HSF PyHEP WG training matters and the Training WG

Eduardo Rodrigues
University of Liverpool, for the PyHEP WG conveners

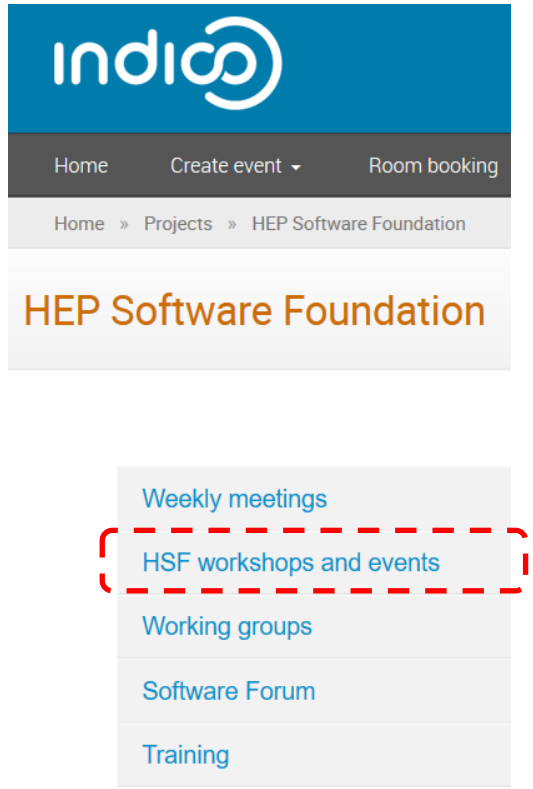
HSF Training WG Meeting, CERN, 9th March 2020

HSF PyHEP WG

- A few quick words on the PyHEP – “Python in HEP” Working Group
- PyHEP series of workshops
- Community projects towards a HEP Python ecosystem and links to Training WG

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

- The "Python in HEP" WG effectively started in early 2018 as an activity group, which I put forward with the proposal of the 1st workshop, held as a pre-CHEP 2018 event
- It became "formally" a WG this year ☺



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

❑ **Lots of ways to communicate !**

- The main channel now has < 100 people registered

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

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

Getting Involved

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

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

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

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

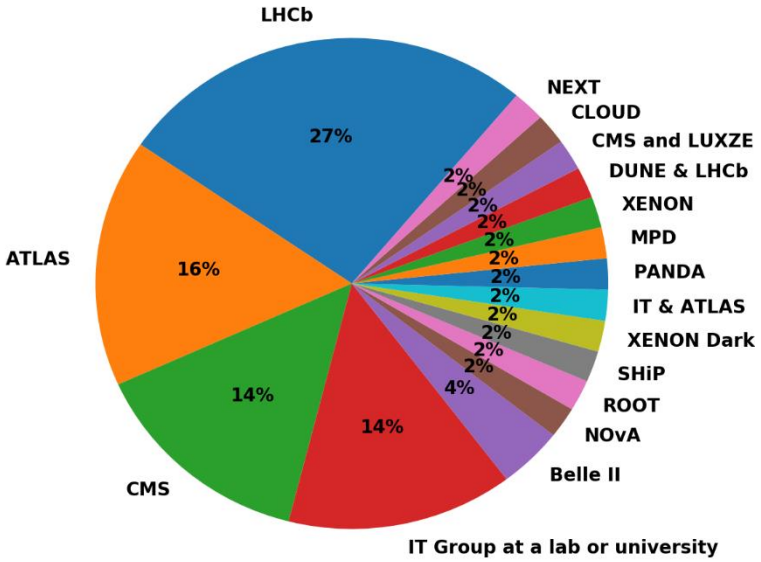
PyHEP Series of Workshops

PyHEP workshops – a new series of workshops

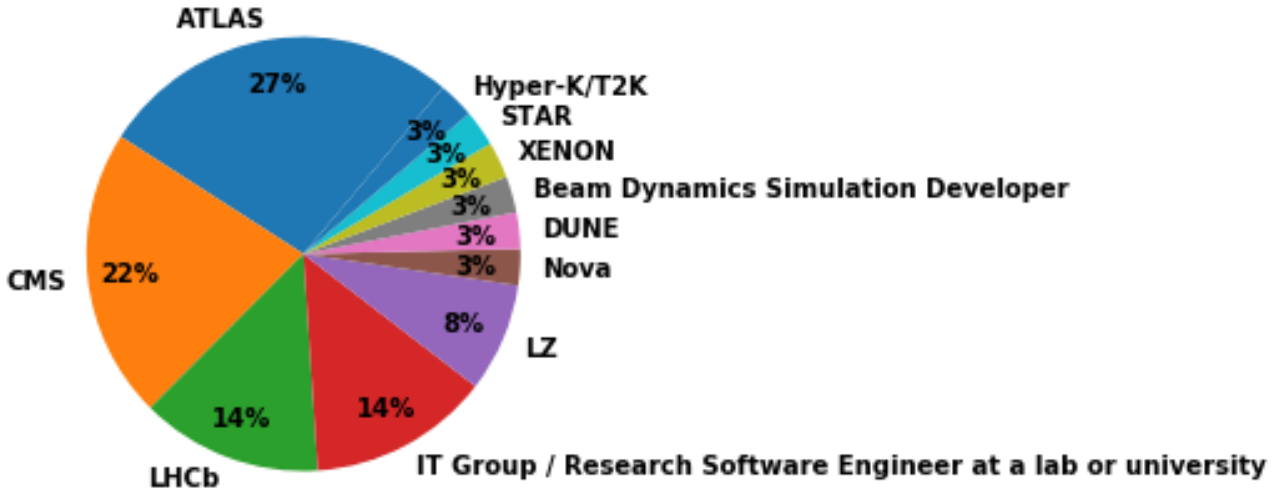
The **PyHEP workshops** are a series of workshops initiated and supported by the HEP Software Foundation (HSF) with the aim to provide an environment to discuss and promote the usage of Python in the HEP community at large. Further information is given on the PyHEP WG website.

❑ **Community diversity – great to see such a very diverse set of participants !**

PyHEP 2018
Sofia, Bulgaria



PyHEP 2019
Abingdon, U.K.



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

PyHEP series of workshops – PyHEP 2020

PyHEP 2020

- To be held in Austin (Texas), U.S.A., in July 11-13
- Next to SciPy 2020 conference, to enhance cross-community exchange
- First official announcement email sent out on Feb. 25th
- See also Indico agenda

Community projects towards a HEP Python ecosystem for data analysis

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

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

Tackle them as a community !

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

❑ 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/

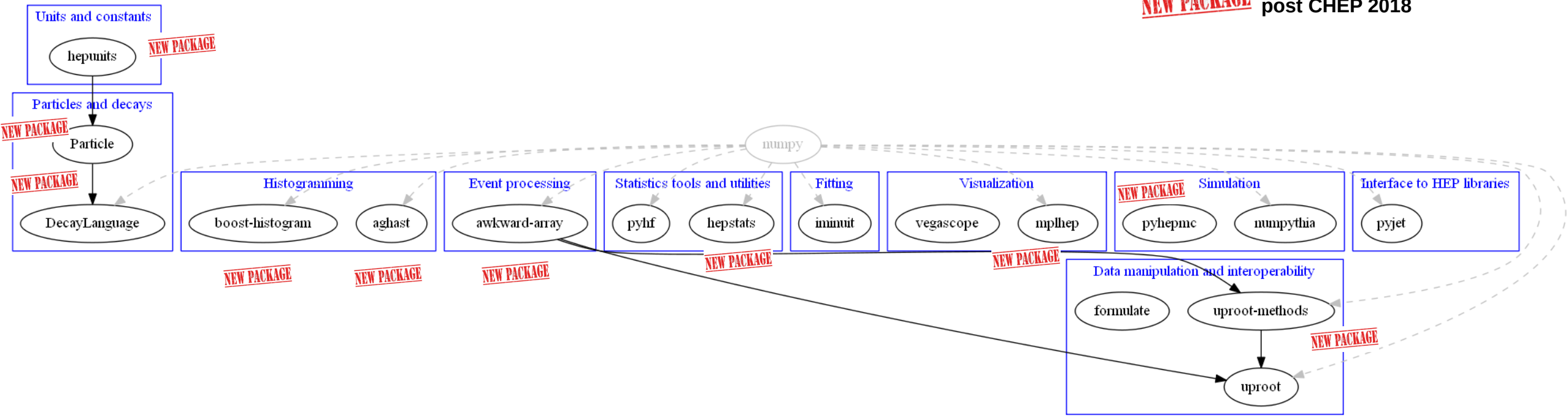
❑ https://scikit-hep.org/

❑ https://github.com/zfit

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

https://scikit-hep.org/

NEW PACKAGE = 1st release post CHEP 2018



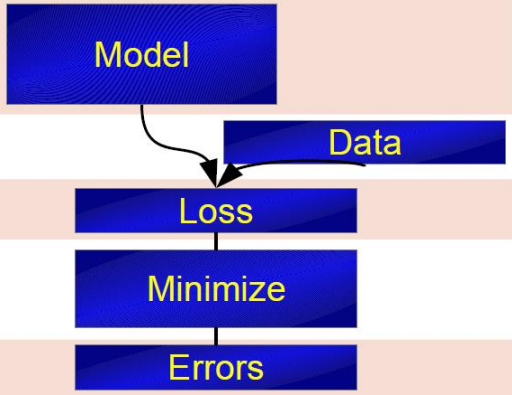
There are other packages: test data, tutorials, org stats, etc.
(and some which tend to now be superseded, hence deprecated …)

The `zfit` project and package

- **Project: provide a stable fitting ecosystem, in close collaboration with the community**
- **`zfit` package:**
  - **Scalable, Pythonic, HEP specific features**
  - **Pure Python, no ROOT dependency, performant (TensorFlow as main backend)**
  - **Highly customisable and extendable**
  - **Depends on iminuit**

- **Simple example:**

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

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

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

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

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

param_errors = result.error()
```



implement custom function

```
from zfit import ztf


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

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

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

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

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

The coffea project

Coffea - Column Object Framework for Effective Analysis

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

Separation of "user code" and "executors"

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

Coffea ***achieved 1 to 3 MHz*** event processing rates

- Using Spark cluster on same site as data at Fermilab

44

Slide taken from Ben Krikler

The FAST-HEP project

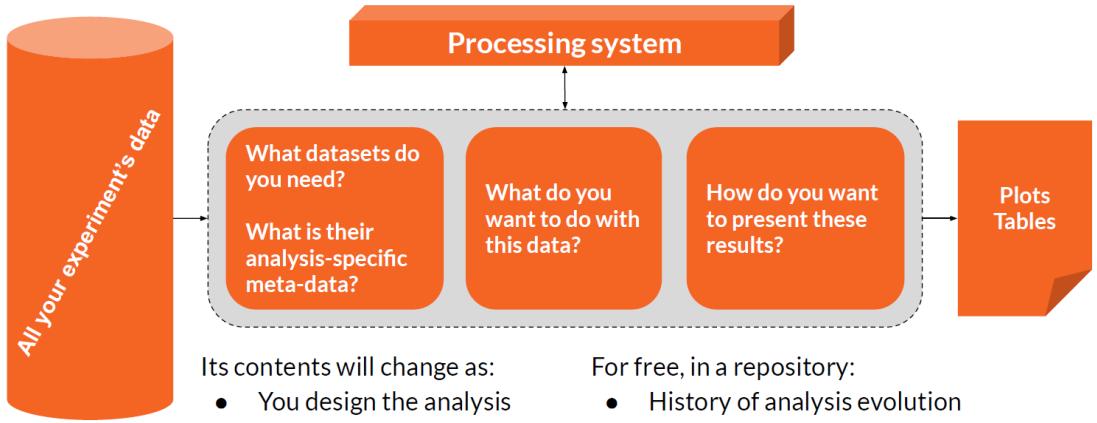
FAST-HEP

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



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

55

The FAST implementation

For tools:
use Python

NumPy

uproot

Awkward Array

NumExpr

at

For data:
use Pandas
Demoed at CHEP 2018

pandas

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

For descriptions:
use YAML...

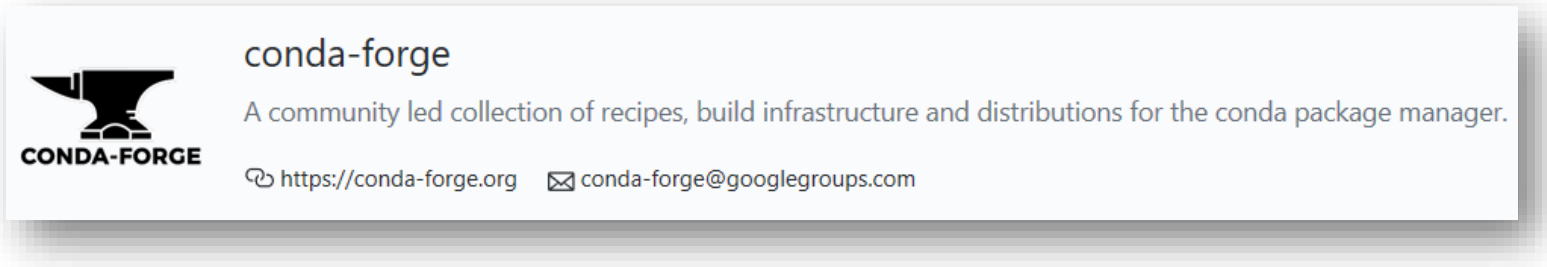
63

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

- Project towards an Analysis Description Language …

Material taken from Ben Krikler

Conda-forge – making it easy for users



- **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`:**