

pyhf: a pure Python statistical fitting library with tensors and autograd

Matthew Feickert

(University of Illinois at Urbana-Champaign)

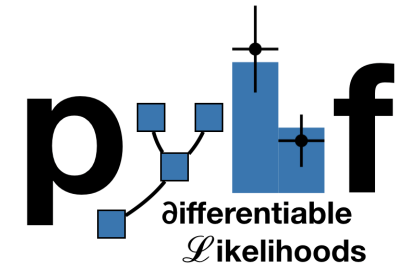


on behalf of the pyhf dev team and IRIS-HEP

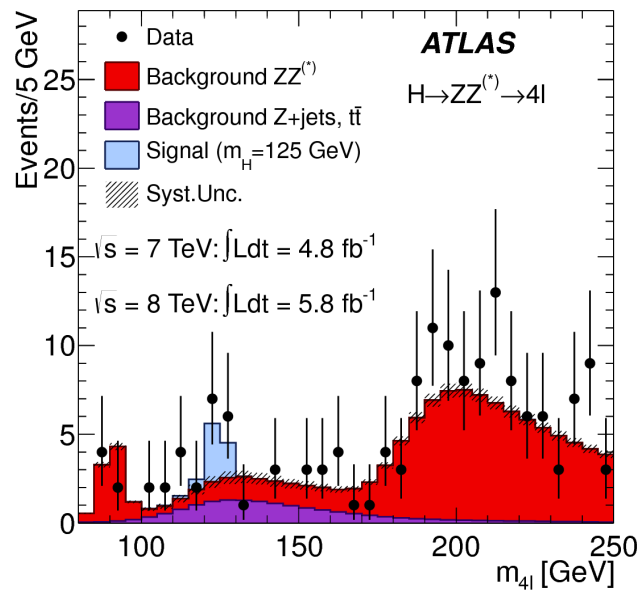
matthew.feickert@cern.ch

NSF 18-Month Review of IRIS-HEP

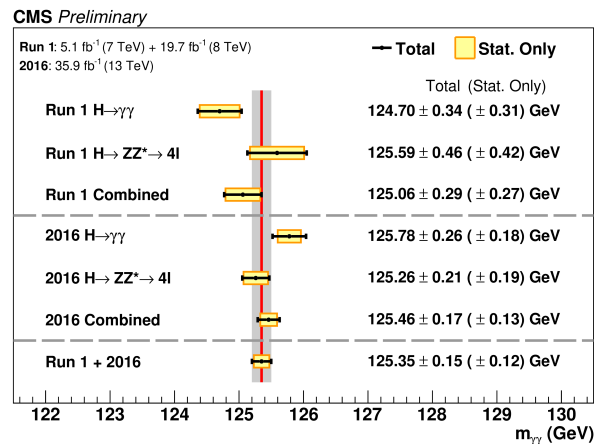
February 27th, 2020



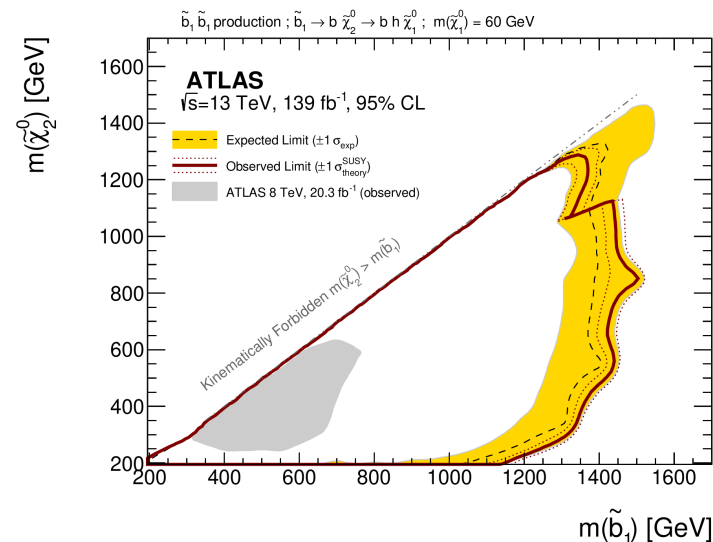
Goals of physics analysis at the LHC



Search for new physics



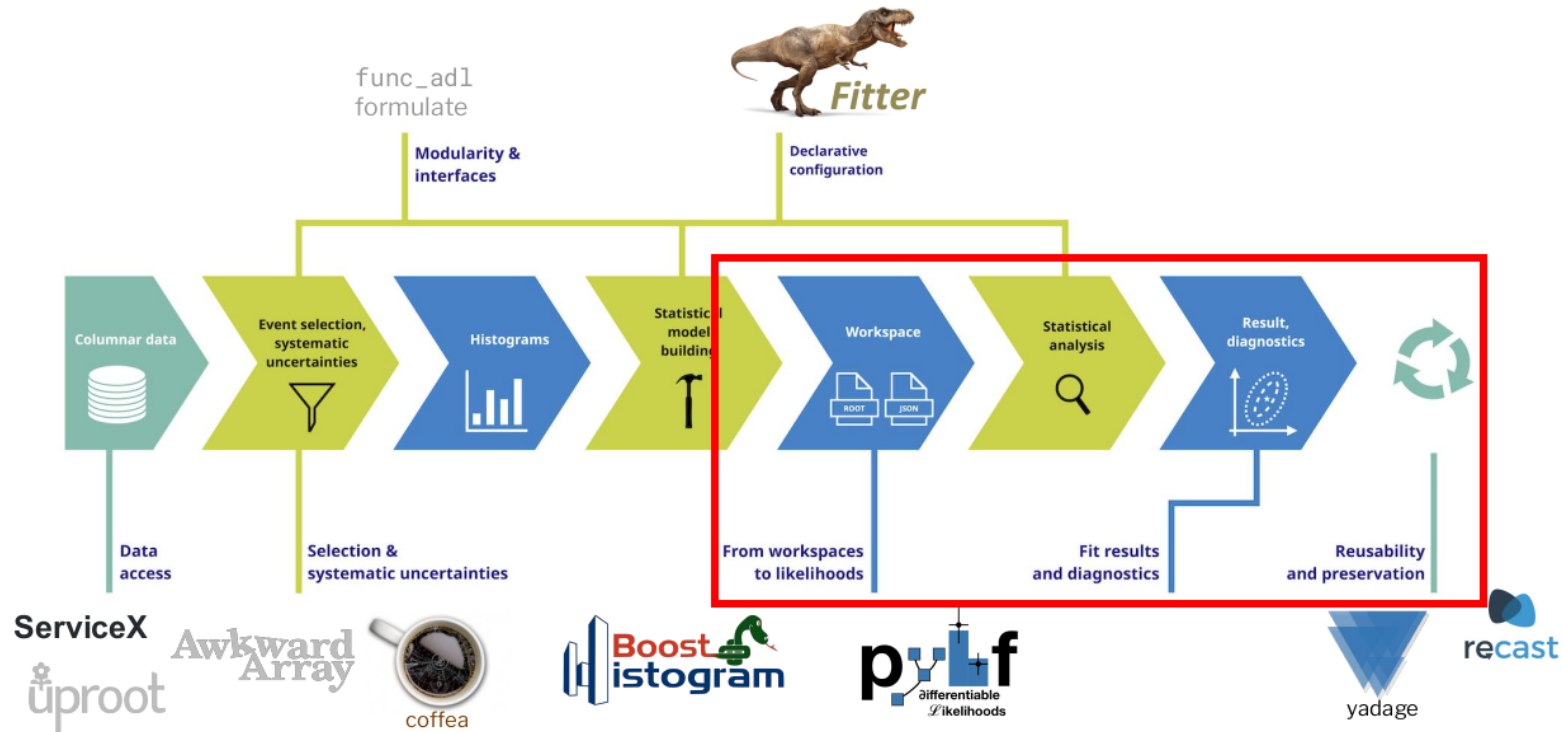
Make precision measurements



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 **literally days**
- **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**

Open source tool for all of HEP

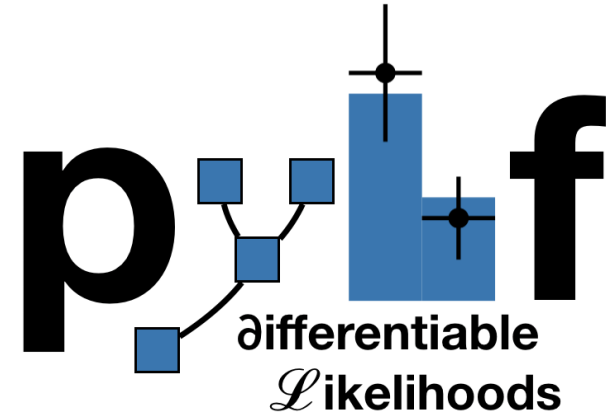
pyhf:

A pure-python implementation of the [HistFactory p.d.f. template](#) that uses [tensors](#) and [autograd](#) to **speed up physics analyses at the LHC**

- Originated from overlap of [DIANA/HEP](#) project fellowship and phenomenology paper. Now [IRIS-HEP](#) supported [Scikit-HEP](#) project.
- Used for reinterpretation in [phenomenology](#) paper, [simplified-model](#) code
- Used internally in [ATLAS](#) for SUSY large scale reinterpretation and combinations
- Working to accommodate [CMS](#) file formats
- [LHCb](#) considering adoption for binned analyses

Scikit-HEP:

A community-driven and oriented project providing Particle Physics a Pythonic ecosystem for data analysis



pyhf team



Lukas Heinrich

CERN



Matthew Feickert

Illinois



Giordon Stark

UCSC SCIPP



Kyle Cranmer

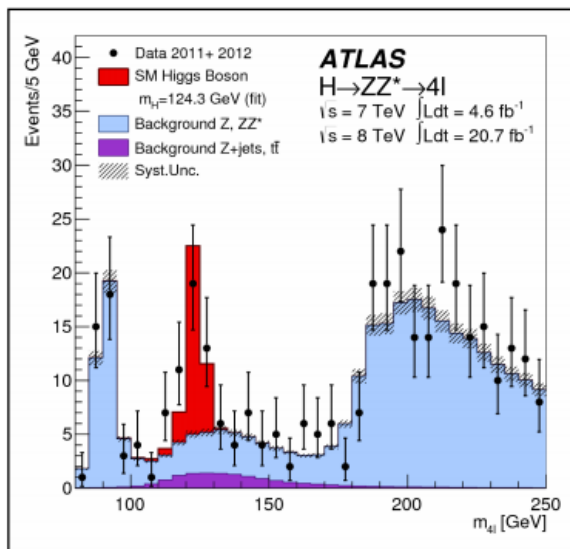
NYU

Core Developers

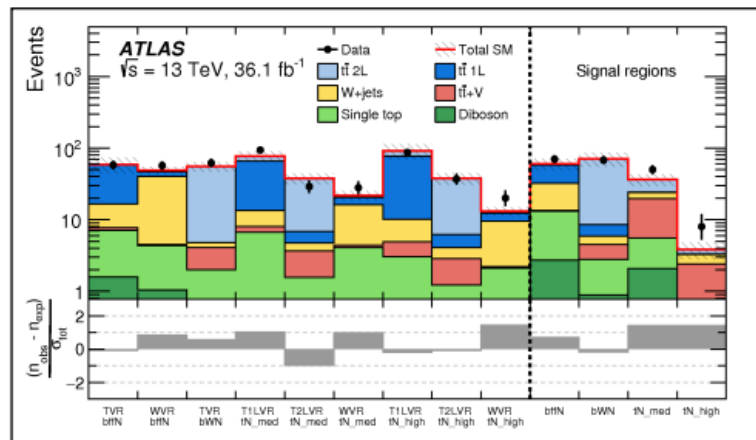
Consulting on Stats

HistFactory

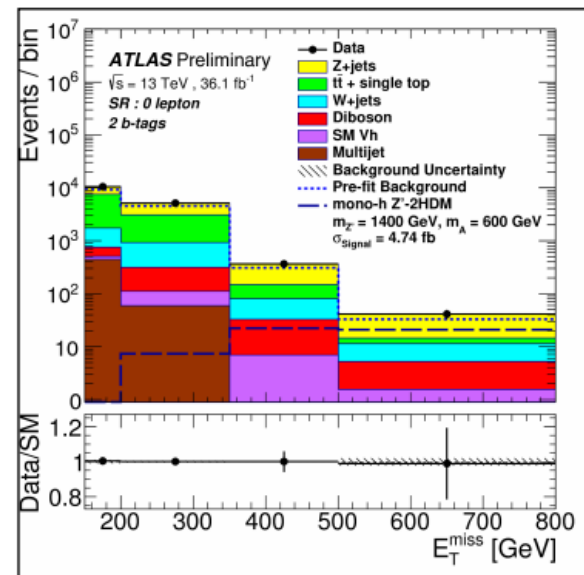
- A flexible p.d.f. template to build binned statistical models
- Developed by Cranmer, Lewis, Moneta, Shibata, and Verkerke ([CERN-OPEN-2012-016](#))
- Widely used by the HEP community for standard model measurements and BSM searches



SM



SUSY



Exotics

HistFactory Template

$$f(\vec{n}, \vec{a} | \vec{\eta}, \vec{\chi}) = \prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}(n_{cb} | \nu_{cb}(\vec{\eta}, \vec{\chi})) \prod_{\chi \in \vec{\chi}} c_{\chi}(a_{\chi} | \chi)$$

This is a **mathematical representation!** Nowhere is any software spec defined

Until now, only the traditional (C++) implementation of HistFactory

Challenges

- Preservation: Likelihood stored in the binary ROOT format
 - Challenge for long-term preservation (i.e. HEPData)
- To start using HistFactory p.d.f.s first have to learn ROOT, RooFit, RooStats
- Difficult to use for reinterpretation

pyhf: HistFactory in pure Python

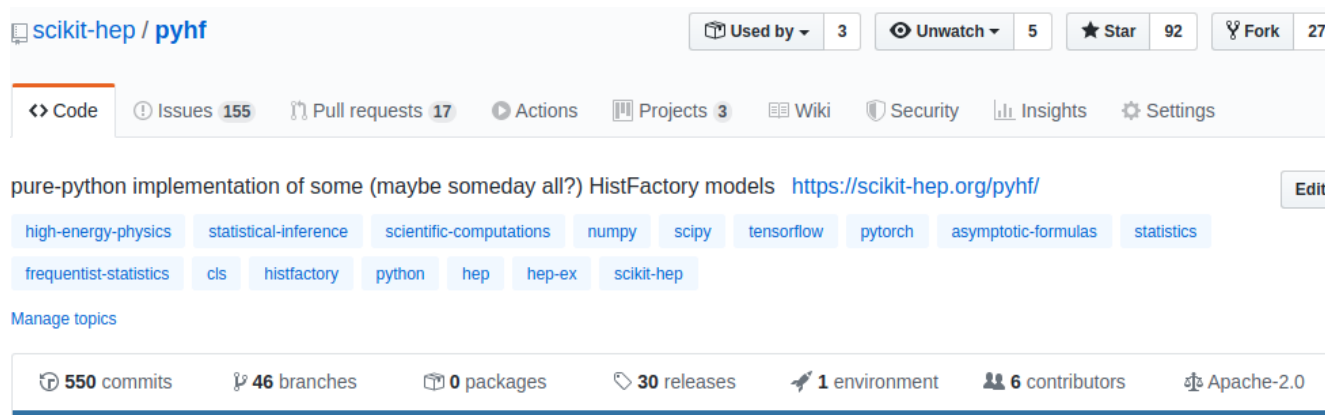
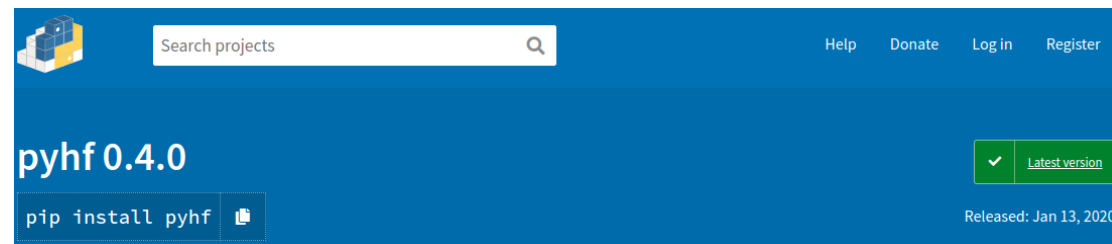
- First non-ROOT implementation of the HistFactory p.d.f. template

- DOI [10.5281/zenodo.1169739](https://doi.org/10.5281/zenodo.1169739)

- pure-Python library as second implementation of HistFactory

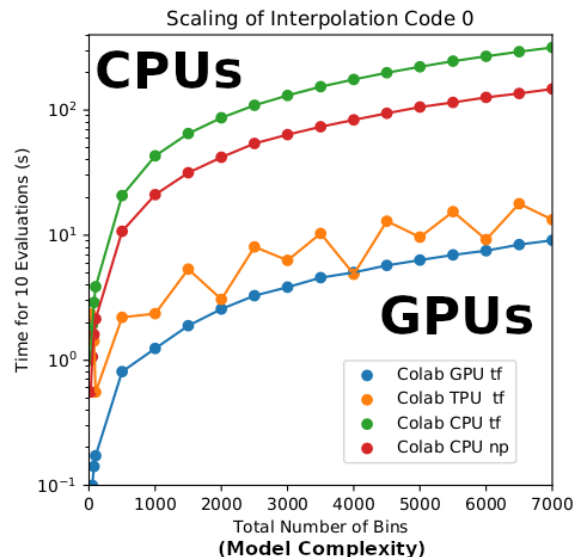
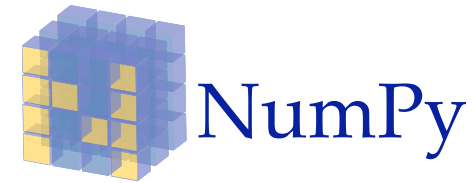
- `$ python -m pip install pyhf`
- No dependence on ROOT!

- Openly developed on GitHub



Machine Learning Frameworks for Computation

- All numerical operations implemented in **tensor backends** through an API of n -dimensional array operations
- Using deep learning frameworks as computational backends allows for **exploitation of auto differentiation (autograd) and GPU acceleration**
- As huge buy in from industry we benefit for free as these frameworks are **continually improved** by professional software engineers



- Preliminary results
- Show hardware acceleration giving **order of magnitude speedup** for some models!
- Hardware acceleration benchmarking planned
- Improvements over traditional
 - 10 hrs to 30 min; 20 min to 10 sec

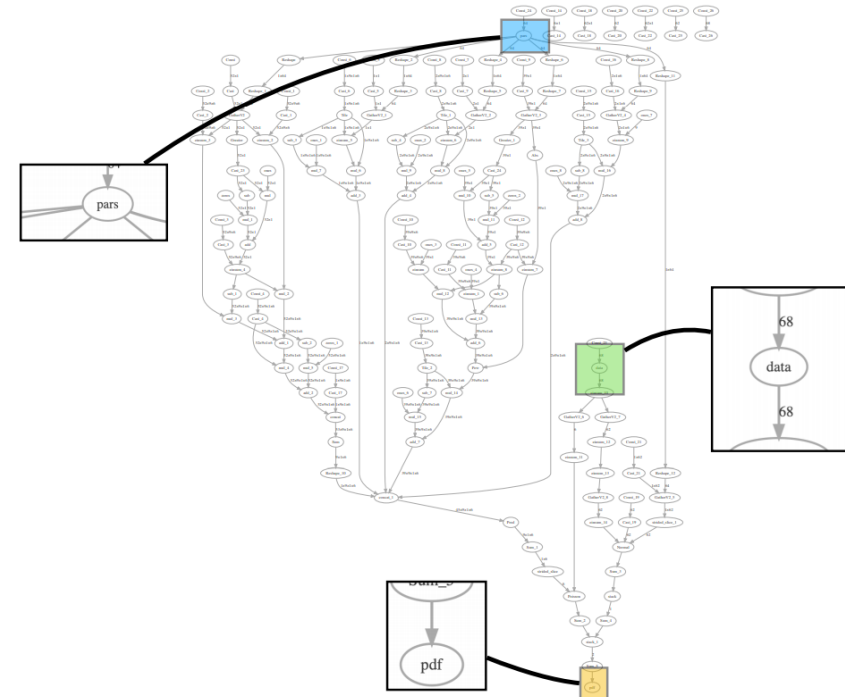
Automatic differentiation

With tensor library backends gain access to **exact (higher order) derivatives** — accuracy is only limited by floating point precision

$$\frac{\partial L}{\partial \mu}, \frac{\partial L}{\partial \theta_i}$$

Exploit **full gradient of the likelihood** with **modern optimizers** to help speedup fit!

Gain this through the frameworks creating **computational directed acyclic graphs** and then applying the chain rule (to the operations)



HistFactory likelihood

JSON spec **fully** describes the HistFactory model

- Human & machine readable
- Industry standard
 - Will be with us forever
- Parsable by every language
 - No lock into Python
- Versionable and easily preserved
 - (HEPData is JSON)
 - Highly compressible

```
{
  "channels": [ # List of regions
    { "name": "singlechannel",
      "samples": [ # List of samples in region
        { "name": "signal",
          "data": [20.0, 10.0],
          # List of rate factors and/or systematic uncertainties
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null } ]
        },
        { "name": "background",
          "data": [50.0, 63.0],
          "modifiers": [ { "name": "uncorr_bkguncrt", "type": "shapesys", "data": [5.0, 12.0] } ]
        }
      ]
    }
  ],
  "observations": [ # Observed data
    { "name": "singlechannel", "data": [55.0, 62.0] }
  ],
  "measurements": [ # Parameter of interest
    { "name": "Measurement", "config": { "poi": "mu", "parameters": [] } }
  ],
  "version": "1.0.0" # Version of spec standard
}
```

Likelihoods 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)

The screenshot shows the HEPData website interface. At the top, there is a search bar and a navigation menu. The main content area displays a search result for "bottom-squark pair production with the ATLAS detector in final states containing Higgs bosons, b -jets and missing transverse momentum". The result includes the ATLAS collaboration, authors, and a DOI link. A sidebar on the right lists "Additional Publication Resources" with a filter and a list of common resources. The main content area also features a "Resources" section with links to "External Link", "C++ File", "gz File", and "gz File".

HEPData Search HEPData

Q Browse all

Hide Publication Information

Search for bottom-squark pair production with the ATLAS detector in final states containing Higgs bosons, b -jets and missing transverse momentum

The ATLAS collaboration

Aad, Georges , Abbott, Brad , Abbott, Dale Charles , Abidinov, Ovsat , Abed Abud, Adam , Abeling, Kira , Abhayasinghe, Deshan Kavishka , Abidi, Syed Haider , Abouzeld, Ossama , Abraham, Nicola

No Journal Information, 2019

<https://doi.org/10.17182/hepdata.89408>

INSPIRE Resources

Abstract

$\tilde{b}_1 \rightarrow b + \tilde{\chi}_2^0$. Each $\tilde{\chi}_2^0$ is assumed to subsequently decay with 100% branching ratio into a Higgs boson (h) like the one in the Standard Model and the lightest neutralino: $\tilde{\chi}_2^0 \rightarrow h + \tilde{\chi}_1^0$. The $\tilde{\chi}_1^0$ is assumed to be the lightest supersymmetric particle (LSP) and is stable. Two signal mass configurations are targeted: the first has a constant LSP mass of 60 GeV; and the second has a constant mass difference between the $\tilde{\chi}_2^0$ and $\tilde{\chi}_1^0$ of 130 GeV. The final states considered contain no charged leptons, three or more b -jets, and large missing transverse momentum. No significant excess of events over the Standard Model background expectation is observed in any of the signal regions considered. Limits at the 95% confidence level are placed in the supersymmetric models considered, and bottom-squarks with mass up to 1.5 TeV are excluded.

Additional Publication Resources

filter

Common Resources 4

Missing Transverse Energy 2

Effective Mass 2

Object Based Missing Transverse Energy significance 2

MaxMin alternative algorithm average $m_{h\text{cand}}$ 2

Leading jet p_T 2

MaxMin algorithm $m_{h\text{cand}}$ 2

Efficiency_SRA_M_m60 2

Acceptance_SRC_28 2

Acceptance_SRC_26 2

Acceptance_SRC_24 2

Acceptance_SRA_M_dm130 2

Acceptance_SRB 2

Acceptance_SRA_L_dm130 2

External Link

Web page with auxiliary material

View Resource

C++ File

Truth code to compute acceptance for all signal regions using the SimpleAnalysis framework

Download

gz File

Archive of full likelihoods in the HistFactory JSON format described in ATL-PHYS-PUB-2019-029. Provided are 3 statistical models labeled RegionA, RegionB and RegionC respectively each in their own sub-directory. For each model the background-only model is found in the file named 'BkgOnly.json'. For each model a set of patches for various signal points is provided

Download

gz File

slha files for the 3 baseline signal points used in the analysis for regions A,B,C

Download

Publications using pyhf

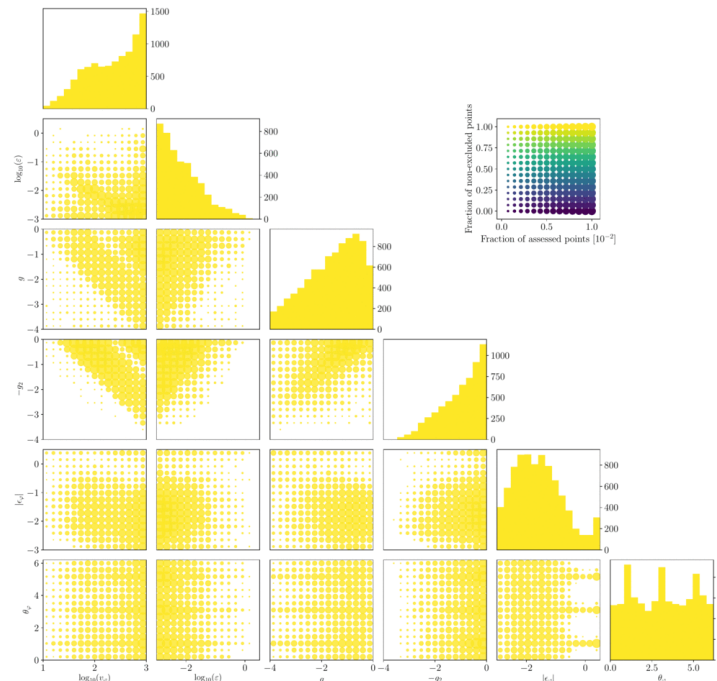
arXiv.org > hep-ph > arXiv:1810.05648

High Energy Physics - Phenomenology

Constraining A_4 Leptonic Flavour Model Parameters at Colliders and Beyond

Lukas Heinrich, Holger Schulz, Jessica Turner, Ye-Ling Zhou

(Submitted on 12 Oct 2018)



ATLAS Note	
Report number	ATL-PHYS-PUB-2019-029
Title	Reproducing searches for new physics with the ATLAS experiment through publication of full statistical likelihoods
Corporate Author(s)	The ATLAS collaboration

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



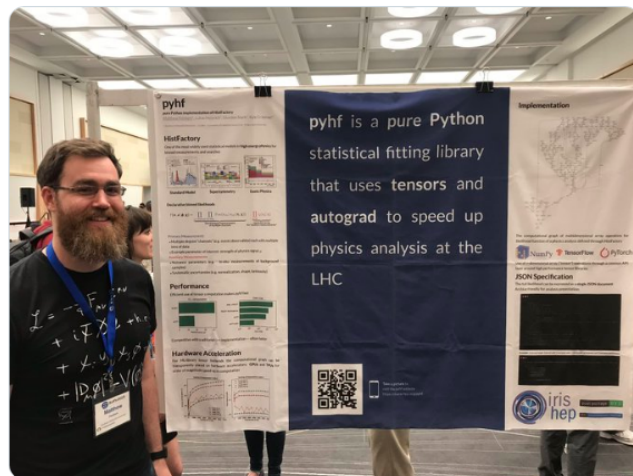
Explore ATLAS open likelihoods on the HEPData platform (Image: CERN)

Building connections to broader communities




Matthew Feickert
@HEPfeickert

Thanks to everyone at #SciPy2019 who came and asked me great questions about pyhf!



SciPy Conference 2019 (2020)




PyHEP 2020

3rd Workshop on Python in High Energy Physics

```
[1]: import particle
from hepunits.units import ctau > 1 cm
# Find all strange baryons
for x in particle.Particle:
    p, pdgid.is_baryon and x.has_strange and p.width > 0 and p.ctau > 1 cm):
        print(x.latex_name)
```

$\Sigma^- \Sigma^+ \Lambda \bar{\Lambda} \Sigma^+ \Sigma^- \Xi^- \Xi^+ \Xi^0 \Xi^0 \Omega^- \bar{\Omega}^+$

July 11–13, 2020 — Austin, Texas (USA)

Co-located with  SciPy2020

PyHEP is a series of workshops initiated and supported by the HEP Software Foundation (HSF) to discuss and promote the use of Python in the HEP community.

PyHEP 2020 will be held on the University of Texas at Austin campus, right next door to SciPy 2020, the primary conference for the scientific Python community at large. SciPy 2020 will be held on July 6–12, making it easy to attend both.

The PyHEP workshop will include

- keynote from the data science domain
- topical sessions
- hands-on tutorials
- plenty of time for discussion

ALL Python skill levels are welcome!




Organizing Committee:



Eduardo Rodrigues — University of Liverpool (Chair)
Ben Webster — University of Bristol (Co-chair)
Jim Plesniak — Princeton University (Co-chair)

Chris Turner — Rice University
Matthew Feickert — University of Illinois at Urbana-Champaign
Peter Ogilby — The University of Texas at Austin

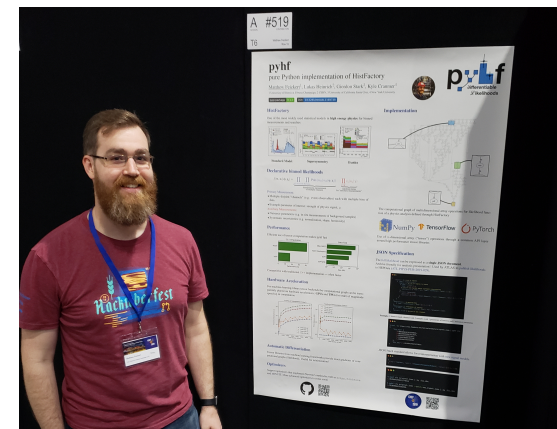
#PyHEP2020
<https://cern.ch/pyhep2020>



Sponsored by

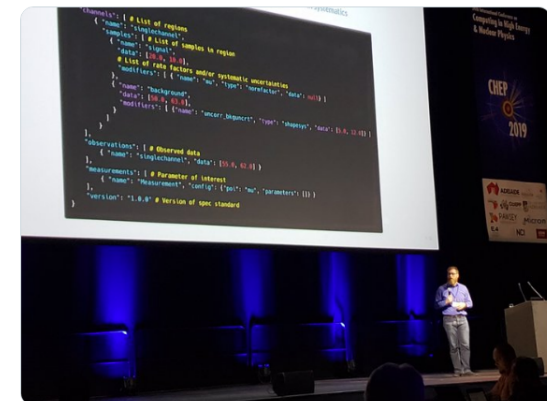


PyHEP 2019 (2020)



Gordon Watts
@SeattleGordon

@HEPfeickert tells us all about likelihoods. And how to preserve them in @HEPData. Another step towards a field standard? indico.cern.ch/event/773049/c...



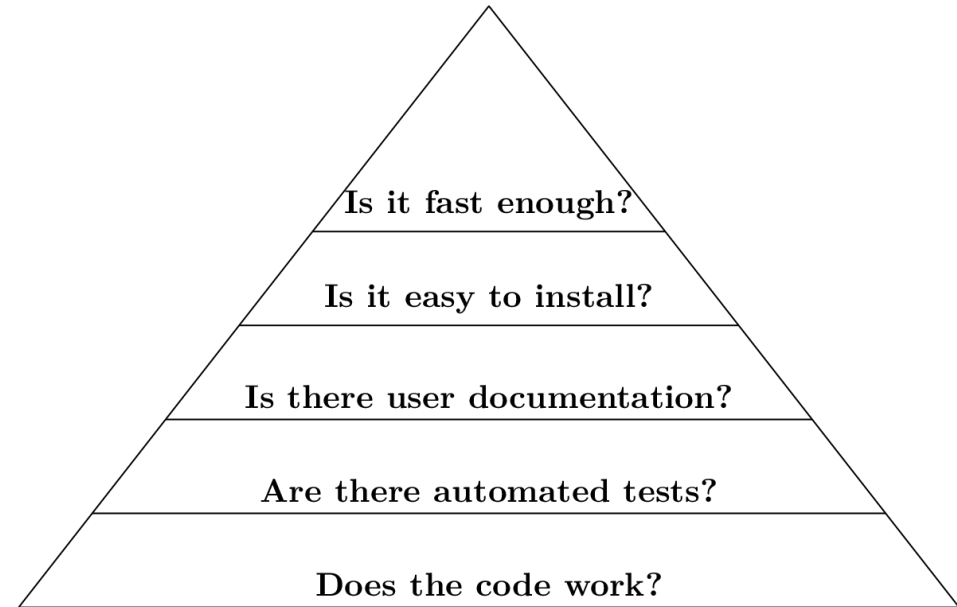
CHEP 2019

Plans and ROADMAP

- [2020 developer ROADMAP](#) designed to align with Analysis System milestones
- Better integration with Analysis Systems pipeline
- Support LHC Full Run-2 analyses using pyhf
- Mentoring HSF Google Summer of Code student to develop hardware acceleration benchmarking code



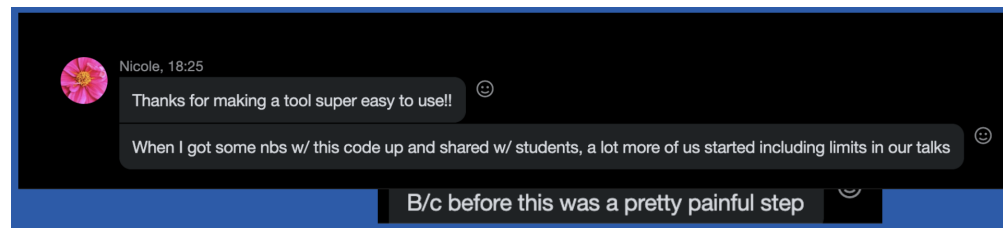
Google Summer of Code



(Seibert's Hierarchy of Needs)

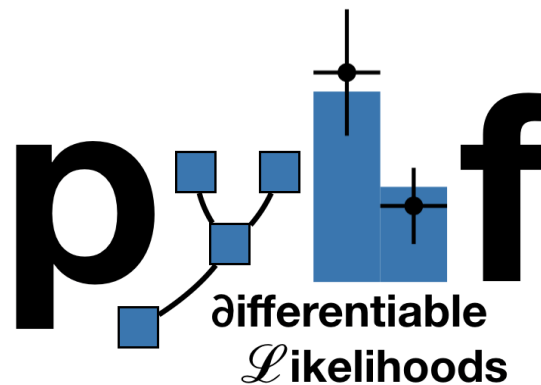
Summary

- Accelerated fitting
 - **reducing time to insight!**
 - Hardware acceleration on GPUs and vectorized operations
 - Backend agnostic acceleration
 - Human acceleration through clean Pythonic API
- Flexible schema great for open likelihood **preservation**
 - JSON: ubiquitous, universal support, versionable
 - Easily describe HistFactory models
 - First full likelihood from an LHC experiment openly published
- Enabling technology for **reinterpretation**
 - pyhf as step in RECAST
 - JSON Patch files for efficient computation of new signal models



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



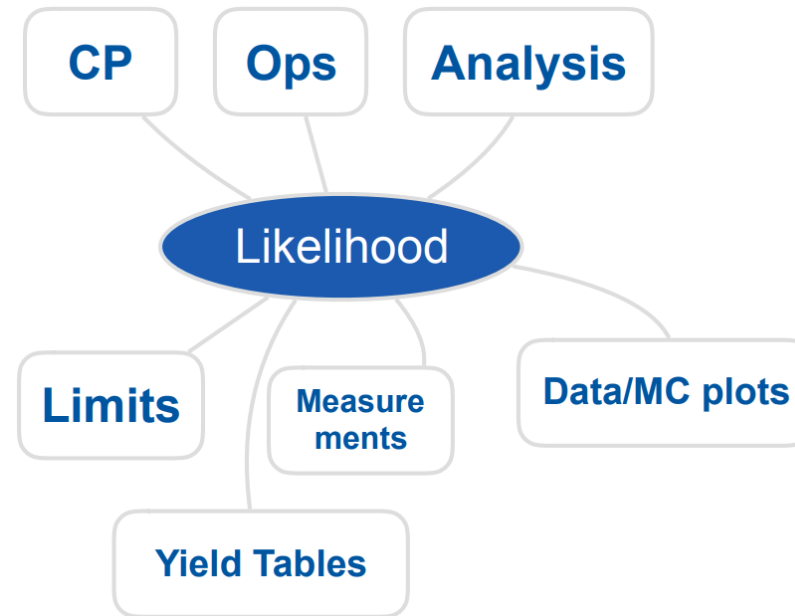
Live demo time!

Just click the button!



Why is the likelihood important?

- High information-density summary of analysis
- Almost everything we do in the analysis ultimately affects the likelihood and is encapsulated in it
 - Trigger
 - Detector
 - Systematic Uncertainties
 - Event Selection
- Unique representation of the analysis to preserve



HistFactory Template

$$f(\vec{n}, \vec{a} | \vec{\eta}, \vec{\chi}) = \prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}(n_{cb} | \nu_{cb}(\vec{\eta}, \vec{\chi})) \prod_{\chi \in \vec{\chi}} c_{\chi}(a_{\chi} | \chi)$$

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

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 simultaneous measurement of multiple channels
- Event rates ν_{cb} from nominal rate ν_{scb}^0 and rate modifiers κ and Δ
- Constraint p.d.f. (+ data) for "auxiliary measurements"
 - encoding systematic uncertainties (normalization, shape, etc)
- \vec{n} : events, \vec{a} : auxiliary data, $\vec{\eta}$: unconstrained pars, $\vec{\chi}$: constrained pars

HistFactory Template

$$f(\vec{n}, \vec{a} | \vec{\eta}, \vec{\chi}) = \prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}(n_{cb} | \nu_{cb}(\vec{\eta}, \vec{\chi})) \prod_{\chi \in \vec{\chi}} c_{\chi}(a_{\chi} | \chi)$$

This is a **mathematical representation!** Nowhere is any software spec defined

- Main Poisson p.d.f. for simultaneous measurement of multiple channels
- Constraint p.d.f. (+ data) for "auxiliary measurements"
 - encoding systematic uncertainties (normalization, shape, etc)

Until now, the only implementation of HistFactory has been in RooStats+RooFit (C++)

Challenges

- Preservation: Likelihood stored in the binary ROOT format
 - Challenge for long-term preservation (i.e. HEPData)
- To start using HistFactory p.d.f.s first have to learn ROOT, RooFit, RooStats
- Difficult to use for reinterpretation

Example pyhf JSON spec

JSON defining a single channel, two bin counting experiment with systematics

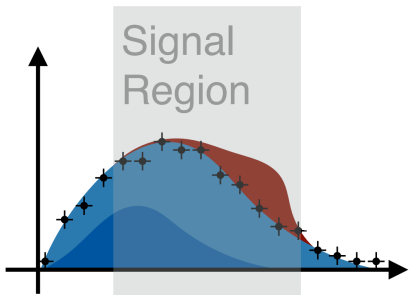
```
{
  "channels": [ # List of regions
    { "name": "singlechannel",
      "samples": [ # List of samples in region
        { "name": "signal",
          "data": [20.0, 10.0],
          # List of rate factors and/or systematic uncertainties
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null } ]
        },
        { "name": "background",
          "data": [50.0, 63.0],
          "modifiers": [ { "name": "uncorr_bkguncrt", "type": "shapesys", "data": [5.0, 12.0] } ]
        }
      ]
    }
  ],
  "observations": [ # Observed data
    { "name": "singlechannel", "data": [55.0, 62.0] }
  ],
  "measurements": [ # Parameter of interest
    { "name": "Measurement", "config": { "poi": "mu", "parameters": [] } }
  ],
  "version": "1.0.0" # Version of spec standard
}
```

JSON Patch for signal model (reinterpretation)

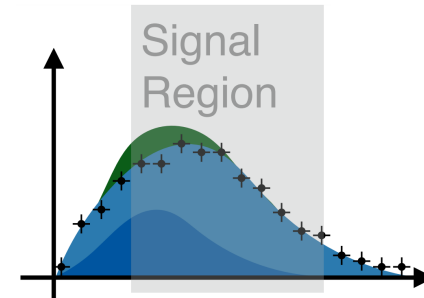
```
$ pyhf cls example.json | jq .CLs_obs
0.053994246621274014

$ cat new_signal.json
[{
  "op": "replace",
  "path": "/channels/0/samples/0/data",
  "value": [10.0, 6.0]
}]

$ pyhf cls example.json --patch new_signal.json | jq .CLs_obs
0.3536906623262466
```



Original analysis (model A)



Recast analysis (model B)

From spec to model with Python API



```
import pyhf
import json

parsed_spec = json.loads(open("example.json").read())
workspace = pyhf.Workspace(parsed_spec)

pdf = workspace.model()
observations = workspace.observations["singlechannel"]
data = observations + pdf.config.auxdata
```

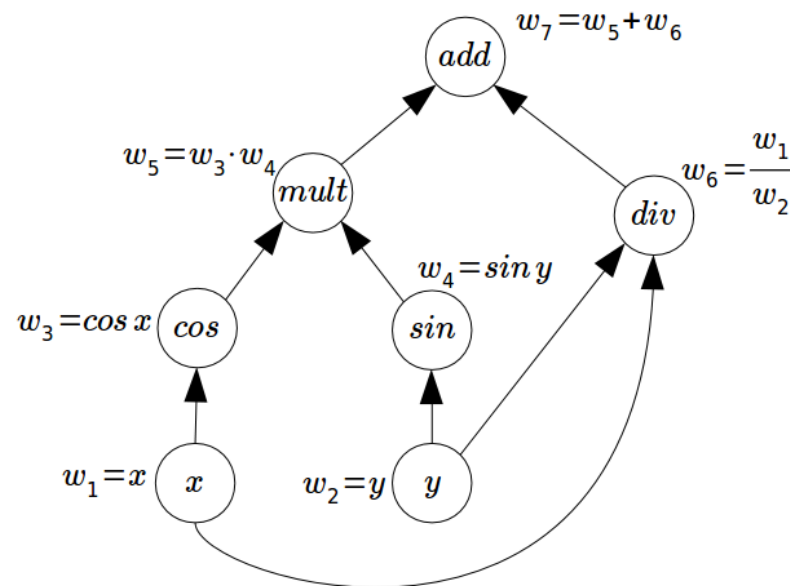
- Build workspace from spec
- From workspace:
 - Construct model
 - Get data (observations + auxiliary measurements)

Automatic differentiation – simple example

With tensor library backends gain access to [exact \(higher order\) derivatives](#) – accuracy is only limited by floating point precision

$$\frac{\partial L}{\partial \mu}, \frac{\partial L}{\partial \theta_i}$$

Gain this through the frameworks creating [computational directed acyclic graphs](#) and then applying the chain rule (to the operations)



Simple example graph

Will pyhf extend to unbinned models?

- No. This is outside the project scope
- pyhf mission goal: To deliver an intuitive and computationally efficient tool for using HistFactory in physics analyses
- [Very](#) interested in seeing Pythonic tools for unbinned fits in the wider particle physics community
- We feel that our efforts would be better spent in focusing on building underlying interface libraries for statistical modeling that offer **common APIs** for the growing number of Pythonic libraries in the ecosystem

Theory users of pyhf



WolfgangWaltenberger commented on Dec 11, 2019

+ 😊 ...

Btw, quick qn. Are you guys serious about the differentiability thing? E.g. can we use it with e.g. pytorch autograd? We would not be using this in the near future, but in the long run, this would be a killer feature!



lukasheinrich commented on Dec 11, 2019 • edited ▼

Member

+ 😊 ...

yes of course we're serious :) this is already possible and used in the autodiff optimizer
<https://github.com/scikit-hep/pyhf/blob/master/src/pyhf/optimize/autodiff.py> https://github.com/scikit-hep/pyhf/blob/master/src/pyhf/optimize/opt_pytorch.py



 **WolfgangWaltenberger** commented on Dec 11, 2019

+ 😊 ...

God I love the 21st century :))

Wolfgang

...

 2

SModelS/MadAnalysis5 developers actively integrating with pyhf

How to combine likelihoods?

- Essentially just concatenate them
- As the channels are different, then just add the lists

```
{
  "channels": [ # List of regions
    { "name": "singlechannel",
      "samples": [ # List of samples in region
        { "name": "signal",
          "data": [20.0, 10.0],
          # List of rate factors and/or systematic uncertainties
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null } ]
        },
        { "name": "background",
          "data": [50.0, 63.0],
          "modifiers": [ { "name": "uncorr_bkguncrt", "type": "shapesys", "data": [5.0, 12.0] } ]
        }
      ]
    }
  ],
  "observations": [ # Observed data
    { "name": "singlechannel", "data": [55.0, 62.0] }
  ],
  "measurements": [ # Parameter of interest
    { "name": "Measurement", "config": { "poi": "mu", "parameters": [] } }
  ],
  "version": "1.0.0" # Version of spec standard
}
```

Is there a pyhf utility to make this easy?

- `$ pyhf spec combine spec1.json spec2.json > combined.json`

JSON Patch for new signal models

```
{
  "channels": [
    { "name": "singlechannel",
      "samples": [
        { "name": "signal",
          "data": [20.0, 10.0],
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null} ]
        },
        # Rest of the model
      ]
    }
  ]
}
```

Original model

```
[{
  "op": "replace",
  "path": "/channels/0/samples/0/data",
  "value": [10.0, 6.0]
}]
```

New Signal (JSON Patch file)

```
{
  "channels": [
    { "name": "singlechannel",
      "samples": [
        { "name": "signal",
          "data": [10.0, 6.0],
          "modifiers": [ { "name": "mu", "type": "normfactor", "data": null} ]
        },
        # Rest of the model
      ]
    }
  ]
}
```

Reinterpretation

Likelihood serialization...

...making good on [19 year old agreement to publish likelihoods](#)

Massimo Corradi

It seems to me that there is a general consensus that what is really meaningful for an experiment is *likelihood*, and almost everybody would agree on the prescription that experiments should give their likelihood function for these kinds of results. Does everybody agree on this statement, to publish likelihoods?

Louis Lyons

Any disagreement ? Carried unanimously. That's actually quite an achievement for this Workshop.

([1st Workshop on Confidence Limits, CERN, 2000](#))

This hadn't been done in HEP until now

- In an "open world" of statistics this is a difficult problem to solve
- What to preserve and how? All of ROOT?
- Idea: Focus on a single more tractable binned model first

Interval estimation

(pseudo)Frequentist confidence intervals

- Use the `CLs method` to construct the interval
 - `CLs` results in `overcoverage by construction` (at low signal-background discrimination)
- pyhf offers an API & CLI for hypothesis test
 - `$ pyhf cls spec.json`
- Can `invert` the tests in order to obtain an interval with the correct coverage properties
- pyhf plan is to factor out inference to another library and focus on modeling

Bayesian credible intervals

- Currently don't support any API for this for same reason
- c.f. [H. Dembinski, PyHEP 2019](#)

```
$ pyhf cls example.json
{
  "CLs_exp": [
    0.0004090387453250841,
    0.0032606968023913925,
    0.02255257597653917,
    0.11898700005707148,
    0.39844667251932997
  ],
  "CLs_obs": 0.053994246621274014
}
```

CLI to `pyhf.utils.hypotest` returns `CLs` values

Interval estimation

(pseudo)Frequentist confidence intervals

- Use the `CLs method` to construct the interval
 - `CLs` results in **overcoverage by construction** (at low signal-background discrimination)
- pyhf offers an API & CLI for hypothesis test
 - `$ pyhf cls spec.json`
- Can **invert** the tests in order to obtain an interval with the correct coverage properties
- pyhf plan is to factor out inference to another library and focus on modeling

```
...
results = invert_interval(mu_tests, cls_obs, cls_exp)

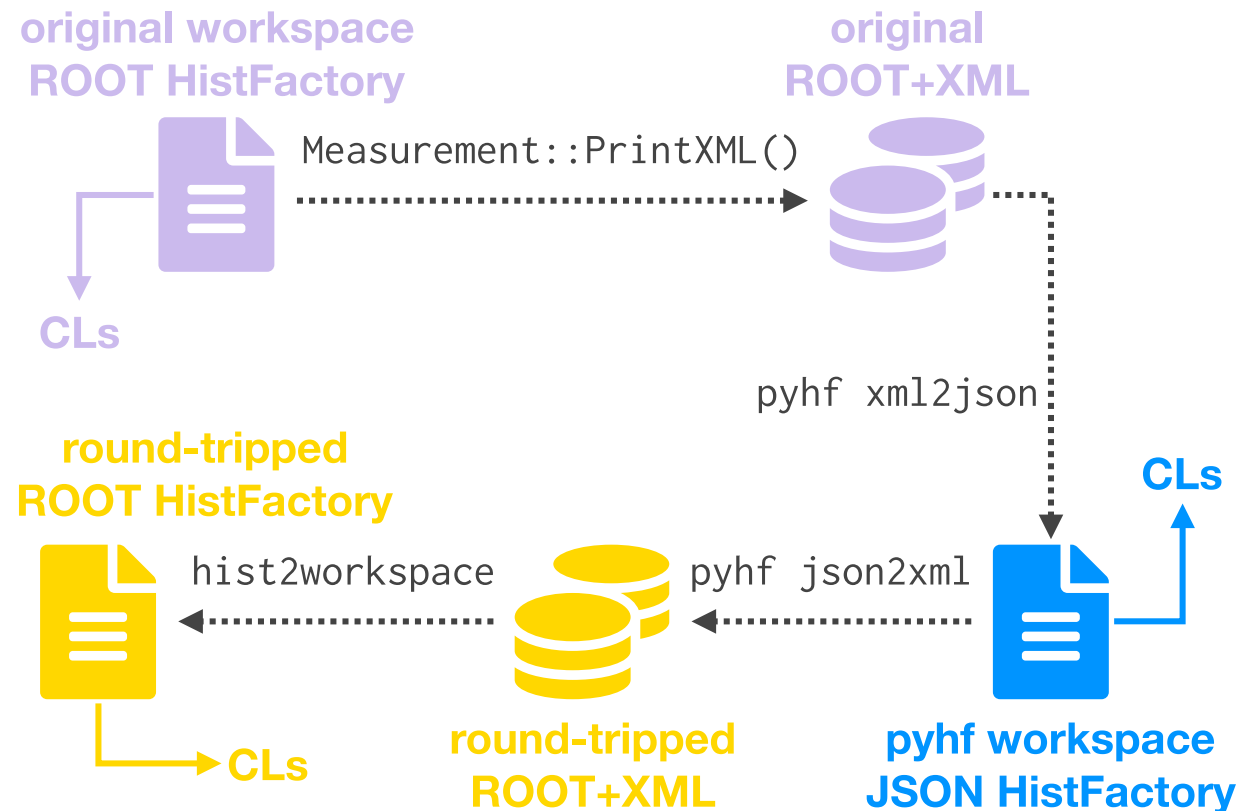
print("Observed Limit: {:.2f}".format(results["obs"]))
print("-----")
for i, n_sigma in enumerate(np.arange(-2, 3)):
    print(
        "Expected Limit{:}: {:.3f}".format(
            "" if n_sigma == 0 else "({} σ)".format(n_sigma), results["exp"][i]
        )
    )

# Observed Limit: 1.02
# -----
# Expected Limit(-2 σ): 0.451
# Expected Limit(-1 σ): 0.612
# Expected Limit: 0.856
# Expected Limit(1 σ): 1.208
# Expected Limit(2 σ): 1.651
```

From demo: invert tests to get expected (**Brazil band**) and observed **95% CL** upper limits on μ

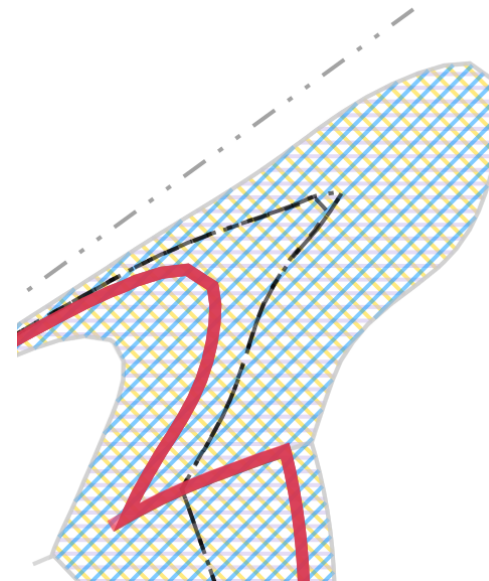
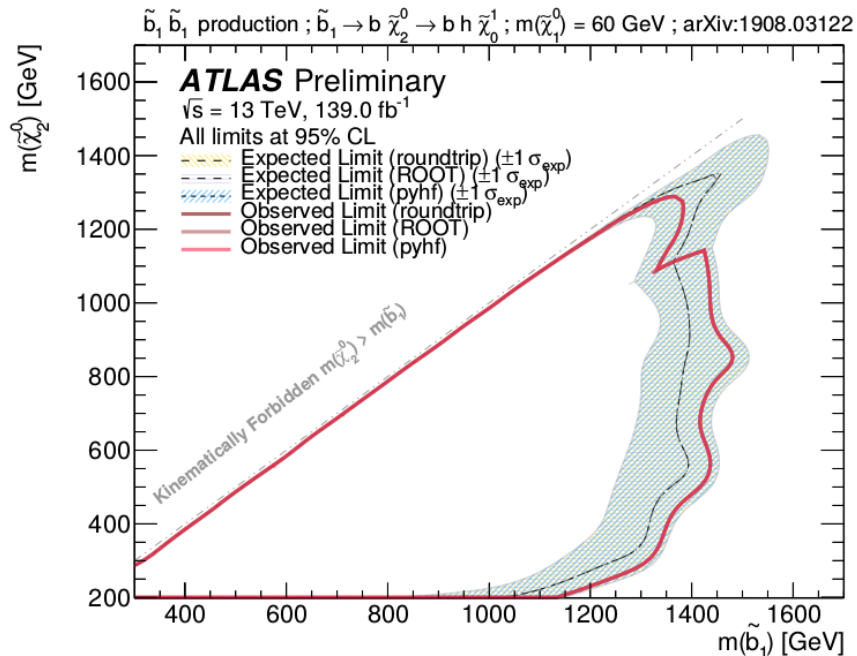
Likelihood serialization and reproduction

- ATLAS PUB note on the JSON schema for serialization and reproduction of results ([ATL-PHYS-PUB-2019-029](#))
 - Contours: ■ original ROOT+XML, ■ pyhf JSON, ■ JSON converted back to ROOT+XML



Likelihood serialization and reproduction

- ATLAS PUB note on the JSON schema for serialization and reproduction of results ([ATL-PHYS-PUB-2019-029](#))
 - Contours: ■ original ROOT+XML, ■ pyhf JSON, ■ JSON converted back to ROOT+XML
 - Overlay of expected limit contours (hatching) and observed lines nice visualization of near perfect agreement
 - Serialized likelihood and reproduced results of ATLAS Run-2 search for sbottom quarks ([CERN-EP-2019-142](#)) and published to HEPData
 - Shown to reproduce results but faster! **ROOT**: 10+ hours **pyhf**: < 30 minutes



References

1. F. James, Y. Perrin, L. Lyons, *Workshop on confidence limits: Proceedings*, 2000.
2. ROOT collaboration, K. Cranmer, G. Lewis, L. Moneta, A. Shibata and W. Verkerke, *HistFactory: A tool for creating statistical models for use with RooFit and RooStats*, 2012.
3. L. Heinrich, H. Schulz, J. Turner and Y. Zhou, *Constraining A_4 Leptonic Flavour Model Parameters at Colliders and Beyond*, 2018.
4. A. Read, *Modified frequentist analysis of search results (the CL_s method)*, 2000.
5. K. Cranmer, *CERN Latin-American School of High-Energy Physics: Statistics for Particle Physicists*, 2013.
6. ATLAS collaboration, *Search for bottom-squark pair production with the ATLAS detector in final states containing Higgs bosons, b-jets and missing transverse momentum*, 2019
7. ATLAS collaboration, *Reproducing searches for new physics with the ATLAS experiment through publication of full statistical likelihoods*, 2019
8. ATLAS collaboration, *Search for bottom-squark pair production with the ATLAS detector in final states containing Higgs bosons, b-jets and missing transverse momentum: HEPData entry*, 2019

