pyhf Roadmap for IRIS-HEP Execution Phase

Matthew Feickert
(University of Illinois at Urbana-Champaign)

matthew.feickert@cern.ch

2020 IRIS-HEP Institute Retreat

May 27th, 2020
pyhf core dev team

Lukas Heinrich
CERN

Matthew Feickert
Illinois
IRIS-HEP

Giordon Stark
UCSC SCIPP
Goals of physics analysis at the LHC

- Search for new physics
- Make precision measurements
- Provide constraints on models through setting best limits

- All require **building statistical models** and **fitting models** to data to perform statistical inference
- Model complexity can be huge for complicated searches
- **Problem:** Time to fit can be **many hours**
- **Goal:** Empower analysts with fast fits and expressive models
Analysis Systems through the lens of `pyhf`

- Accelerating fitting (reducing time to **insight** (statistical inference)!)  
- Flexible schema great for open likelihood **preservation**  
  - Likelihood serves as high information-density summary of analysis  
- An enabling technology for **reinterpretation**
Accomplishments in Year 2
Full likelihoods (3) preserved on HEPData

- Background-only model JSON stored
- Signal models stored as JSON Patch files
- Together are able to fully preserve the full model (with own DOI! DOI: 10.17182/hepdata.89408.v1/r2)
- c.f. Matthew's CHEP 2019 talk, Lukas's LHCP 2020 talk
Publications using pyhf

New open release allows theorists to explore LHC data in a new way

The ATLAS collaboration releases full analysis likelihoods, a first for an LHC experiment

9 JANUARY, 2020 | By Katarina Anthony
Rapid adoption in ATLAS...

- Impressive appetite for `pyhf` in ATLAS analyses
- Much of SUSY, $H H \rightarrow 4b$ limit setting
  - Giordon: SUSY Run-2 Summaries subconvener
  - Lukas: ATLAS Modeling Group convener
- Upcoming: ATLAS Stats Forum recommendation

Thanks for making a tool super easy to use! When I got some [Jupyter] notebooks with this code up and shared with students a lot more of us started including limits in our talks. Before this was a pretty painful step!

— Nicole Hartman (SLAC), ATLAS Ph.D. Student

SUSY EWK 3L RPV analysis (ATLAS-CONF-2020-009): Exclusion curves as a function of mass and branching fraction to $Z$ bosons
• **SModelS** team has implemented a **SModelS/pyhf** interface
  - tool for interpreting simplified-model results from the LHC
  - designed to be used by theorists

• Have produced comparison for *Search for direct stau production in events with two hadronic tau leptons in \( \sqrt{s} = 13 \text{ TeV} \) pp collisions with the ATLAS detector (ATLAS-SUSY-2018-04) published likelihood
  - Compare simplified likelihood (**SModelS**)
  - to full likelihood (**pyhf**)

---

---

So here is one of our first reasonable validation plots. It’s preliminary, the black line is ATLAS-SUSY-2018-04 official exclusion curve. The grey line is **SModelS** using **pyhf**, running over the published data. — Wolfgang Waltenberger, CMS/SModelS
In working to add `percentile` method across all backends (as part of toys in v0.5.0) discovered discrepancy between NumPy implementation and TensorFlow Probability (TFP)
  - Through research between NumPy and TFP source code found a bug in TFP!
  - Confirmed by dev team in discussion on GitHub Issue
  - Agreed with dev team I would write a PR, which was reviewed and merged in timely manner

Along with Henry and Jim, now have upstream contributions to open source directly originating from IRIS-HEP work

`pyhf` will need this bug fix in the next TFP release, and thousands of other projects will benefit

Bonus: Continuing goodwill development
Roadmap for Year 3 Execution
In a word: Stability

```
$ git diff setup.cfg
diff --git a/setup.cfg b/setup.cfg
index 7d082227..8b7f9b9a 100644
--- a/setup.cfg
+++ b/setup.cfg
@@ -1,6 +1,6 @@
 [metadata]
 name = pyhf
-version = 0.5.1
+version = 1.0.0

description = (partial) pure Python HistFactory implementation
long_description = file: README.rst
long_description_content_type = text/x-rst
@@ -15,7 +15,7 @@ project_urls =
    Source = https://github.com/scikit-hep/lfhv
    Tracker = https://github.com/scikit-hep/lfhv/issues
classifiers =
- Development Status :: 4 - Beta
+ Development Status :: 5 - Production/Stable
License :: OSI Approved :: Apache Software License
Intended Audience :: Science/Research
Topic :: Scientific/Engineering
```
Adoption by analyses

- Any analysis that wants to use pyhf for full Run 2 should be able to

Requirements:

- pyhf becomes mature in its feature set
  - Stat Config
  - Non-asymptotic calculators (toys in v0.5.0)
  - Norm factor expressions
- Validation across all backends against HistFactory
  - pyhf GitHub org setup to help streamline process
  - Reproduction of published analyses on HEPData
- Documented examples
  - Case studies
  - Public knowledge base (pyhf Stack Overflow)
  - Rosetta stone (and what can't be done) between ROOT HistFactory and pyhf
• Preliminary results (old) show hardware acceleration giving order of magnitude speedup for some models!

• Improvements over traditional
  - 10 hrs to 30 min; 20 min to 10 sec

• Hardware acceleration benchmarking important to find edges
Integration into Analysis Ecosystems Pipeline

- Most obvious connections:
  - **ServiceX**: direct data transform and delivery
    - Illinois team dynamic between Ben and Matthew
  - **cabinetry**: general interfacing to other tools
    - c.f. Alex’s poster from 2020 Poster Session for more details

Cabinetry is a tool to build and steer (profile likelihood) template fits with applications in high energy physics in mind. It acts as an interface to many powerful tools to make it easier for an analyzer to run their statistical inference pipeline. An incomplete list of interesting tools to interact:

- **ServiceX** for data delivery.
- **coffea** for histogram processing.
- **uproot** for reading ROOT files
- for building likelihood functions (captured in so-called workspaces in RooFit) and inference:
  - **RooFit** to model probability distributions
  - **RooStats** for statistical tools
  - **HistFactory** to implement a subset of binned template fits
  - **pyhf** for a pythonic take on HistFactory.
  - **zfit** for a pythonic take on RooFit
  - **MadMiner** for likelihood-free inference techniques
Successful Application: Years 4/5
Reducing time to insight: Fitting as a service

pyhf HistFactory model spec is pure JSON: Very natural to use a REST web API for remote fitting!

1. pyhf installed on different clusters with GPUs around the world
2. User hits a REST API with JSON pyhf workspace as a request
3. pyhf fits the workspace on the cluster on demand
4. Returns fit results over REST API to user
Growing number of analyses publishing full likelihoods to HEPData

At the moment each likelihood is the collection of many individual signal patch files

Introduce concept of "patchsets" to reduce all of this two files:
- Background only file
- Signal patchset file

Would use `hepdata-validator` to resolve all files to inline JSON

Allows for entire likelihood to be natively supported in HEPData (no more tarballs required)
Grand Challenge Integration
Analysis Systems Grand Challenge

Following up on Kyle's presentation yesterday

Grand Analysis Challenge

End-to-end analysis optimization including systematics on a realistically sized HL-LHC end-user analysis dataset + observed limit & reinterpretation afterburner

- Focus on vertical slice through tools with game-changing functionality while operating on a realistically sized analysis dataset
- ~200TB for MC samples [maybe larger; 200TB was only data (see DC2 for details)]
- Multiple analysis regions, cuts, histogramming specification, and systematic variations declared using cabinetry and func_adl specifications (as in a typical SUSY search for example)
- ServiceX to perform event selection and deliver histograms for pyhf model
- Optimize analysis by using automatic differentiation to compute d(Expected limit)/d(analysis parameters), which are back-propagated from from output of stats tool, through pyhf running in fitting service, back to ServiceX running at analysis facility, and through the event selection & histogramming code.
  - Demonstrates forward looking functionality (using autodiff technique that powers deep learning in a physics context and emerging “Differentiable Programming” paradigm) [see grad-hep, neos]
  - Replace “Graduate-student descent” with gradient descent
  - Requires multiple passes over the data (caching)
  - Requires distributed optimization / passing gradients across protocol
- Once optimized: apply analysis “data” for observed limit and reinterpret efficiently using active learning [excursion]
ServiceX to pyhf

ServiceX to perform event selection and deliver histograms for pyhf model

- Should be relatively easy to translate from ServiceX output to pyhf JSON model, but probably don't want to
- Moving the translation from pyhf to cabinetry seems like a more robust solution
- cabinetry has ability to be a powerful tool, but to pyhf translation is most interesting
  - ServiceX to cabinetry: data delivery
  - cabinetry to pyhf: constructing of likelihood
- If useful, Matthew could join contribution efforts
- Alex has pointed out this is even mostly doable now with TRExFitter
  - ServiceX feeding histograms to TRExFitter
  - Convert XML to JSON with pyhf xml2json
  - Fit with pyhf
**pyhf: Fitting as a service**

Optimize analysis by using automatic differentiation to compute 
\( d(\text{Expected limit}) / d(\text{analysis parameters}) \), which are back-propagated from output of stats tool, through **pyhf running in fitting service**, back to ServiceX running at analysis facility, and through the event selection & histogramming code.

- As already covered, fitting with **pyhf** can be scaled up on demand and run almost anywhere
  - Local machine, cluster, AWS
- **pyhf** being built on frameworks that automatically handle gradients allows for this to happen naturally
- Should get taken care of as a natural part of **pyhf** development
Summary

- **Accomplishments**
  - Published and preserved full likelihoods
  - Become hugely popular and adopted inside ATLAS
  - Establishing connections for growth with SModelS and HEPData

- **Year 3 Execution**
  - Reach stable API and v1.0.0 release
  - Provide analysis support
  - Benchmark and profile hardware acceleration benefits

- **Vision for Year 4/5**
  - Globally deployed and scalable "fitting as a service" using REST web API
  - Have native support in HEPData for analysis preservation

- **Grand Challenge**
  - Integrate with cabinetry for ServiceX translation
  - Exploit fitting as a service + gradients for differentiable AS pipeline
HistFactory Template

\[ P(n_c, x_c, a_p | \phi_p, \alpha_p, \gamma_p) = \prod_{c \in \text{channels}} \left[ \text{Pois}(n_c | \nu_c) \prod_{e=1}^{n_c} f_c(x_e | \alpha) \right] G(L_0 | \lambda, \Delta_L) \prod_{p \in S+\Gamma} f_p(a_p | \alpha_p) \]

**Use:** Multiple disjoint channels (or regions) of binned distributions with multiple samples contributing to each with additional (possibly shared) systematics between sample estimates

**Main pieces:**

- **Main Poisson p.d.f. for bins observed in all channels**
- **Constraint p.d.f. (+ data) for "auxiliary measurements"**
  - encoding systematic uncertainties (normalization, shape, etc)
References


The end.