High Energy Physics Data Analysis Workflow in Python ecosystem

KyungEon Choi

University of Texas at Austin
Department of Physics

Seminar @ Chonnam National University
Feb 6-7, 2023

Outline

Outline

Caveats & Goals

Caveats

- This talk covers only a small fraction of many activities and developments in the high energy physics computing community.
- The focus will be non-traditional tools (a.k.a. non-ROOT) and workflows

Goals

- Understand why we want new tools
- Familiarize with new computing tools from both industry-driven and HEP-driven
- Walk-through hands-on materials
- Implement into YOUR (analysis) workflow

Caveats & Goals

Caveats

- This talk covers only a small fraction of many activities and developments in the high energy physics computing community.
- The focus will be non-traditional tools (a.k.a. non-ROOT) and workflows

Goals

- Understand why we want new tools
- Familiarize with new computing tools from both industry-driven and HEP-driven
- Walk-through hands-on materials
- Implement into YOUR (analysis) workflow

Please stop and ask any questions any time!!

https://github.com/kyungeonchoi/analysis_in_python

WWW was born at CERN in 1989

Big data, data science, etc. have been friends of high energy physicists

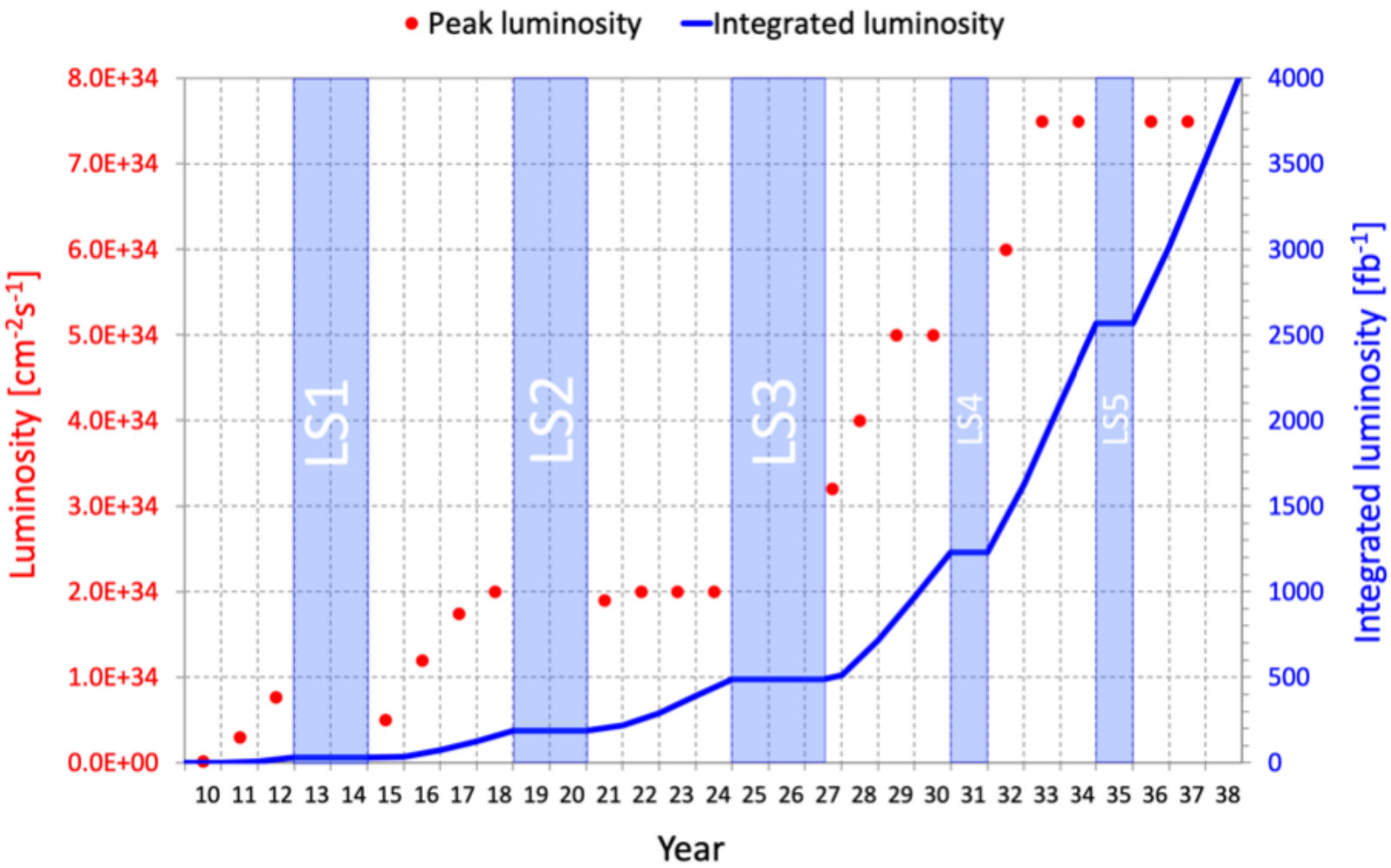
Why computing in 2023?

Two main driving forces

- High-Luminosity LHC (with Long Shutdowns)
- Boom in data science

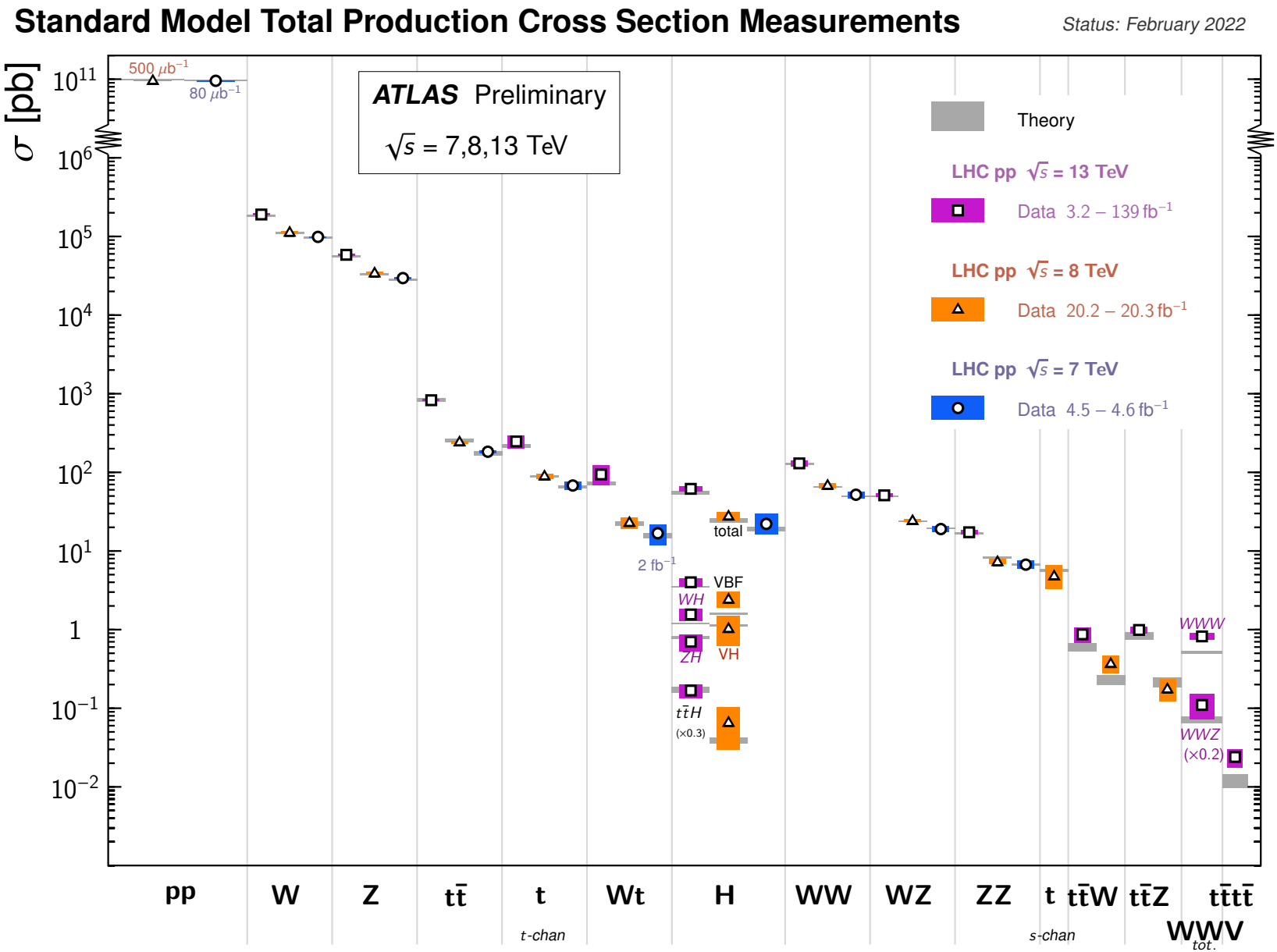
allow us to explore modern data science stacks

LHC Luminosity towards High-Luminosity LHC era

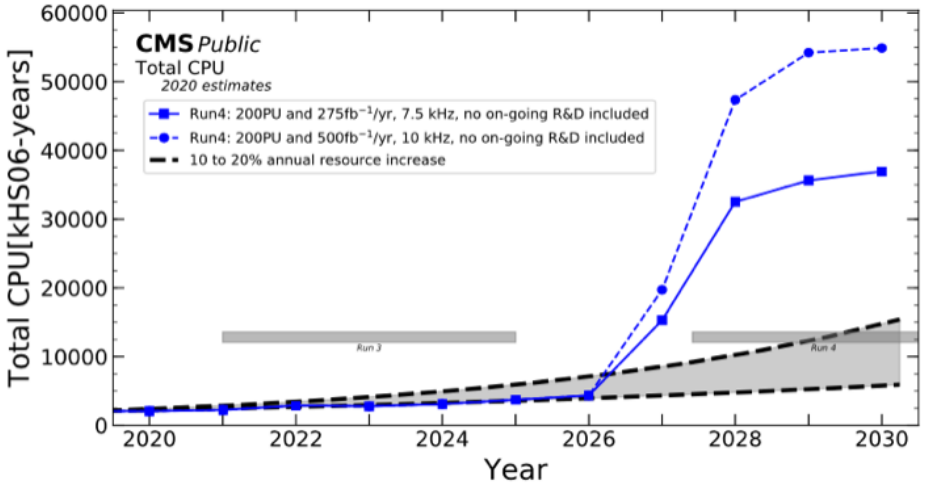


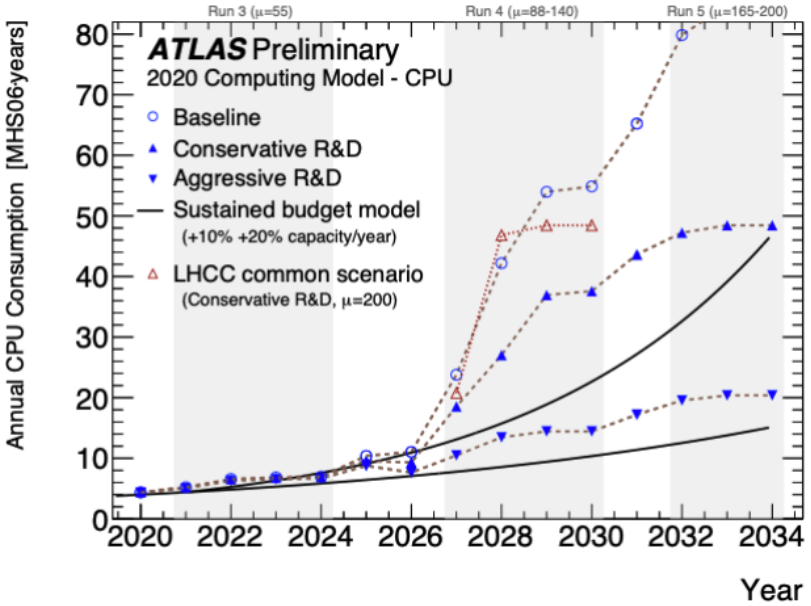
CMS NOTE-2021/001

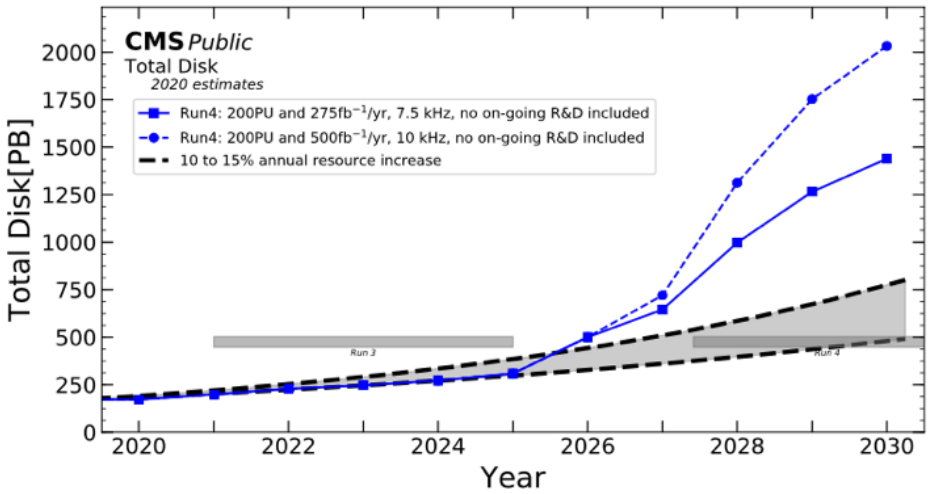
Why do we want to accumulate more data?

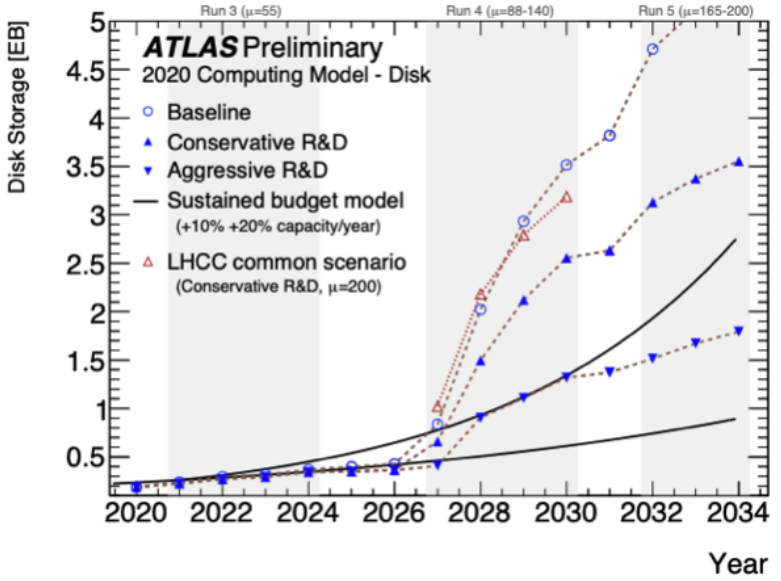


So, how much computing power do we need?



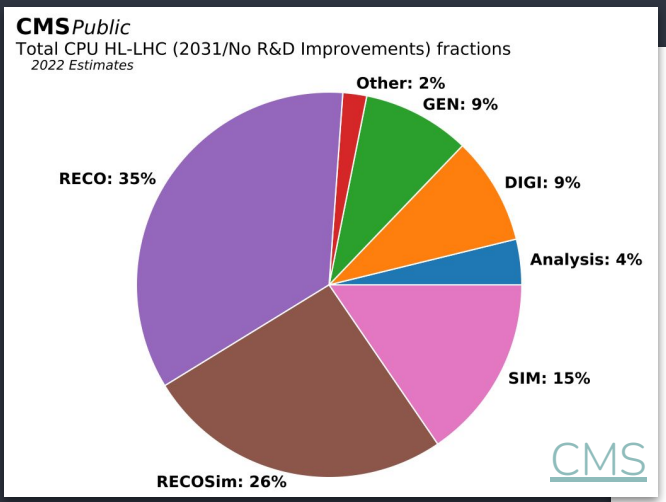


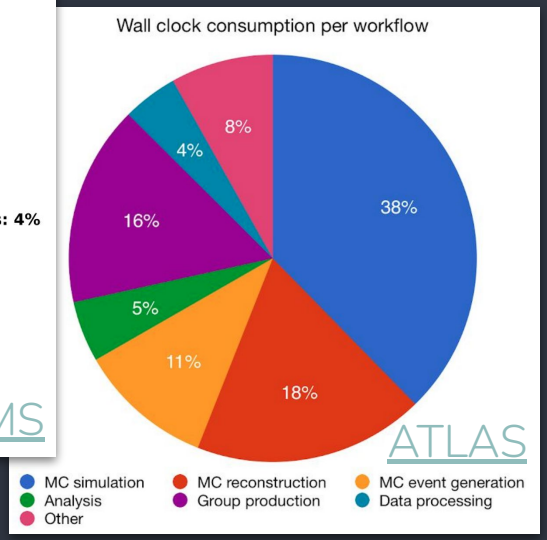




Why we care?

CPU time

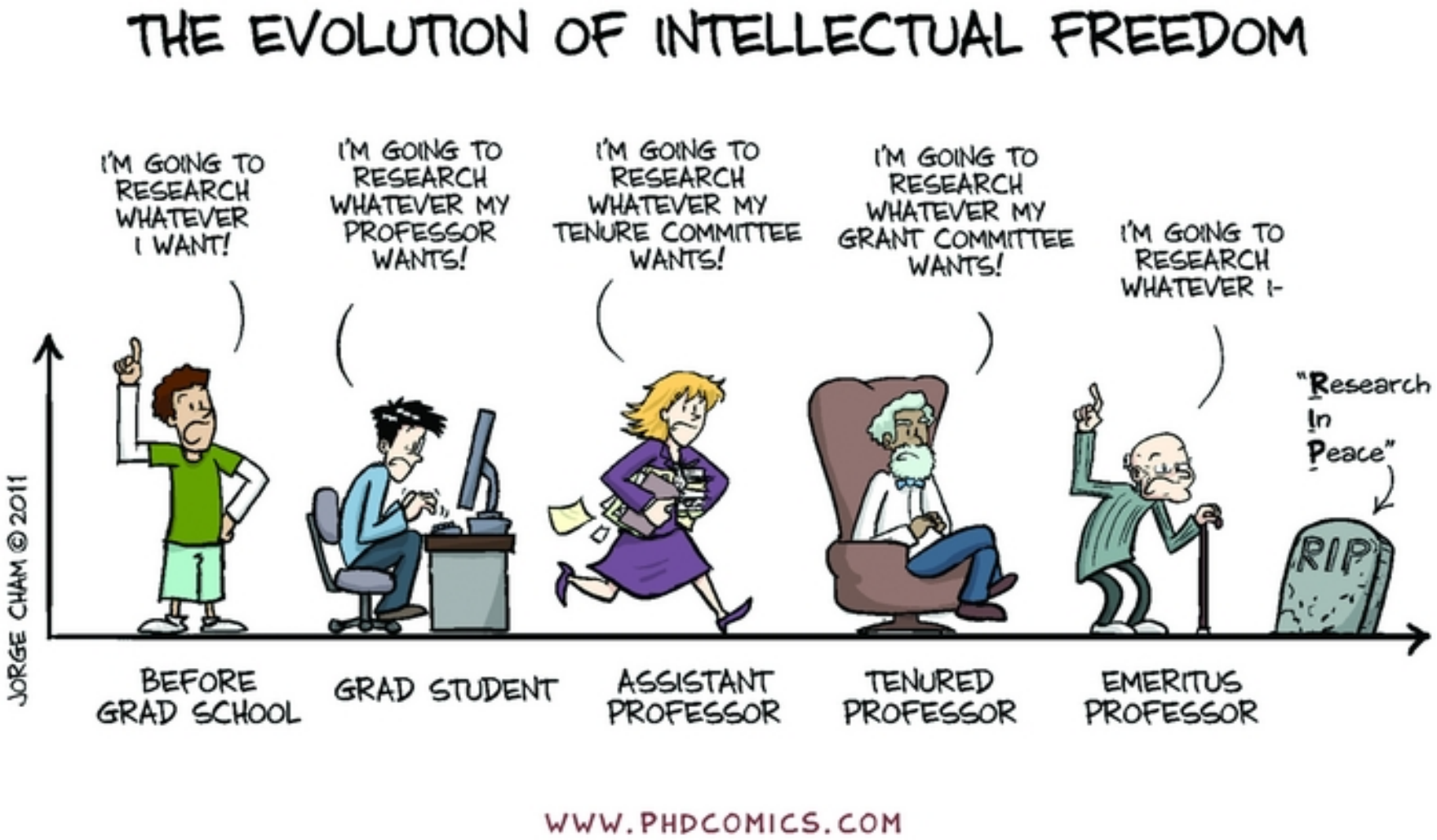




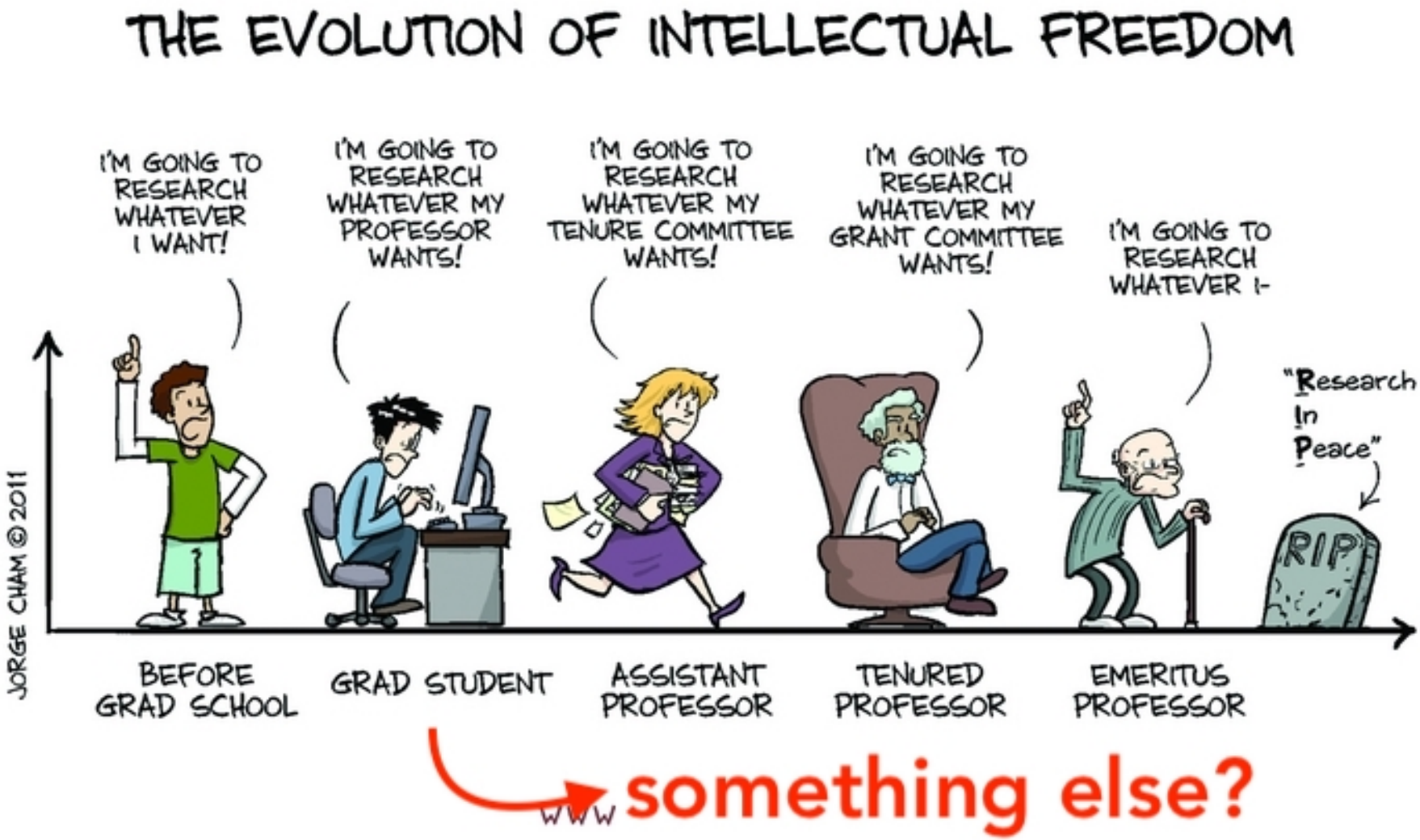
PhD student time

E.Guiraud, ACAT 2022

Why I care?


THE EVOLUTION OF INTELLECTUAL FREEDOM
I'M GOING TO RESEARCH WHATEVER I WANT!
I'M GOING TO RESEARCH WHATEVER MY PROFESSOR WANTS!
I'M GOING TO RESEARCH WHATEVER MY TENURE COMMITTEE WANTS!
I'M GOING TO RESEARCH WHATEVER MY GRANT COMMITTEE WANTS!
I'M GOING TO RESEARCH WHATEVER I-
"Research In Peace"
RIP
JORGE CHAM © 2011
BEFORE GRAD SCHOOL
GRAD STUDENT
ASSISTANT PROFESSOR
TENURED PROFESSOR
EMERITUS PROFESSOR
WWW.PHDCOMICS.COM

Why I care?


THE EVOLUTION OF INTELLECTUAL FREEDOM
I'M GOING TO RESEARCH WHATEVER I WANT!
I'M GOING TO RESEARCH WHATEVER MY PROFESSOR WANTS!
I'M GOING TO RESEARCH WHATEVER MY TENURE COMMITTEE WANTS!
I'M GOING TO RESEARCH WHATEVER MY GRANT COMMITTEE WANTS!
I'M GOING TO RESEARCH WHATEVER I-
"Research In Peace"
RIP
JORGE CHAM © 2011
BEFORE GRAD SCHOOL
GRAD STUDENT
ASSISTANT PROFESSOR
TENURED PROFESSOR
EMERITUS PROFESSOR
WWW
something else?

symmetry topics follow +

From physics to data science

05/21/19 | By Sarah Charley

Four physicists share their journeys through academia into industry and offer words of wisdom for those considering making a similar move.

Throughout his higher education, Jamie Antonelli had always envisioned himself as one day becoming a physics professor. All of his role models were professors; all of his peers were working to become professors; all of his research was preparing him for a career as a professor.

"I was living in a bubble," Antonelli says. "I was keeping my head down and following the same path as everyone around me instead of taking an honest look at my future."

Every year, a few hundred students like Antonelli graduate with PhDs in particle physics. And every year, only about a dozen permanent positions open up at universities and research institutions. As Antonelli and his peers navigated cycles of applications and rejections, he was hit with a hard truth: Most PhD physicists will leave academia.

https://www.symmetrymagazine.org/article/from-physics-to-data-science

Outline

ROOT - "The" HEP software

- Developed at CERN and 20 years of solid development
- **All-in-one**:
  - save data in a compressed binary form in a ROOT file
  - access data from PC or grid system
  - analyze data using mathematical and statistical tools
  - publish results by generating beautiful plots
- C++ ecosystem
- ROOT + your favorate batch system (e.g. HTcondor) is still the default software framework in HEP community

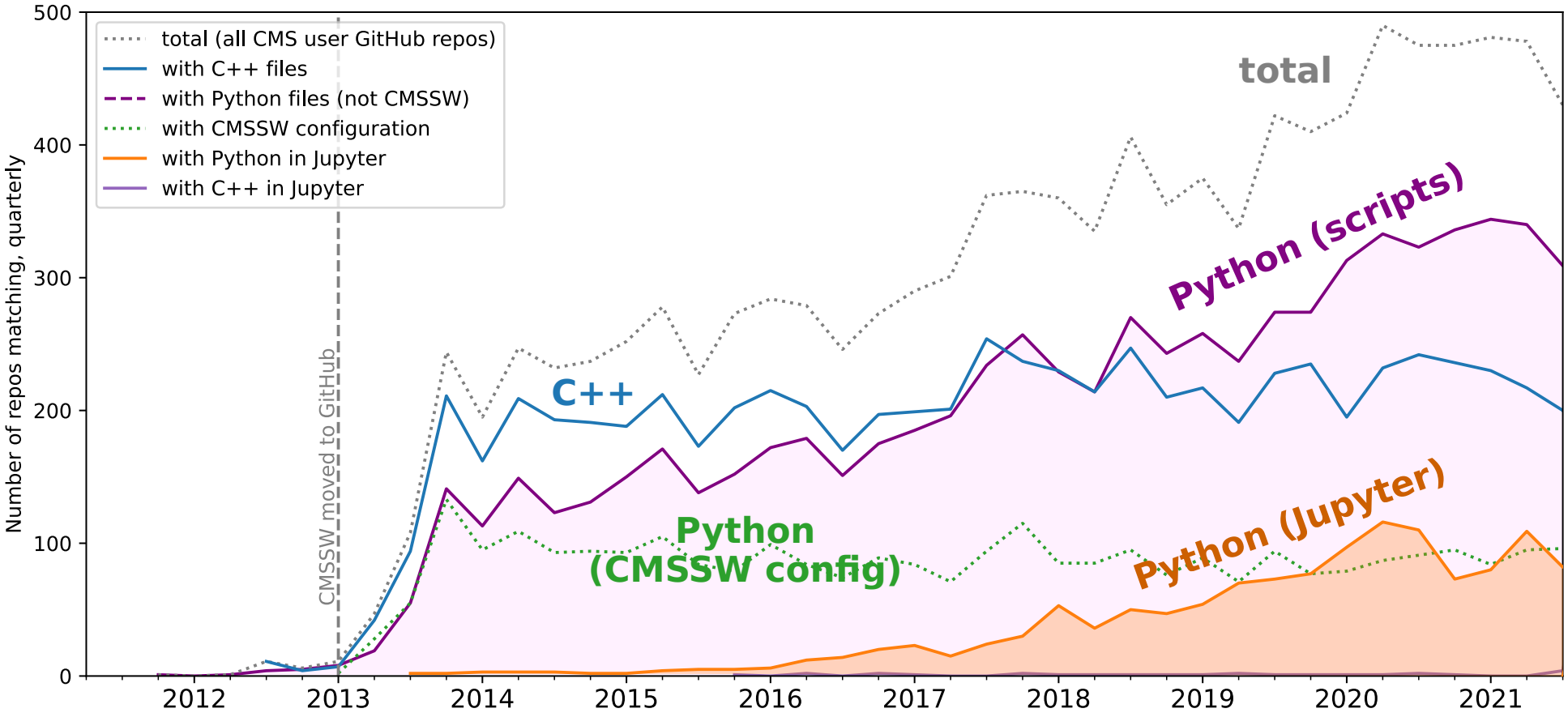
Scope of the New HEP computing Tools in this talk

- Community-driven project for HEP Python ecosystem
- https://scikit-hep.org

- Software R&D institute for all aspsects of HEP computing
- Mostly ATLAS and CMS institutions
- **Not only python tools** but many others such as machine learning, data management, analysis facility and so on

Rapid rise of Python for analysis in HEP

Source: "import XYZ" matches in GitHub repos for users who fork CMSSW.

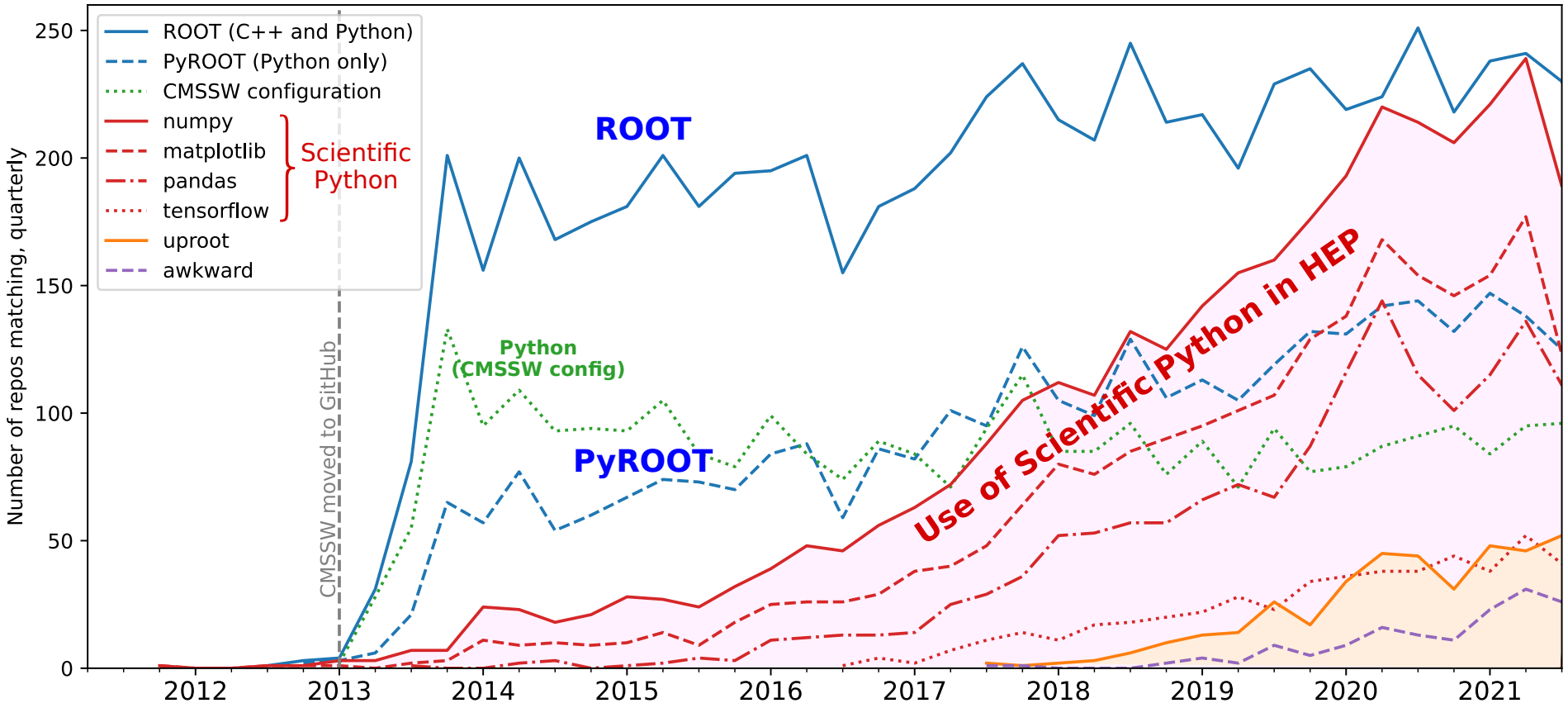


(CMSSW is the CMS experiment's "offline" software framework)

J.Pivarski, M. Feickert

Explosion of Scientific Python (NumPy, etc.) use recent since 2018

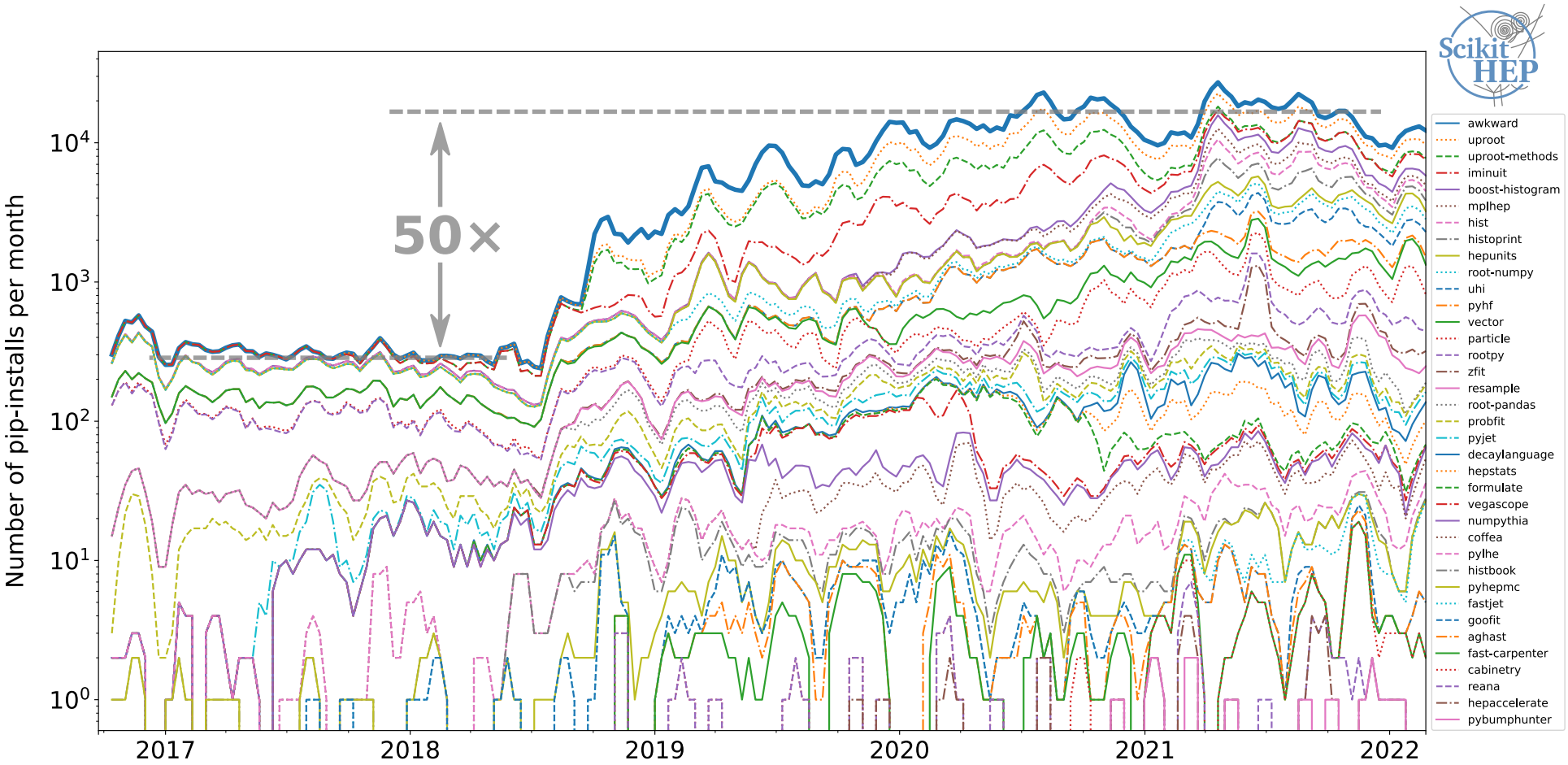
Source: "import XYZ" matches in GitHub repos for users who fork CMSSW.



J.Pivarski, M. Feickert

Growth tightly coupled to the rise of Scikit-HEP supported by IRIS-HEP

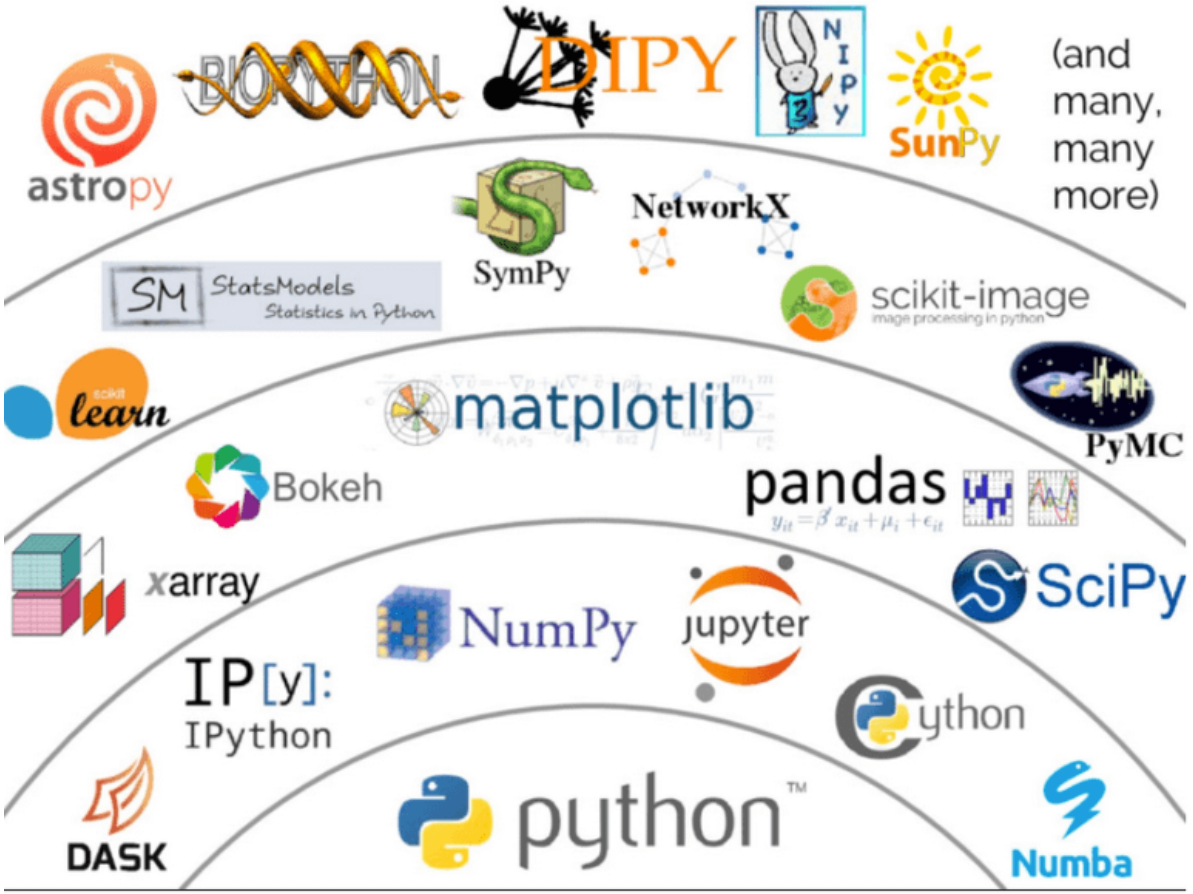
Source: "pip install XYZ" download rate for MacOS/Windows (no batch jobs).



J.Pivarski, M. Feickert

Python ecosystem for data science

In his PyCon 2017 keynote, Jake VanderPlas gave us the iconic "PyData ecosystem" image



Python ecosystem for HEP

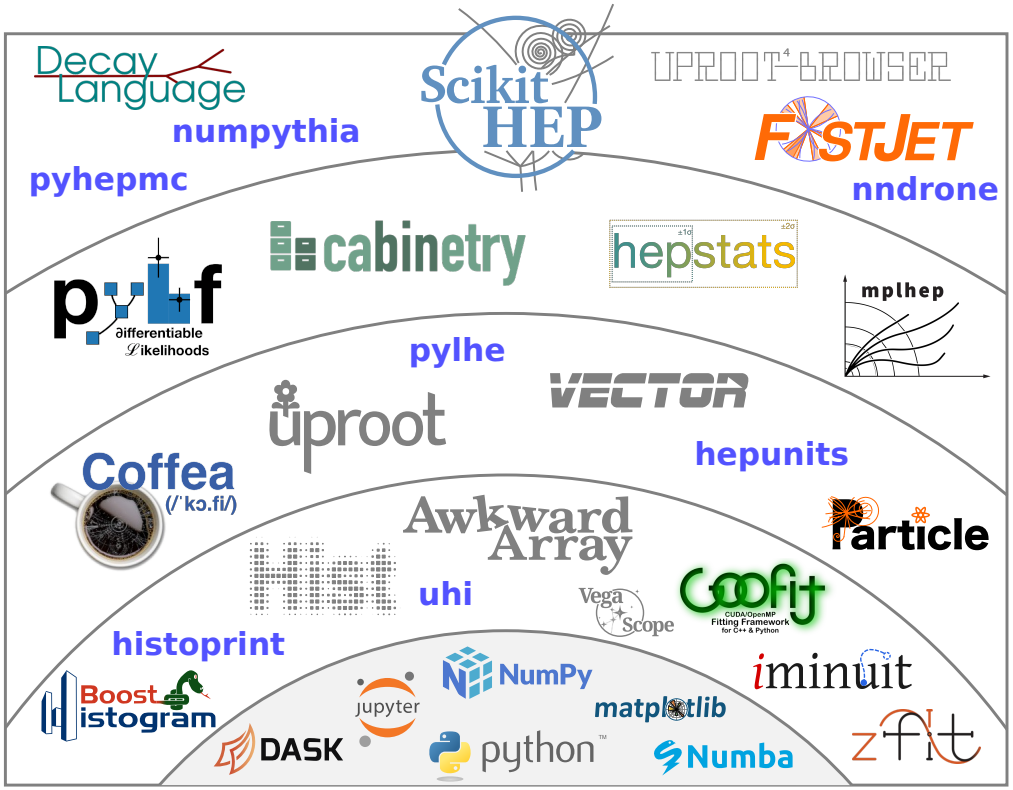
5 HEP-specific UI applications or packaged algorithms

4 HEP-specific for common problems

3 HEP-specific, foundational

2 needed to create, but not really HEP-specific

1 non-HEP software we depend on



J.Pivarski, M. Feickert

iris hep

About ▾ Connect ▾ Activities ▾ Fellows Jobs

Institute for Research and Innovation in Software for High Energy Physics (IRIS-HEP)

Computational and data science research to enable discoveries in fundamental physics

IRIS-HEP is a software institute funded by the National Science Foundation. It aims to develop the state-of-the-art software cyberinfrastructure required for the challenges of data intensive scientific research at the High Luminosity Large Hadron Collider (HL-LHC) at CERN, and other planned HEP experiments of the 2020's. These facilities are discovery machines which aim to understand the fundamental building blocks of nature and their interactions. Full Overview

News and Featured Stories:

Developing an inclusive space for women in software at IRIS-HEP

Pattern-Recognition Experts Connect the Dots at Princeton

Upcoming Events:

Feb 8–10, 2023 *Virtual*
HSF/IRIS-HEP Software Basics Training (Virtual)

May 17–19, 2023 *Virtual*
HSF/IRIS-HEP Software Basics Training (Virtual)

View all past events

Upcoming Topical Meetings:

No meetings currently scheduled. Check back again soon!

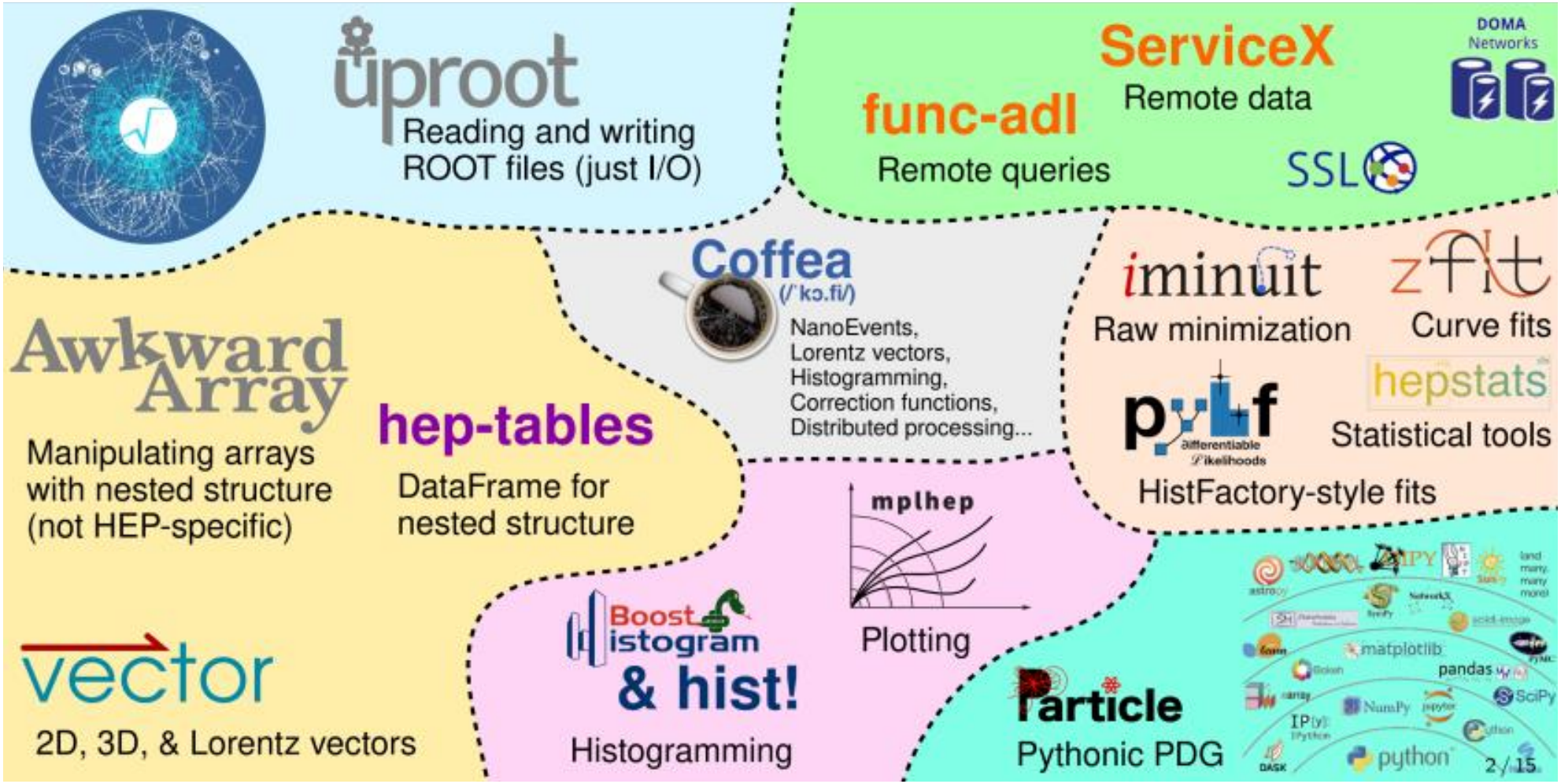
View all • Indico (recordings)

Related projects:

ATLAS • CMS • LHCb • USATLAS • U.S. ATLAS Operations Program • USCMS • U.S. CMS Operations Program • OSG • PATh • SOTERIA • SciAuth • EWMS •

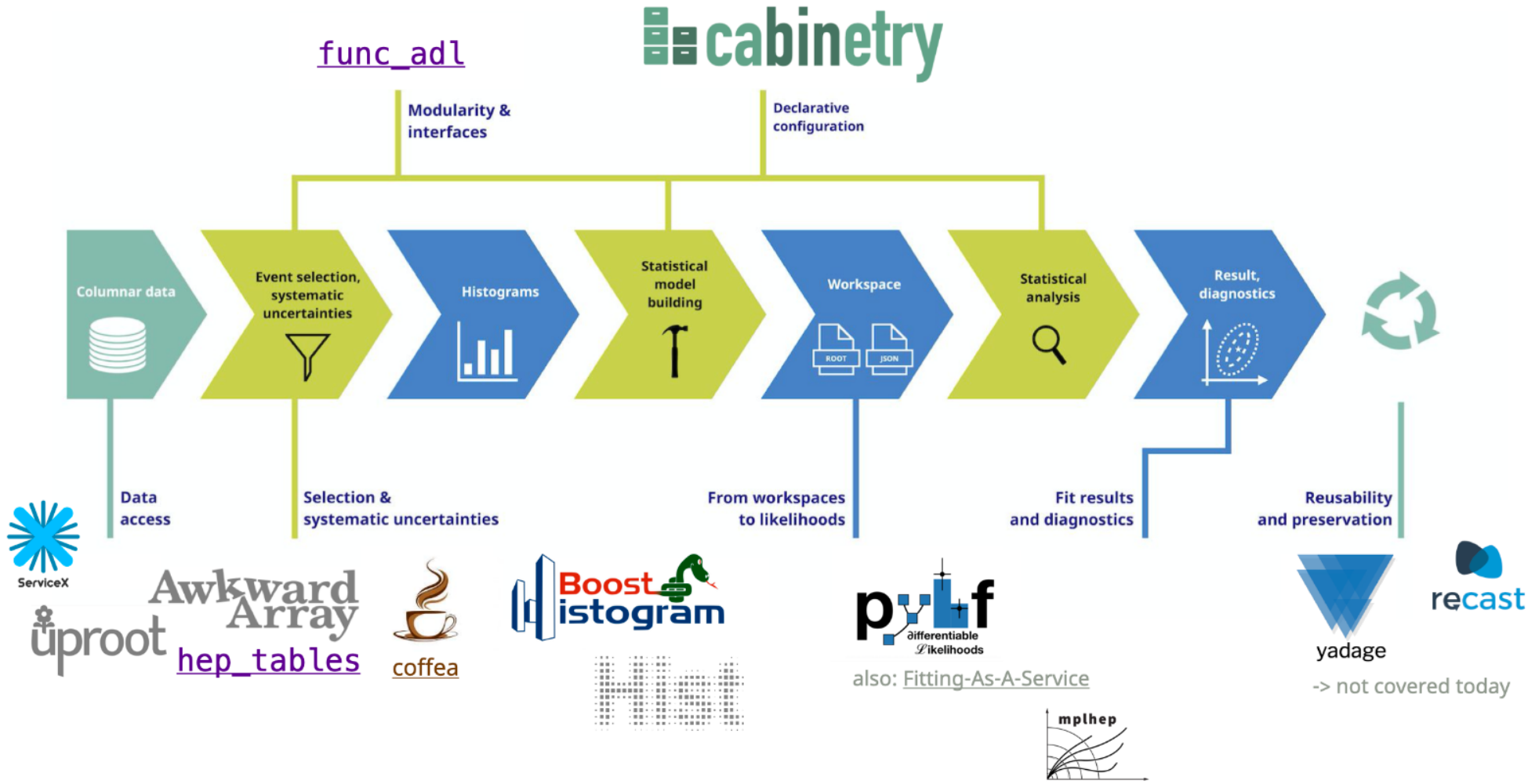
https://iris-hep.org

Building blocks from IRIS-HEP + Scikit-HEP



J.Pivarski

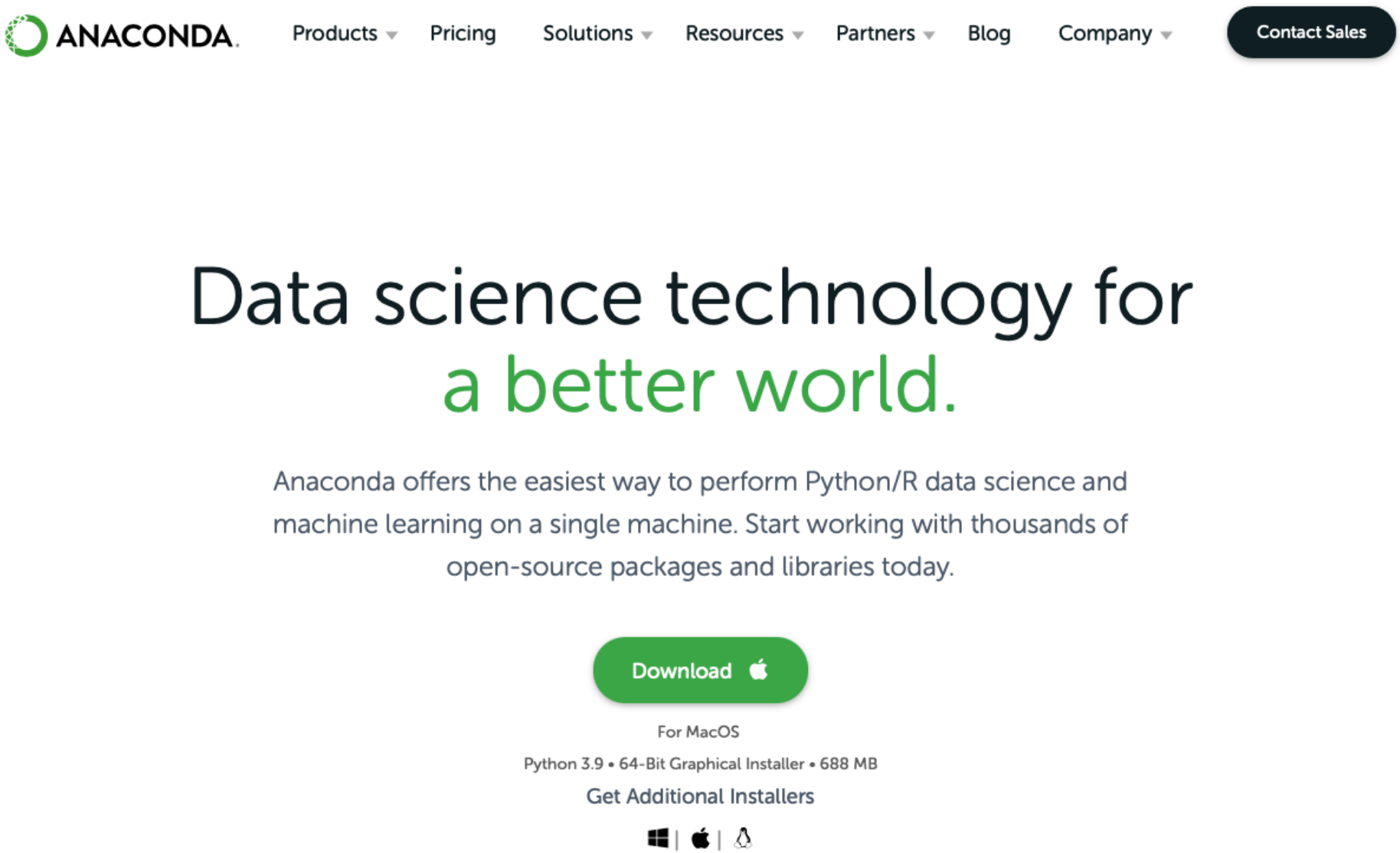
Physics Analysis Workflow in Python ecosystem



A. Held

Outline

ANACONDA - First thing is first



https://www.anaconda.com

uproot

latest

Search docs

Release history

TUTORIALS

Getting started guide

Uproot 3 → 4+ cheat-sheet

MAIN INTERFACE

uproot.open

uproot.iterate

uproot.concatenate

uproot.dask

uproot.create

uproot.recreate

uproot.update

uproot.ReadOnlyFile

uproot.ReadOnlyDirectory

uproot.TTree

uproot.TBranch

uproot.WritableFile

uproot.WritableDirectory

uproot.WritableTree

uproot

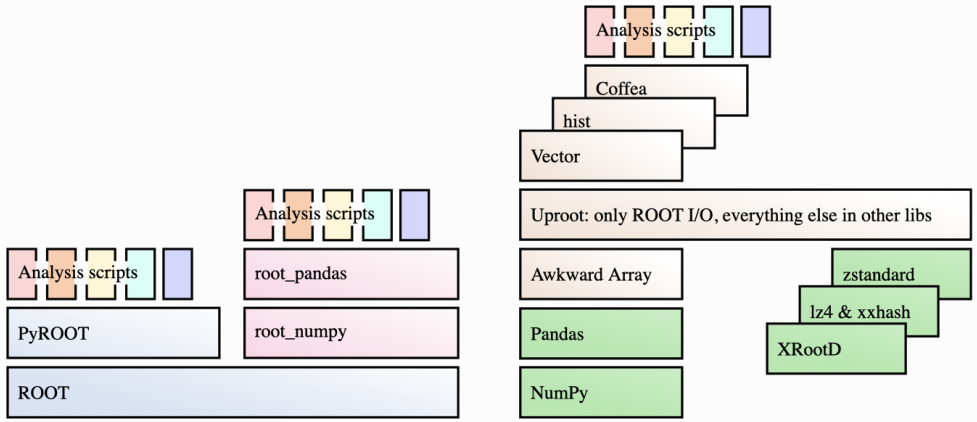
pypi package 5.0.2 | conda-forge v4.3.7 | python 3.7–3.11 | license BSD 3-Clause

tests https://github.com/badges/shields/issues/8671

Scikit-HEP Project | NSF 1836650 | DOI 10.5281/zenodo.4340632 | docs online | chat online

Uproot is a library for reading and writing ROOT files in pure Python and NumPy.

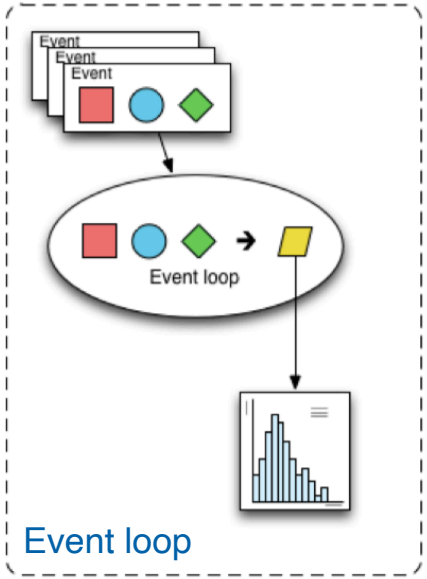
Unlike the standard C++ ROOT implementation, Uproot is only an I/O library, primarily intended to stream data into machine learning libraries in Python. Unlike PyROOT and root_numpy, Uproot does not depend on C++ ROOT. Instead, it uses Numpy to cast blocks of data from the ROOT file as Numpy arrays.



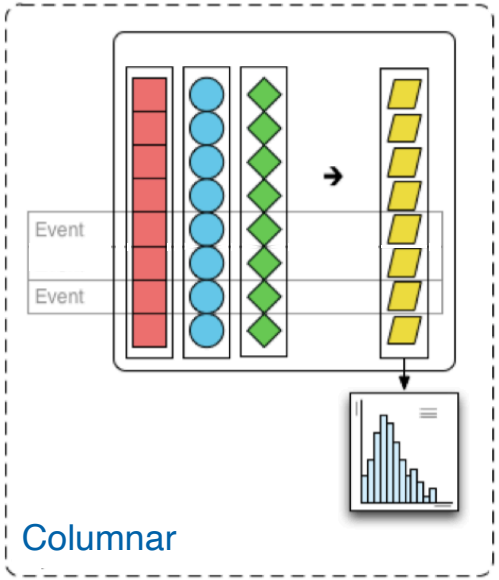
- readthedoc page
- GitHub Page

What is columnar analysis?

- Event loop analysis:
  - Load relevant values for a specific event into local variables
  - Evaluate several expressions
  - Store derived values
  - Repeat (explicit outer loop)



- Columnar analysis:
  - Load relevant values for many events into contiguous arrays
  - Evaluate several **array programming** expressions
    - Implicit *inner* loop**s**
  - Store derived values



Fermilab

Awkward-arrays

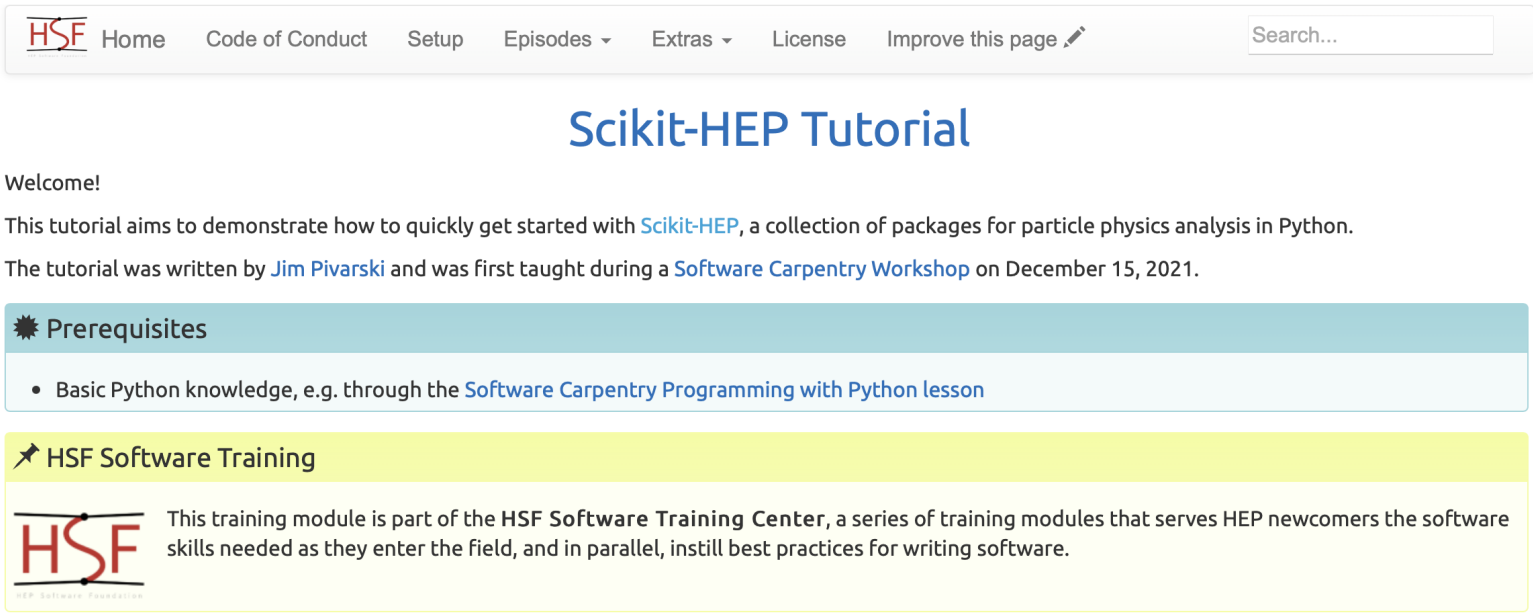
```
>>> ak.Array([
...     [1, 2, 3],
...     [4]
... ])
<Array [[1, 2, 3], [4]] type='2 * var * int64'>
```

```
x = np.array([1, 2, 3])
y = np.array(
    [
        [4, 5, 6],
        [7, 8, 9],
    ]
)
np.broadcast_arrays(x, y)
```

- `Uproot` and `Awkward-arrays` are so-called foundation libraries
- event-at-a-time vs. array-at-a-time
- https://awkward-array.org

Hands-on: Uproot, Awkward-arrays

https://hsf-training.github.io/hsf-training-scikit-hep-webpage/



- (Too) many (nice) tutorials → (maybe) not easy to choose where or what to start
- `Uproot` and `Awkward-arrays` are foundation libraries
- Let's walk-through the following sections of Scikit-HEP Tutorial
  1. Introduction: Python background
  2. Basic file I/O with Uproot
  3. TTree details
  4. Jagged, ragged, Awkward Arrays

```
1    python -m pip install scikit-hep-testdata
```

Hist, mplhep

Physicists have created at least 20 histogram libraries in Python, most 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)

hist

[1]:
```
from hist import Hist
import hist
```

[2]:
```
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),
)
```

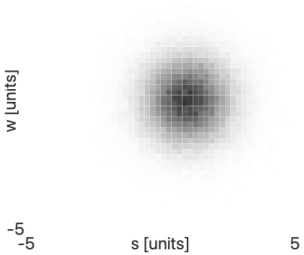
[3]:
```
import numpy as np

s_data = np.random.normal(size=100_000) + np.ones(100_000)
w_data = np.random.normal(size=100_000)

# normal fill
h.fill(s_data, w_data)
```

[3]:
```
Regular(50, -5, 5, underflow=False, overflow=False,
name='S', label='s [units]')
Regular(50, -5, 5, underflow=False, overflow=False,
name='W', label='w [units]')

Double() Σ=100000.0
```

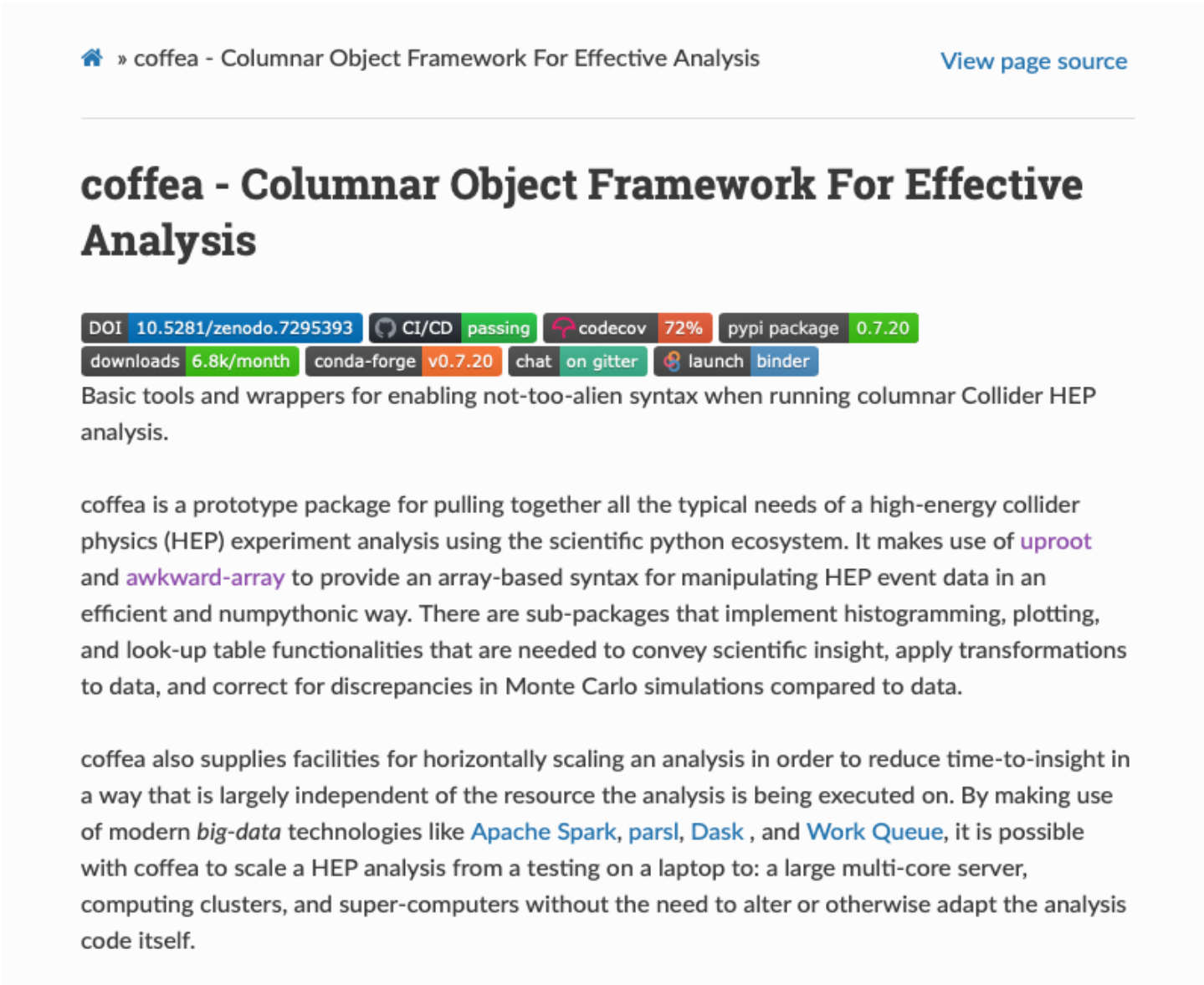


mplhep

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

# Load style sheet
plt.style.use(hep.style.CMS)  # or ATLAS/LHCb2

h, bins = np.histogram(np.random.random(1000))
fig, ax = plt.subplots()
hep.histplot(h, bins)
```



» coffea - Columnar Object Framework For Effective Analysis View page source

coffea - Columnar Object Framework For Effective Analysis

DOI 10.5281/zenodo.7295393 CI/CD passing codecov 72% pypi package 0.7.20
downloads 6.8k/month conda-forge v0.7.20 chat on gitter launch binder

Basic tools and wrappers for enabling not-too-alien syntax when running columnar Collider HEP analysis.

coffea is a prototype package for pulling together all the typical needs of a high-energy collider physics (HEP) experiment analysis using the scientific python ecosystem. It makes use of uproot and awkward-array to provide an array-based syntax for manipulating HEP event data in an efficient and numpythonic way. There are sub-packages that implement histogramming, plotting, and look-up table functionalities that are needed to convey scientific insight, apply transformations to data, and correct for discrepancies in Monte Carlo simulations compared to data.

coffea also supplies facilities for horizontally scaling an analysis in order to reduce time-to-insight in a way that is largely independent of the resource the analysis is being executed on. By making use of modern *big-data* technologies like Apache Spark, parsl, Dask , and Work Queue, it is possible with coffea to scale a HEP analysis from a testing on a laptop to: a large multi-core server, computing clusters, and super-computers without the need to alter or otherwise adapt the analysis code itself.

Keywords

- NanoEvents
- Processors
- CMS tools
- Scale-out

Hands-on

- NanoEvents
- coffea processor
- or Together

Outline

Physics Analysis Workflow in Python ecosystem

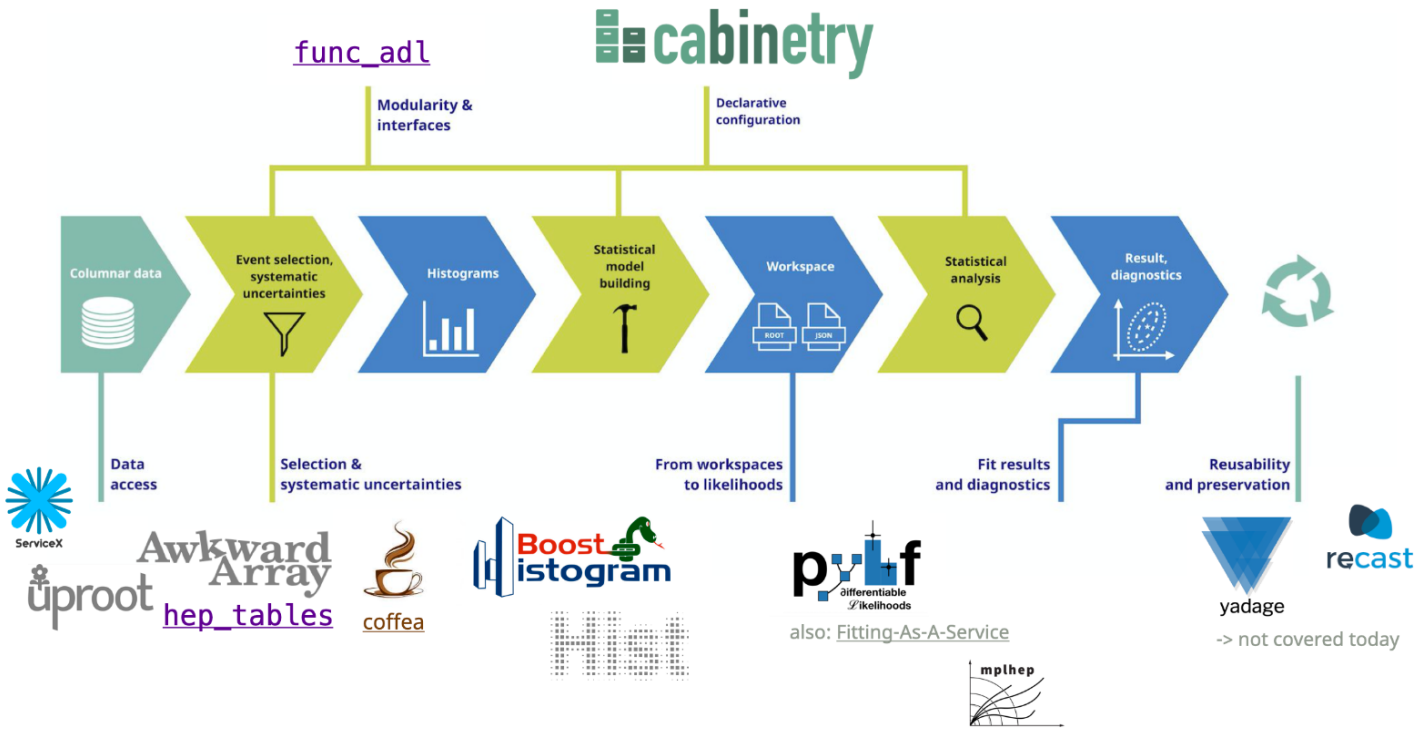
CMS Open Data $t\bar{t}$: from data delivery to statistical inference

We are using 2015 CMS Open Data in this demonstration to showcase an analysis pipeline. It features data delivery and processing, histogram construction and visualization, as well as statistical inference.

This notebook was developed in the context of the IRIS-HEP AGC tools 2022 workshop. This work was supported by the U.S. National Science Foundation (NSF) Cooperative Agreement OAC-1836650 (IRIS-HEP).

This is a **technical demonstration**. We are including the relevant workflow aspects that physicists need in their work, but we are not focusing on making every piece of the demonstration physically meaningful. This concerns in particular systematic uncertainties: we capture the workflow, but the actual implementations are more complex in practice. If you are interested in the physics side of analyzing top pair production, check out the latest results from ATLAS and CMS! If you would like to see more technical demonstrations, also check out an ATLAS Open Data example demonstrated previously.

This notebook implements most of the analysis pipeline shown in the following picture, using the tools also mentioned there:



Backup