Pythonic Data Analysis

Jim Pivarski

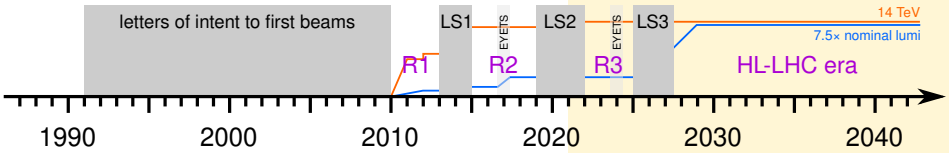
Princeton University – IRIS-HEP

May the 4th be with you

Part 1: Trends in data analysis software

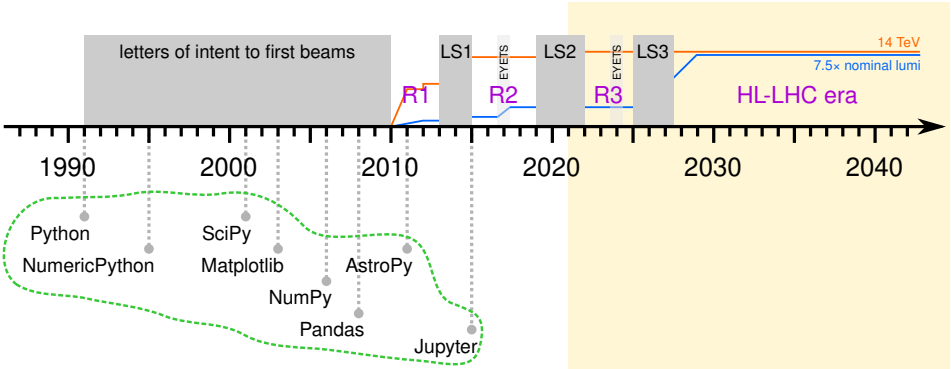
Part 2: Status of Pythonic HEP software

Pythonic data analysis in the HL-LHC era?

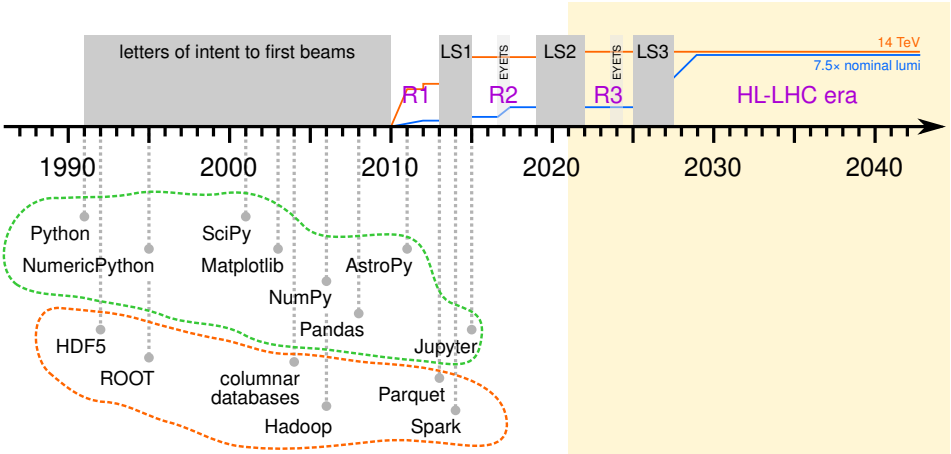


future

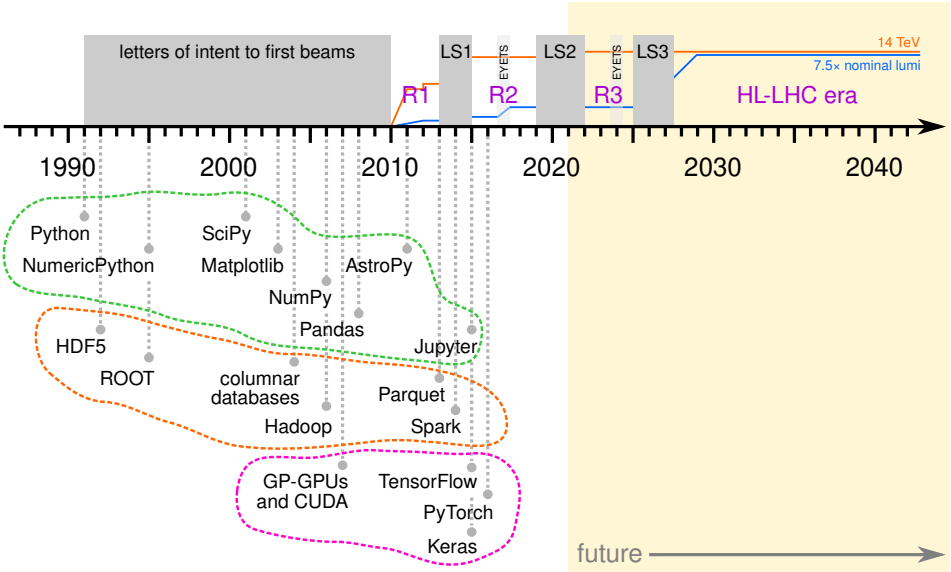
Pythonic data analysis in the HL-LHC era?



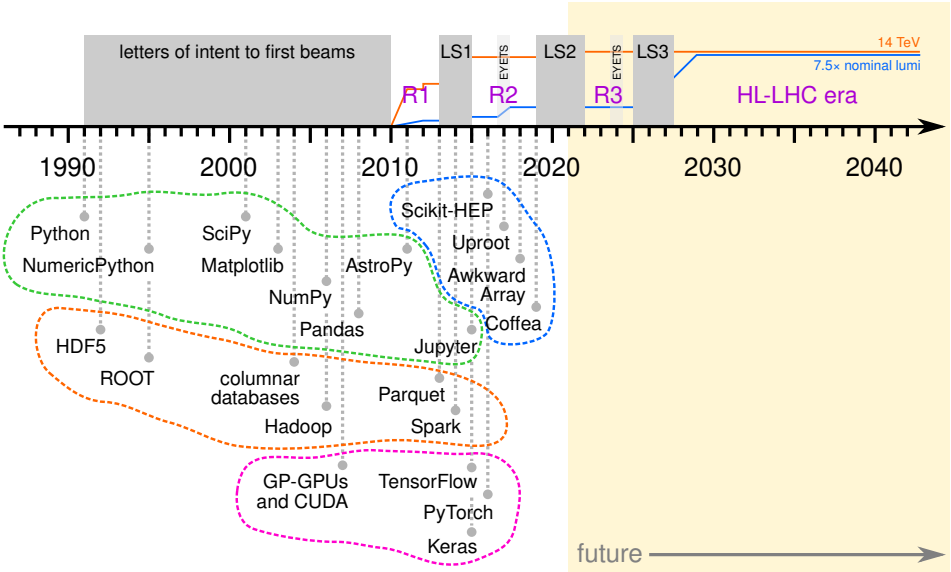
Pythonic data analysis in the HL-LHC era?



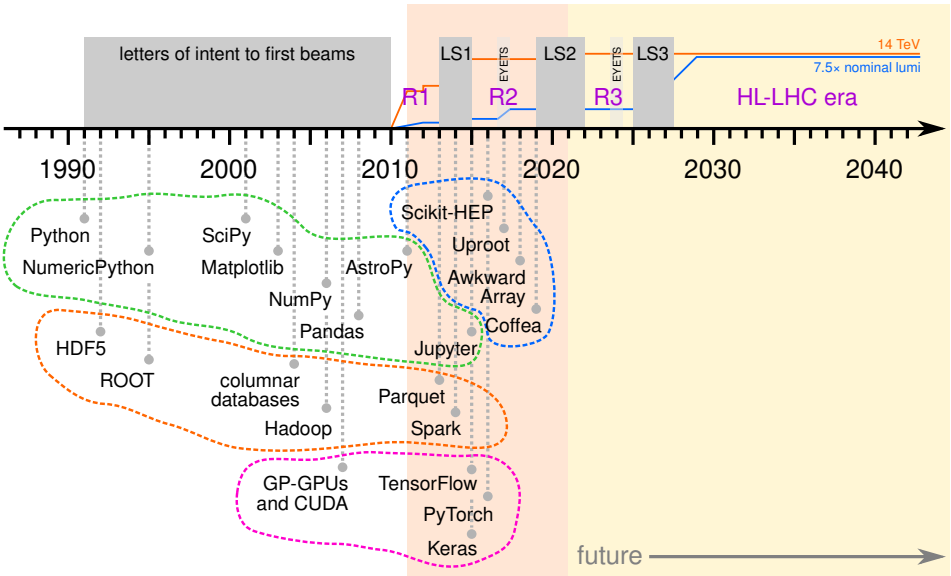
Pythonic data analysis in the HL-LHC era?



Pythonic data analysis in the HL-LHC era?

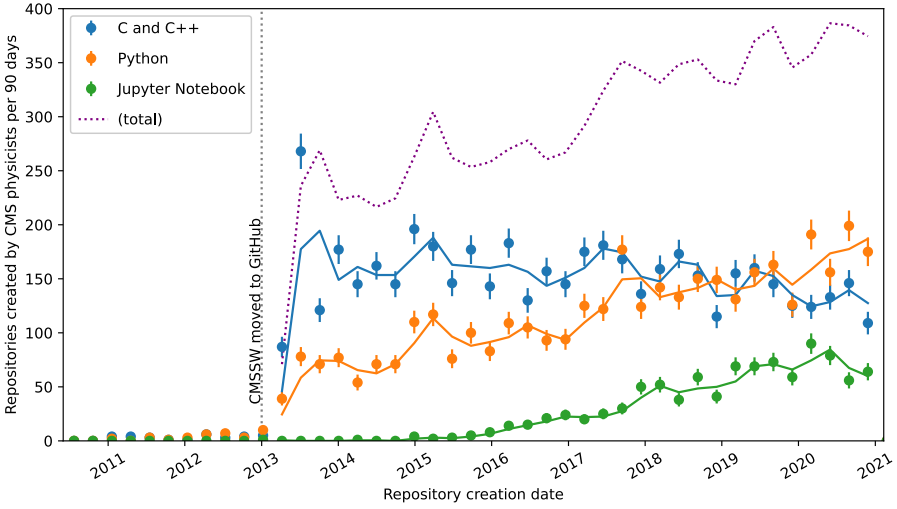


Pythonic data analysis in the HL-LHC era?

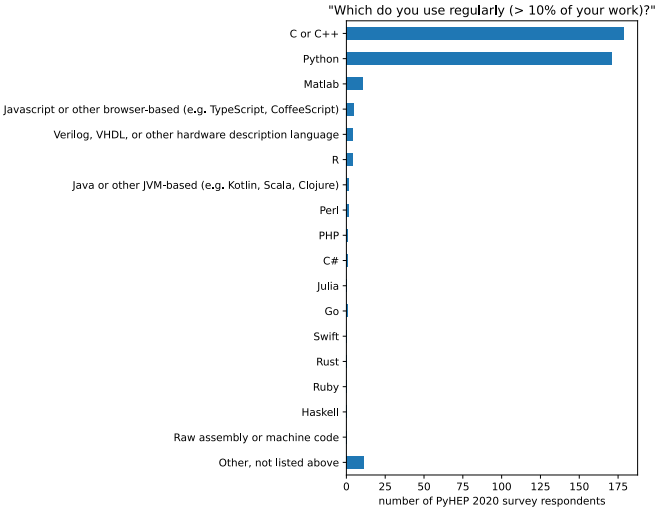


What were the physicists doing in this decade?

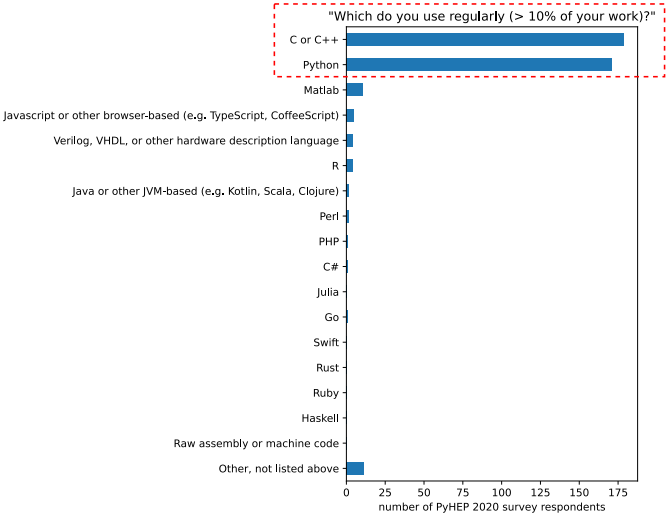
Primary language of GitHub repos created by users who forked CMSSW:



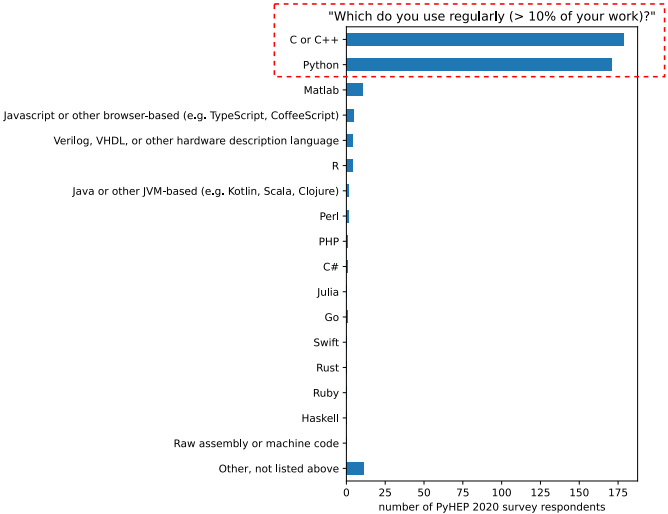
Consistent with survey results (PyHEP 2020 participants)

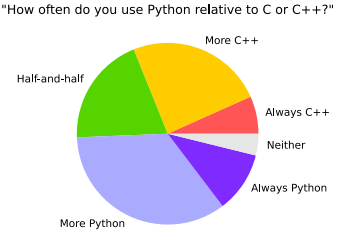


Consistent with survey results (PyHEP 2020 participants)

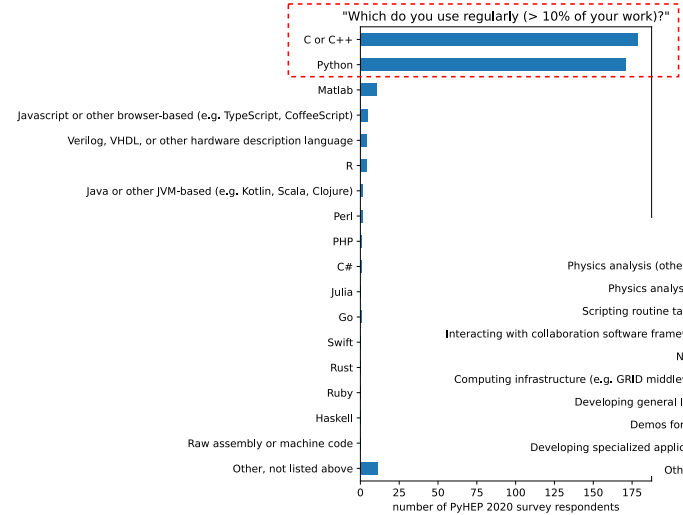


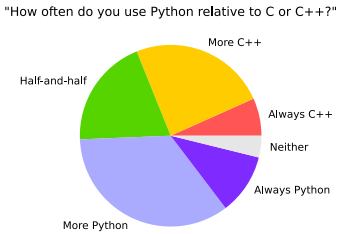
Consistent with survey results (PyHEP 2020 participants)

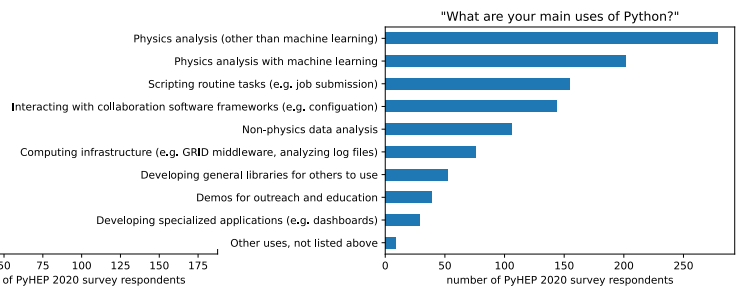




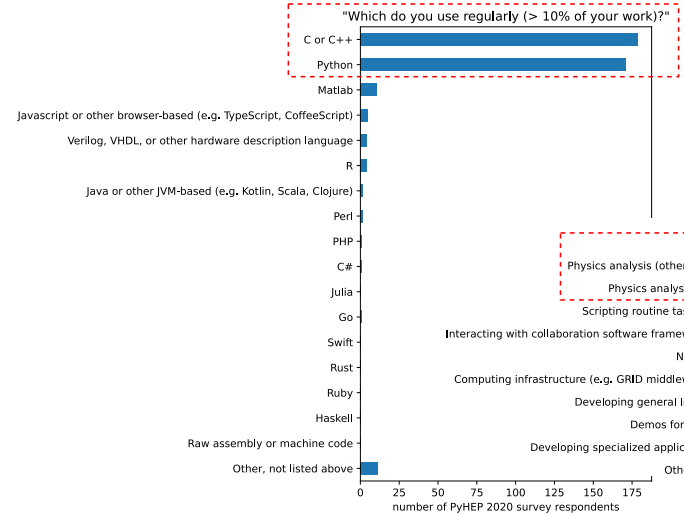
Consistent with survey results (PyHEP 2020 participants)

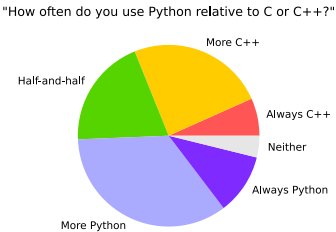


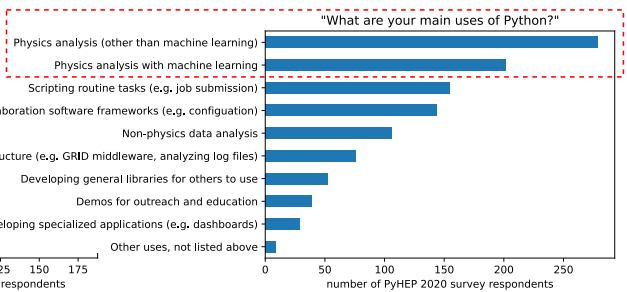




Consistent with survey results (PyHEP 2020 participants)

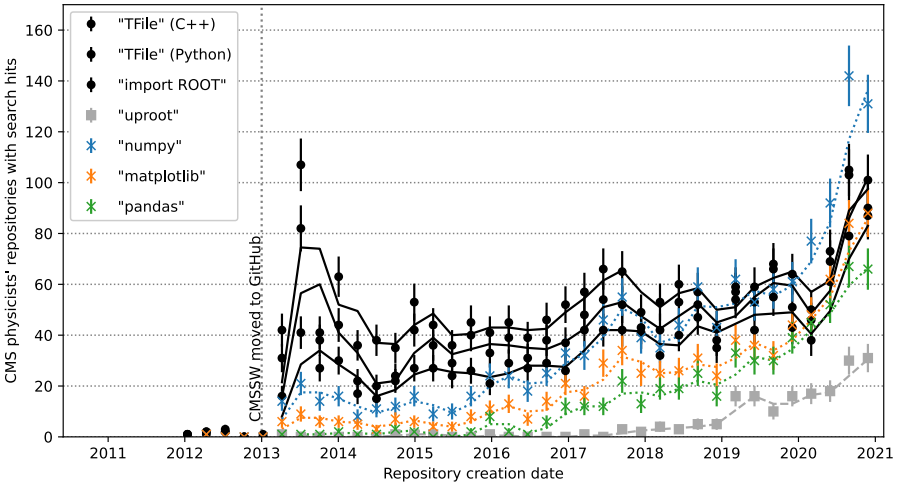




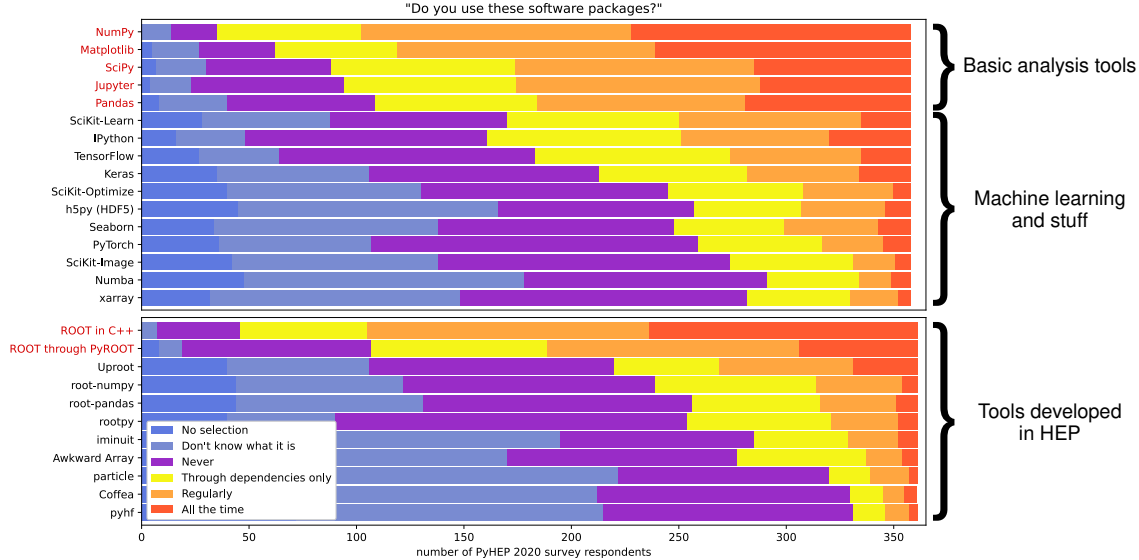


Pythonic analysis is mainstream, and the trend preceded Uproot

GitHub repos matching search strings: Pythonic analysis is as common as "TFile".



This is also consistent with self-reported usage



Part 1: Trends in data analysis software

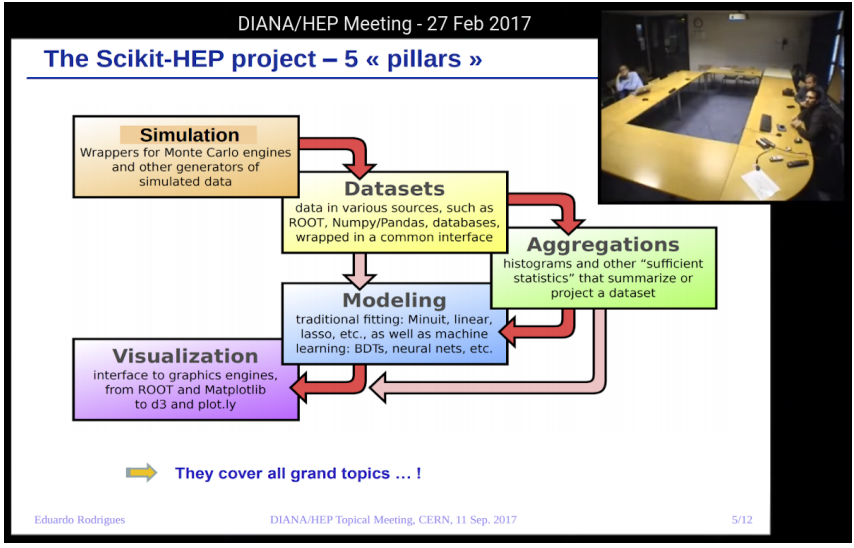
Part 2: Status of Pythonic HEP software

Not a single group, but mostly shared vision

Most of the developers of Pythonic HEP software

- aim for small package granularity, providing tools that address a well-defined class of problems at one level of abstraction,
- aim for interoperability with each other and the larger Pythonic ecosystem,
- avoid overlapping functionality, by communicating through HSF and IRIS-HEP channels,
- focus on domain-specific problems that won't be addressed by non-HEP software *or* focus on connecting HEP-specific tools to the larger ecosystem.

Scikit-HEP: a clearinghouse for Pythonic HEP software



Originally conceived as a core package with "affiliates" like AstroPy; now it's "just affiliates."

A common brand for packages that work together and with the Python ecosystem.

Scikit-HEP's 5 pillars today:

Simulation: other than `numpythia` and `pyhepmc`, this is mostly left to non-Python packages (which fill files that can be read with Python).

Datasets: `root-numpy`, `uproot`, `pylhe`, `awkward`, as well as `numpy` and `pandas`. Some use HDF5/`h5py` because of its recognition by ML tools.

Aggregations: `boost-histogram`/`hist`, `coffea`, `fast-carpenter`.

Modeling/fitting: by far, the most covered: `pyhf` (dozens of publications), `iminuit`, `zfit`, `hepstats`, `goofit`, `SModelS`, direct use of Pandas, ML...

Visualization: most modeling tools output to `matplotlib`, `mplhep` is widely used as a dependency. `coffea`, `hist`, `histoprint`.

Should also add Distributed computing: some `pyspark`, but more `dask`, often through `coffea` (future `coffea-casa`/ServiceX).

Should also add Acceleration/JIT-compilation: `numba`, `jax`, `ROOT.RDataFrame`.

Should also add HEP domain-specific: corrections in `coffea`, Lorentz vectors in `vector`, PDG in `particle`, jet clustering in `pyjet`...

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Aggregations: need better interop between boost-histograms and ROOT histograms. Also, common usage is tending toward "superhistograms," collections that describe systematic variations that ought to have shared metadata.

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Aggregations: need better interop between boost-histograms and ROOT histograms. Also, common usage is tending toward "superhistograms," collections that describe systematic variations that ought to have shared metadata.

Modeling/fitting: well-covered: needs are highly specialized and practitioners write their own packages. `scikit-hep/cookie` simplifies package-creation.

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Aggregations: need better interop between boost-histograms and ROOT histograms. Also, common usage is tending toward "superhistograms," collections that describe systematic variations that ought to have shared metadata.

Modeling/fitting: well-covered: needs are highly specialized and practitioners write their own packages. `scikit-hep/cookie` simplifies package-creation.

Visualization: in rapid development now, and mostly centralized in a few packages.

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Aggregations: need better interop between boost-histograms and ROOT histograms. Also, common usage is tending toward "superhistograms," collections that describe systematic variations that ought to have shared metadata.

Modeling/fitting: well-covered: needs are highly specialized and practitioners write their own packages. `scikit-hep/cookie` simplifies package-creation.

Visualization: in rapid development now, and mostly centralized in a few packages.

Distributed computing: early experiments with a wide variety of options are mostly narrowing on Dask, though I think we should keep an eye on Ray. The Query System concept is in rapid development in IRIS-HEP/ServiceX.

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Aggregations: need better interop between boost-histograms and ROOT histograms. Also, common usage is tending toward "superhistograms," collections that describe systematic variations that ought to have shared metadata.

Modeling/fitting: well-covered: needs are highly specialized and practitioners write their own packages. `scikit-hep/cookie` simplifies package-creation.

Visualization: in rapid development now, and mostly centralized in a few packages.

Distributed computing: early experiments with a wide variety of options are mostly narrowing on Dask, though I think we should keep an eye on Ray. The Query System concept is in rapid development in IRIS-HEP/ServiceX.

Acceleration: Awkward Arrays and Lorentz vectors can be JIT-compiled with Numba; histogramming is next. Awkward Arrays should also become interoperable with RDataFrame so we have both Python and C++ JIT-compilation.

Status/readiness for the HL-LHC:

Simulations: don't need to be in Python (much like event reconstruction).

Datasets: mostly covered, especially as minimalist formats like NanoAOD catch on. Pythonic access to RNTuple will be needed and is in development.

Aggregations: need better interop between boost-histograms and ROOT histograms. Also, common usage is tending toward "superhistograms," collections that describe systematic variations that ought to have shared metadata.

Modeling/fitting: well-covered: needs are highly specialized and practitioners write their own packages. `scikit-hep/cookie` simplifies package-creation.

Visualization: in rapid development now, and mostly centralized in a few packages.

Distributed computing: early experiments with a wide variety of options are mostly narrowing on Dask, though I think we should keep an eye on Ray. The Query System concept is in rapid development in IRIS-HEP/ServiceX.

Acceleration: Awkward Arrays and Lorentz vectors can be JIT-compiled with Numba; histogramming is next. Awkward Arrays should also become interoperable with RDataFrame so we have both Python and C++ JIT-compilation.

HEP domain-specific: `coffea` covers each use-case before it gets its own library.

Breakdown by strategy

Domain-specific stuff we do

- (super)histograms as fillable objects
- HEP-style plots (pulls, Brazil, efficiency...)
- ansatz fitting, limit setting, discovery significance
- applying corrections, clustering, Lorentz vector manipulation

Connections to externals

- ROOT, LHE files $\leftrightarrow$ arrays, Pandas, ML
- using ML packages to do HEP fits: `pyhf/combinetf`
- extending Numba for jagged arrays, Lorentz vectors, histogramming
- events $\rightarrow$ histograms workflow in distributed computing frameworks
- building Query Systems out of standard parts

External libraries we use

- machine learning
- distributed computing
- JIT-compilation
- autodifferentiation
- interactive notebooks

Conclusions

The question is not quite, "Will the whole ecosystem be ready for the HL-LHC?"

Many pieces are in-use *now*, and they're growing to fill real analysis needs *now*.

- I personally know of ∼10 analyses using Python almost exclusively (CMS $H \to \gamma\gamma$, $\to \mu\mu$, $\to c\bar{c}$, di-Higgs, top EFT, Dazsle analyses...).
- There is a wave of such analyses gradually approaching publication.

From current trends, it looks like Pythonic analysis will either predominate in Run 3 or stay evenly mixed with C++, it is today.