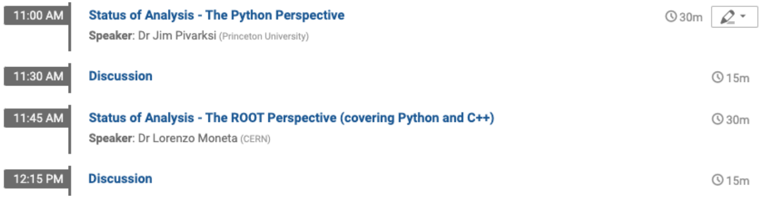
Status of Analysis — The Python Perspective

Jim Pivarski

Princeton University – IRIS-HEP

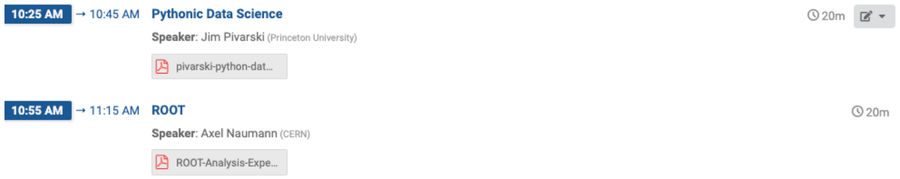
September 28, 2022

Something I want to address in this talk



11:00 AM	**Status of Analysis - The Python Perspective** **Speaker**: Dr Jim Pivarksi (Princeton University)	30m
11:30 AM	**Discussion**	15m
11:45 AM	**Status of Analysis - The ROOT Perspective (covering Python and C++)** **Speaker**: Dr Lorenzo Moneta (CERN)	30m
12:15 PM	**Discussion**	15m

Something I want to address in this talk



10:25 AM → 10:45 AM **Pythonic Data Science** 20m

Speaker: Jim Pivarski (Princeton University)

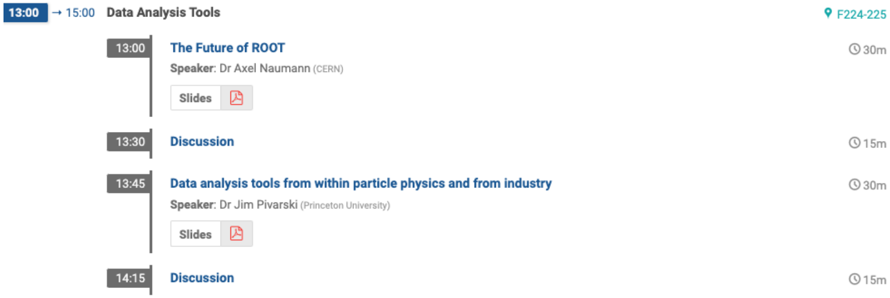
pivarski-python-dat...

10:55 AM → 11:15 AM **ROOT** 20m

Speaker: Axel Naumann (CERN)

ROOT-Analysis-Expe...

Something I want to address in this talk



13:00 → 15:00 **Data Analysis Tools** F224-225

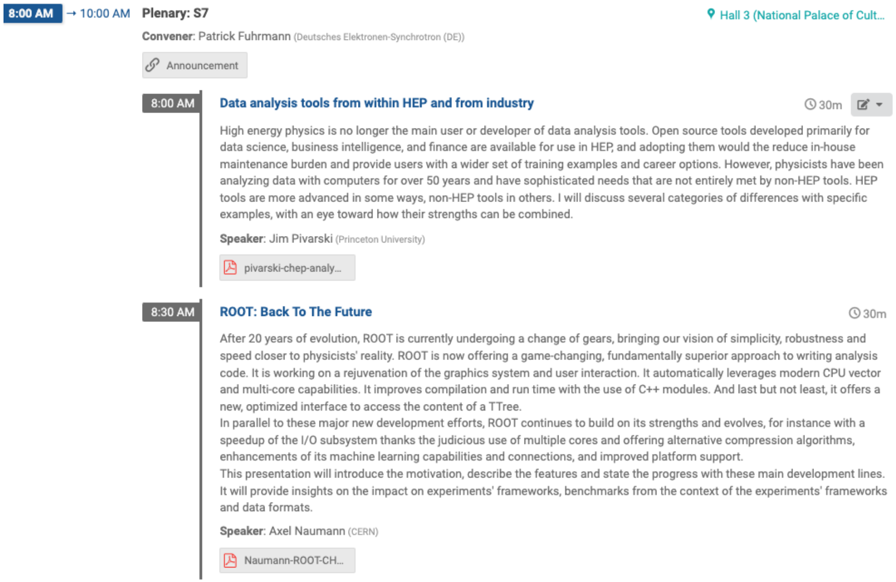
13:00 **The Future of ROOT** 30m
Speaker: Dr Axel Naumann (CERN)
Slides

13:30 **Discussion** 15m

13:45 **Data analysis tools from within particle physics and from industry** 30m
Speaker: Dr Jim Pivarski (Princeton University)
Slides

14:15 **Discussion** 15m

Something I want to address in this talk



8:00 AM → 10:00 AM **Plenary: S7** Hall 3 (National Palace of Cult...

Convener: Patrick Fuhrmann (Deutsches Elektronen-Synchrotron (DE))

Announcement

8:00 AM **Data analysis tools from within HEP and from industry** 30m

High energy physics is no longer the main user or developer of data analysis tools. Open source tools developed primarily for data science, business intelligence, and finance are available for use in HEP, and adopting them would the reduce in-house maintenance burden and provide users with a wider set of training examples and career options. However, physicists have been analyzing data with computers for over 50 years and have sophisticated needs that are not entirely met by non-HEP tools. HEP tools are more advanced in some ways, non-HEP tools in others. I will discuss several categories of differences with specific examples, with an eye toward how their strengths can be combined.

Speaker: Jim Pivarski (Princeton University)

pivarski-chep-analy...

8:30 AM **ROOT: Back To The Future** 30m

After 20 years of evolution, ROOT is currently undergoing a change of gears, bringing our vision of simplicity, robustness and speed closer to physicists' reality. ROOT is now offering a game-changing, fundamentally superior approach to writing analysis code. It is working on a rejuvenation of the graphics system and user interaction. It automatically leverages modern CPU vector and multi-core capabilities. It improves compilation and run time with the use of C++ modules. And last but not least, it offers a new, optimized interface to access the content of a TTree.
In parallel to these major new development efforts, ROOT continues to build on its strengths and evolves, for instance with a speedup of the I/O subsystem thanks the judicious use of multiple cores and offering alternative compression algorithms, enhancements of its machine learning capabilities and connections, and improved platform support.
This presentation will introduce the motivation, describe the features and state the progress with these main development lines. It will provide insights on the impact on experiments' frameworks, benchmarks from the context of the experiments' frameworks and data formats.

Speaker: Axel Naumann (CERN)

Naumann-ROOT-CH...

It might be an efficient way to summarize all the analysis activities:

Python $\bigcup$ ROOT.

It might be an efficient way to summarize all the analysis activities:

$$\text{Python} \bigcup \text{ROOT}.$$

But sometimes people read into it an implicit "versus."

ROOT has had a Python interface since 2004.

ROOT has had a Python interface since 2004.

- PyROOT is old enough to vote!

ROOT has had a Python interface since 2004.

- PyROOT is old enough to vote!

Nor is the alternative strictly “Python.”

ROOT has had a Python interface since 2004.

- PyROOT is old enough to vote!

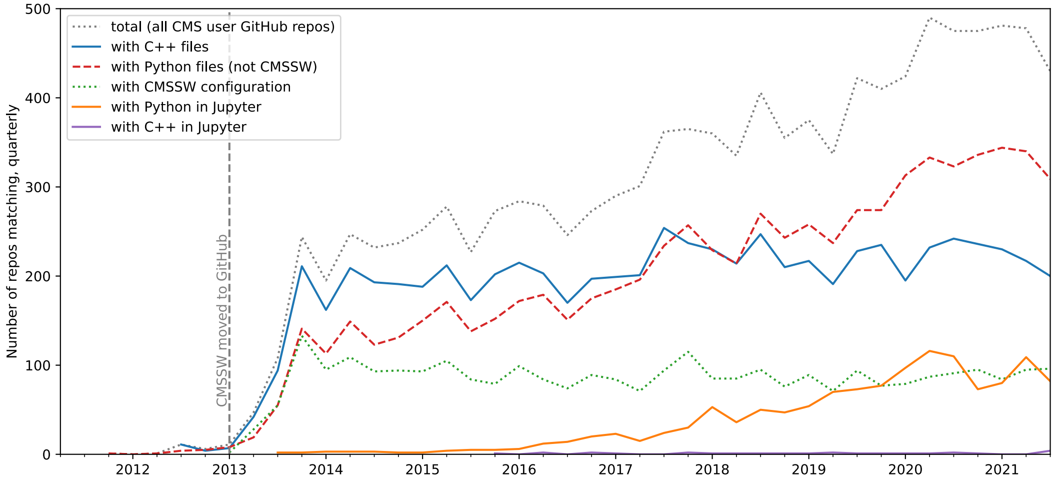
Nor is the alternative strictly “Python.”

- I’ve also tried to characterize it as “industry” or “data science.”

What I'm trying to talk about here is a social trend, and using a programming language as a proxy for it.

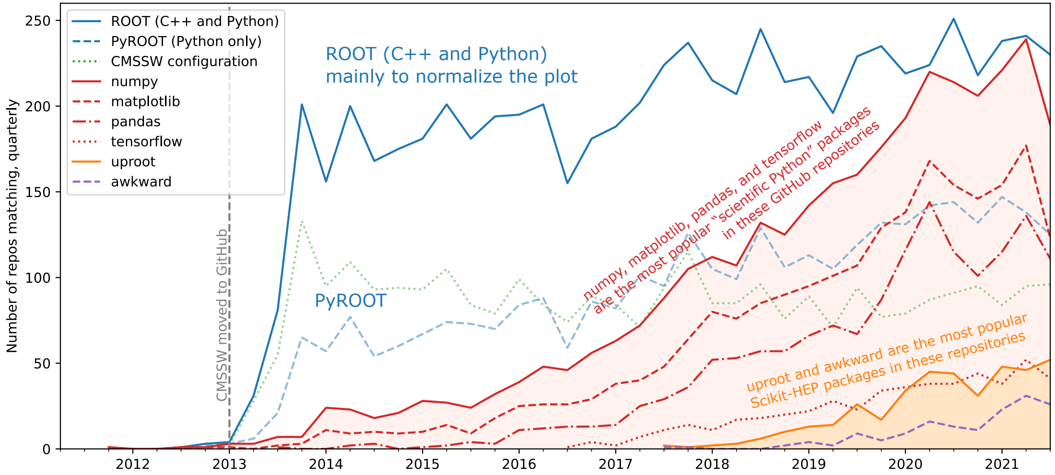
Analysis of 11 635 GitHub repos created by 2 172 CMS physicists

Software for the CMS experiment is on GitHub and only CMS physicists need to fork it. Using that, we can examine all of those physicists' non-fork, public repositories.



Analysis of 11 635 GitHub repos created by 2 172 CMS physicists

By regex searching for "`import [A-Za-z_][A-Za-z_0-9]*`" etc., we can count the number of physicist repositories that use different packages over time.



There is a rise in Python usage (including PyROOT).

There is a rise in Python usage (including PyROOT).

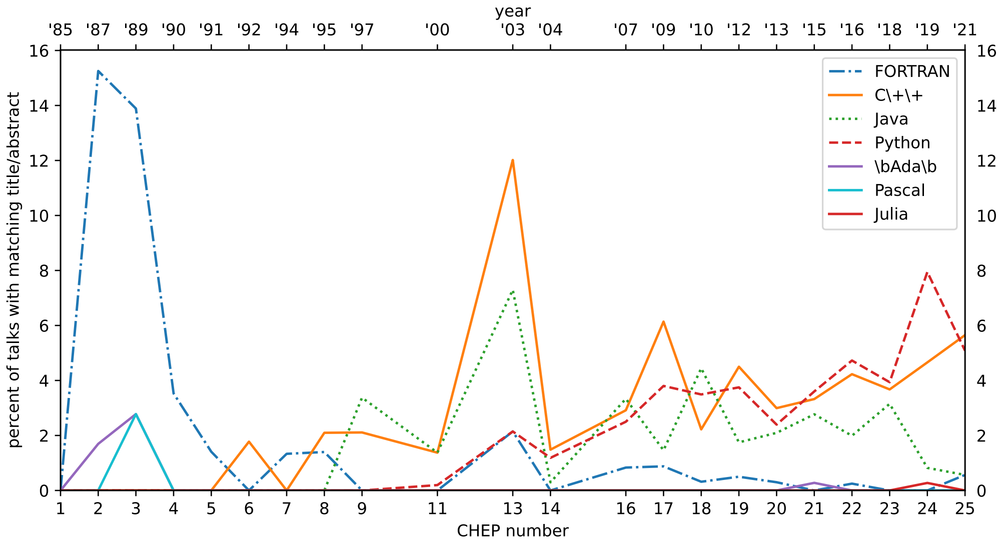
But the biggest thing that happened in the past 5 years is

```
import numpy, matplotlib, pandas, tensorflow
```

What we're seeing is the use of data analysis tools developed by scientists from other fields, the Big Data industry, quants, ...

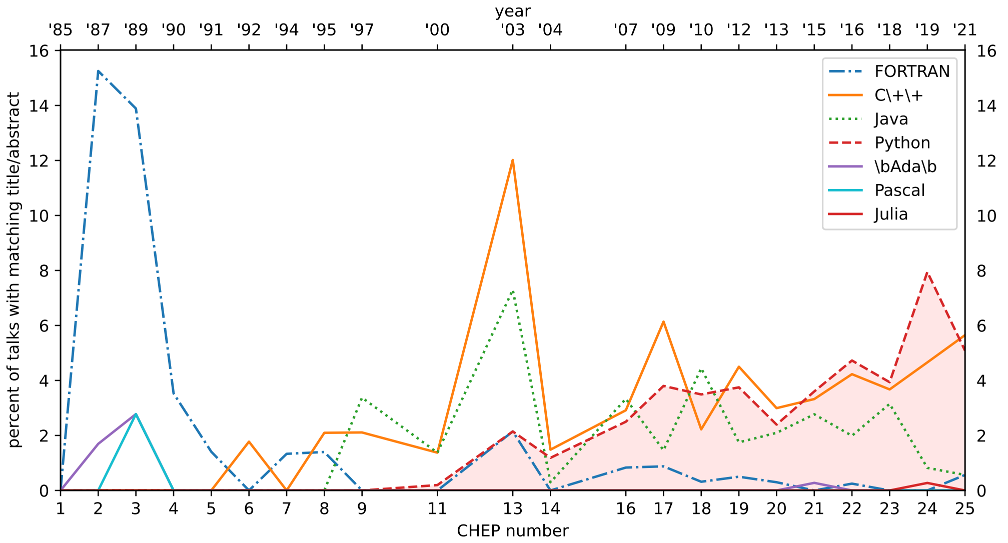
In fact, Python, the language, has been in HEP for a long time

Regex matches to all Computing in High Energy Physics (CHEP) titles and abstracts.

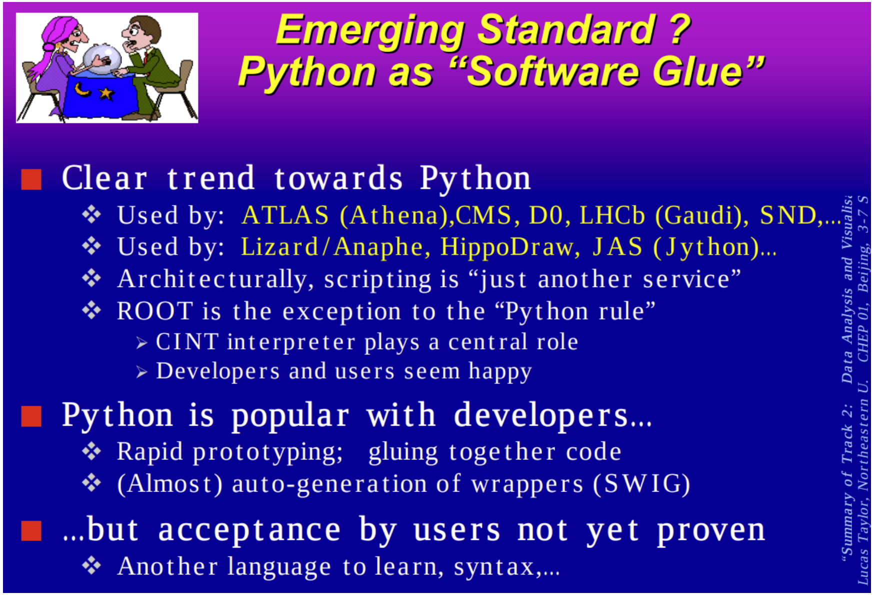


In fact, Python, the language, has been in HEP for a long time

Regex matches to all Computing in High Energy Physics (CHEP) titles and abstracts.



From Lucas Taylor's summary of data analysis track, CHEP 2001



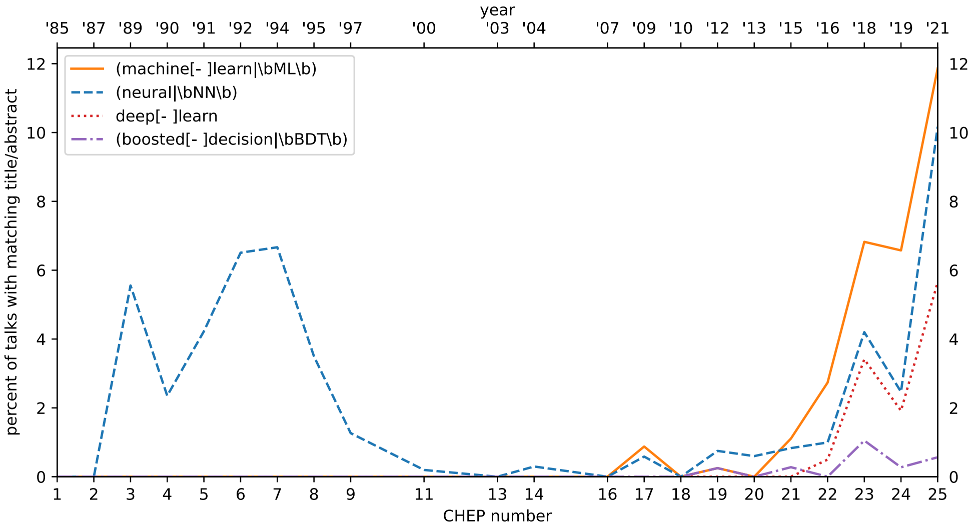
Emerging Standard ?
Python as "Software Glue"

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

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

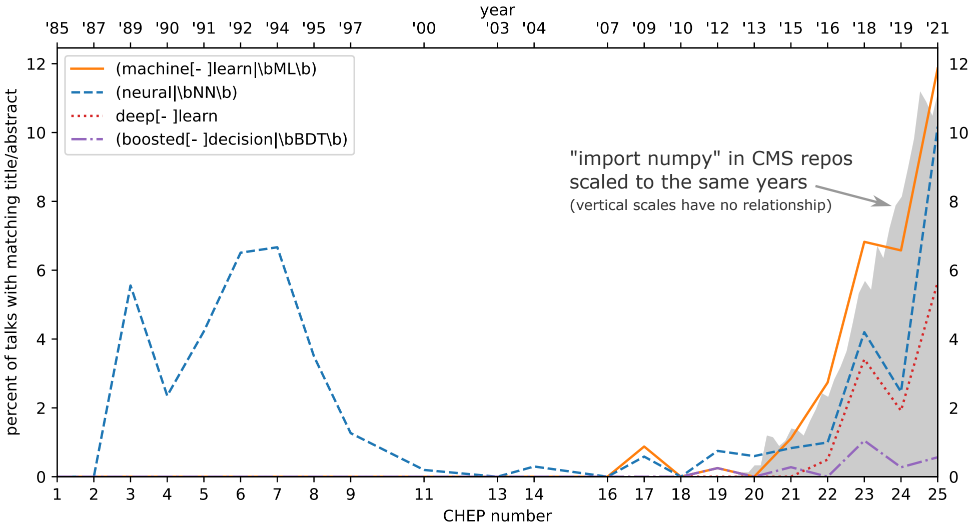
The *change* was likely driven (or started) by machine learning

Regex matches to all Computing in High Energy Physics (CHEP) titles and abstracts.

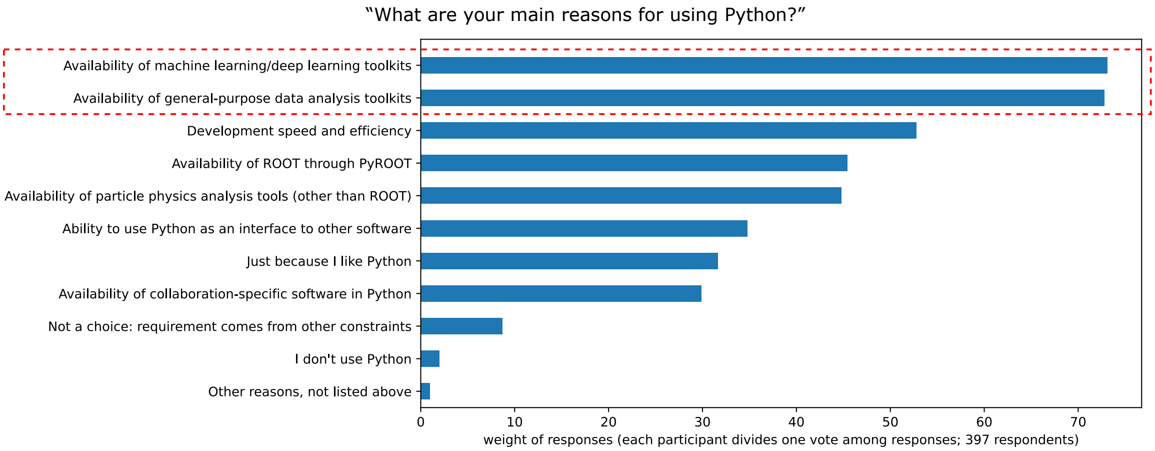


The *change* was likely driven (or started) by machine learning

Regex matches to all Computing in High Energy Physics (CHEP) titles and abstracts.



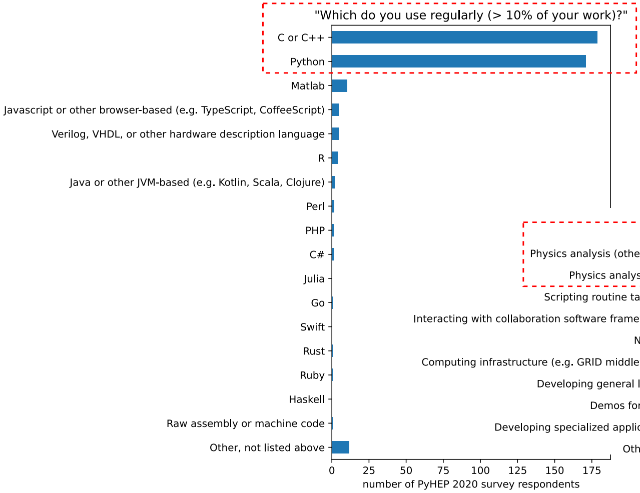
PyHEP 2020 survey

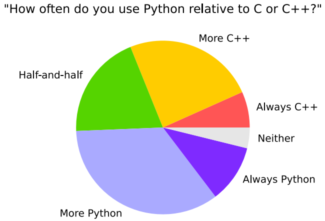


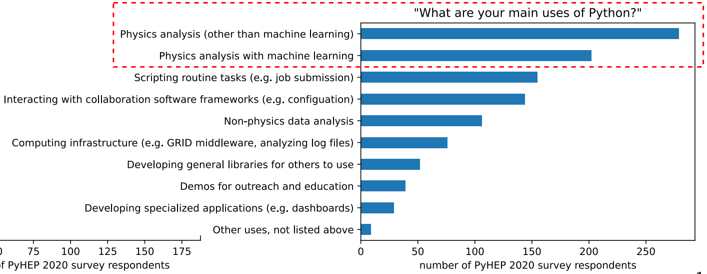
Probably prompted by outside developments in machine learning, particle physicists are now mixing tools from other fields into their analyses.

But they are using C++ *and* Python, ROOT *and* other toolkits

Again, from the PyHEP 2020 survey ($N = 406$):







Mood in the community is for inclusiveness, too

From the PyHEP 2022 Slack:

- "Is the scikit-hep collection already a complete replacement for the traditional ROOT approach? . . . I would love to mostly avoid ROOT."
- "The goal is not to replace ROOT but to provide alternative analysis styles that can be complimentary for the work you want to do. So I don't think anyone wants to try to reimpliment *everything* ROOT does, but LHCb has done an entire analysis using only Scikit-HEP tools: https://inspirehep.net/literature/1889335 https://indico.cern.ch/event/1126109/contributions/4780169"
- "But let me stress is once more as these matters come often. The whole ecosystem can do a lot in various ways, and indeed ROOT and Scikit-HEP take different routes/philosophies. But they are alternative approaches, not replacements. In the end we all win by having several approaches/packages."

(Lots of eyeballs, hearts, thumbs-up, smiles, etc.)

PyHEP talks about connecting ROOT-based systems to others

Uhepp: Sharing plots in a self-contained format	Frank Sauerburger
Lessons learned converting a production-grade Python CMS analysis to distributed RDataFrame	Tommaso Tedeschi and Vincenzo Padulano
Awkward RDataFrame Tutorial	Ianna Osborne
Enabling Dask Interoperability with XRootD Storage Systems	Scott Demarest
Using C++ From Numba, Fast and Automatic	Baidyanath Kundu
Pyhf to Combine Converter	Petey Ridolfi

(Not counting talks about ROOT and HepMC3 *files*, Minuit, Pythia, and others.)

But that doesn't mean that we *avoid* overlaps with ROOT or *only* bridge ROOT functionality to the outside world.

But that doesn't mean that we *avoid* overlaps with ROOT or *only* bridge ROOT functionality to the outside world.

We add particle physics functionality to the scientific Python ecosystem in a Pythonic way:

But that doesn't mean that we *avoid* overlaps with ROOT or *only* bridge ROOT functionality to the outside world.

We add particle physics functionality to the scientific Python ecosystem in a Pythonic way:

- following Python conventions and practices (as they evolve)

But that doesn't mean that we *avoid* overlaps with ROOT or *only* bridge ROOT functionality to the outside world.

We add particle physics functionality to the scientific Python ecosystem in a Pythonic way:

- following Python conventions and practices (as they evolve)
- share data between packages by referencing arrays

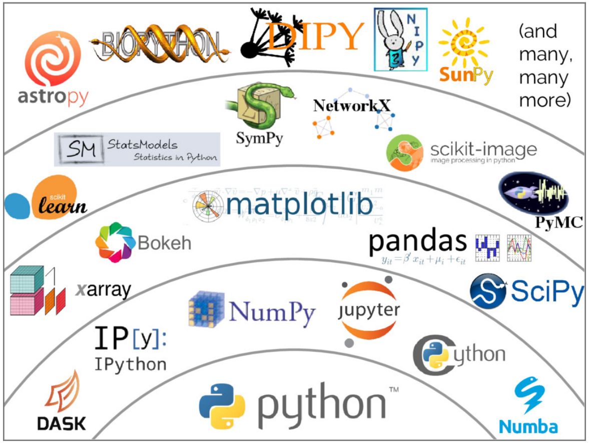
But that doesn't mean that we *avoid* overlaps with ROOT or *only* bridge ROOT functionality to the outside world.

We add particle physics functionality to the scientific Python ecosystem in a Pythonic way:

- following Python conventions and practices (as they evolve)
- share data between packages by referencing arrays
- preference for small-scoped, cooperating packages

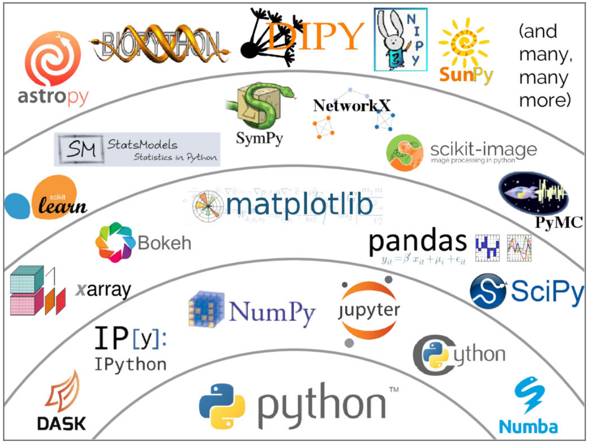
This is our model: pieces that can be plugged together

Scientific Python ecosystem

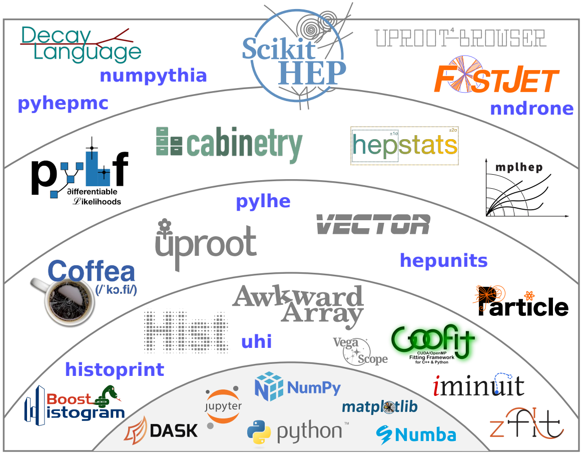


This is our model: pieces that can be plugged together

Scientific Python ecosystem

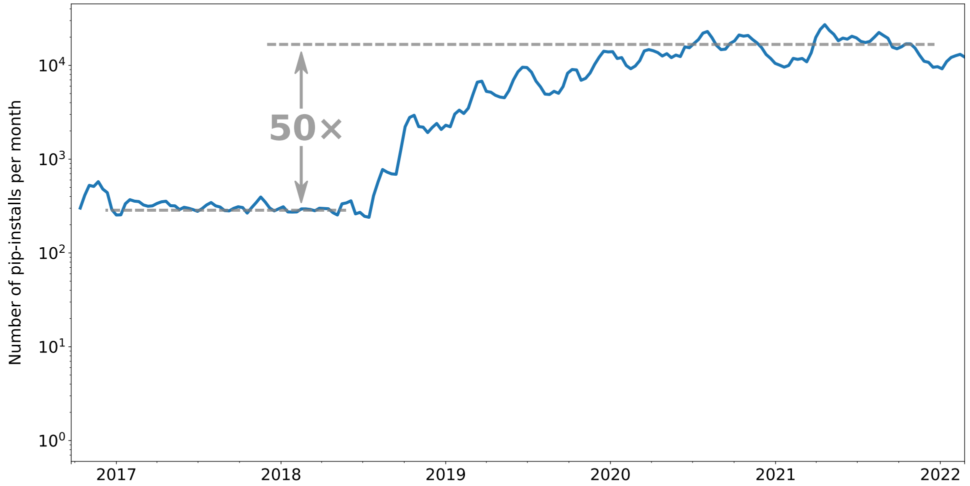


Scikit-HEP ecosystem



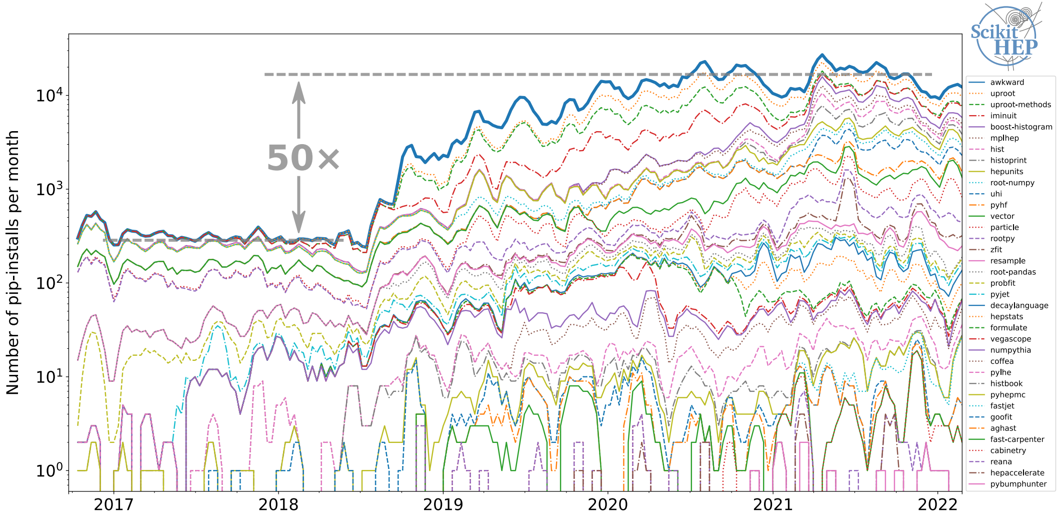
Since we started, packages grew in popularity *and number*

Download rates of Scikit-HEP packages (on Mac & Windows: no batch jobs)



Since we started, packages grew in popularity *and number*

Download rates of Scikit-HEP packages (on Mac & Windows: no batch jobs)



Advantages of small packages

Disadvantages of small packages

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.

Disadvantages of small packages

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.
- Recognition: author of that package is highly visible.

Disadvantages of small packages

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.
- Recognition: author of that package is highly visible.
- Prevents scope-creep: pressure for features outside the target area can be refused without guilt.

Disadvantages of small packages

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.
- Recognition: author of that package is highly visible.
- Prevents scope-creep: pressure for features outside the target area can be refused without guilt.

Disadvantages of small packages

- Coordination: what if no one wants to cover a given target area?

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.
- Recognition: author of that package is highly visible.
- Prevents scope-creep: pressure for features outside the target area can be refused without guilt.

Disadvantages of small packages

- Coordination: what if no one wants to cover a given target area?
- Coordination: what if everyone wants to cover the same target area?

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.
- Recognition: author of that package is highly visible.
- Prevents scope-creep: pressure for features outside the target area can be refused without guilt.

Disadvantages of small packages

- Coordination: what if no one wants to cover a given target area?
- Coordination: what if everyone wants to cover the same target area?
- Coordination: what if packages in neighboring target areas can't share data?

Advantages of small packages

- Agile: when a new area needs to be filled or an old package ceases to be maintained/liked, a new one can move in without historical constraints.
- Recognition: author of that package is highly visible.
- Prevents scope-creep: pressure for features outside the target area can be refused without guilt.

Disadvantages of small packages

- Coordination: what if no one wants to cover a given target area?
- Coordination: what if everyone wants to cover the same target area?
- Coordination: what if packages in neighboring target areas can't share data?
- Communication: where do users get help solving problems that cross target areas?

Case study showing the worst and the best: histograms

NumPy, Matplotlib, Pandas, etc. can plot histograms, but not the way we need in particle physics: as fillable objects, with (negative) weights, error propagation, . . .

Case study showing the worst and the best: histograms

NumPy, Matplotlib, Pandas, etc. can plot histograms, but not the way we need in particle physics: as fillable objects, with (negative) weights, error propagation, . . .

So we need to make our own histogram package. And it seems easy, right?

Case study showing the worst and the best: histograms

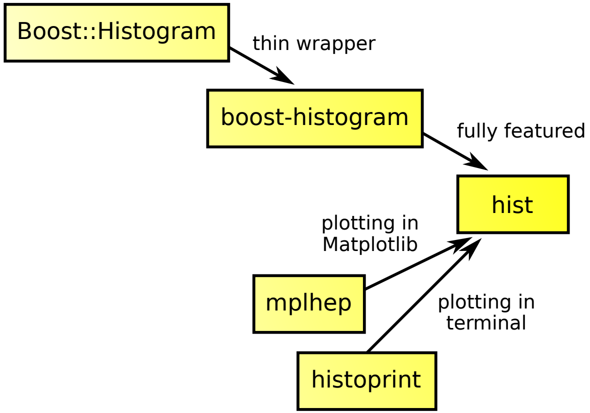
NumPy, Matplotlib, Pandas, etc. can plot histograms, but not the way we need in particle physics: as fillable objects, with (negative) weights, error propagation, . . .

So we need to make our own histogram package. And it seems easy, right?

Physicists have created at least 20 histogram packages in Python, mostly single-author.

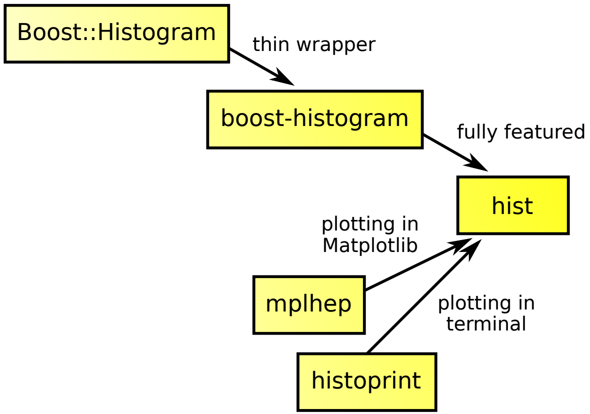
- PyROOT (2004–now)
- PAIDA (2004–2007)
- Plothon (2007–2008)
- SVGFig (2008–2009)
- YODA (2008–now)
- DANSE (2009–2011)
- rootpy (2011–2019)
- SimpleHist (2011–2015)
- pyhistogram (2015)
- multihist (2015–now)
- matplotlib-hep (2016)
- QHist (2017–2019)
- Physt (2016–now)
- Histogrammar (2016–now)
- HistBook (2018–2019)
- Coffea.hist (2019–2022)
- boost-histogram (2019–now)
- mplhep (2019–now)
- histoprint (2020–now)
- hist (2020–now)

In Scikit-HEP, several packages converged to work together



Originally, each of these was developed independently by a single author.

In Scikit-HEP, several packages converged to work together

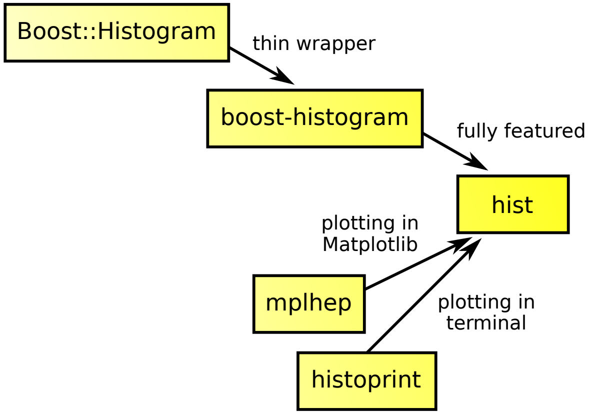


Originally, each of these was developed independently by a single author.

They each provide a piece of functionality users can get through

```python
import hist
```

In Scikit-HEP, several packages converged to work together



Originally, each of these was developed independently by a single author.

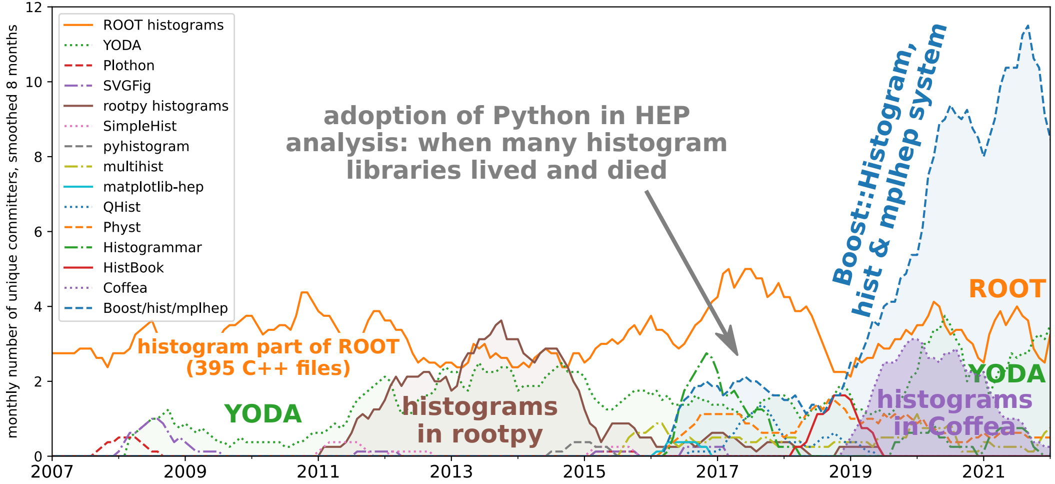
They each provide a piece of functionality users can get through

```
import hist
```

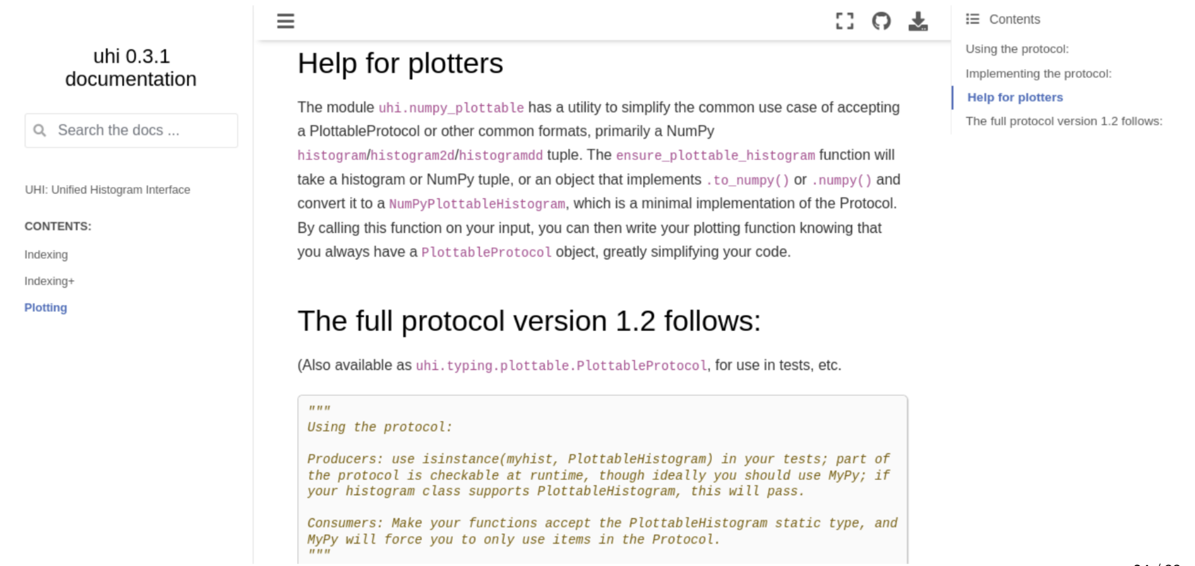
Now, 47 developers have contributed to these packages, and 20 contributed to more than one.

Histogram proliferation and convergence

Number of unique developers contributing to each package per month (in git).
Measures concentration of *developer activity*, not users.



Now we define the interfaces: agreed-upon protocols



uhi 0.3.1 documentation

Search the docs ...

UHI: Unified Histogram Interface

CONTENTS:

Indexing

Indexing+

Plotting

Contents

Using the protocol:

Implementing the protocol:

Help for plotters

The full protocol version 1.2 follows:

Help for plotters

The module `uhi.numpy_plottable` has a utility to simplify the common use case of accepting a PlottableProtocol or other common formats, primarily a NumPy `histogram`/`histogram2d`/`histogramdd` tuple. The `ensure_plottable_histogram` function will take a histogram or NumPy tuple, or an object that implements `.to_numpy()` or `.numpy()` and convert it to a `NumPyPlottableHistogram`, which is a minimal implementation of the Protocol. By calling this function on your input, you can then write your plotting function knowing that you always have a `PlottableProtocol` object, greatly simplifying your code.

The full protocol version 1.2 follows:

(Also available as `uhi.typing.plottable.PlottableProtocol`, for use in tests, etc.

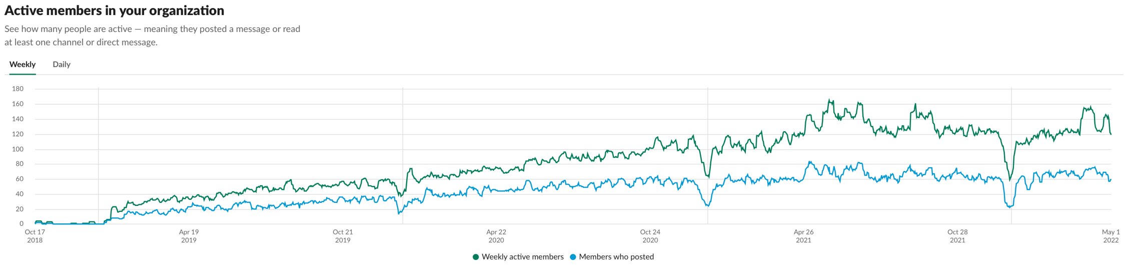
```
"""
Using the protocol:

Producers: use isinstance(myhist, PlottableHistogram) in your tests; part of
the protocol is checkable at runtime, though ideally you should use MyPy; if
your histogram class supports PlottableHistogram, this will pass.

Consumers: Make your functions accept the PlottableHistogram static type, and
MyPy will force you to only use items in the Protocol.
"""
```

Addressing the disadvantages: developer coordination

IRIS-HEP Slack: ∼150 active members weekly, not all in IRIS-HEP, not all in HEP



GitHub: also very active.

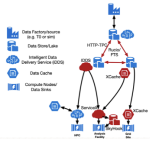
I need a visualization that demonstrates not just the volume, but the "off-diagonal" interactions between developers of different packages and between the particle physics community and the wider community.

Addressing the disadvantages: integration testing at scale

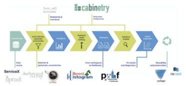
IRIS-HEP and the Analysis Grand Challenge

- **AGC**: *“Analysis Grand Challenge”*
  - historically, an integration exercise
    - test realistic end-to-end analysis pipelines aimed at HL-LHC use
    - combine technologies being developed in various ares of IRIS-HEP & adjacent ecosystem
    - identify & address performance bottlenecks and usability issues
  - organized jointly with the US ATLAS & US CMS operations programs

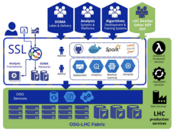
Data Organization, Management and Access (DOMA):
data delivery



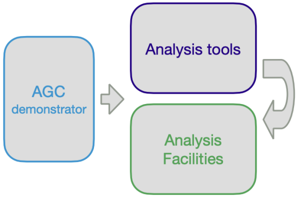
Analysis Systems (AS):
tools



Scalable Systems Laboratory (SSL):
deployment techniques and resources



AGC combining IRIS-HEP focus areas



Addressing the disadvantages: user communication

The problem is that we have *too many* ways to answer questions from users.

- GitHub issues and discussions: best so far, but distributed per-package
- Gitter: low-barrier chat, but also per-package
- Mattermost: CERN credentials are a barrier, but most LHC experiments are here
- Slack: required invitation is a barrier; mostly developers, anyway
- StackOverflow: good for cross-package discussions, but too diffuse in non-scientific world (when non-physicists answer questions, they're usually wrong)
- Scientific-Python.org Discourse and Discord: options under consideration

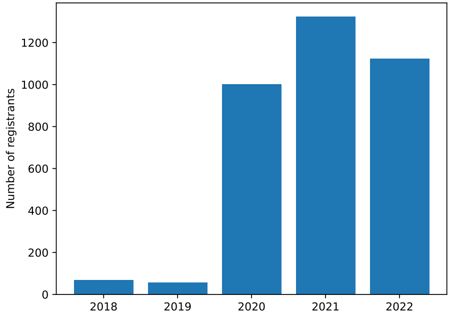
We would benefit by converging on one and sending users to that single forum.

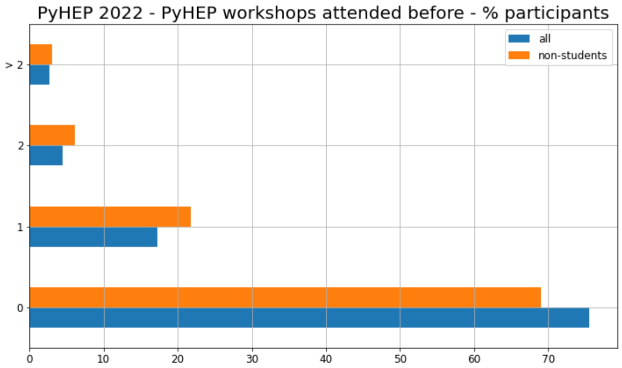
PyHEP workshop evolved from developer-focus to user-focus

2018

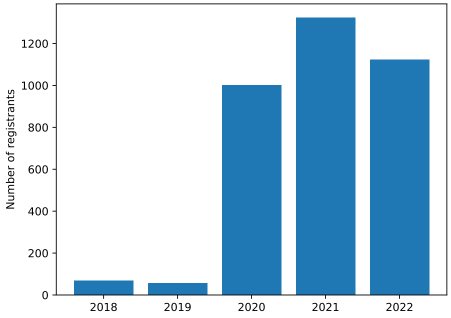
2021

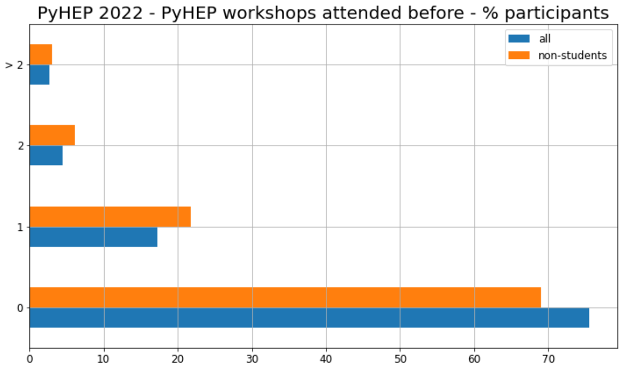
Can you guess what happened?





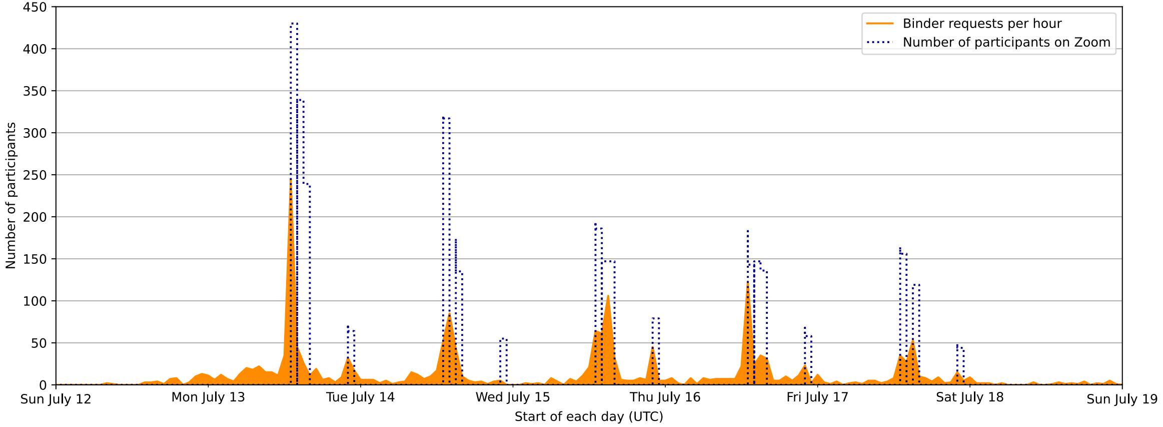
Can you guess what happened?



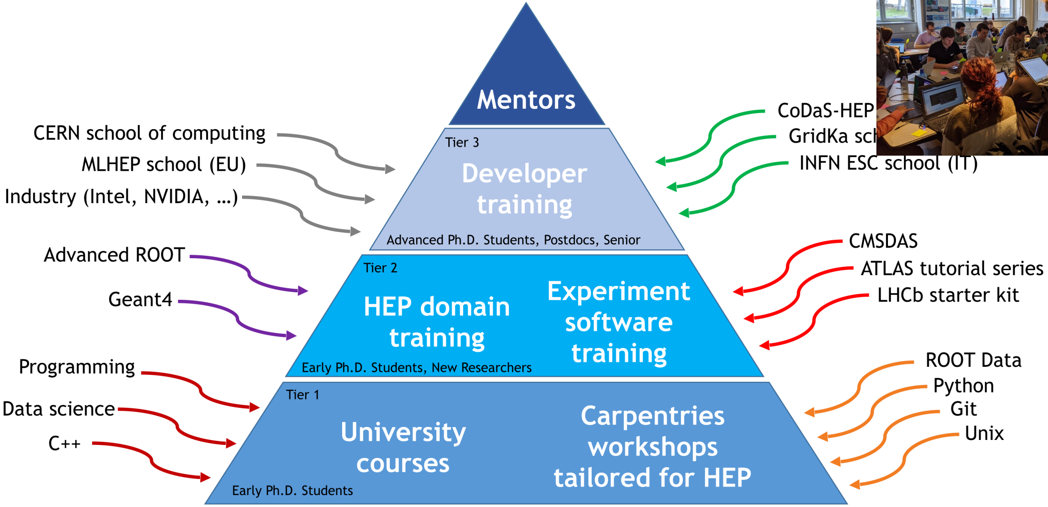


By going virtual, we discovered a cohort of newcomers who can't/wouldn't travel.

About 20–40% of the 1000 registrants attend/run the notebooks



In addition, there's the HSF/IRIS-HEP software training



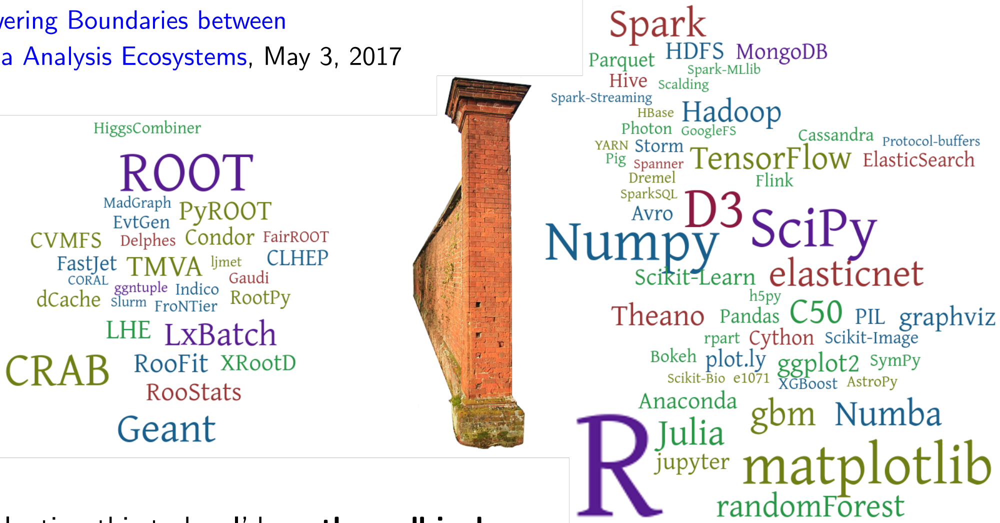
Where are we as a community?

Well, we're using Python for data analysis a lot more.

But more importantly, we're sharing (both ways) with the broader scientific community and making everything work together.

My slide from Future Trends in Nuclear Physics Computing 2017

Lowering Boundaries between
Data Analysis Ecosystems, May 3, 2017



Evaluating this today, I'd say **the wall is down.**