PROCEEDINGS OF SCIENCE

pyhf: pure-Python implementation of HistFactory with tensors and automatic differentiation

Matthew Feickert,[a,*] **Lukas Heinrich**[b] **and Giordon Stark**[c]

[a] *University of Wisconsin-Madison, 447 Lorch St., Madison, WI, USA*

[b] *Technical University Munich, Arcisstraße 21, 80333 München, Germany*

[c] *University of California Santa Cruz SCIPP, Santa Cruz, CA, USA*

E-mail: matthew.feickert@cern.ch, lukas.heinrich@cern.ch, giordon.holtsberg.stark@cern.ch

The HistFactory p.d.f. template is per-se independent of its implementation in ROOT and it is useful to be able to run statistical analysis outside of the ROOT, RooFit, RooStats framework. pyhf is a pure-Python implementation of that statistical model for multi-bin histogram-based analysis and its interval estimation is based on the asymptotic formulas of “Asymptotic formulae for likelihood-based tests of new physics”. pyhf supports modern computational graph libraries such as TensorFlow, PyTorch, and JAX in order to make use of features such as auto-differentiation and GPU acceleration. In addition, pyhf’s JSON serialization specification for HistFactory models has been used to publish 23 full probability models from published ATLAS collaboration analyses to HEPData.

41st International Conference on High Energy physics - ICHEP2022
6-13 July, 2022
Bologna, Italy

*Speaker

1. Introduction

Measurements in High Energy Physics (HEP) aim to determine the compatibility of observed events with theoretical predictions. The relationship between them is often formalised in a statistical *model* $f(\boldsymbol{x}|\boldsymbol{\phi})$ describing the probability of data $\boldsymbol{x}$ given model parameters $\boldsymbol{\phi}$. Given observed data, the *likelihood* $\mathcal{L}(\boldsymbol{\phi})$ then serves as the basis to test hypotheses on the parameters $\boldsymbol{\phi}$. For measurements based on binned data (*histograms*), the HistFactory [1] family of statistical models has been widely used for likelihood construction in both Standard Model (SM) measurements (e.g. Refs. [2, 3]) as well as searches for new physics (e.g. Ref. [4]) and reinterpretation studies (e.g. Ref. [5]). pyhf [6, 7] is presented as the first pure-Python implementation of the HistFactory specification. In addition to providing a Python and command line API for HistFactory model building and inspection, it leverages modern open source n-dimensional array libraries to take advantage of automatic differentiation and hardware acceleration to accelerate the statistical inference and reduce the time to analyst insight.

2. HistFactory **Formalism**

HistFactory statistical models — described in depth in Ref. [8] and Ref. [9] — center around the simultaneous measurement of disjoint binned distributions (*channels*) observed as event counts $\boldsymbol{n}$. For each channel, the overall expected event rate is the sum over a number of physics processes (*samples*). The sample rates may be subject to parametrised variations, both to express the effect of *free parameters* $\boldsymbol{\eta}$ and to account for systematic uncertainties as a function of *constrained parameters* $\boldsymbol{\chi}$, whose impact on the expected event rates from the nominal rates is limited by *constraint terms*. In a frequentist framework these constraint terms can be viewed as *auxiliary measurements* with additional global observable data $\boldsymbol{a}$, which paired with the channel data $\boldsymbol{n}$ completes the observation $\boldsymbol{x} = (\boldsymbol{n}, \boldsymbol{a})$. The full parameter set can be partitioned into free and constrained parameters $\boldsymbol{\phi} = (\boldsymbol{\eta}, \boldsymbol{\chi})$, where a subset of the free parameters are declared *parameters of interest* (POI) $\boldsymbol{\psi}$ (e.g. the *signal strength*) and all remaining parameters as *nuisance parameters* $\boldsymbol{\theta}$.

$$f(\boldsymbol{x}|\boldsymbol{\phi}) = f(\boldsymbol{x}|\overset{\text{free}}{\overset{\downarrow}{\boldsymbol{\eta}}}, \underset{\text{constrained}}{\underset{\uparrow}{\boldsymbol{\chi}}}) = f(\boldsymbol{x}|\overset{\text{parameters of interest}}{\overset{\downarrow}{\boldsymbol{\psi}}}, \underset{\text{nuisance parameters}}{\underset{\uparrow}{\boldsymbol{\theta}}}) \tag{1}$$

The overall structure of a HistFactory probability model is then a product of the analysis-specific model term describing the measurements of the channels and the analysis-independent set of constraint terms:

$$f(\boldsymbol{n}, \boldsymbol{a}\,|\,\boldsymbol{\eta}, \boldsymbol{\chi}) = \underbrace{\prod_{c\,\in\,\text{channels}}\ \prod_{b\,\in\,\text{bins}_c} \text{Pois}\left(n_{cb}\,|\,\nu_{cb}\left(\boldsymbol{\eta}, \boldsymbol{\chi}\right)\right)}_{\substack{\text{Simultaneous measurement}\\ \text{of multiple channels}}} \underbrace{\prod_{\chi\in\boldsymbol{\chi}} c_{\chi}(a_{\chi}\,|\,\chi)}_{\substack{\text{constraint terms}\\ \text{for “auxiliary measurements”}}}\quad, \tag{2}$$

where within a certain integrated luminosity one observes n_{cb} events given the expected rate of events $\nu_{cb}(\boldsymbol{\eta}, \boldsymbol{\chi})$ as a function of unconstrained parameters $\boldsymbol{\eta}$ and constrained parameters $\boldsymbol{\chi}$. The latter has corresponding one-dimensional constraint terms $c_{\chi}(a_{\chi}|\,\chi)$ with auxiliary data a_{χ} constraining the parameter χ. The expected event rates ν_{cb} are defined as

$$\nu_{cb}(\boldsymbol{\phi}) = \sum_{s \in \text{samples}} \nu_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi}) = \sum_{s \in \text{samples}} \underbrace{\left(\prod_{\kappa \in \boldsymbol{\kappa}} \kappa_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right)}_{\text{multiplicative modifiers}} \left(\nu_{scb}^{0}(\boldsymbol{\eta}, \boldsymbol{\chi}) + \underbrace{\sum_{\Delta \in \boldsymbol{\Delta}} \Delta_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})}_{\text{additive modifiers}}\right) \quad (3)$$

from constant *nominal rate* ν_{scb}^{0} and a set of multiplicative and additive *rate modifiers* $\boldsymbol{\kappa}(\boldsymbol{\phi})$ and $\boldsymbol{\Delta}(\boldsymbol{\phi})$.

3. pyhf

Through adoption of open source n-dimensional array ("tensor" in the machine learning world) computational Python libraries, pyhf decreases the abstractions between a physicist performing an analysis and the statistical modeling without sacrificing computational speed. By taking advantage of tensor calculations and hardware acceleration, pyhf can achieve comparable or better performance than the C++ implementation of HistFactory on data from real LHC analyses in most situations. pyhf's default computational backend is built from NumPy and SciPy, and supports TensorFlow, PyTorch, and JAX as alternative backend choices. These alternative backends support hardware acceleration on GPUs, and in the case of JAX JIT compilation, as well as auto-differentiation allowing for calculating the full gradient of the likelihood function — all contributing to speeding up fits.

3.1 JSON Schema

The structure of the JSON specification of HistFactory models [8] used by pyhf closely follows the original XML-based specification [1]. The JSON specification for a HistFactory *workspace* is a primary focus of Ref. [8], but a workspace can be summarised as consisting of a set of channels (an analysis region) that include samples and possible parameterised modifiers, a set of measurements (including the POI), and observations (the observed data). Listing 1 demonstrates a simple workspace representing the measurement of a single two-bin channel with two samples: a signal sample and a background sample. The signal sample has an unconstrained normalisation factor μ, while the background sample carries an uncorrelated shape systematic. The background uncertainties for the bins are 10% and 20% respectively. Use of this JSON specification has allowed for the publication of 23 full statistical models from ATLAS analyses to HEPData at the time of writing in 2022. This has been a significant step forward in enabling reinterpretation and recasting of LHC results by the broader particle physics community [10].

3.2 Enabling Analysis Ecosystems

In addition to being used in ATLAS analyses, and in the flavor physics community [11, 12], pyhf has been used as a computational engine for reinterpretation studies by the particle physics phenomenology community [13, 14] and as the inference engine for Scikit-HEP library cabinetry [15], as well as other more analysis specific open source projects [16, 17]. The adoption of pyhf as a library for other projects to build upon has large implications for establishing standards and providing improvements across ecosystems of analysis tools. Of particular note, the Institute for Research and

```
{
  ”channels”: [
    { ”name”: ”singlechannel”,
      ”samples”: [
        { ”name”: ”signal”,
          ”data”: [5.0, 10.0],
          ”modifiers”: [ { ”name”: ”mu”, ”type”: ”normfactor”, ”data”: null} ]
        },
        { ”name”: ”background”,
          ”data”: [50.0, 60.0],
          ”modifiers”: [ {”name”: ”uncorr_bkguncrt”, ”type”: ”shapesys”, ”data”: [5.0,12.0]} ]
        }
      ]
    }
  ],
  ”observations”: [
    {”name”: ”singlechannel”, ”data”: [50, 60]}
  ],
  ”measurements”: [
    { ”name”: ”Measurement”, ”config”: {”poi”: ”mu”, ”parameters”: []} }
  ]
}
```

Listing 1: A toy example of a 2-bin single channel workspace with two samples. The signal sample has expected event rates of 5.0 and 10.0 in each bin, while the background sample has expected event rates of 50.0 and 60.0 in each bin. An experiment provided the observed event rates of 50.0 and 60.0 for the bins in that channel. The uncorrelated shape systematic on the background has 10% and 20% uncertainties in each bin, specified as absolute uncertainties on the background sample rates. A single measurement is defined which specifies μ as the POI [8].

Innovation in Software for High Energy Physics (IRIS-HEP) [18] has adopted pyhf as a core part of its Analysis Systems pipeline — a demonstrator model for modern distributed computing for experiments in the high-luminosity LHC (HL-LHC) era — which has provided rigorous testing of its interoperability with other tools. Improvements to pyhf directly impact all the areas highlighted in Figure 1. In addition to its computational abilities, pyhf is highly portable given its pure-Python nature and use of dependencies, like SciPy, that are broadly trusted in computational science and have been built for ubiquitous architectures. This allows for full pyhf runtimes to be natively used in novel environments, such as the Pyodide port of CPython to WebAssembly/Emscripten. While Pyodide is not optimal for serious computational use cases, the ability to use the full pyhf API allows for creation of statistical linting and visualization tools that use the same tooling as in production while leveraging interactivity of web native platforms enabled by Pyodide and the PyScript framework.

4. Conclusions

pyhf is the first pure-Python implementation of the HistFactory specification that leverages modern open source n-dimensional array libraries as computational backends to exploit automatic differentiation and hardware acceleration to speed up fits and reduce the time to scientific insight. It provides a Python and command line API for building, inspection, and to perform statistical inference for HistFactory models, and its JSON model serialization has enabled publication of full

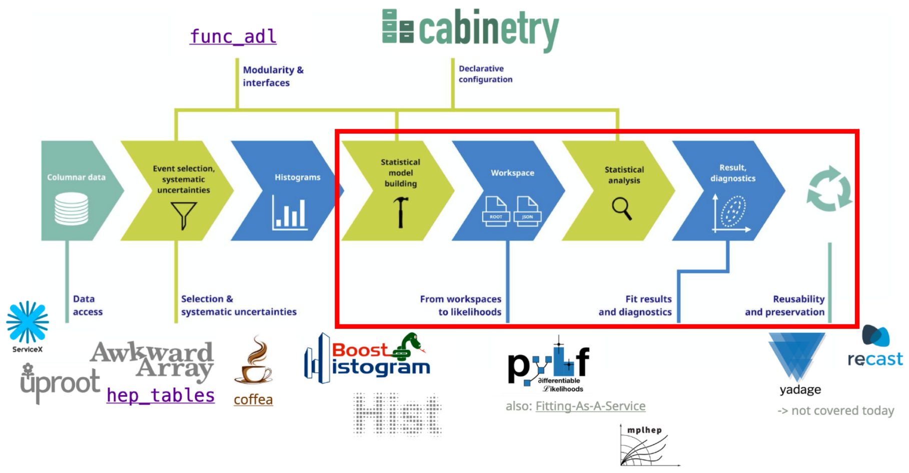
func_adl
cabinetry
Modularity & interfaces
Declarative configuration
Columnar data
Event selection, systematic uncertainties
Histograms
Statistical model building
Workspace
ROOT
JSON
Statistical analysis
Result, diagnostics
Data access
Selection & systematic uncertainties
From workspaces to likelihoods
Fit results and diagnostics
Reusability and preservation
ServiceX
Awkward Array
uproot
hep_tables
coffea
Boost histogram
Hist
pyhf
differentiable likelihoods
also: Fitting-As-A-Service
mplhep
yadage
-> not covered today
recast

Figure 1: Overview of the IRIS-HEP Analysis Systems pipeline for analysis in the HL-LHC era. The red outline indicates the areas of the pipeline in which pyhf is used either directly or as an underlying library.

statistical models from the ATLAS collaboration and improved reinterpretations. As pyhf is an open source library that has been built as part of the Scikit-HEP community project it has been readily adopted by a growing number of other libraries and tools as a computational and inference engine, allowing for improvements in the library API and computational backends to propagate to the broader user community. Growing community support and interaction, adoption across the broader particle physics community, and rigorous testing from LHC experiments and IRIS-HEP systems has demonstrated that pyhf has become a key component of the growing ecosystem of Pythonic open source scientific tools in particle physics.

Acknowledgments

Matthew Feickert's contributions to this work were supported by the U.S. National Science Foundation (NSF) Cooperative Agreement OAC-1836650 (IRIS-HEP).

References

[1] K. Cranmer, G. Lewis, L. Moneta, A. Shibata and W. Verkerke, *HistFactory: A tool for creating statistical models for use with RooFit and RooStats*, Tech. Rep. CERN-OPEN-2012-016 (Jan, 2012).

[2] ATLAS Collaboration, *Measurements of Higgs boson production and couplings in diboson final states with the ATLAS detector at the LHC*, Phys. Lett. B **726** (2013) 88.

[3] LHCb Collaboration, *Dalitz plot analysis of $B^0 \to \overline{D}^0 \pi^+\pi^-$ decays*, Phys. Rev. D **92** (2015) 032002.

[4] ATLAS Collaboration. ATLAS-CONF-2018-041, 2018.

[5] L. Heinrich, H. Schulz, J. Turner and Y.-L. Zhou, *Constraining A_4 leptonic flavour model parameters at colliders and beyond*, *JHEP* **04** (2019) 144.

[6] L. Heinrich, M. Feickert and G. Stark, "pyhf: v0.7.0." 10.5281/zenodo.1169739.

[7] L. Heinrich, M. Feickert, G. Stark and K. Cranmer, *pyhf: pure-python implementation of histfactory statistical models*, *Journal of Open Source Software* **6** (2021) 2823.

[8] ATLAS Collaboration. ATL-PHYS-PUB-2019-029, 2019.

[9] Feickert, Matthew, Heinrich, Lukas and Stark, Giordon, *Likelihood preservation and statistical reproduction of searches for new physics*, *EPJ Web Conf.* **245** (2020) 06017.

[10] K. Cranmer et al., *Publishing statistical models: Getting the most out of particle physics experiments*, *SciPost Phys.* **12** (2022) 037 [2109.04981].

[11] Belle II Collaboration, *Search for $B+\to K+\nu\nu^-$ Decays Using an Inclusive Tagging Method at Belle II*, *Phys. Rev. Lett.* **127** (2021) 181802 [2104.12624].

[12] Belle Collaboration, *Search for a dark leptophilic scalar produced in association with $\tau^+\tau^-$ pair in e^+e^- annihilation at center-of-mass energies near 10.58 GeV*, 2207.07476.

[13] G. Alguero, S. Kraml and W. Waltenberger, *A SModelS interface for pyhf likelihoods*, *Comput. Phys. Commun.* **264** (2021) 107909 [2009.01809].

[14] G. Alguero, J. Heisig, C.K. Khosa, S. Kraml, S. Kulkarni, A. Lessa et al., *New developments in SModelS*, *PoS* **TOOLS2020** (2021) 022 [2012.08192].

[15] Alexander Held, "cabinetry: v0.5.1." 10.5281/zenodo.4742752.

[16] Mason Proffitt, "abcd-pyhf: v0.0.5."

[17] Nathan Simpson and Lukas Heinrich, *neos: End-to-End-Optimised Summary Statistics for High Energy Physics*, 2203.05570.

[18] IRIS-HEP, "Institute for Research and Innovation in Software for High Energy Physics (IRIS-HEP)." https://iris-hep.org/.