

PyHEP2022 Workshop Report

Graeme Stewart and Vincenzo Padulano



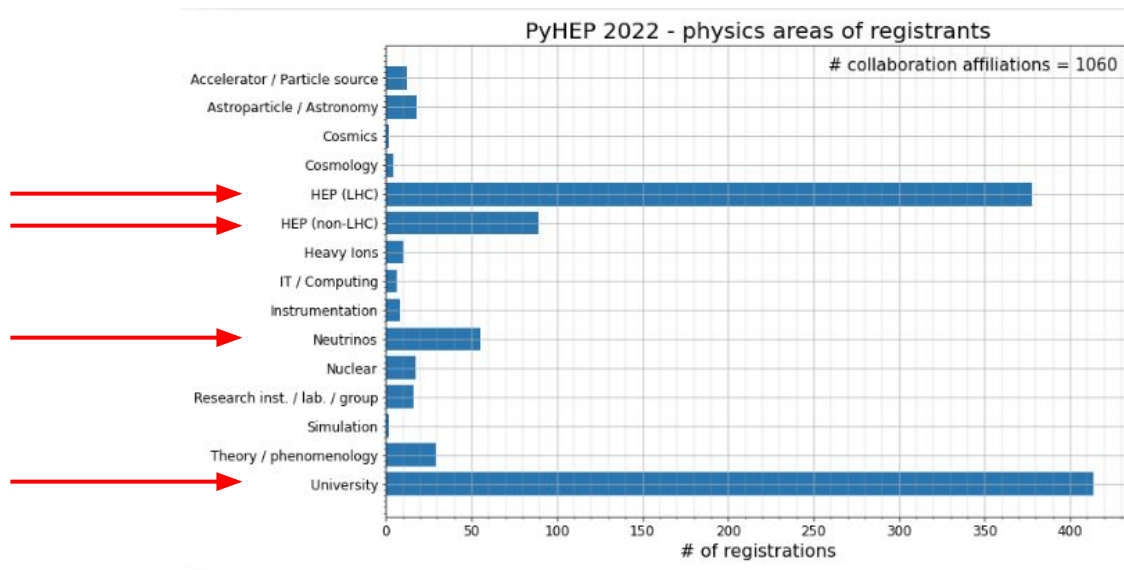
2022-10-03

PyHEP Workshops

- Series of workshops started at CHEP2018
 - Successful event, so continued annually
 - Organised by the [PyHEP HSF Working Group](#) - *Eduardo Rodrigues* was the founder and continues to play a critical role
- Motivation
 - Increasing interest in using tooling from Python data science
 - Recognition of Python as a first class language for analysis in HEP
 - Update the community on latest developments and discuss challenges
- First two workshops were mainly for package developers
- Since 2020, organised as an online event
 - Because, COVID...
 - But registrations jumped from 55 to 1000
 - Clearly tapped a significant vein of interest, particularly from students

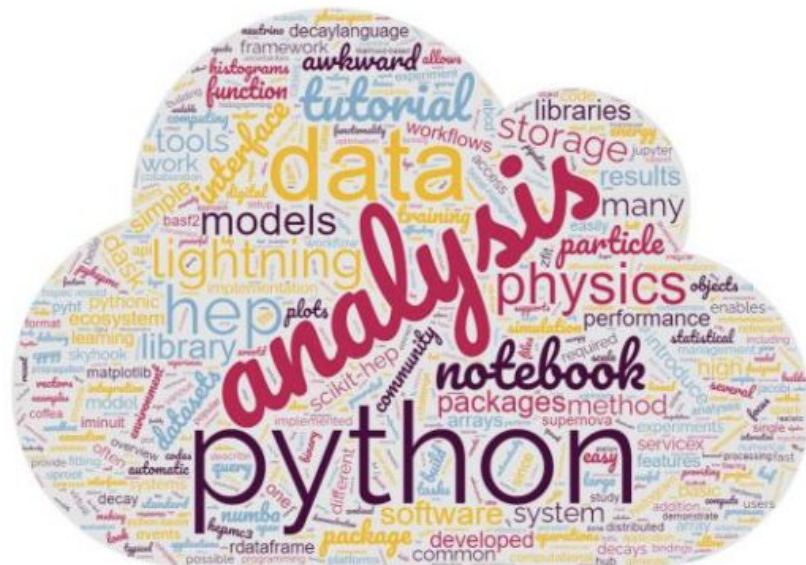
PyHEP2022

- Decided again for a virtual workshop
 - Low barrier for entry at the cost of lack of coffee-based interactions
 - 1122 people registered

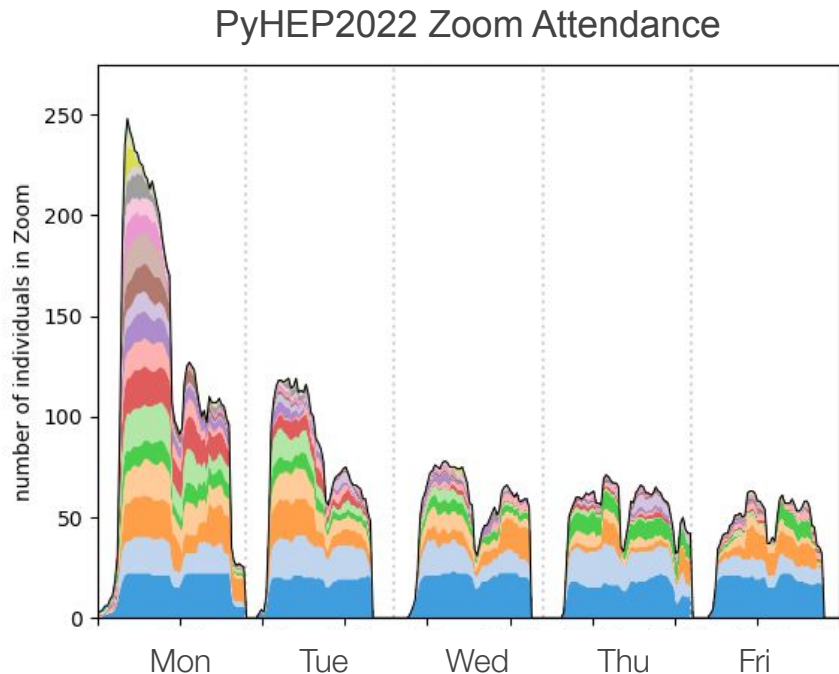


Format

- Open Abstract Call - 42 submissions (a record!)
- Three presentation options
 - Tutorial - 1 hour based on Jupyter notebook
 - Notebook Talk - 30 minutes, demonstration and discussion, also from a notebook
 - Lightning talk - 10 minutes, mixture of slides and notebooks used
- Bonus features
 - Slack for discussions
 - Sli.do for questions
 - RemotelyGreen for social mixing
 - Hackashop for leveling up from user to contributor



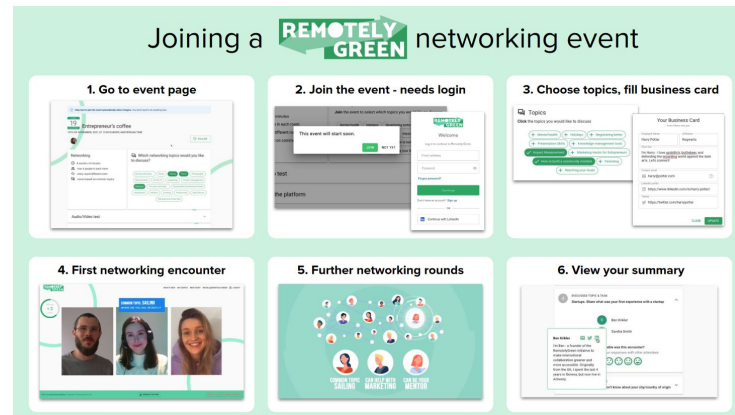
Registrations to Reality...



- Peak attendance almost 250
- Drops off day-by-day, to 50-70 people
 - Which is still a pretty active community
- In total, **421** people did attend at some point in the workshop
 - So a bit buffet style!
- Not clear why more than half the registrants didn't show up
 - We try to understand this through the survey
 - Seems that people who had previously attended were more likely to show up

Hackashop and Remote Mingling...

- We tried to overcome the virtual workshop disadvantages by using the RemotelyGreen social mixer app
 - Company founded by some of our CERN colleagues
 - Session was only very lightly attended (~22)
 - Positive feedback from those who did join, but didn't get wide traction - too busy? Not interesting enough? YACT?
- Also wanted to engage users with developers, through an online hackashop event
 - Introductory talk by Aman Goel, followed by breakout rooms with projects
 - Also, around 20 people joined, but this was much more expected
 - At least some people now working on PRs with projects



Workshop Highlights

Training and Features

- With significant user interaction, tutorials and training in generic Python topics are a popular part of the workshop
- [Level up your Python](#) (Henry Schreiner), is an excellent [intermediate training resource](#) (all notebooks)
 - Core: Logging, debugging, profiling, generators, decorators, packaging
 - Numerical Python: numpy, panda, numba, PyBind11

Decorators

This is likely the simplest syntactic sugar you'll see today, but maybe one with some of the furthest reaching consequences. Let's say you have a bit of code that looks like this:

```
def f(): ...  
f = g(f)
```

So `g` is a function that takes a function and (hopefully) returns a function, probably a very similar one since you are giving it the same name as the old "f". In Python 2.5, we gained the ability to write this instead:

```
@g  
def f(): ...
```

That's it. The thing after the `@` "decorates" (or transforms) the function you are defining and the output is saved with the name `f`.

Training and Features

- HSF Training Experience

- Providing generic training to HEP community
- Working with The Carpentries, thanks to IRIS-HEP funding
- *Building a training community*

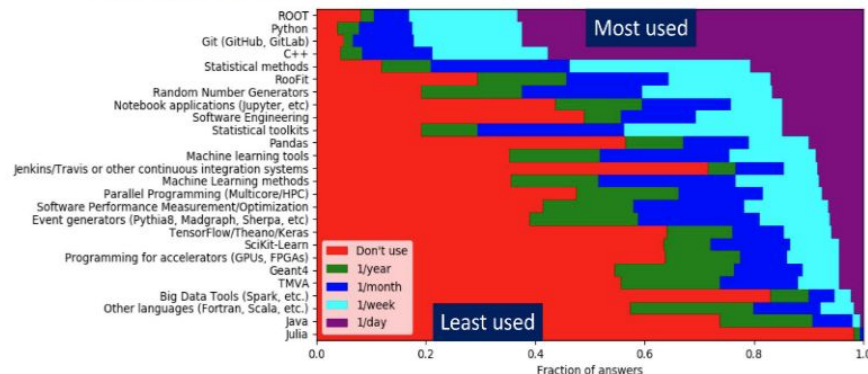
- Using C++ from Numba

- How to use [cppyy](#) (as used in PyROOT) with numba
 - Mix-in C++ code and classes into numba jited functions

- Forward look: What's new in Python 3.11

- Faster! (avg. 25% speed-up over Python 3.10)
- Better error messages and exception 'notes'
- Better static typing
- AsyncIO.Taskgroups, Tomllib, WebAssembly (pyodide)

Frequency of use of some scientific software/computing tools

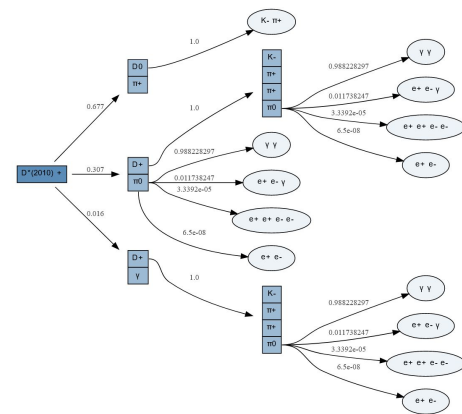


```
cppyy.cppdef("""
template<typename T>
T square(T t) { return t*t; }
""")

@numba.jit(nopython=True)
def tsa(a):
    total = type(a[0])(0)
    for i in range(len(a)):
        total += cppyy.gbl.square(a[i])
    return total
```

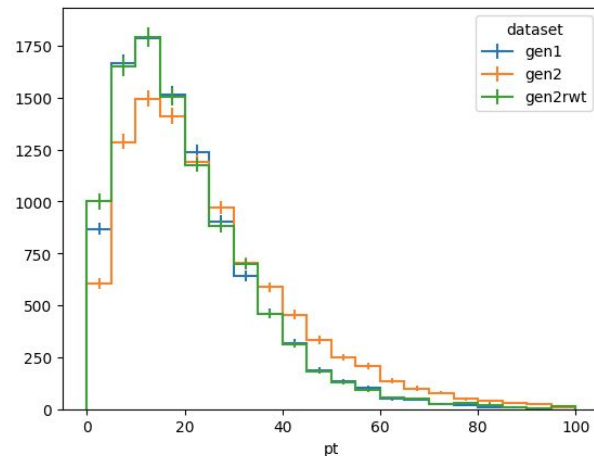
HEP Tooling in Python

- Continued improvements in Python tooling for HEP, making things more convenient
 - Most of these tools joined to the [SciKit-HEP](#) project
- Particle, Decay Language, PhaseSpace
 - Particle - Python interface to PDG data, including ‘translation’ between different PIDs
 - DecayLanguage - Python interface to particle decays
 - Phase Space - Simulate n-body decays, interfaces with DecayLanguage



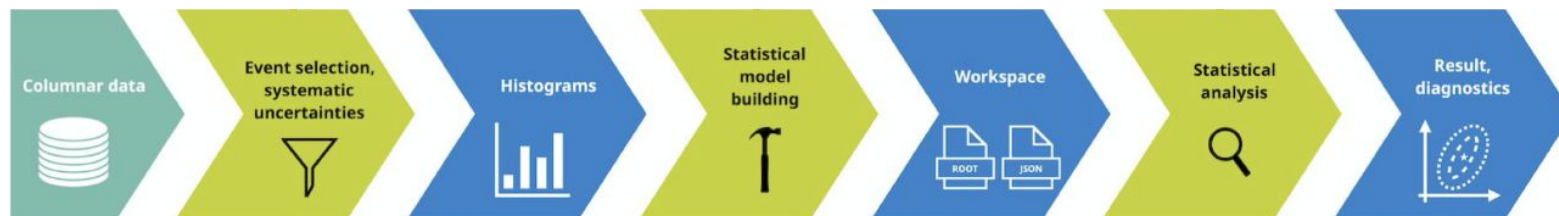
HEP Tools

- Correctionlib
 - Well-structured JSON data format for correction factors that are typical in HEP
 - C++ and Python
 - Very good runtime performance
 - N.B. Also discussed that the Analysis Ecosystem II workshop
- Nicer Pythonic interfaces to important HEP packages
 - pyhepmc3
 - Hand crafted Pythonic interface (cf. autogenerated)
 - Pythia8
 - Updated interface to Pythia, using PyBind11



Example of generator reweighting, implemented with CorrectionLib

Analysis Workflows



Data pipelines

- [ServiceX+FuncADL](#)
- [Skyhook](#)
- [Spark](#)
- [coffea+WorkQueue](#)
- [Analysis Grand Challenge](#)

Examples from experiments

- [Belle II](#)
- [LHCb](#)
- [CMS analysis with RDF+Dask](#)

Tools and interoperability

- [Scikit-HEP tools](#)
- [Vector](#)
- [XRootD+Dask](#)
- [uproot+Dask](#)
- [Awkward RDataFrame](#)

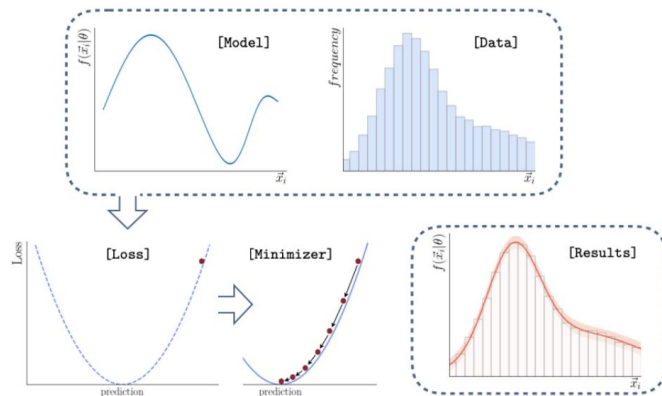
Analysis Workflows - selected highlights

- Many presentations regarding different tools in the Scikit-HEP family
 - Low-level array views
 - **Vector** manipulations
 - Remote I/O: Dask understands **XRootD** → enables end-to-end pipeline with **Dask collections**
- Data pipelines depend on experiment/site
 - Thus need for common analysis interfaces
- ServiceX embeds data delivery as part of the analysis notebook
 - **.yaml** configuration file to specify **dataset** files/trees
 - Data **caching** is available, path to local cache directory must be supplied
- Skyhook's vision: zero-copy in-memory views on distributed datasets
 - Avoids serialization costs in principle
 - Relies on a solid organization of data lake and the associated object store (**Ceph**)

Fitting and Histograms

Statistical analysis

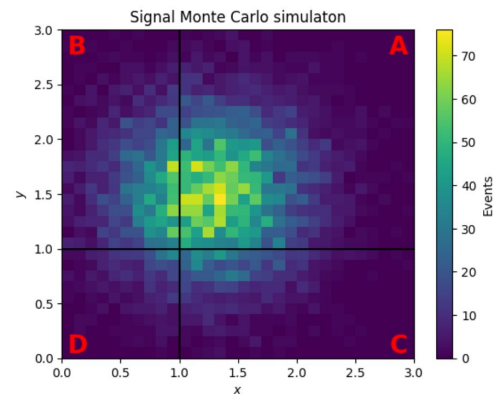
- [iminuit](#)
- [jacobi](#)
- [zfit](#)
- [abcd_pyhf](#)
- [pyhf<->combine](#)



Depiction of fitting workflow

Histograms

- [Scikit-HEP histogram tools](#)
- [Uhepp](#)




Signals with
regions ABCD

Fitting and Histograms

- **jacobi**
 - Numerical differentiation of any Python callable,
 - Numerically robust, although not as fast as other solutions
- **iminuit**
 - Minimizer wrapper around Minuit2 (no ROOT dependency)
 - Used also in other packages below
- **zfit:**
 - General purpose fitting tool
 - Particularly flexible, can be used with other packages (iminuit, hepstats).
 - Compatible with Universal Histogram Interface
- **abcd_pyhf:**
 - **pyhf wrapper**, specialized for background estimation
- **pyhf and combine analyses can be converted both ways**
 - CMS **data card** to json translation layer
 - Extra “shapes files” to store binned data (ROOT file)

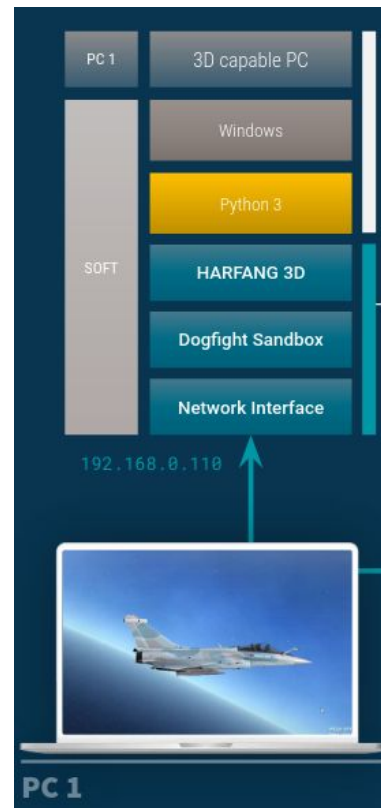
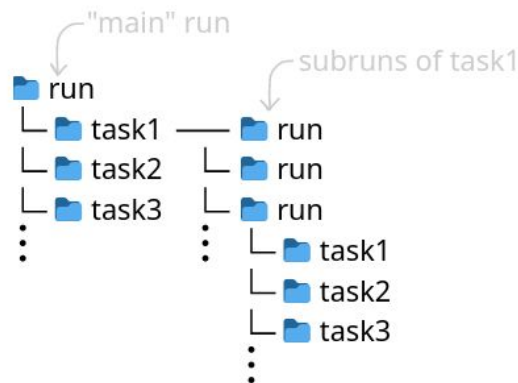
Fitting and histograms

- Uhepp: a format for sharing histograms
 - **Data + graphics** settings in the same format
 - Bin edges, values, metadata stored in semi-structured file 
 - Also provides a **web service** to upload/download histograms in the application
- boost.histogram + Hist
 - A lot of **overlap** between the two (Hist is based on boost.histogram)
 - Again a lot of interoperability with **plotting** libraries
 - (matplotlib, inline html in the notebook, terminal-friendly representation)

```
▼ root:
  ► badge:
  ▼ bins:
    ► edges: [] 41 items
    ► rebin: [] 11 items
  ► metadata:
  ► ratio: [] 1 item
  ► ratio_axis:
  ► stacks: [] 2 items
    type: "histogram"
  ► variable:
    version: "0.4"
    y_axis: {}
  ▼ yields:
    ► bkg:
    ▼ data:
      ► base: [] 42 items
      ► stat: [] 42 items
    ► sig:
```


Bonus Features

- The SuperNova Early Warning System & Software for Studying Supernova Neutrinos
- 3D and VR Industrial Use Cases in Python
- The Bureaucrat
 - Task + file manager aimed at HEP workflows
- EOS - A software for Flavor Physics Phenomenology
 - Flavor physics theory predictions in a Jupyter notebook
- Scalable, Sparse IO with larcv
 - Sparse IO layer, data format, Python/C++, parallel I/O with MPI. Aimed at ML pipelines



Summary

- PyHEP2022 brought 100s of package developers and users together again
 - Videos now all posted to [YouTube](#)
 - Survey results indicate a positive view of the event overall
- Participation and wide range of topics shows this piece of our ecosystem remains very vibrant
 - Continuing importance of Python in the future for the field
- User experience is always emphasized
 - Lots of areas to engage in
 - Extreme interoperability between contents of presentations, frequently overlapping

