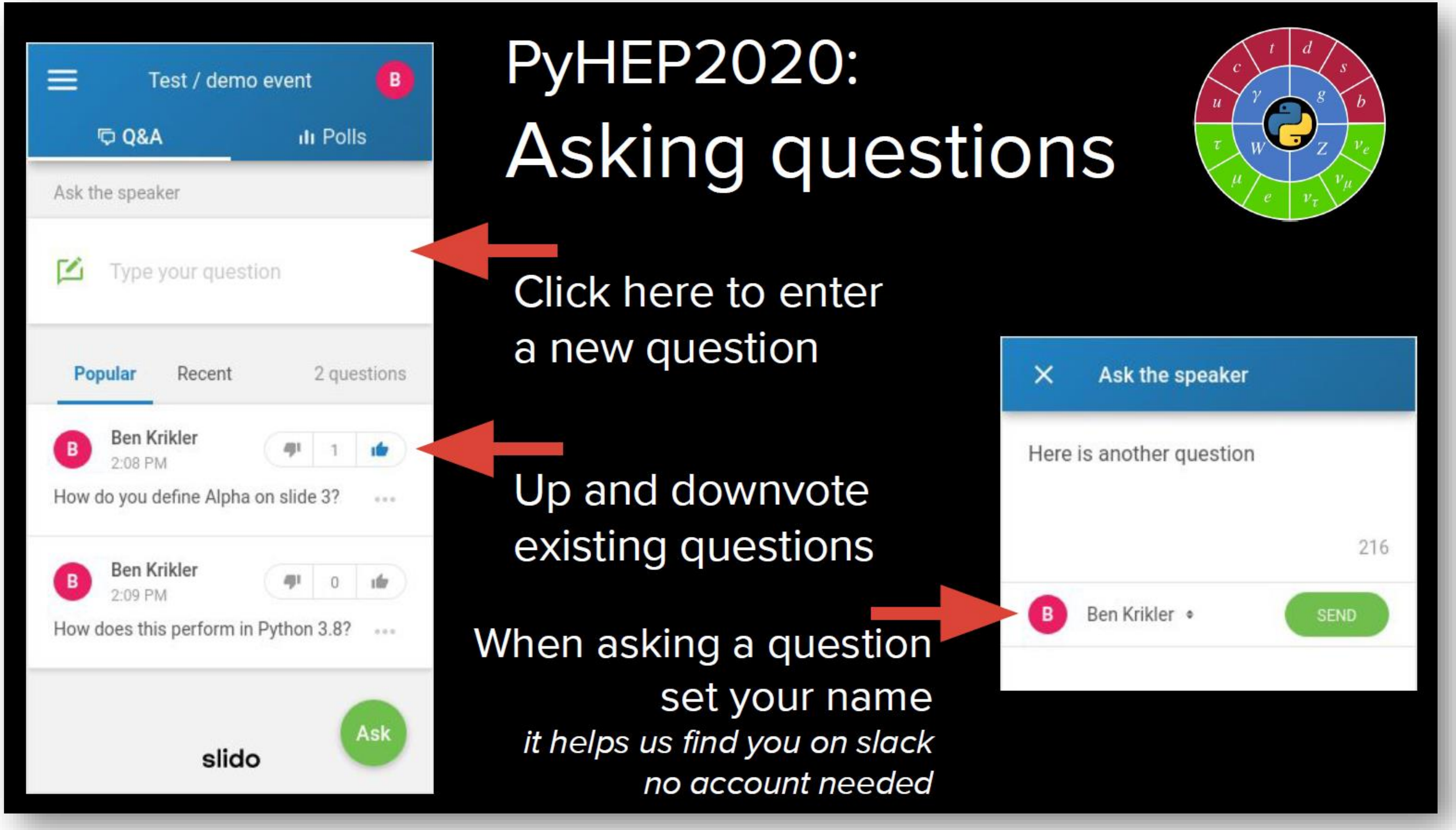
HEP computing

Join at
slido.com
#3142020

slido

PyHEP 2020 logistics – how does slido work for Q&As

slido ✓ Easy to use ✓ Works with your live video ✓ No app downloads



PyHEP2020:

Asking questions

Test / demo event

Q&A | Polls

Ask the speaker

Type your question

Popular | Recent | 2 questions

Ben Krikler 2:08 PM 1
How do you define Alpha on slide 3?

Ben Krikler 2:09 PM 0
How does this perform in Python 3.8?

Ask

slido

Click here to enter a new question

Up and downvote existing questions

When asking a question set your name

it helps us find you on slack
no account needed

Ask the speaker

Here is another question

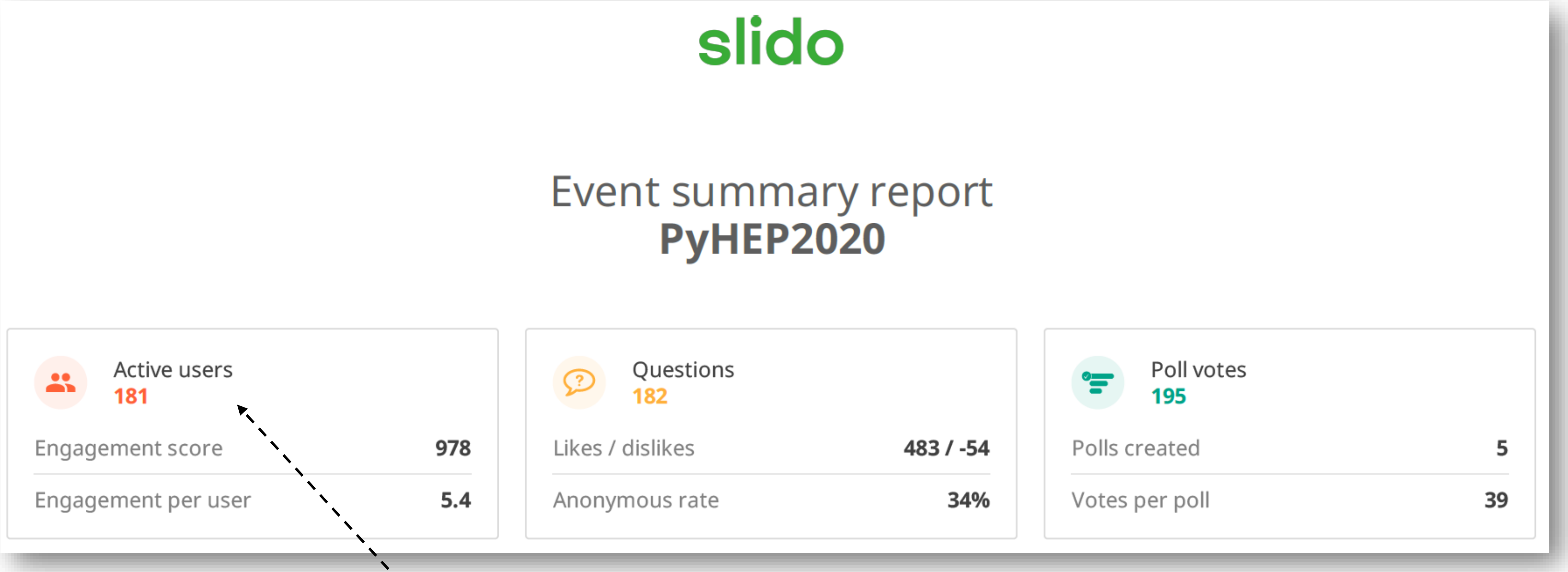
216

Ben Krikler

SEND

PyHEP 2020 logistics – slido for Q&A post-talk sessions

Was slido a success? Yes !



slido

Event summary report
PyHEP2020

Active users	181
Engagement score	978
Engagement per user	5.4

Questions	182
Likes / dislikes	483 / -54
Anonymous rate	34%

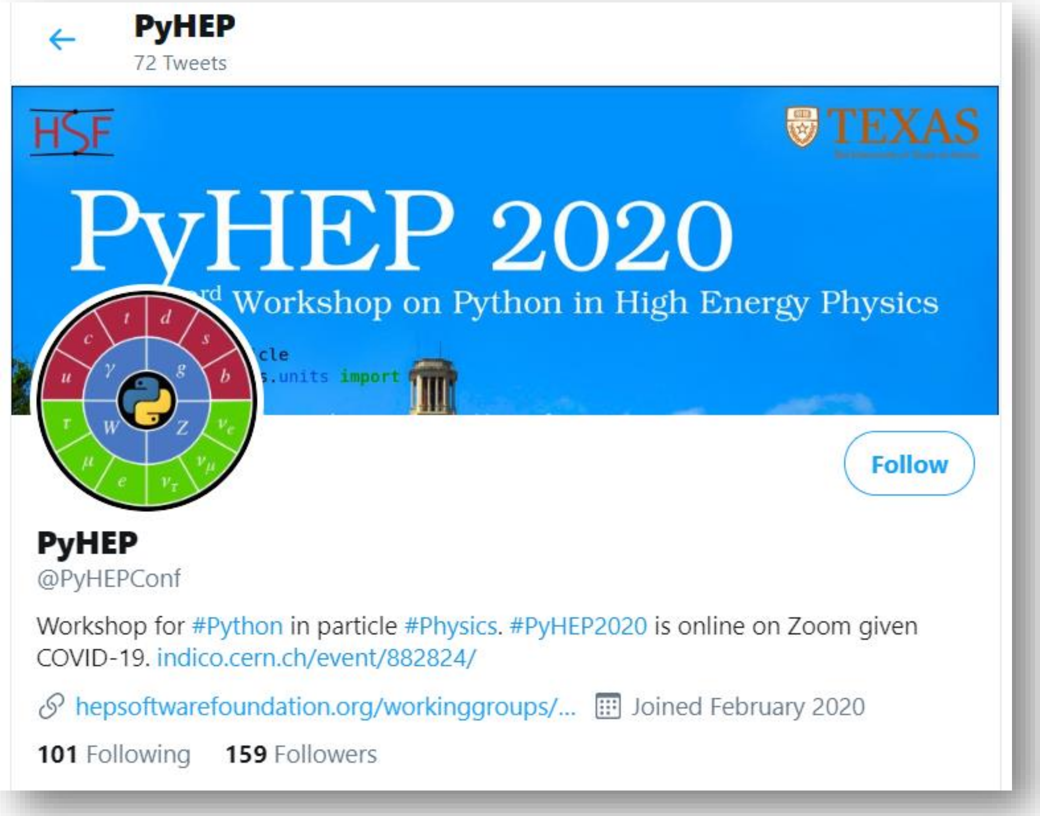
Poll votes	195
Polls created	5
Votes per poll	39

With 413 joined participants in total

PyHEP 2020 logistics – we are even on

@PyHEPConf

#PyHEP2020


PyHEP
72 Tweets

HSF
TEXAS
PyHEP 2020
3rd Workshop on Python in High Energy Physics
cle
s.units import
t d c s u γ g b τ W Z ν_e μ e ν_τ ν_μ

Follow

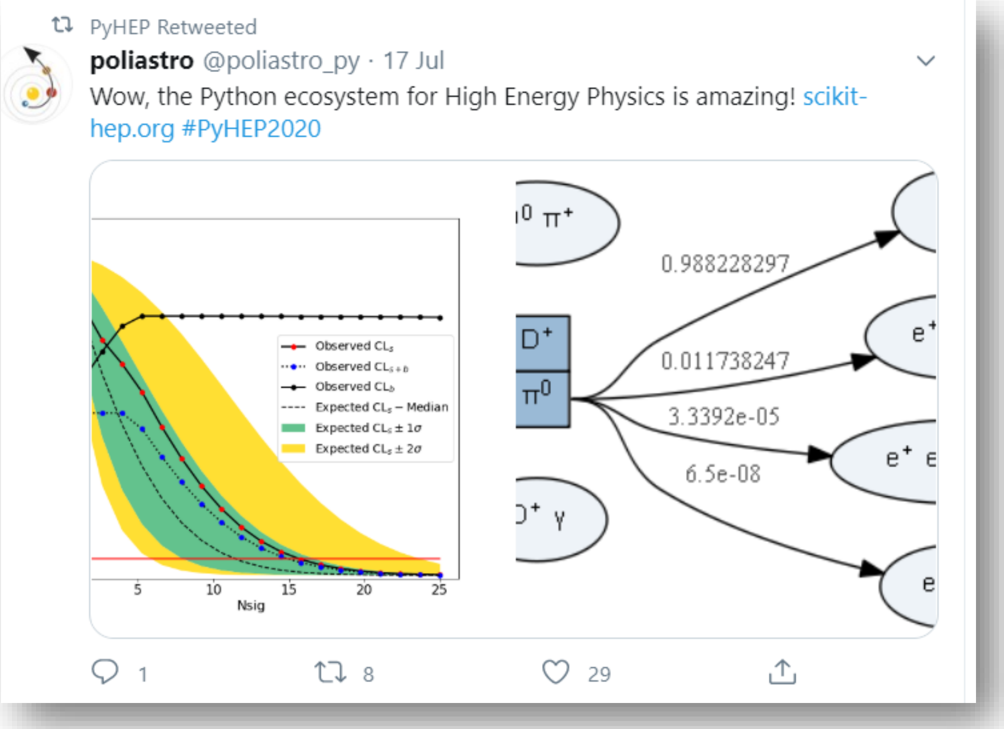
PyHEP
@PyHEPConf

Workshop for #Python in particle #Physics. #PyHEP2020 is online on Zoom given COVID-19. indico.cern.ch/event/882824/

hepsoftwarefoundation.org/workinggroups/... Joined February 2020

101 Following **159** Followers

A testimony from an astroparticle colleague …


PyHEP Retweeted

poliastro @poliastro_py · 17 Jul

Wow, the Python ecosystem for High Energy Physics is amazing! scikit-hep.org #PyHEP2020

1 8 29

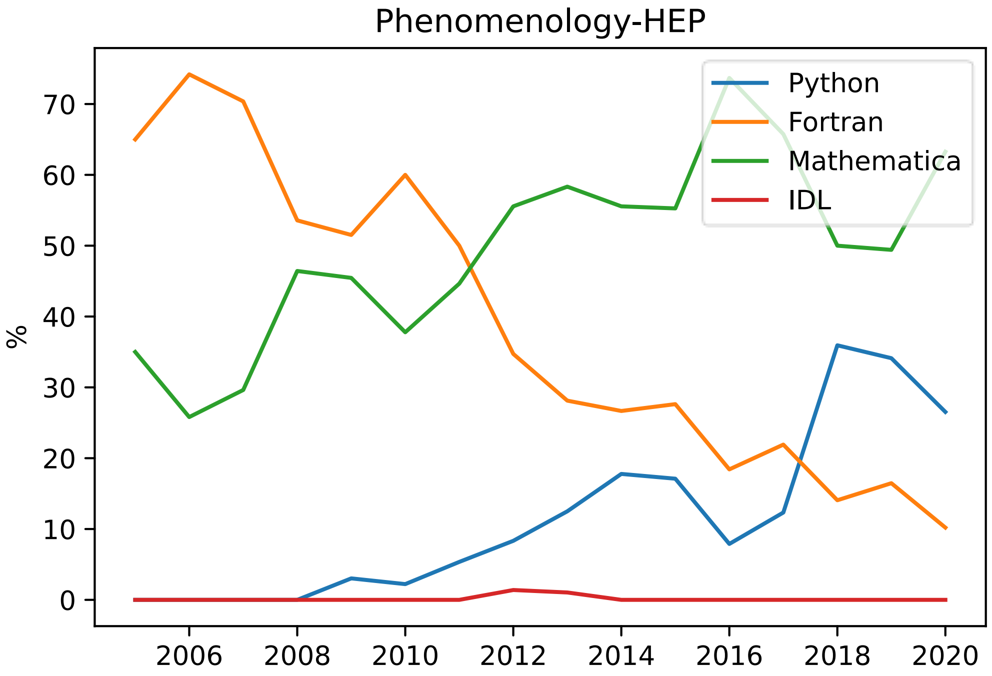
PyHEP 2020 – "Proceedings"

- ❑ No (standard) proceedings per se
- ❑ All presentational material posted on workshop agenda
  and later given a DOI with Zenodo, in a dedicated "pyhep2020 community" – formal citation, replaces proceedings
  - Indico contains links to slides, notebook repositories, Binder launch buttons, YouTube recordings
- ❑ With Zenodo + Binder, all code from the workshop should be reproducible into the future
  ⇒ "living workshop proceedings"!
- ❑ Recordings on YouTube are in some way an alternative to proceedings

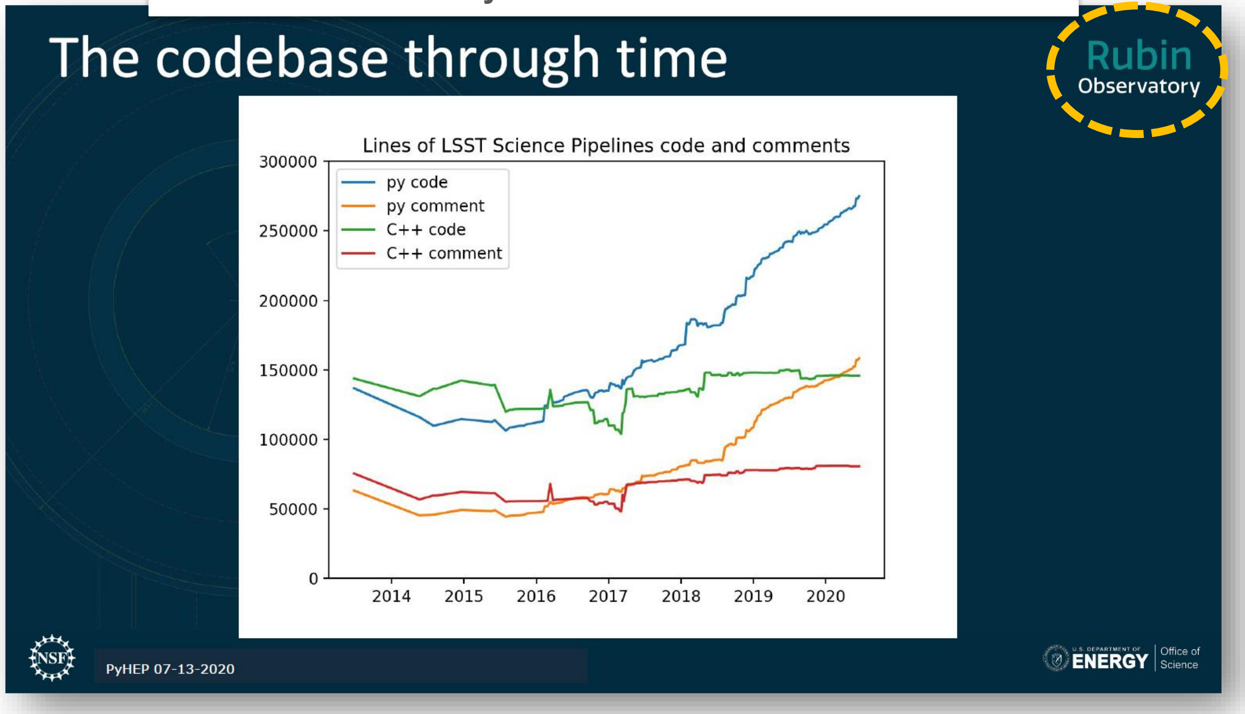
PyHEP 2020 – keynote presentations

- ❑ **Python on the rise not just in experimental particle physics**

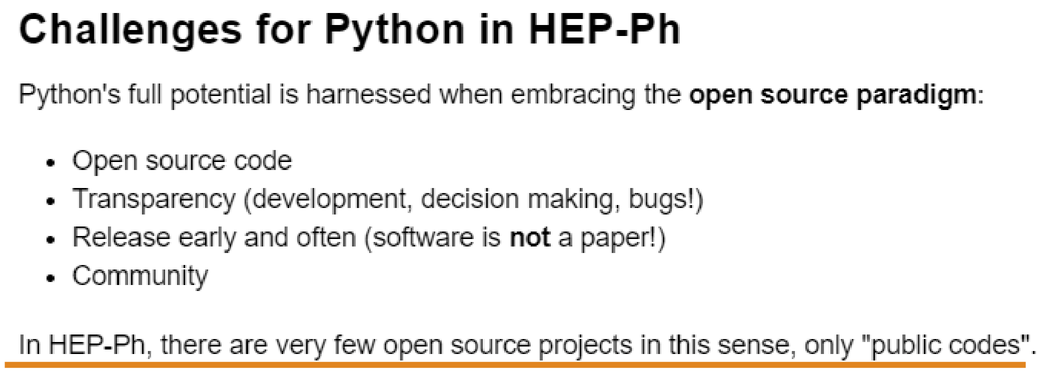
David Straub (flavour phenomenologist)
“Python & HEP: a perfect match, in theory”



Nate Lust (astronomy)
“Rubin Observatory: The software behind the science”


The codebase through time

PyHEP 07-13-2020


Challenges for Python in HEP-Ph

Python's full potential is harnessed when embracing the **open source paradigm**:

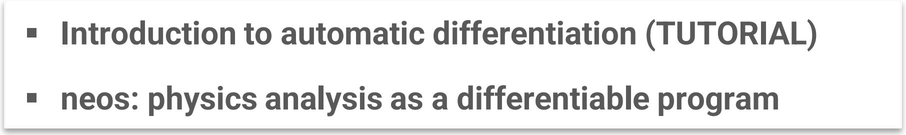
- Open source code
- Transparency (development, decision making, bugs!)
- Release early and often (software is **not** a paper!)
- Community

In HEP-Ph, there are very few open source projects in this sense, only "public codes".

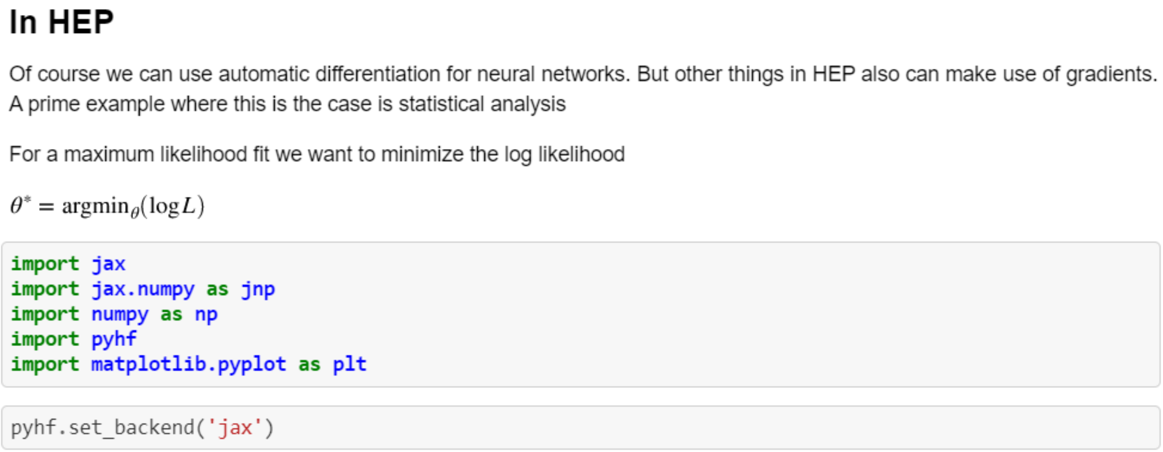
PyHEP 2020 – on trends

“ Automatic differentiation is a method to compute exact derivatives of functions implements as **programs**. It's a widely applicable method and famously is used in many Machine learning optimization problems.”

- ❑ **Auto-differentiation, specifically in the context of differentiable analysis, came out as an unforeseen “theme“ and a new direction**
  - 1 tutorial and 1 talk on the subject



- Introduction to automatic differentiation (TUTORIAL)
- neos: physics analysis as a differentiable program



In HEP

Of course we can use automatic differentiation for neural networks. But other things in HEP also can make use of gradients. A prime example where this is the case is statistical analysis

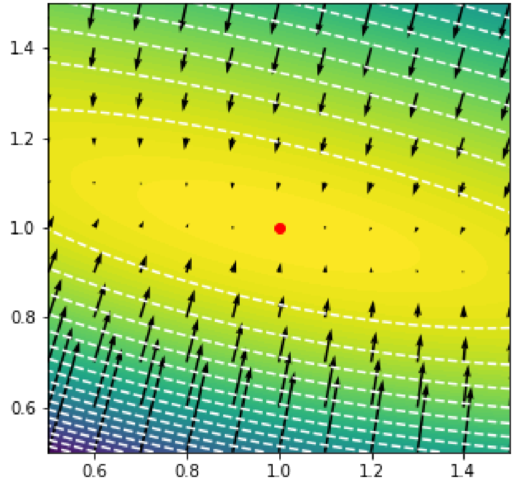
For a maximum likelihood fit we want to minimize the log likelihood

$\theta^* = \mathrm{argmin}_\theta(\log L)$

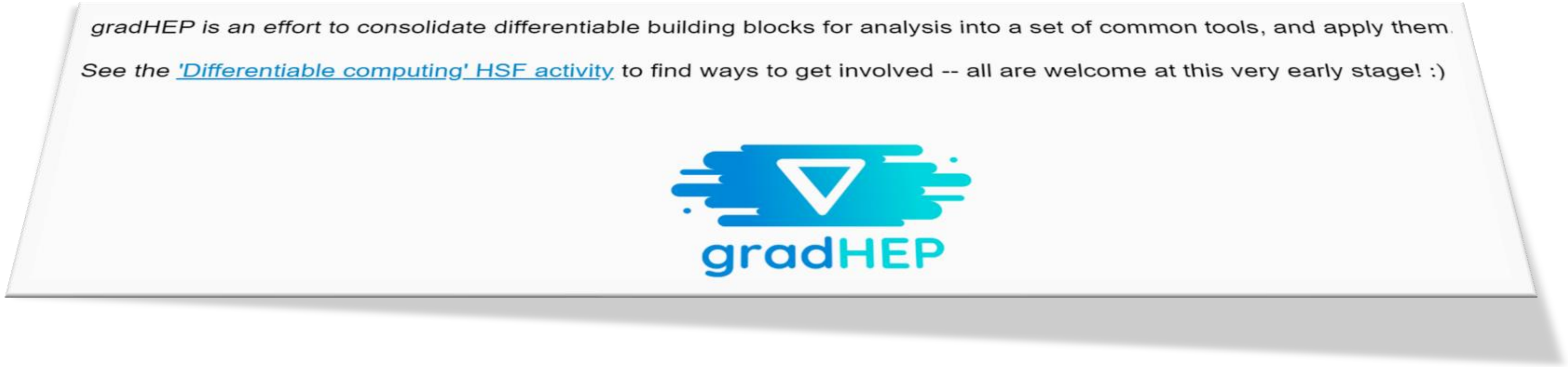
```python
import jax
import jax.numpy as jnp
import numpy as np
import pyhf
import matplotlib.pyplot as plt
```

```python
pyhf.set_backend('jax')
```

Define the model, fit … and plot:



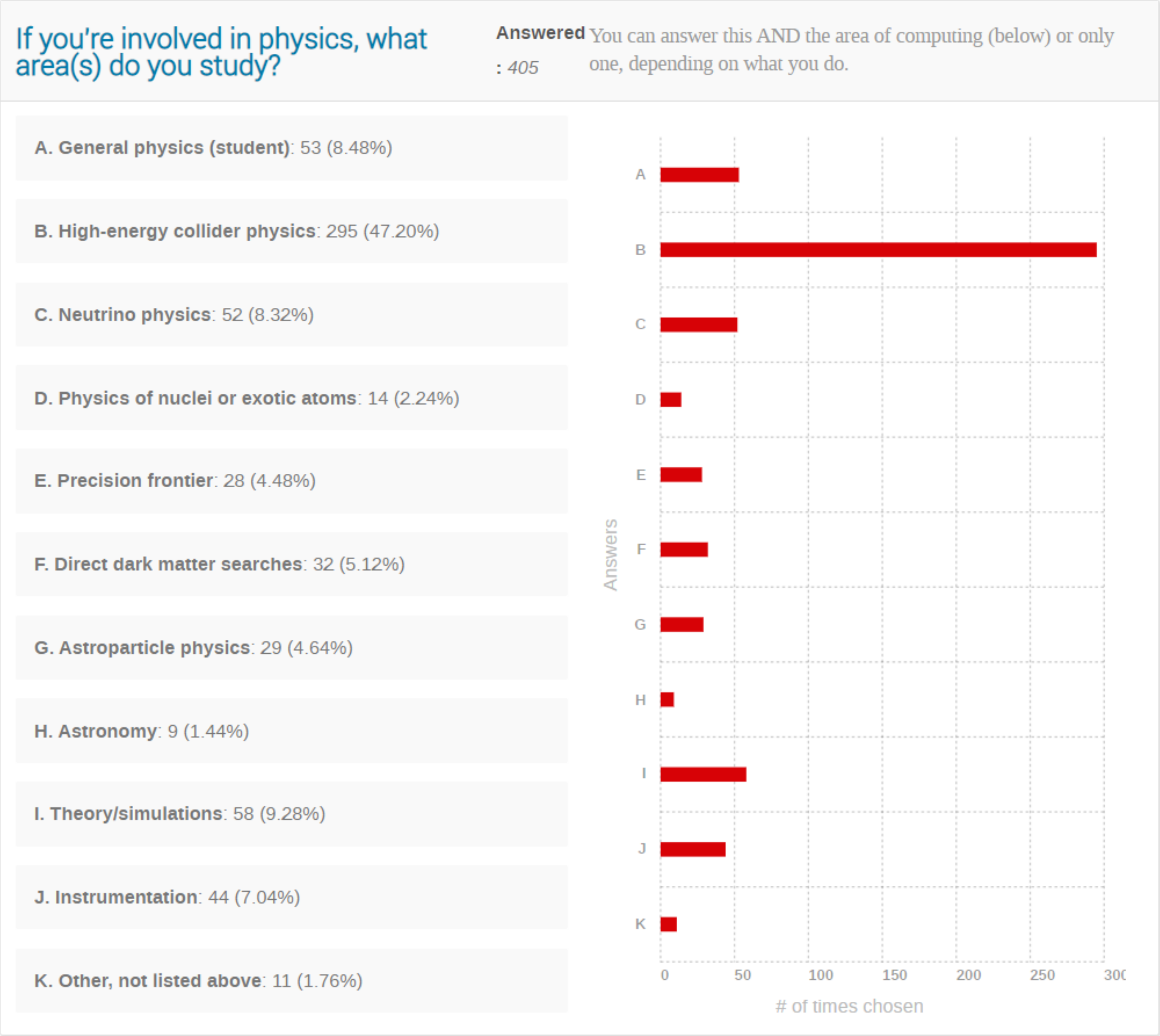
(Taken from the tutorial)



gradHEP is an effort to consolidate differentiable building blocks for analysis into a set of common tools, and apply them.

See the 'Differentiable computing' HSF activity to find ways to get involved -- all are welcome at this very early stage! :)

PyHEP 2020 – background of participants …



If you're involved in physics, what area(s) do you study?

Answered: *405*

You can answer this AND the area of computing (below) or only one, depending on what you do.

A. General physics (student): 53 (8.48%)

B. High-energy collider physics: 295 (47.20%)

C. Neutrino physics: 52 (8.32%)

D. Physics of nuclei or exotic atoms: 14 (2.24%)

E. Precision frontier: 28 (4.48%)

F. Direct dark matter searches: 32 (5.12%)

G. Astroparticle physics: 29 (4.64%)

H. Astronomy: 9 (1.44%)

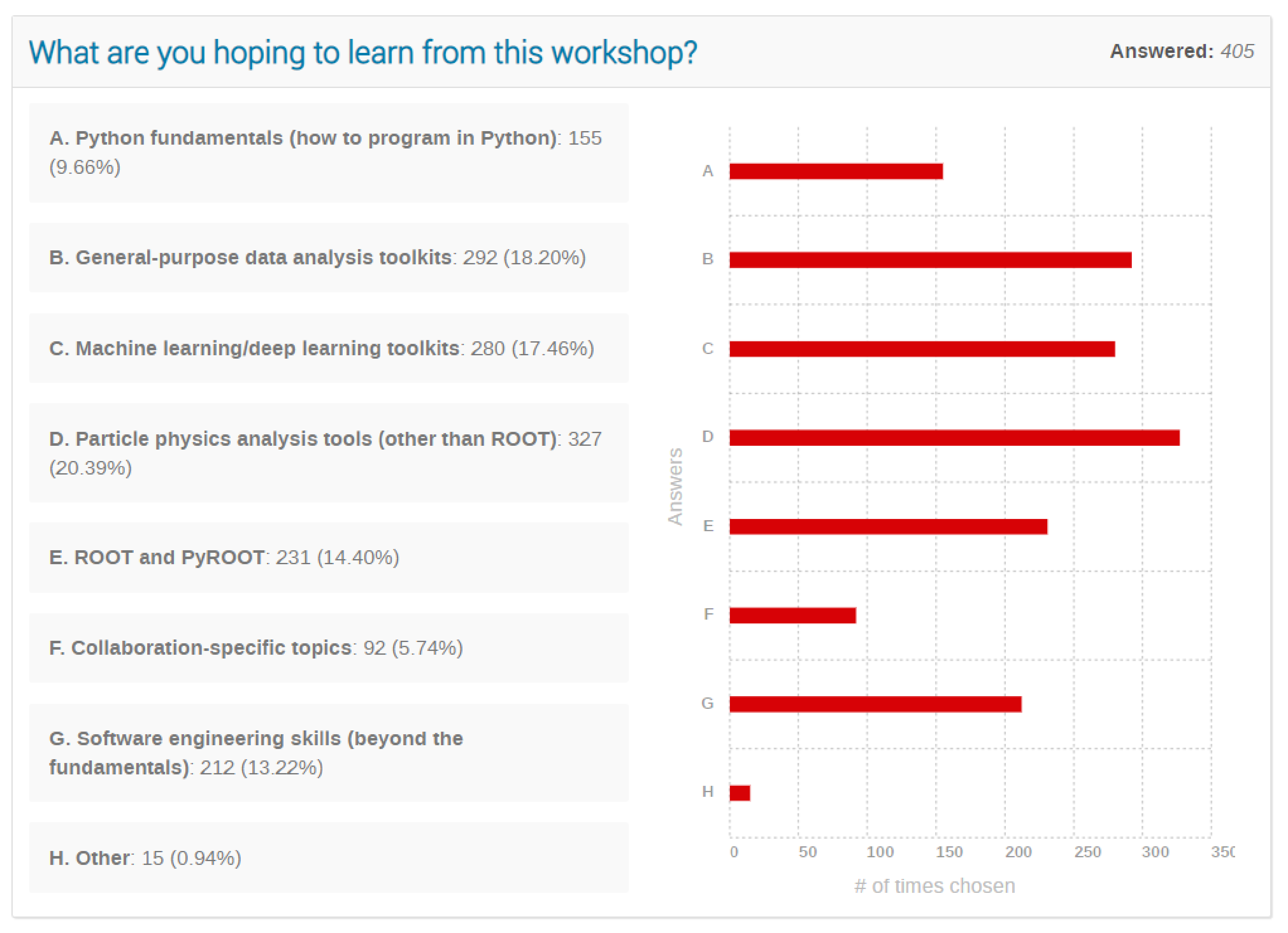
I. Theory/simulations: 58 (9.28%)

J. Instrumentation: 44 (7.04%)

K. Other, not listed above: 11 (1.76%)

Taken from the pre-workshop survey (408 respondents)

PyHEP 2020 – … and their hopes



What are you hoping to learn from this workshop?

Answered: *405*

A. Python fundamentals (how to program in Python): 155 (9.66%)

B. General-purpose data analysis toolkits: 292 (18.20%)

C. Machine learning/deep learning toolkits: 280 (17.46%)

D. Particle physics analysis tools (other than ROOT): 327 (20.39%)

E. ROOT and PyROOT: 231 (14.40%)

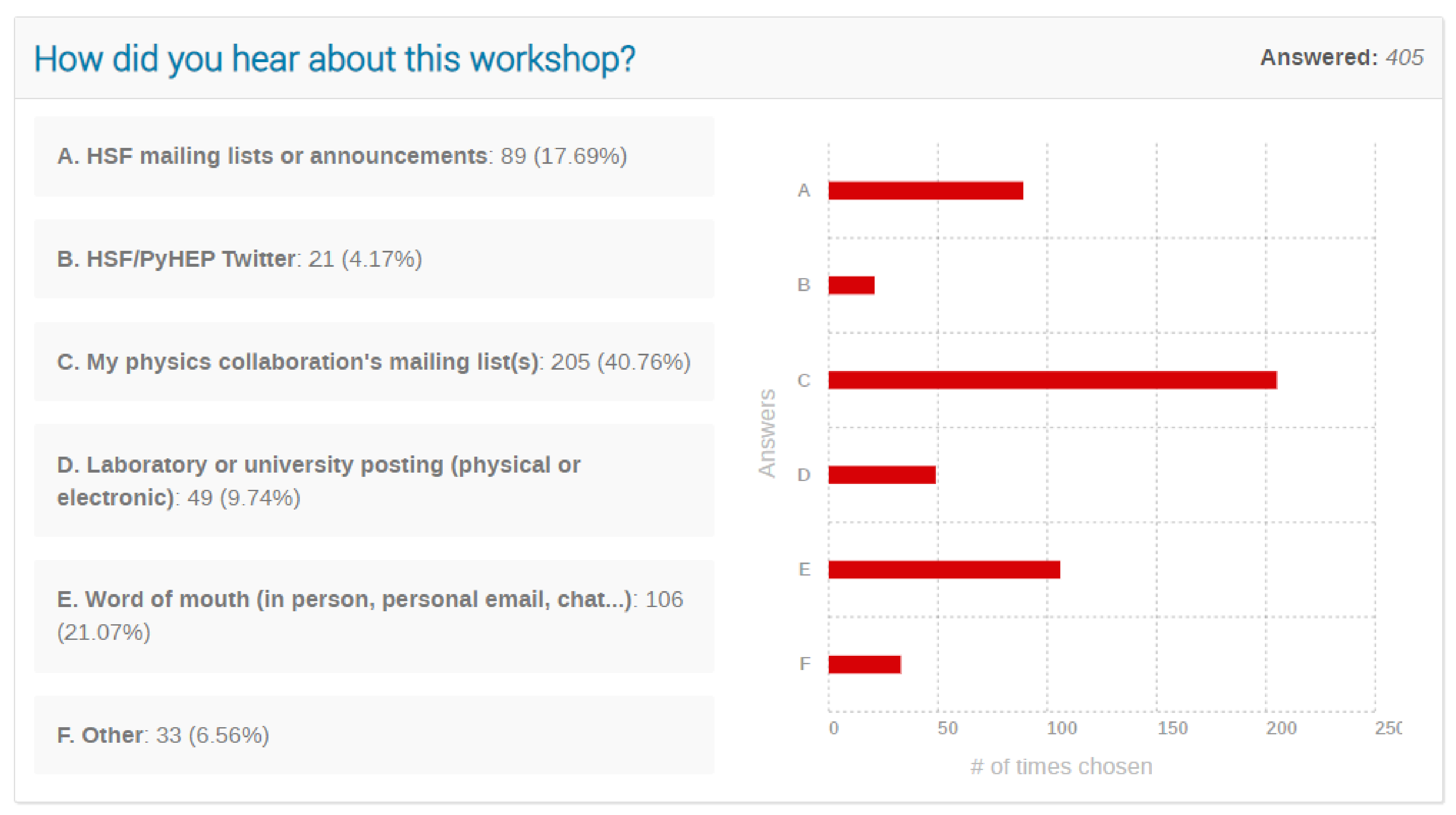
F. Collaboration-specific topics: 92 (5.74%)

G. Software engineering skills (beyond the fundamentals): 212 (13.22%)

H. Other: 15 (0.94%)

Taken from the pre-workshop survey (408 respondents)

PyHEP 2020 organisational aspects – multi-channel advertising is crucial



How did you hear about this workshop?

Answered: *405*

A. HSF mailing lists or announcements: 89 (17.69%)

B. HSF/PyHEP Twitter: 21 (4.17%)

C. My physics collaboration's mailing list(s): 205 (40.76%)

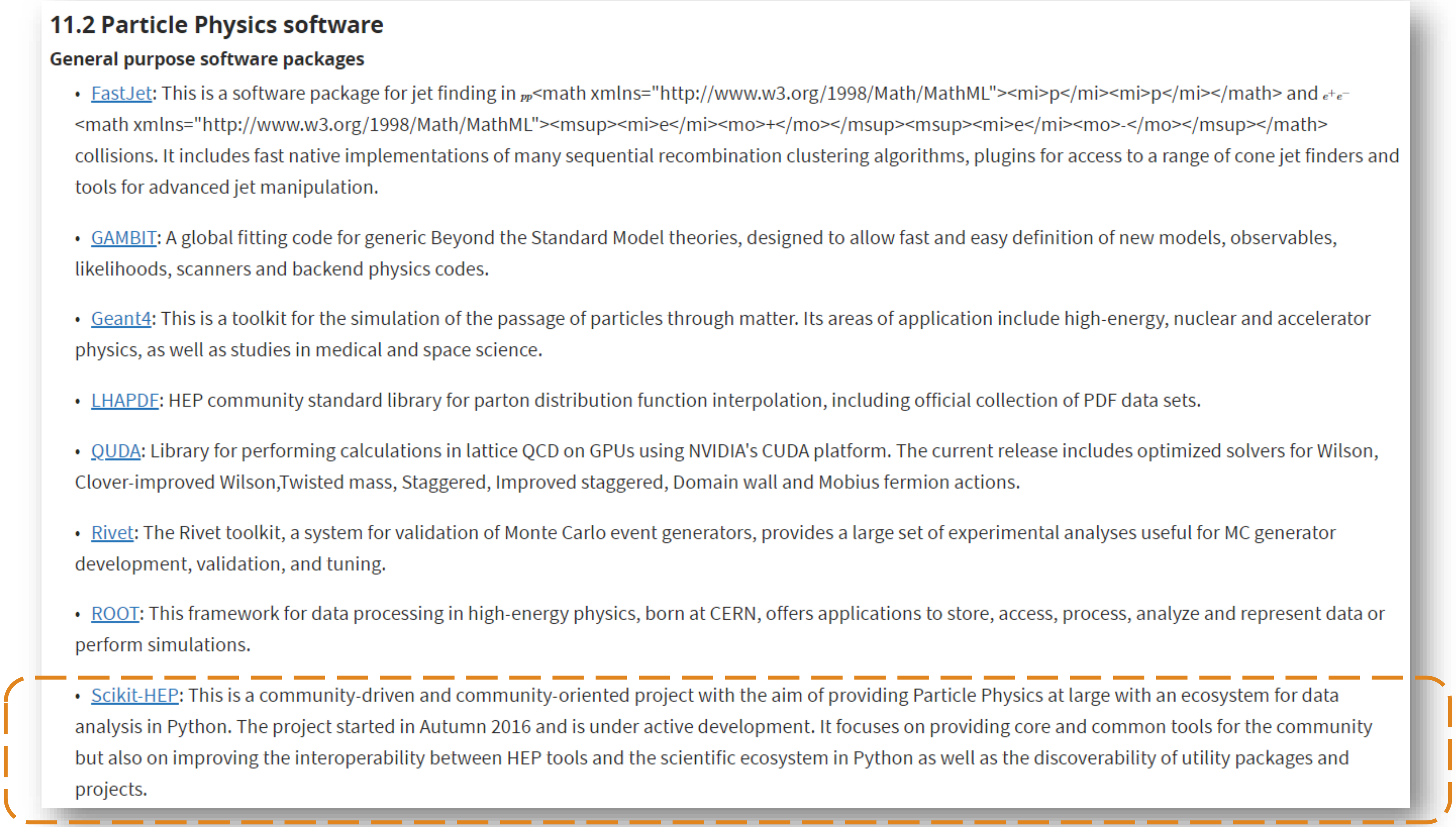
D. Laboratory or university posting (physical or electronic): 49 (9.74%)

E. Word of mouth (in person, personal email, chat...): 106 (21.07%)

F. Other: 33 (6.56%)

Taken from the pre-workshop survey (408 respondents)

Scikit-HEP made it to the PDG !



11.2 Particle Physics software

General purpose software packages

- FastJet: This is a software package for jet finding in pp<math xmlns="http://www.w3.org/1998/Math/MathML"><mi>p</mi><mi>p</mi></math> and e^+e^- <math xmlns="http://www.w3.org/1998/Math/MathML"><msup><mi>e</mi><mo>+</mo></msup><msup><mi>e</mi><mo>-</mo></msup></math> collisions. It includes fast native implementations of many sequential recombination clustering algorithms, plugins for access to a range of cone jet finders and tools for advanced jet manipulation.

- GAMBIT: A global fitting code for generic Beyond the Standard Model theories, designed to allow fast and easy definition of new models, observables, likelihoods, scanners and backend physics codes.

- Geant4: This is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high-energy, nuclear and accelerator physics, as well as studies in medical and space science.

- LHAPDF: HEP community standard library for parton distribution function interpolation, including official collection of PDF data sets.

- QUDA: Library for performing calculations in lattice QCD on GPUs using NVIDIA's CUDA platform. The current release includes optimized solvers for Wilson, Clover-improved Wilson,Twisted mass, Staggered, Improved staggered, Domain wall and Mobius fermion actions.

- Rivet: The Rivet toolkit, a system for validation of Monte Carlo event generators, provides a large set of experimental analyses useful for MC generator development, validation, and tuning.

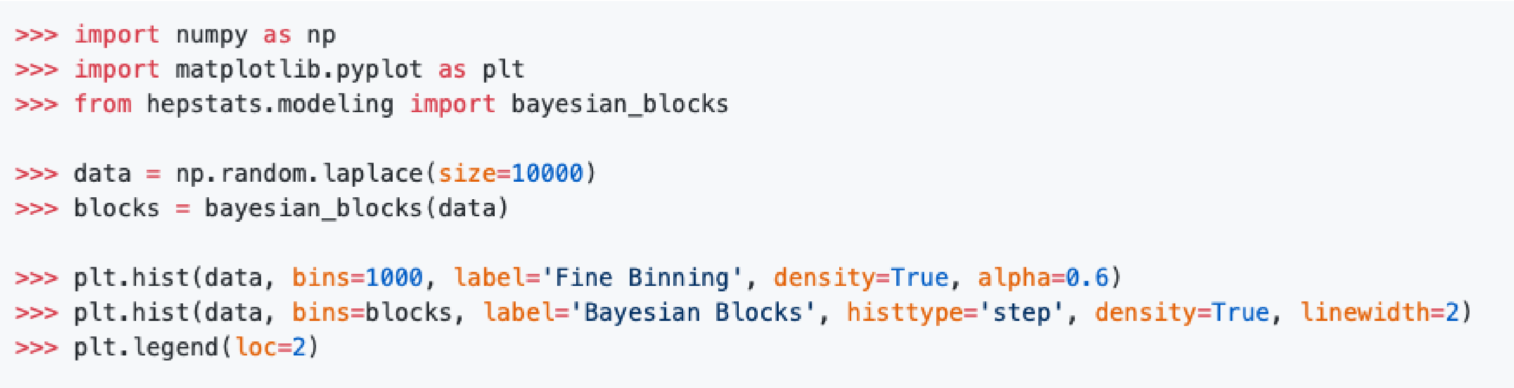
- ROOT: This framework for data processing in high-energy physics, born at CERN, offers applications to store, access, process, analyze and represent data or perform simulations.

- Scikit-HEP: This 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 under active development. It focuses on providing core and common tools for the community but also on improving the interoperability between HEP tools and the scientific ecosystem in Python as well as the discoverability of utility packages and projects.

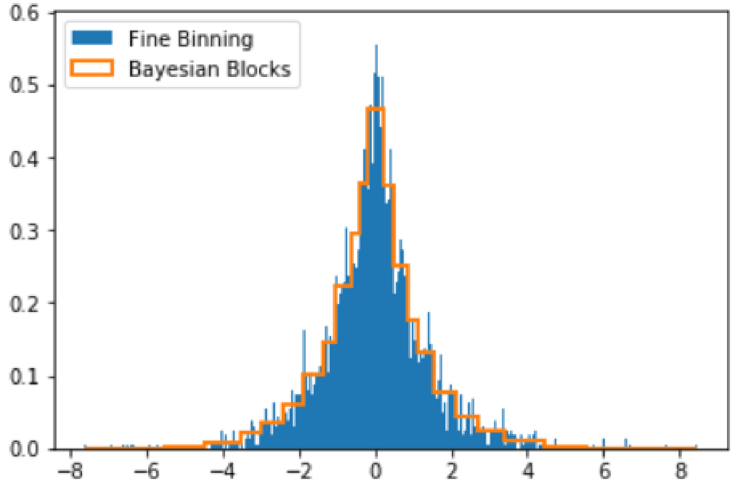
Taken from Online Particle Physics Information (see PDG)

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

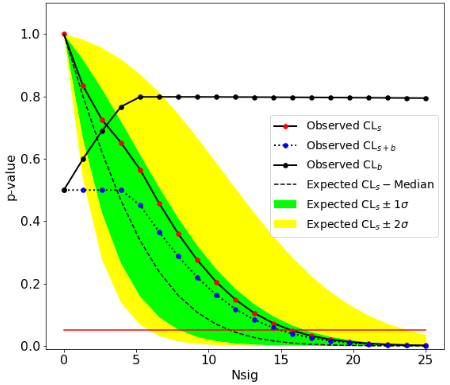


```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from hepstats.modeling import bayesian_blocks

>>> 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)
```

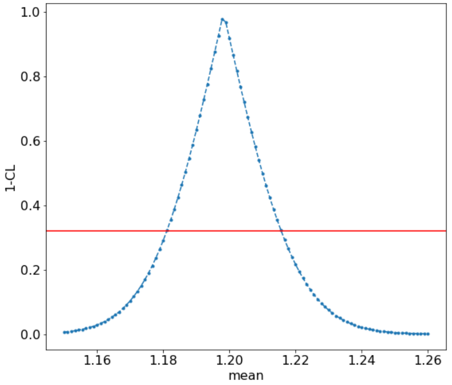


- Likelihood-based hypothesis tests, upper limit and confidence interval calculations
  - Works with a fitting library providing models, likelihood, etc.
  - Built on a common interface, used by zfit, and does not depend on a fitting backend

Upper limit on signal yield:

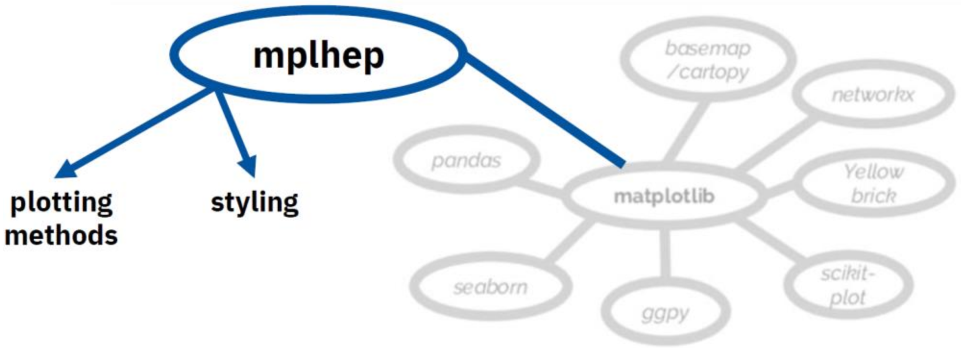


1-CL plot for the mean of a peak:

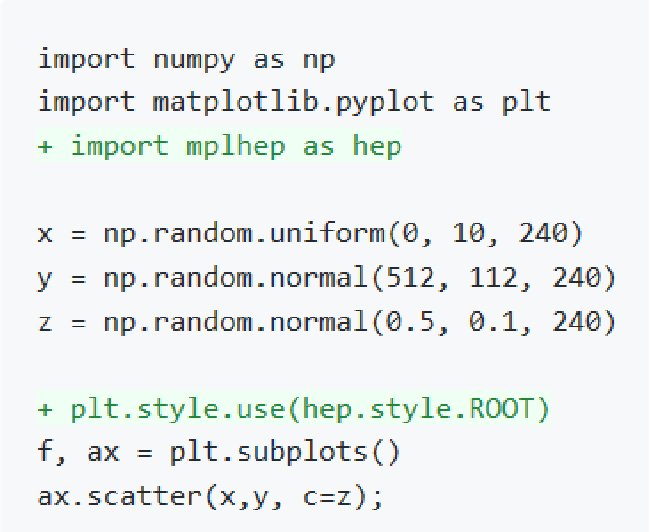


mplhep package – helper visualisation tool for HEP atop Matplotlib

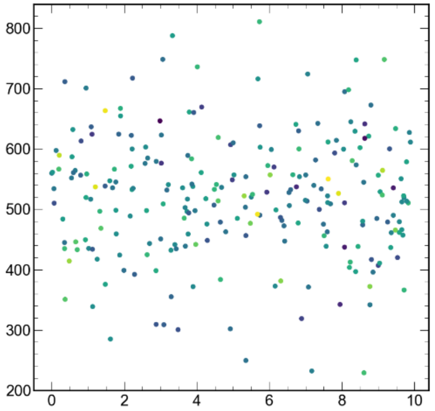
- ❑ Matplotlib is a key tool for visualisation in the data science domain
- ❑ But it not provide all that HEP wants
  - Requires a lot of tinkering
- ❑ mplhep idea:
  - Keep matplotlib as a versatile and well-tested backend
  - Provide a new domain-specific API



Minimal Example

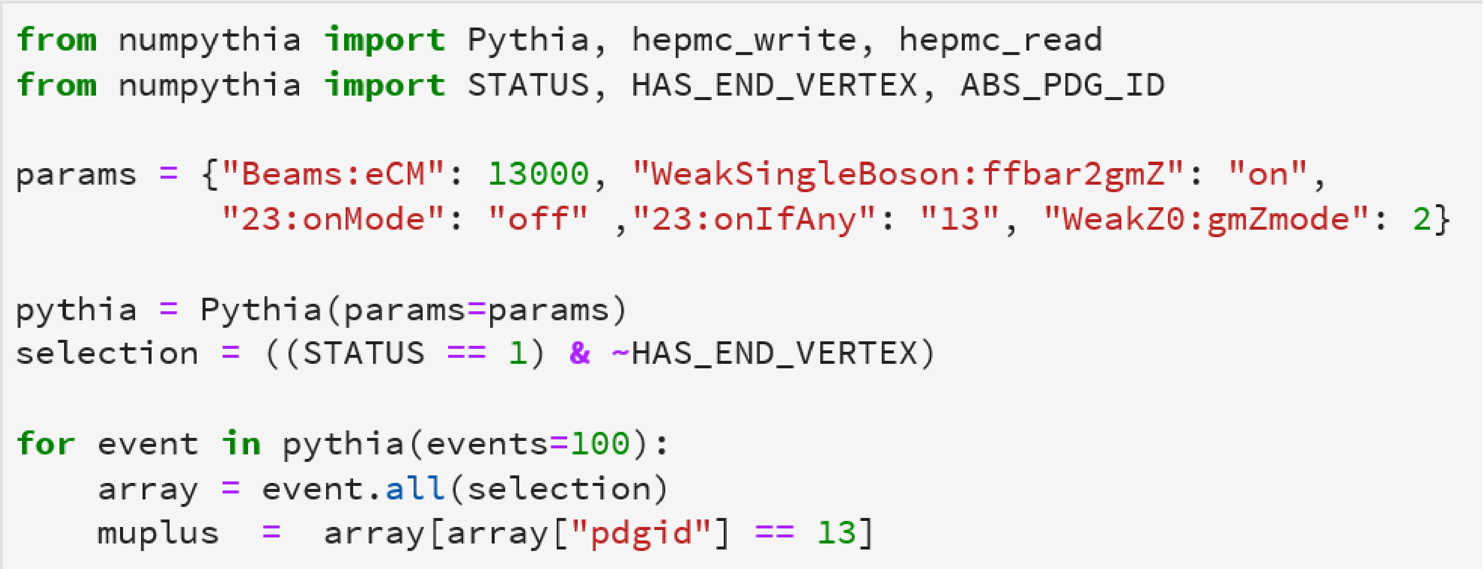


```
import numpy as np
import matplotlib.pyplot as plt
+ import mplhep as hep

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);
```



Simulation & jet clustering – `numpythia` and `pyjet` packages

❑ **Generate events with Pythia and pipe them into NumPy arrays**

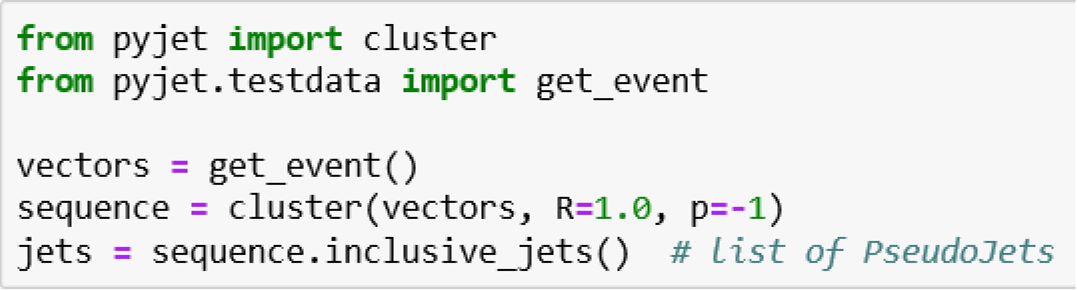


```
from numpythia import Pythia, hepmc_write, hepmc_read
from numpythia import STATUS, HAS_END_VERTEX, ABS_PDG_ID

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

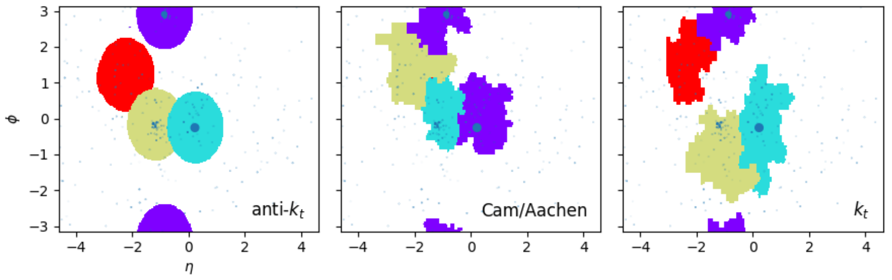
❑ **Possible to feed those events into FastJet using `pyjet`**



```
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
```

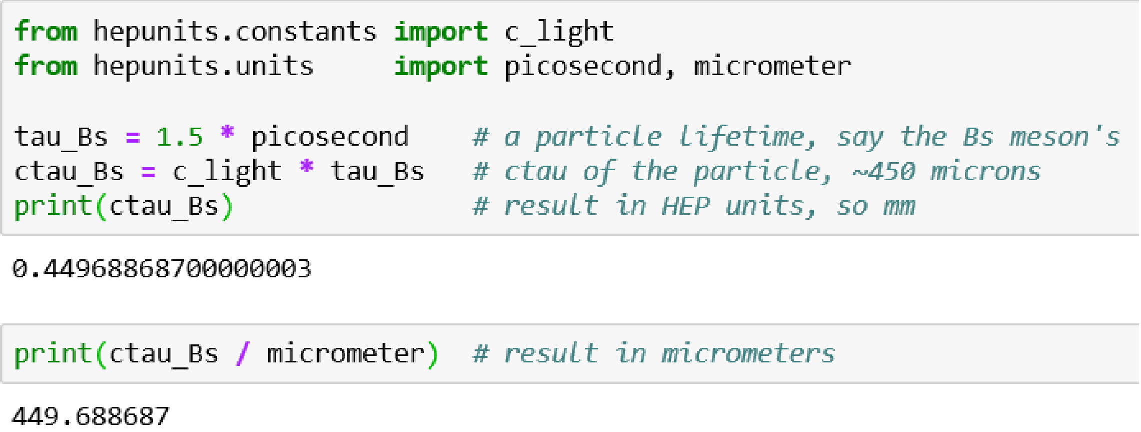
pyjet

Interface between FastJet and NumPy



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:**

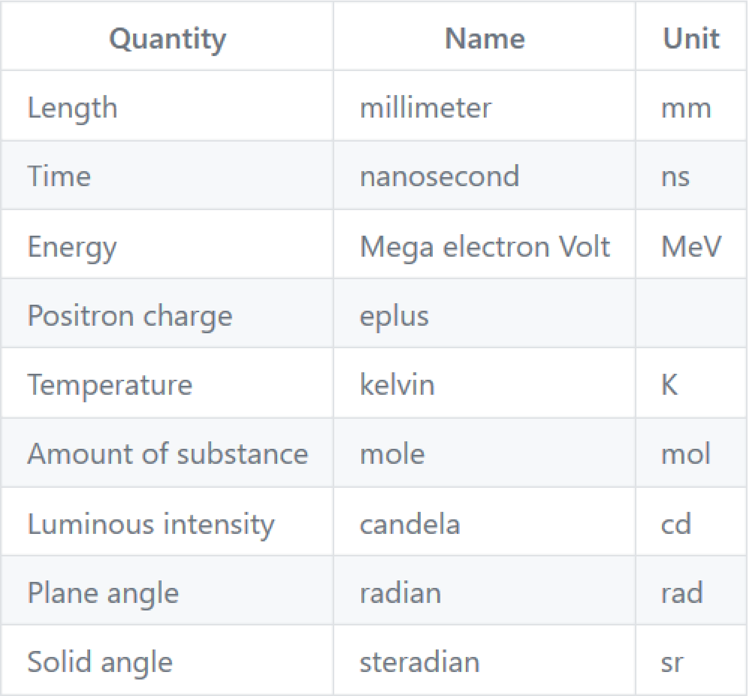


```
from hepunits.constants import c_light
from hepunits.units     import picosecond, micrometer

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
```

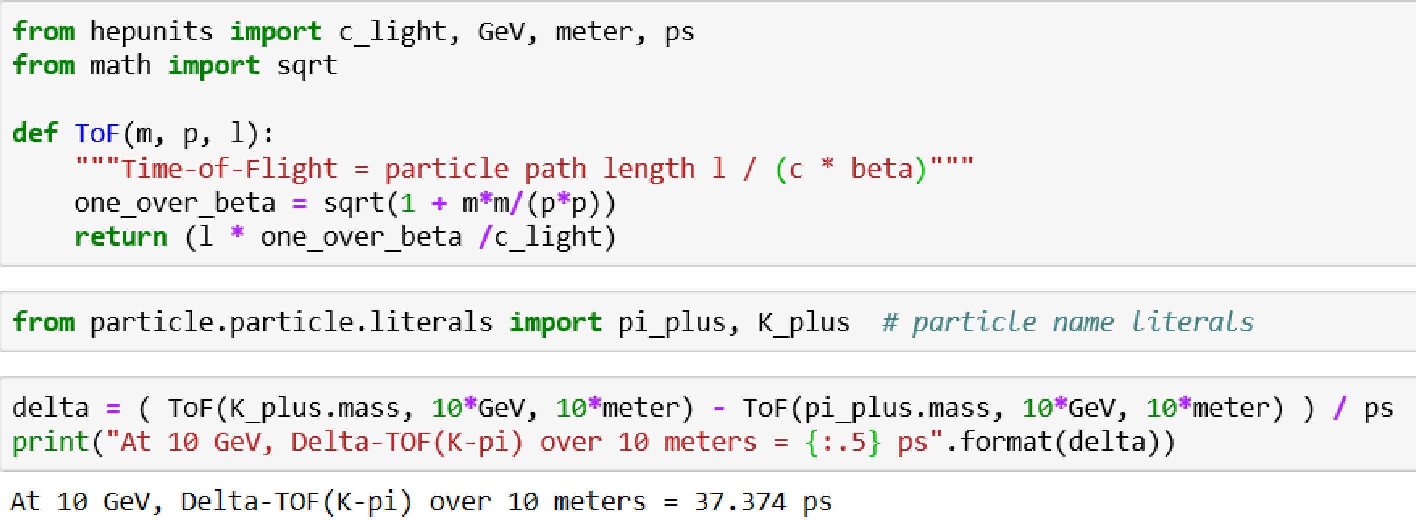
```
print(ctau_Bs / micrometer)  # result in micrometers
```

```
449.688687
```



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

- ❑ **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
```

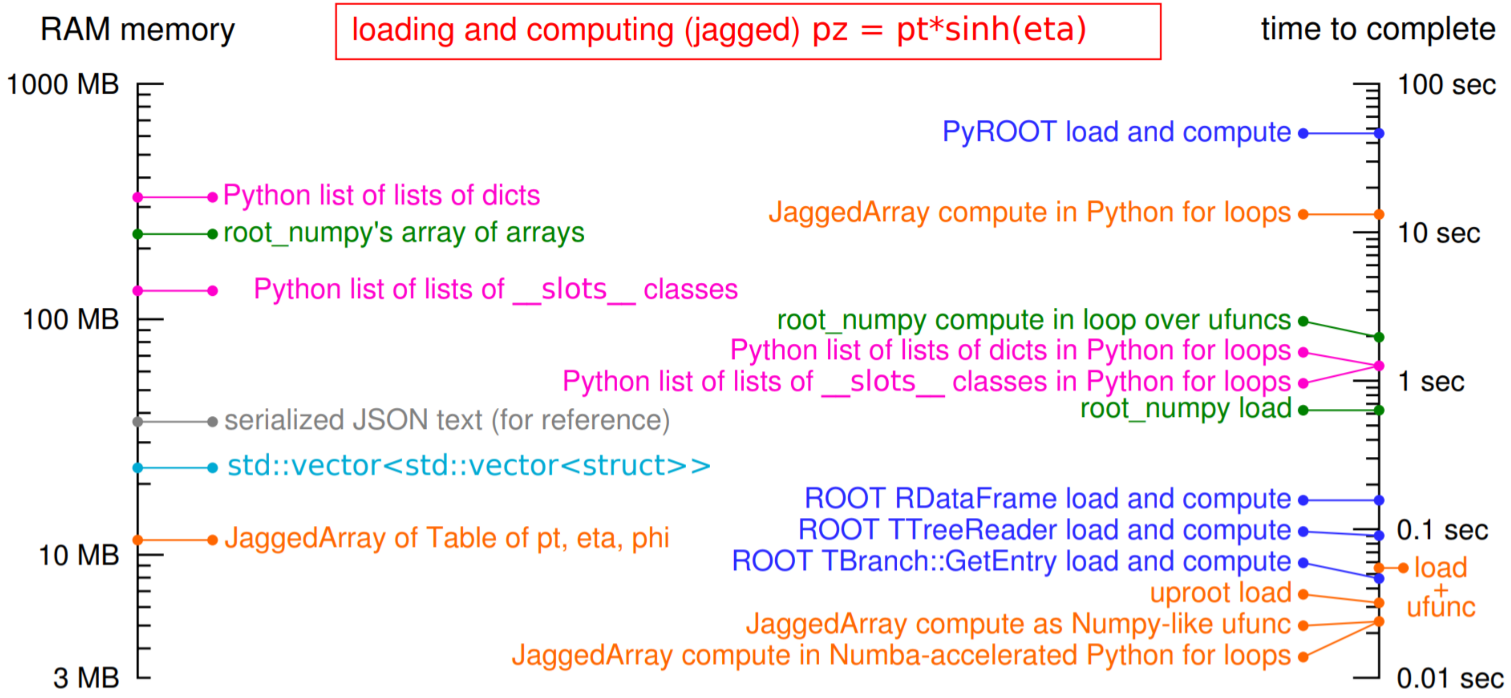
```
delta = ( ToF(K_plus.mass, 10*GeV, 10*meter) - ToF(pi_plus.mass, 10*GeV, 10*meter) ) / ps
print("At 10 GeV, Delta-TOF(K-pi) over 10 meters = {:.5} ps".format(delta))
```

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

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

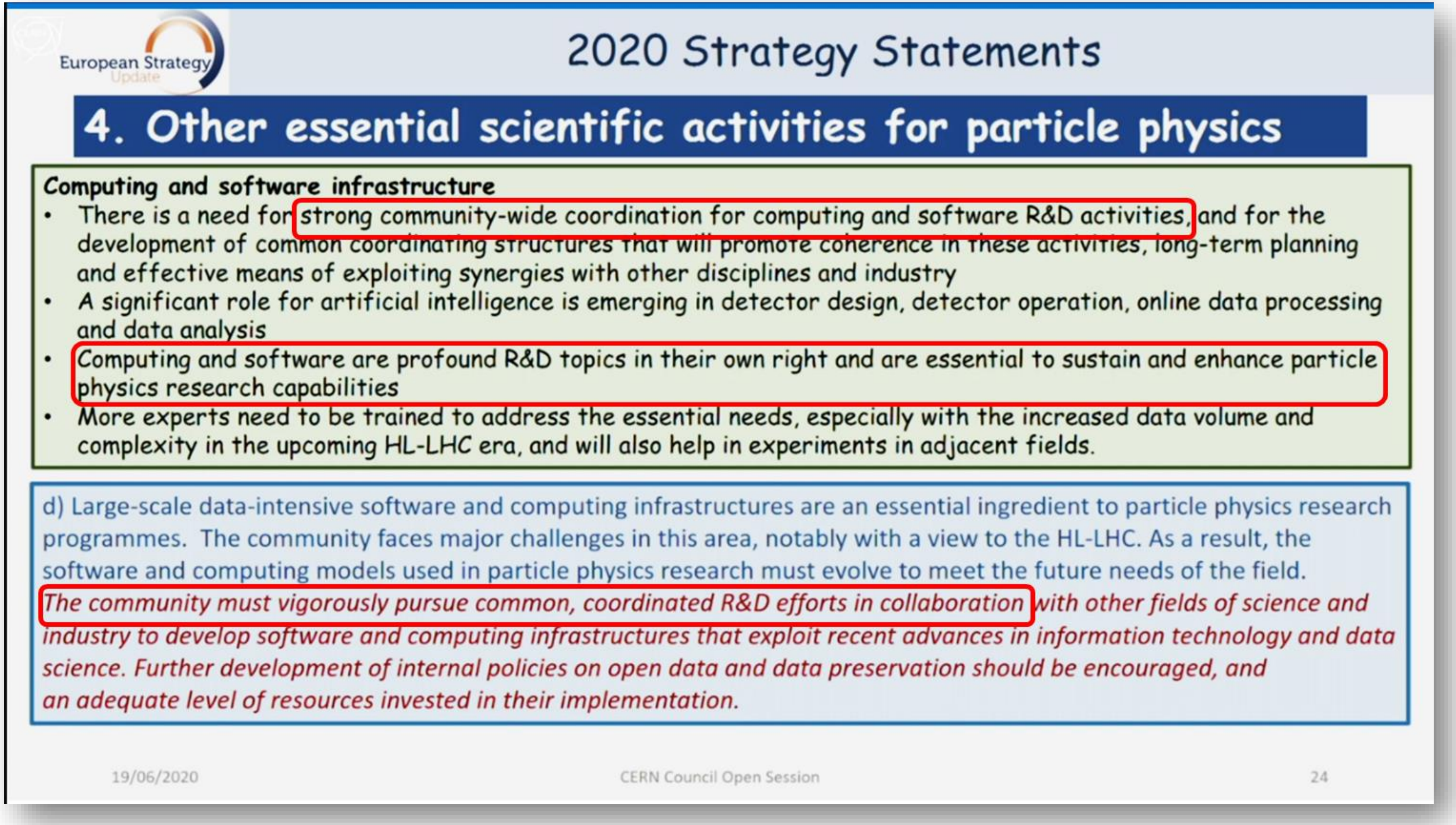
- ❑ **NOPE !**

"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

HEP Software & Computing – we are being listened to!



European Strategy Update

2020 Strategy Statements

4. Other essential scientific activities for particle physics

Computing and software infrastructure

- There is a need for strong community-wide coordination for computing and software R&D activities, and for the development of common coordinating structures that will promote coherence in these activities, long-term planning and effective means of exploiting synergies with other disciplines and industry
- A significant role for artificial intelligence is emerging in detector design, detector operation, online data processing and data analysis
- Computing and software are profound R&D topics in their own right and are essential to sustain and enhance particle physics research capabilities
- More experts need to be trained to address the essential needs, especially with the increased data volume and complexity in the upcoming HL-LHC era, and will also help in experiments in adjacent fields.

d) Large-scale data-intensive software and computing infrastructures are an essential ingredient to particle physics research programmes. The community faces major challenges in this area, notably with a view to the HL-LHC. As a result, the software and computing models used in particle physics research must evolve to meet the future needs of the field. *The community must vigorously pursue common, coordinated R&D efforts in collaboration with other fields of science and industry to develop software and computing infrastructures that exploit recent advances in information technology and data science. Further development of internal policies on open data and data preservation should be encouraged, and an adequate level of resources invested in their implementation.*

19/06/2020 CERN Council Open Session 24