PyHEP.dev 2024 Questions and Discussions

Whole-workshop summary document: PyHEP.dev 2024 summary document

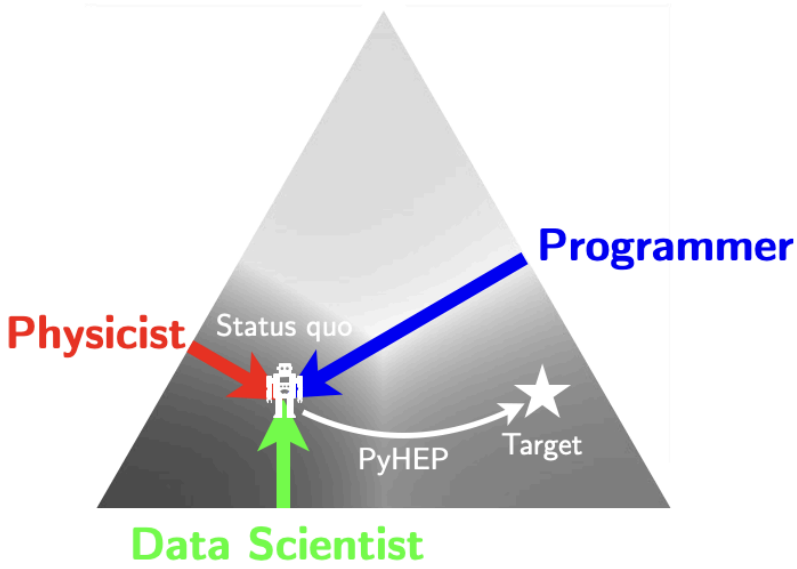
Monday 2024-08-26

Discussion: What is a HEP analysis? What does PyHEP cover?

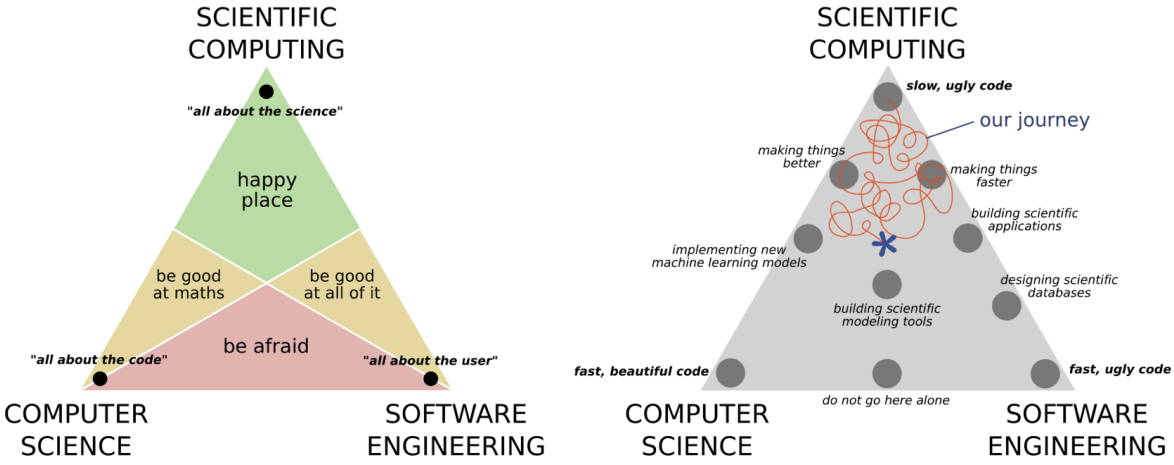
- What is the difference between data analysis and systems administration
- Yaroslav: data analysis is transforming data to such forms, which could be analyzed by humans. In other words, creating plots and tables from data, performing statistical tests, etc.
- Eduardo: Data analysis tends to be much more exploratory and iterative as well. Generally one does not know how to reach the final measurement when one starts, in terms of actual steps.
- Martin: Data analysis is an art of getting physics from the data and we should be thinking deeply about multidimensional analysis and data products. Don't limit the thinking to the traditional 2-dimensional visualizations of observables or parameter space, but be thinking about how we can use more information.
    - Jim: Follow up: At what level should the physicist be thinking about? Should we be responsible for thinking about compression levels? 5-8 years ago there was a lot of discussion about where there should be a separation of responsibilities and roles. (Matthew: It feels like we are also needing to be thinking about communication between people like "us" developers at PyHEP and the broader particle physics community)
- Marcel: ….[Matthew forgot the plot here a bit about where the discussion was but]
    - Oksana and Matthew: Coffea-casa: Provides a very clear deployment of best practices and standards as well as tooling and packages. Gives a basic setup that abstracts away the technical difficulties from new users, but allows for deeper inspection if interested.
    - Giordon: Need to think about how to provide a high level experience like coffea-casa as well as how these things can be broken down into the pieces.
        - Ianna: Is this documentation limited?
        - This matters for people outside the LHC environment as well. E.g. Belle II, FASER, etc.
        - Nikolai: At Belle II we've also not yet made such a shift from C++ to more Python (in the sense of pushing "pure" python analysis further up the

analysis chain). Belle II uses a very configurable C++ framework and then just Pandas dataframes is generally enough.

- Jim: Striking bit in the discussion between the high level physicist view and the low level "computer" details is the rethinking of this separation. E.g. the 200 Gbps scale really required some intensive thinking at the low level (e.g. the coordination between the compute providers and the rearrangement of hardware was required). Might want to keep the separation between interface and lower level optimization, so might want to allow the student to start from the high level (e.g. Jupyter) interface but then be able to hack and dive lower level into interfaces and extensions.
    - Jonas: Important to also point out that the "hackability" here is key but given that there needs to be builder and maintainers we need to raise the ability level, and we need to have funding agencies be aware of this. If there isn't funding for this work and positions associated with it, then the support will go away.
    - ALL: career paths for maintenance aside development of tools has never been so key to the success of the field.
    - Jim: Maintenance mode is important to consider in the software project life cycle.
    - Giordon: Most of this software at the LHC level is being written by experts or Research Software Engineers (RSE)s. Might want to consider making it clearer that users from other experiments should be contributing as well (and become developers).
- Stefan: There's an overlap between some of the software between astrophysics and particle physics
- Matthew: Scientific Python aims to bring these ecosystems together under a single umbrella
- Saransh: There are orgs that "sponsor" pydata or scientific python projects -
    - Quansight - provides dedicated software developers (maintainers) to pydata ecosystem and earns by providing consulting to industry
    - NumFOCUS - Matthew: provides services that one would have to pay for without their support, such as legal services
    - Anaconda, Quanstack



-
- Saransh: something similar from https://agilescientific.com/blog/2018/1/10/what-is-scientific-computing



Discussion: Challenges - AGC & 200Gbps

We started with a demo of AGC and dask-awkward.

“Why are you ignoring warnings?” “Because of the Coffea → Vector transition and various issues in PHYSLITE.”

Showed how Uproot and uproot.dask work. The dask-awkward array is delayed, can be visualized, and its fields and type can be investigated.

With histograms (dask-histogram).

“The histogram doesn’t have a name. Can you give it one?” Yes.

Request: static type checking when *defining* Awkward behaviors.

Live example with computing dielectron mass.

Dask graph visualization is for understanding and interpreting the Dask graph *optimization*. (Lindsey Gray has given some very good talks on this at IRIS-HEP training events, e.g. https://indico.cern.ch/event/1376945/contributions/5787148/)

Example with dask.bag to show a more interesting graph (tree reduction).

“How does uproot.dask choose partitions?” Manually: there are arguments that control it. (docs: step_size and steps_per_file.) Or Coffea’s dataset_tools preprocess.

“Difference between dask.array and dask.bag (and dask-awkward)?” They’re all different Dask high-level collections: array is NumPy-like, dask-awkward is Awkward-like, and bag is Python objects.

Looked at a big task and its profiling plots.

- https://www.polarsignals.com/blog/posts/2023/03/28/how-to-read-icicle-and-flame-graphs

“How do we interpret the profiling output?” These are the places where Uproot (1) waits for input, (2) decompresses, and (3) converts the TBasket data into short arrays. Interestingly, it does not include (4) where the short arrays are concatenated into one big array.

“How did you get the Dask network to scale up so quickly?” Kubernetes; it’s not just HTCondor. Also, Alex was first on this cluster; the second user would wait.

We all tried diagnosing the performance oddities in the plot.

Locality: this example has data sources close to the U. Chicago site where it’s computed.

“Can the Dask workers have different memory limits?” No. Benjamin says yes: Dask is very flexible and we’ve used this.

Several use-cases for heterogeneity came up. Benjamin, Matthew Feickert, and Nikolai Hartmann need to talk.

Matthew: coiled.io/Dask team should help us with these issues. (Maybe we should talk on GitHub first https://docs.dask.org/en/latest/support.html but I think making sure that these issues are at least recorded with some examples, even if just showing them that we have an issue that comes from our problems that are not common in minimal failing examples)

Tuesday 2024-08-27

Discussion: Building and evaluating likelihoods

Big topics presented and discussed: fitting packages and their interoperability via serialisation in the HS3 format, computational backends.

- RooFit: open-world
    - Jonas Rembser's standalone variation (https://github.com/guitargeek/roofit)
- RooStats: open-world, inference library
- HistFactory (ROOT)
- TRExFitter (ROOT, HistFactory + beyond, currently still ATLAS-internal, docs for ATLAS)
- pyhf
    - Language: Python
    - Models: HistFactory (closed world)
    - Maintenance: Actively maintained by
    - Used by: ATLAS, IRIS-HEP, phenomenology, Belle II, MicroBooNE, lots of future colliders (μCollider, FCC, ILC, CEPC, Snowmass, etc…)
- zfit
    - Language: Python
    - Models: open-world, custom
    - Many different minimizers
    - Used by: LHCb, Belle II, many smaller experiments

- evermore [https://github.com/pfackeldey/evermore]
    - Language: Python
    - closed world (focus: binned template)

- hepstats:
    - open-world inference library [https://github.com/scikit-hep/hepstats]
- CMS Combine [https://cms-analysis.github.io/HiggsAnalysis-CombinedLimit/]
- Tons of ATLAS frameworks (HistFitter is public, very long list of others (many internal), many ROOT, some pyhf-based)
    - xRooFit (this is being upstreamed into ROOT increasingly, more of a higher level API than a framework compared to some others)
- Goofit [https://github.com/GooFit/GooFit]
- iminuit [https://github.com/scikit-hep/iminuit], initially mostly wrapping Minuit2 but now also other minimizers + interfaces + template fit utilities. Used a lot outside HEP, Astro being a big user.
- Cabinetry [https://github.com/scikit-hep/cabinetry], model building + inference tooling for HistFactory models using pyhf as backend
- Gammapy [https://gammapy.org/]
    - Minimization via iminuit, scipy or sherpa
    - Defines likelihood on its own: Either Cash for s+b in each bin or WStat for designated signal and background regions (always binned analysis)
    - Likelihood extension via priors/NPs since one of the latest version
    - Asymptotic limits only in extra packages for dedicated analysis [https://github.com/StFroese/TITRATE]
- abcd_pyhf
    - Mason's implementation of ABCD method in pyhf (https://github.com/masonproffitt/abcd-pyhf)
- Combiner (https://gitlab.cern.ch/tdado/combiner/), model building and inference tooling for measurement combinations [RooFit/RooStats based relying on iminuit for the fitting]
- BAT.jl [https://github.com/bat/BAT.jl]
- RooUnfold [https://gitlab.cern.ch/RooUnfold/RooUnfold]
- BLUE (https://blue.hepforge.org/) - "best linear unbiased estimator" (measurement combination)
- HAverager (https://github.com/HAverager) - combination of histograms/averaging

Apart from model building, minimization is crucial for point estimates & frequentist inference. General problem: difference ***stopping criteria*** between minimizers. iminuit uses EDM, stable, reliable but expensive.

- SciPy minimizers [https://docs.scipy.org/doc/scipy/tutorial/optimize.html]
    - many minimizers
    - old Fortran code (LFortran migration)
    - interface "broken" (all in one call)
    - Improved (object based) interface proposed (PR), conclusion: new library
- NLopt [https://nlopt.readthedocs.io/en/latest/]
    - many minimizers
- iminuit [https://github.com/scikit-hep/iminuit], mostly wrapping Minuit2
    - stateful (which is maybe not ideal)
- zfit minimizers [https://zfit.readthedocs.io/en/latest/user_api/zfit.minimize.html] (example)

- wraps & standardizes SciPy, NLopt, iminuit & ipopt minimizers
- same stopping criterion

Serializing arbitrary math in JSON: https://dmg.org/pfa/
Also, XLA (https://jax.readthedocs.io/en/latest/export/index.html) and LLVM IR, STAN (https://mc-stan.org/)

What if they get too big? BSON, Awkward/Parquet,
Jim: If there is a JSON Schema for HS3 then it can be read into an Awkward Array and then serialized to Parquet. 🤔

Vincenzo: Not a fitting expert, but from the API/maintainability point of view I strongly support the creation of high-level specifications such as HS3. There is the aspect that introducing more features becomes increasingly less maintainable, and being able to write entire computation graphs in a file seems like a big challenge. If this becomes interesting for a large enough part of the community, I would advise that this is done by writing it into an easily machine-readable format such as JSON (it's what HS3 does already after all). Then probably we would end up with large files that in the computations part of the model become not human-readable. But, they would be machine-readable, so that libraries could implement simple helpers to convert back the computations in a nicely readable way for the user.

Josue: Big point: binned vs unbinned

Nikolai: PyTorch serialization only dumps numerical values, not code (like Python pickle, not dill). Maybe it´s good to keep this separation (user has to keep track of code along with the data).

Alex: let´s be pragmatic, make something people will use. Danger is lofty goals, covering everything in one format.

Yaroslav: very important, serializing code is dangerous. Pickle states that unpickling can execute arbitrary code on one's system, and one can't check that in advance. It could be extremely unsafe and bad practice to distribute serialized code.

Matthew: the win is improving the communication/interoperability between theory and experiment. This is an enabler of new physics results, not just a nice-to-have. Cross-experiment statistical combinations.

- Example, see SModelS: https://smodels.github.io/docs/ListOfAnalyses

Eduardo: Incremental improvements/pragmatism, easier to read, easier to publish, easier to share with theorists, phenomenologists doing spectroscopy, PDG, … We also have to teach people how to use these tools.

Alex: the big challenge in combinations don´t go away just by having common serialization. It´s the physics.

Aside: Matt Haberland is the sole maintainer of SciPy Stats inside of SciPy. He explicitly wants more people to engage with scipy.stats and communicate what they can and can't do, and is interested in getting feedback from how and why particle physics people do things differently.

- (Jonas) Talked at SciPy with Matt and the general gist seems that they prefer not to have *more* inside scipy but instead rather less and rather have other specialized libraries that can do this.
    - Matthew: Yes, agreed. The original comment is more about him wanting to understand what are some of the reasons why we have had to create our own ecosystem of tools when it comes to Frequentist testing systems and if there are ways that support can be incorporated into SciPy, without adding huge amounts of additional tooling.
- (Giordon) I can provide some feedback but the main issue is that scipy.stats feels like an afterthought with respect to the rest of scipy and feels like a package within a package with things being a little hard to discover (why does PyMC exist for example if scipy.stats has everything?).
- Another limitation of scipy.stats is error estimation/propagation/Hessian calculation.

Summary:

- Serialisation of Probability Models
    - HS3 (arbitrary models, inference)
    - STAN (probability models)
    - Serialisation of Computational Graph
        - PyTorch (only dumps numerical values)
        - JAX (graph)
        - Tensorflow (graph)
        - StableHLO (XLA) (code)
        - LLVM IR (code)
        - PFA (arbitrary math)
- Model Building
    - Open World
        - RooFit (+ xRooFit)
        - zfit
    - Closed World
        - pyhf
            - abcd_pyhf
            - cabinetry
        - evermore
        - *RooStats-dependence*

            - CMS Combine
            - TRExFitter
            - Combiner
            - RooUnfold
            - HistFactory
        - gammapy
        - BLUE
        - HAverager
- Minimizers
    - scipy
    - minuit
    - NLopt
    - zfit.minimize
    - pyhf.optimizer
- Inference
    - hepstats
    - RooStats
    - Goofit
    - gammapy
    - pyhf
    - cabinetry
    - spey

Discussion: Statistical model serialisation

Wednesday 2024-08-28

Plothist discussion email signup:

- Scikit-hep-admins
- plothist devs (tristan.fillinger@kek.jp, cyrraz.code@protonmail.com)
- mplhep devs (andrzej.novak@cern.ch, Jonas.Eschle@cern.ch)
- Matthew Feickert (matthew.feickert@cern.ch)
- Saransh Chopra (s.chopra@ucl.ac.uk)
- Giordon Stark (gstark@cern.ch)
- Jim Pivarski (pivarski@princeton.edu)
- Eduardo Rodrigues (eduardo.rodrigues@cern.ch)

Consensus reached to merge plothist and mplhep in one new package, while keeping the mplhep name.

Discussion: Workflows

- Luigi (b2luigi, law):
    - widely used
    - Spotify open-sourced it
- Snakemake:
    - too much like make, have to make a build directory
    - maybe governance issues
        - Pointed out by NumFOCUS in their initial affiliation application - they worked on it and are now in (June 2024), so good improvements on this
    - Can include modules provided by others [https://snakemake.readthedocs.io/en/stable/snakefiles/deployment.html][https://github.com/adonath/snakemake-workflow-fermi-lat]
    - More widely used in the sciences, bioinformatics
- CWL: too static
- AirFlow (https://airflow.apache.org/) :
    - Large overhead in setting it up
    - Targets more static and recurring workflow structures
-

Discussion: Histogramming

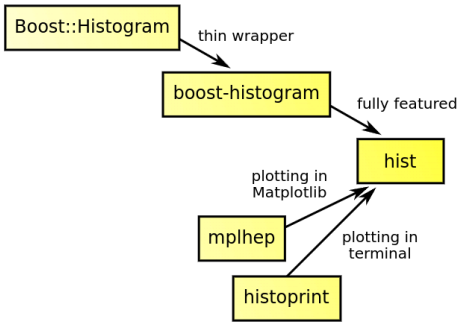
- HDF5 implementation for boost-histogram - https://github.com/aryamanjeendgar/UHI-serialization
- Serialization in JSON Scheme in UHI - https://uhi.readthedocs.io/en/latest/serialization.html

Implementations of histograms

Today, Hist/boost-histogram seem to be the most widely used, Henry presentation on ecosystem (2019)

- (dead) https://github.com/Bilokin/histogrammer
- (dead) https://github.com/cbourjau/pyhistogram
- (dead) qhist (for ROOT) - https://gitlab.com/ckhurewa/qhist
- (dead) pygram11 - https://github.com/douglasdavis/pygram11
- YODA (latest presentation at ACAT2024) cg_yoda2_acat2024.pdf
- hdrhistogram - https://github.com/HdrHistogram/HdrHistogram

- (dead) multihist - https://github.com/jelleaalbers/multihist
- (only 1D & 2D numpy-like histo) fast-histogram - https://github.com/astrofrog/fast-histogram
- (dead) paida - https://pypi.org/project/paida/
- (dead) SimpleHist - https://github.com/ndevenish/simplehistogram
- Histoprint - https://github.com/scikit-hep/histoprint
- (dead) matplotlib-hep - https://github.com/ibab/matplotlib-hep
- (similar features?) physt - https://github.com/janpipek/physt



Current ecosystem -
Also the PyHEP22 (more recent) talk - PyHEP2022 Histograms as Objects

---

Thursday 2024-08-29

Discussion: RDataframe/Coffea analyses (at scale)

Looking at the bulk of the recent years' presentations of analyses with some sort of distributed workflow, Dask has been a big hit. Also HTCondor is often mentioned. But there are other tools, such as TaskVine.

Discussion: Future of PyHEP.dev

Friday 2024-08-30

Discussion: Paper-writing