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

Matthew Feickert

(University of Wisconsin-Madison)

matthew.feickert@cern.ch

International Conference on High Energy Physics (ICHEP) 2022

July 8th, 2022

American Family Insurance Data Science Institute
UNIVERSITY OF WISCONSIN–MADISON

pyhf ∂ifferentiable $\mathscr{L}$ikelihoods

pyhf team

Lukas Heinrich

Technical University of Munich

Matthew Feickert

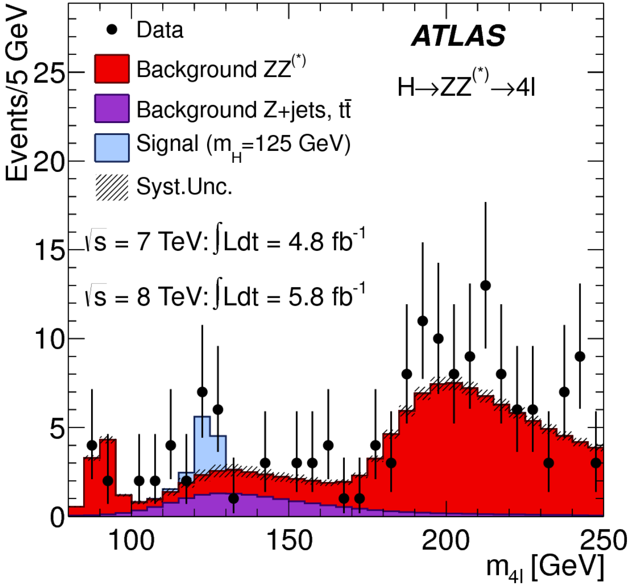
University of Wisconsin-Madison
(Illinois for work presented today)

Giordon Stark

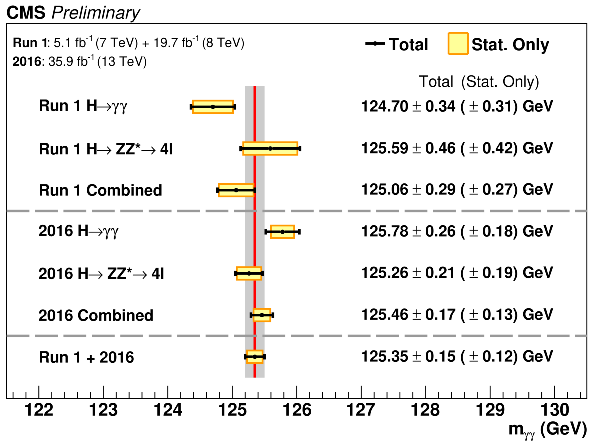
University of California Santa Cruz
SCIPP

plus more than 20 contributors

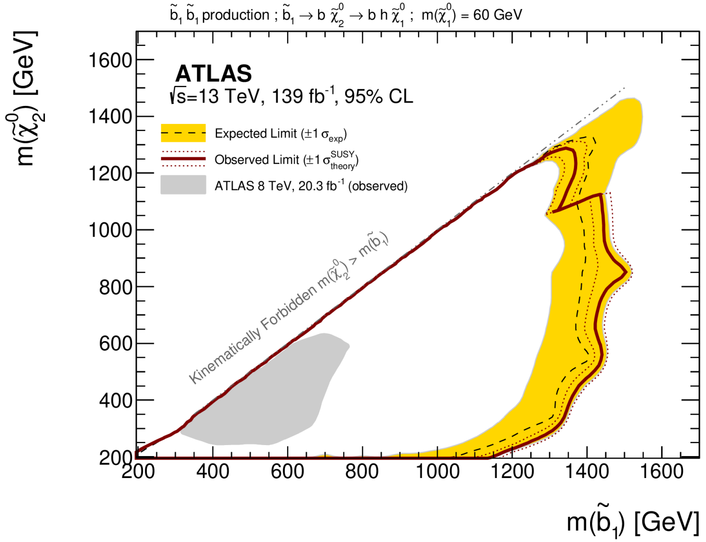
Goals of physics analysis at the LHC



Search for new physics



Make precision measurements

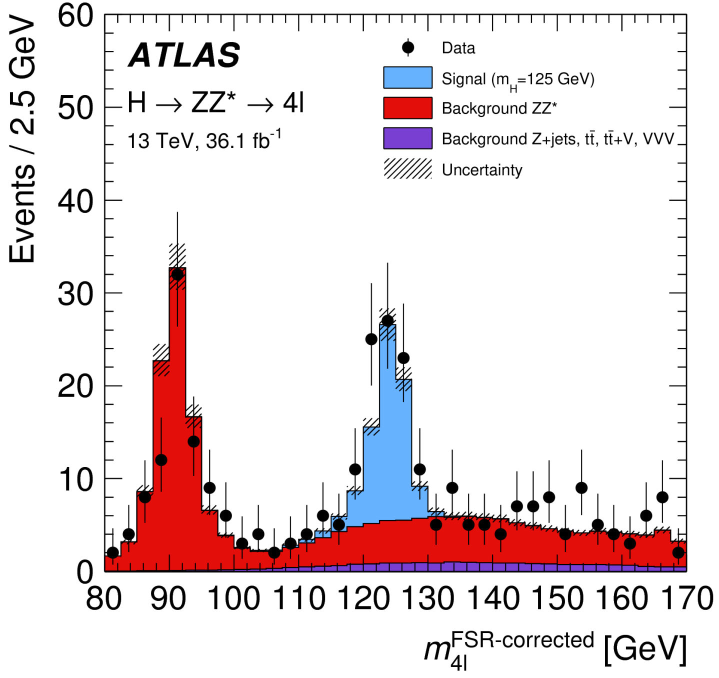


Provide constraints on models through setting best limits

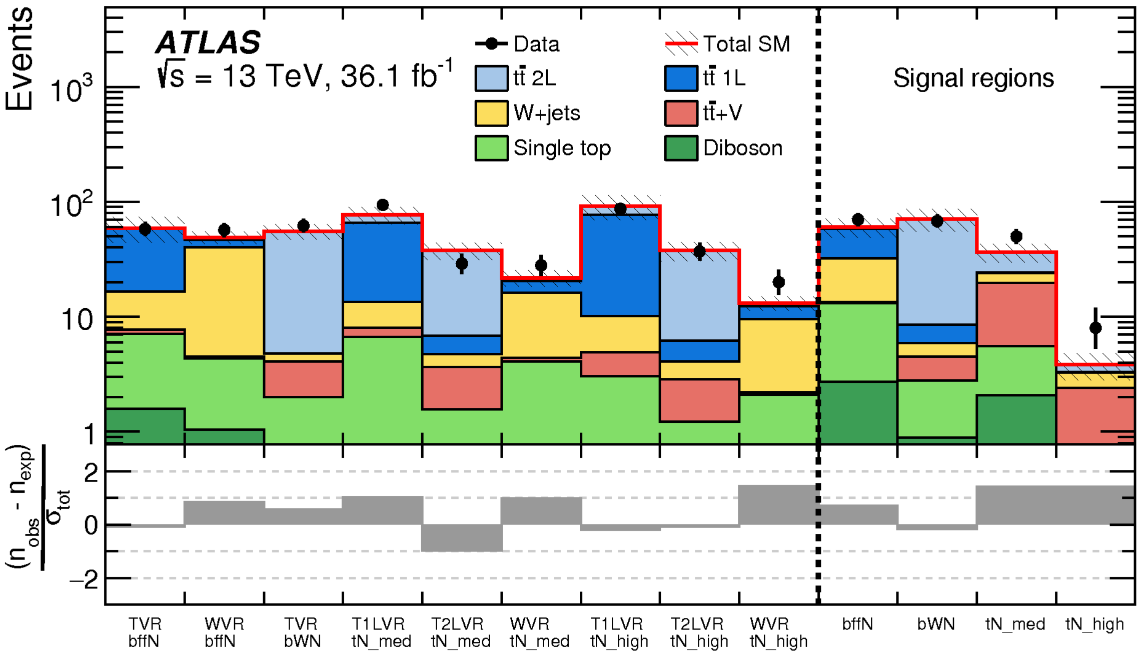
- All require **building statistical models** and **fitting models** to data to perform statistical inference
- Model complexity can be huge for complicated searches
- Problem: Time to fit can be **many hours**
- Goal: Empower analysts with fast fits and expressive models

HistFactory Model

- A flexible probability density function (p.d.f.) template to build statistical models in high energy physics
- Developed in 2011 during work that lead to the Higgs discovery [CERN-OPEN-2012-016]
- Widely used by ATLAS for **measurements of known physics** and **searches for new physics**



Standard Model



Beyond the Standard Model

HistFactory Template: at a glance

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

$\vec{n}$: events, $\vec{a}$: auxiliary data, $\vec{\eta}$: unconstrained pars, $\vec{\chi}$: constrained pars

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

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 $\nu_{cb}(\vec{\eta}, \vec{\chi})$ (nominal rate ν_{scb}^0 with rate modifiers)
  - encode systematic uncertainties (e.g. normalization, shape)
- Constraint p.d.f. (+ data) for "auxiliary measurements"

HistFactory Template: at a second glance

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

$\vec{n}$: events, $\vec{a}$: auxiliary data, $\vec{\eta}$: unconstrained pars, $\vec{\chi}$: constrained pars

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

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:

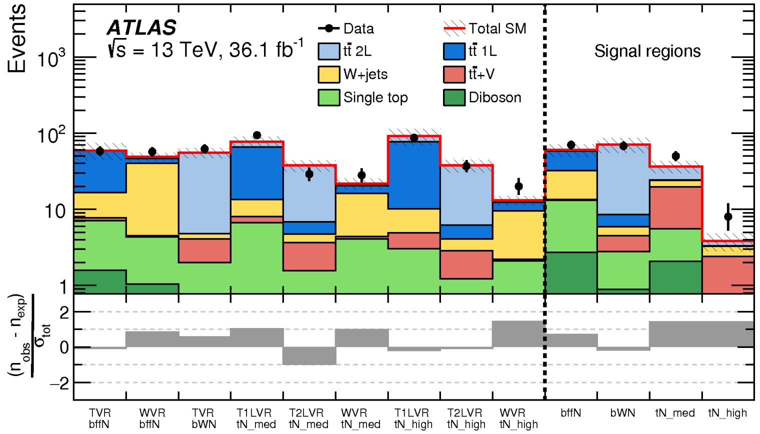
- Main Poisson p.d.f. for simultaneous measurement of multiple channels
- Event rates $\nu_{cb}(\vec{\eta}, \vec{\chi})$ (nominal rate ν_{scb}^{0} with rate modifiers)
    - encode systematic uncertainties (e.g. normalization, shape)
- Constraint p.d.f. (+ data) for "auxiliary measurements"

HistFactory Template: grammar

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

Mathematical grammar for a simultaneous fit with:

- multiple "channels" (analysis regions, (stacks of) histograms) that can have multiple bins
- with systematic uncertainties that modify the event rate $\nu_{cb}(\vec{\eta}, \vec{\chi})$
- coupled to a set of constraint terms



Example: **Each bin** is separate (1-bin) channel, each **histogram** (color) is a sample and share a **normalization systematic** uncertainty

HistFactory Template: implementation

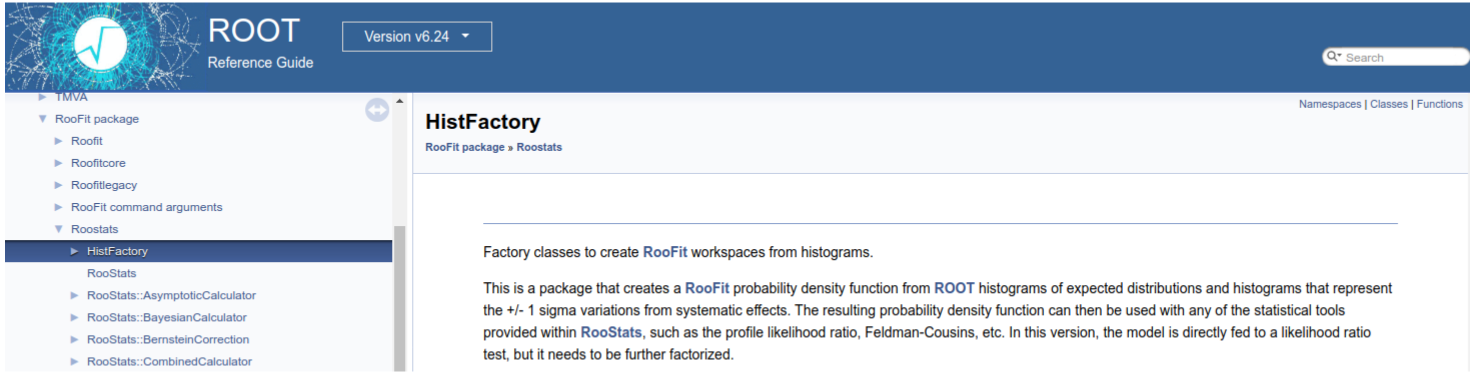
$$f\left(\text{data}|\text{parameters}\right) = f\left(\vec{n}, \vec{a}|\vec{\eta}, \vec{\chi}\right) = \prod_{c\,\in\,\text{channels}} \prod_{b\,\in\,\text{bins}_c} \text{Pois}\left(n_{cb}|\nu_{cb}\left(\vec{\eta}, \vec{\chi}\right)\right) \prod_{\chi\,\in\,\vec{\chi}} c_{\chi}\left(a_{\chi}|\chi\right)$$

$\vec{n}$: events, $\vec{a}$: auxiliary data, $\vec{\eta}$: unconstrained pars, $\vec{\chi}$: constrained pars

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

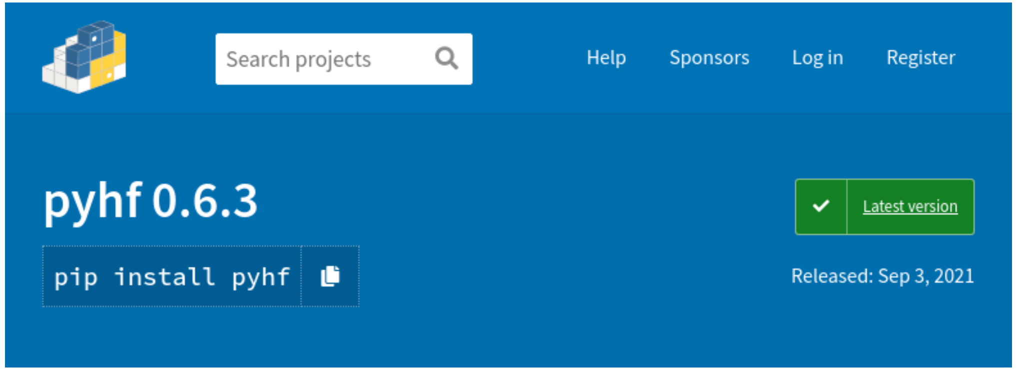
This is a mathematical representation! Nowhere is any software spec defined

Until 2018 the only implementation of HistFactory has been in `ROOT`



pyhf: HistFactory in pure Python

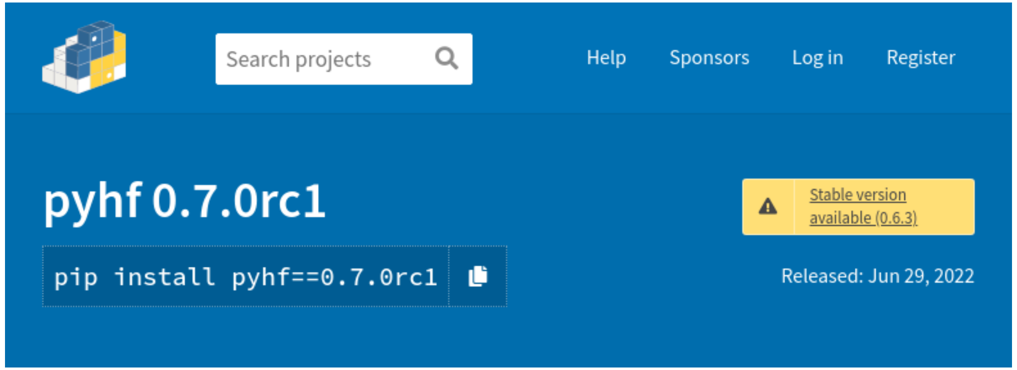
- First non-ROOT implementation of the HistFactory p.d.f. template
  - DOI 10.5281/zenodo.1169739
- pure-Python library as second implementation of HistFactory
  - `$ python -m pip install pyhf`
  - No dependence on ROOT!



- Open source tool for all of HEP
  - IRIS-HEP supported Scikit-HEP project
  - Used in ATLAS SUSY, Exotics, and Top groups in 22 published analyses (inference and published models)
  - Used by Belle II (DOI: 10.1103/PhysRevLett.127.181802)
  - Used in analyses and for reinterpretation by phenomenology community, `SModelS` (DOI: 10.1016/j.cpc.2021.107909), and `MadAnalysis 5` (arXiv:2206.14870)
  - Ongoing IRIS-HEP supported Fellow work to provide conversion support to CMS Combine as of Summer 2022!

`pyhf`: HistFactory in pure Python

- First non-ROOT implementation of the HistFactory p.d.f. template
  - DOI 10.5281/zenodo.1169739
- pure-Python library as second implementation of HistFactory
  - `$ python -m pip install --pre pyhf`
  - No dependence on ROOT!



- Open source tool for all of HEP
  - IRIS-HEP supported Scikit-HEP project
  - Used in ATLAS SUSY, Exotics, and Top groups in 22 published analyses (inference and published models)
  - Used by Belle II (DOI: 10.1103/PhysRevLett.127.181802)
  - Used in analyses and for reinterpretation by phenomenology community, `SModelS` (DOI: 10.1016/j.cpc.2021.107909), and `MadAnalysis 5` (arXiv:2206.14870)
  - Ongoing IRIS-HEP supported Fellow work to provide conversion support to CMS Combine as of Summer 2022!

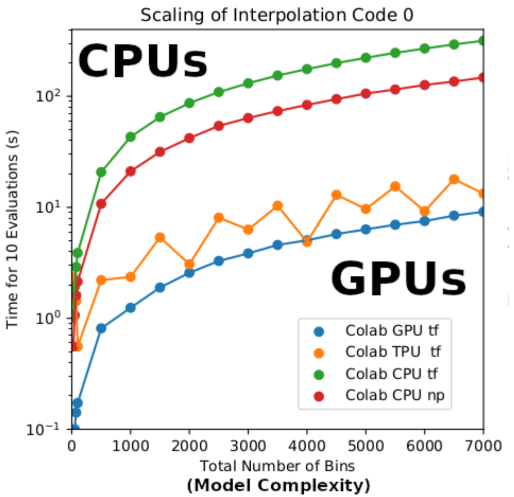
Machine Learning Frameworks for Computation

- 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 (physicists are not)

NumPy

PyTorch

TensorFlow



- Hardware acceleration giving **order of magnitude speedup** in interpolation for systematics!
  - does suffer some overhead
- Noticeable impact for large and complex models
  - hours to minutes for fits

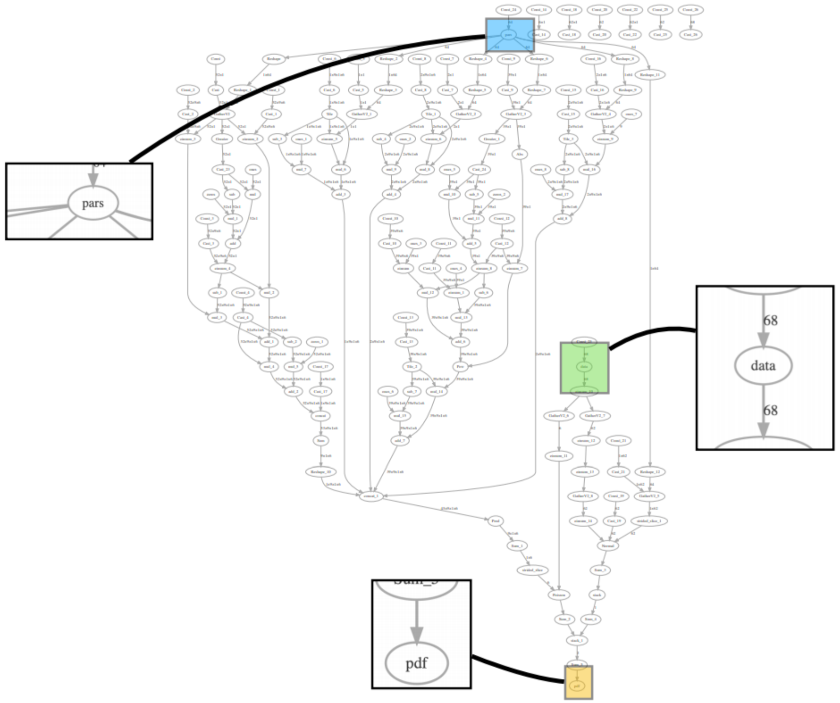
Automatic differentiation

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)



JSON spec fully describes the HistFactory model

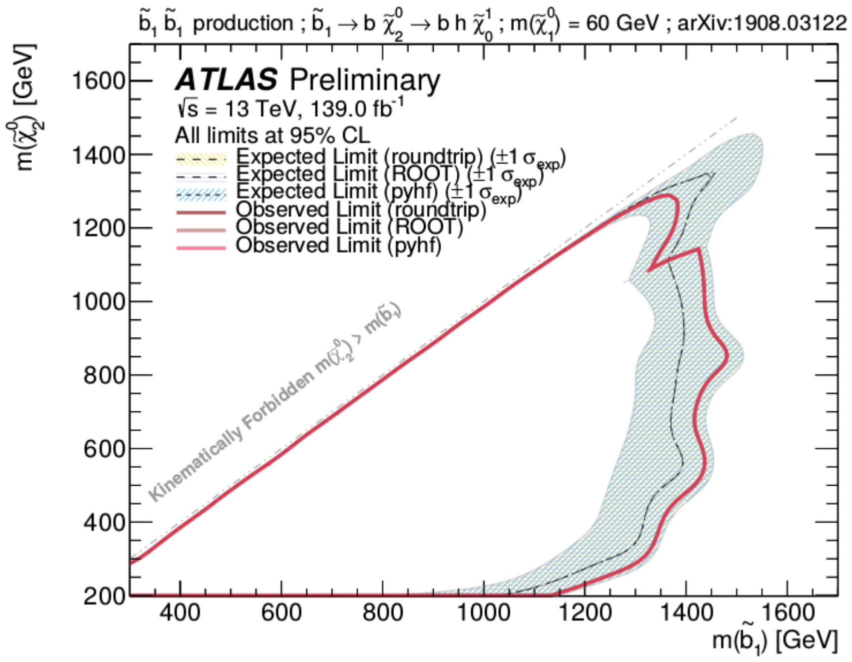
- Human & machine readable **declarative** statistical models
- Industry standard
  - Will be with us forever
- Parsable by every language
  - Highly portable
  - Bidirectional translation with ROOT
- Versionable and easily preserved
  - JSON Schema describing HistFactory specification
  - Attractive for analysis preservation
  - 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
}
```

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

ATLAS validation and publication of models

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



(ATLAS, 2019)

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)

(CERN, 2020)

Large community adoption followed (2020 on)

Charged lepton flavor violation at the EIC

Vincenzo Cirigliano, Kaori Fuyuto, Christopher Lee, Emanuele Mereghetti and Bin Yan

events in a likelihood analysis using pyhf [71–73],

E-mail: cirigliano@lanl.gov, kfuyuto@lanl.gov, clee@lanl.gov, emereghetti@lanl.gov, binyan@lanl.gov

EIC
DOI:10.1007/JHEP03(2021)256

Lepton flavor violation and dilepton tails at the LHC

Andrei Angelescu[1,a], Darius A. Faroughy[2,b], Olcyr Sumensari[3,4,c]

sonian distributions. The 95% confidence level (CL) upper limits were extracted using the CL_s method [48] with the pyhf package [49]. For High Luminosity (HL) projections,

LHC BSM DOI:10.1140/epjc/s10052-020-8210-5

Sensitivity of future hadron colliders to leptoquark pair production in the di-muon di-jets channel

value of μ at which CL_s = 0.05. We compute the CL_s values using pyhf [64], a Python implementation of HistFactory [65]. By comparison with the theoretical

FCC
DOI:10.1140/epjc/s10052-020-7722-3

How to discover QCD Instantons at the LHC

Simone Amoroso[1], Deepak Kar[2], Matthias Schott[3,a]

signal region selection are used to perform a counting experiment using the pyhf package [56]. The systematic uncer-

LHC QCD DOI:10.1140/epjc/s10052-021-09412-1

Search for $B^+ \to K^+ \nu\bar{\nu}$ Decays Using an Inclusive Tagging Method at Belle II

statistical analysis to determine the signal yields is ned with the PYHF package [43,44], which constructs

E. Bertholet, M. Bessner, S. Bettarini, F. Bianchi, T. Bilka, D. Biswas, A. Bozek, M. Bracko,

Belle-II
DOI: 10.1103/PhysRevLett.127.181802

μ-Collider DOI: 10.1007/JHEP06(2021)133

Hunting wino and higgsino dark matter at the muon collider with disappearing tracks

Rodolfo Capdevilla,[a,b] Federico Meloni,[c] Rosa Simoniello[d] and Jose Zurita[e]

The pyhf software package [94, 95] was used
e expected discovery p-value and to set limits

Search for chargino–neutralino pair production in final states with three leptons and missing transverse momentum in $\sqrt{s}$ = 13 TeV pp collisions with the ATLAS detector

the results from the signal regions of the contributing searches, which
The combination is implemented in the pyhf framework [171, 172],

The ATLAS Collaboration

ATLAS
arXiv:2106.01676

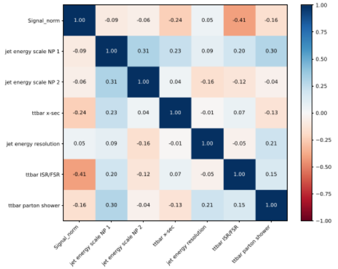
Extending and visualization: cabinetry

- **pyhf** focuses on the modeling (library not a framework)
- Leverage the design of the **Scikit-HEP ecosystem** and close communication between pyhf dev team and cabinetry lead dev Alexander Held
- **cabinetry** designs & steers template profile likelihood fits
- Uses pyhf as the inference engine
- Provides common visualization for inference validation

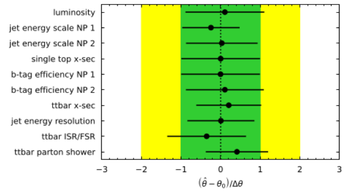
cabinetry

- Implementations for all common inference tasks exist
  - includes associated visualizations
  - results validated against TRExFitter

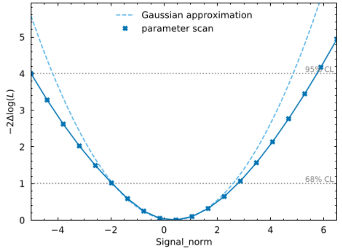
parameter correlations



nuisance parameter pulls



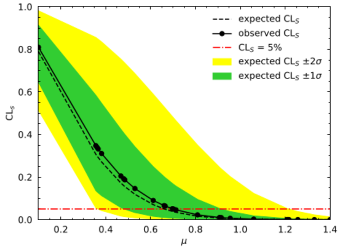
likelihood scans



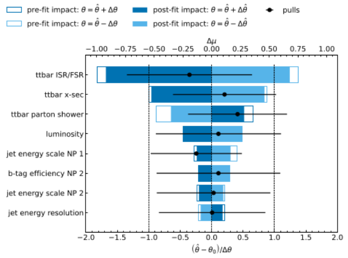
discovery significance

```
$ cabinetry significance workspaces/example_workspace.json
INFO - cabinetry.fit - calculating discovery significance
INFO - cabinetry.fit - observed p-value: 1.13053295%
INFO - cabinetry.fit - observed significance: 2.280
INFO - cabinetry.fit - expected p-value: 0.42110716%
INFO - cabinetry.fit - expected significance: 2.635
```

upper parameter limits

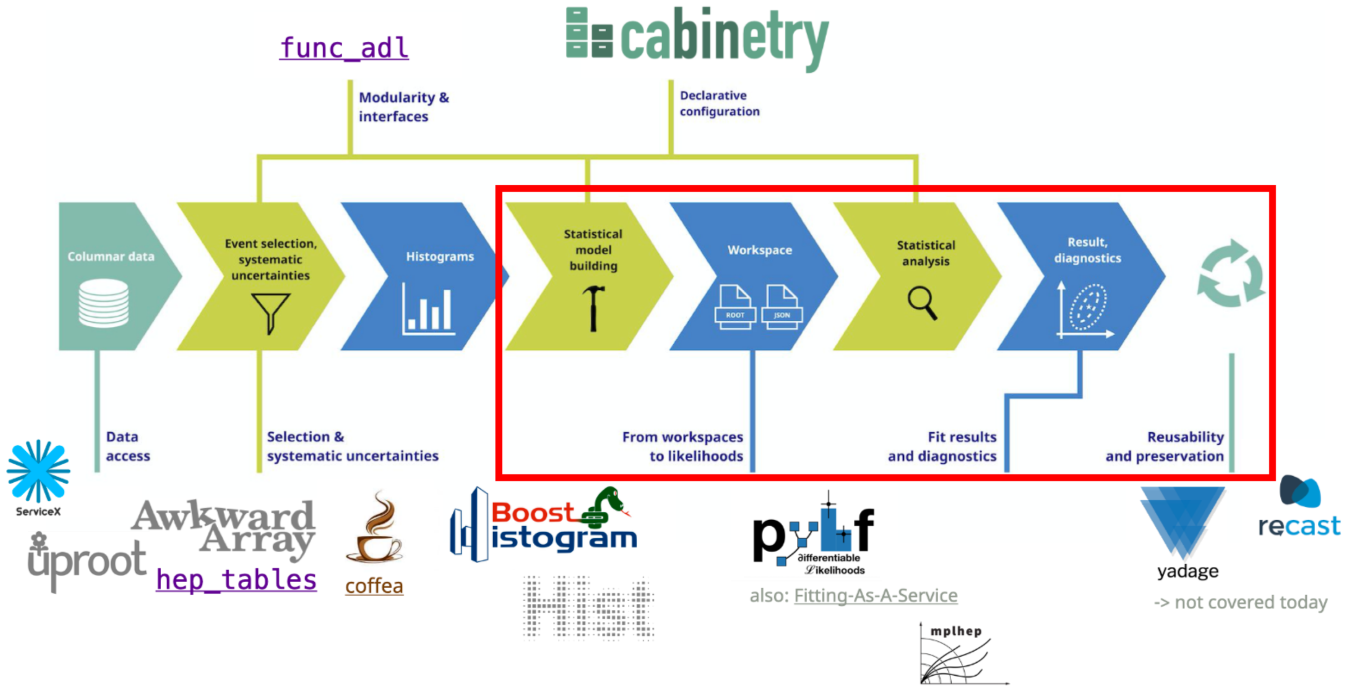


nuisance parameter impacts



Alexander Held, ATLAS SUSY Workshop 2021

Core part of IRIS-HEP Analysis Systems pipeline



- Analysis Systems pipeline: deployable stack of experiment agnostic infrastructure
  - c.f. demonstration at IRIS-HEP Analysis Grand Challenge Tools Workshop 2022
- Accelerating fitting (reducing time to **insight** (statistical inference)!) (`pyhf` + `cabinetry`)
- An enabling technology for **reinterpretation** (`pyhf` + RECAST)

Browser native ecosystem as of April 2022

```
Pyolite: A WebAssembly-powered Python kernel backed by Pyodide

[*]: import piplite
     await piplite.install(["pyhf==0.6.3", "requests"])
     %matplotlib inline
     import pyhf

[ ]:
```

Pyodide CPython port to WebAssembly/Emscripten powering JupyterLite kernel

Browser native ecosystem as of April 2022

```
Pyolite: A WebAssembly-powered Python kernel backed by Pyodide

[1]: import piplite
     await piplite.install(["pyhf==0.6.3", "requests"])
     %matplotlib inline
     import pyhf

[ ]:
```

Pyodide CPython port to WebAssembly/Emscripten powering JupyterLite kernel

Browser native ecosystem as of April 2022

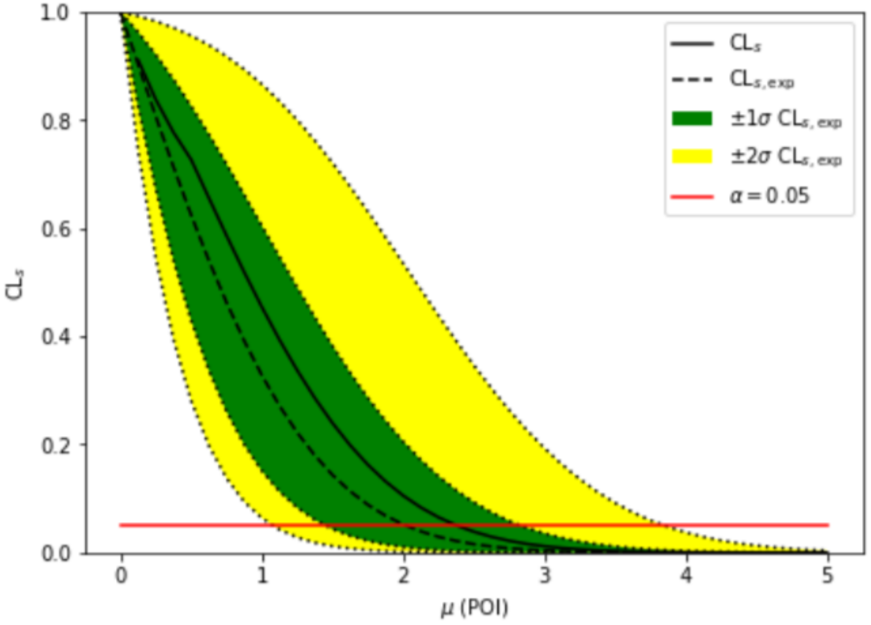
```
Pyolite: A WebAssembly-powered Python kernel backed by Pyodide

[1]: import piplite
     await piplite.install(["pyhf==0.6.3", "requests"])
     %matplotlib inline
```

```
[ ]: import pyhf
     import numpy as np
     import matplotlib.pyplot as plt
     from pyhf.contrib.viz import brazil
     pyhf.set_backend("numpy")
     model = pyhf.simplemodels.uncorrelated_background(
         signal=[10.0], bkg=[50.0], bkg_uncertainty=[7.0]
     )
     data = [55.0] + model.config.auxdata
     poi_vals = np.linspace(0, 5, 41)
     results = [
         pyhf.infer.hypotest(
             test_poi, data, model, test_stat="qtilde", return_expected_set=True
         )
         for test_poi in poi_vals
     ]
     fig, ax = plt.subplots()
     fig.set_size_inches(7, 5)
     brazil.plot_results(poi_vals, results, ax=ax);
```

Pyodide CPython port to WebAssembly/Emscripten powering JupyterLite kernel

Browser native ecosystem as of April 2022

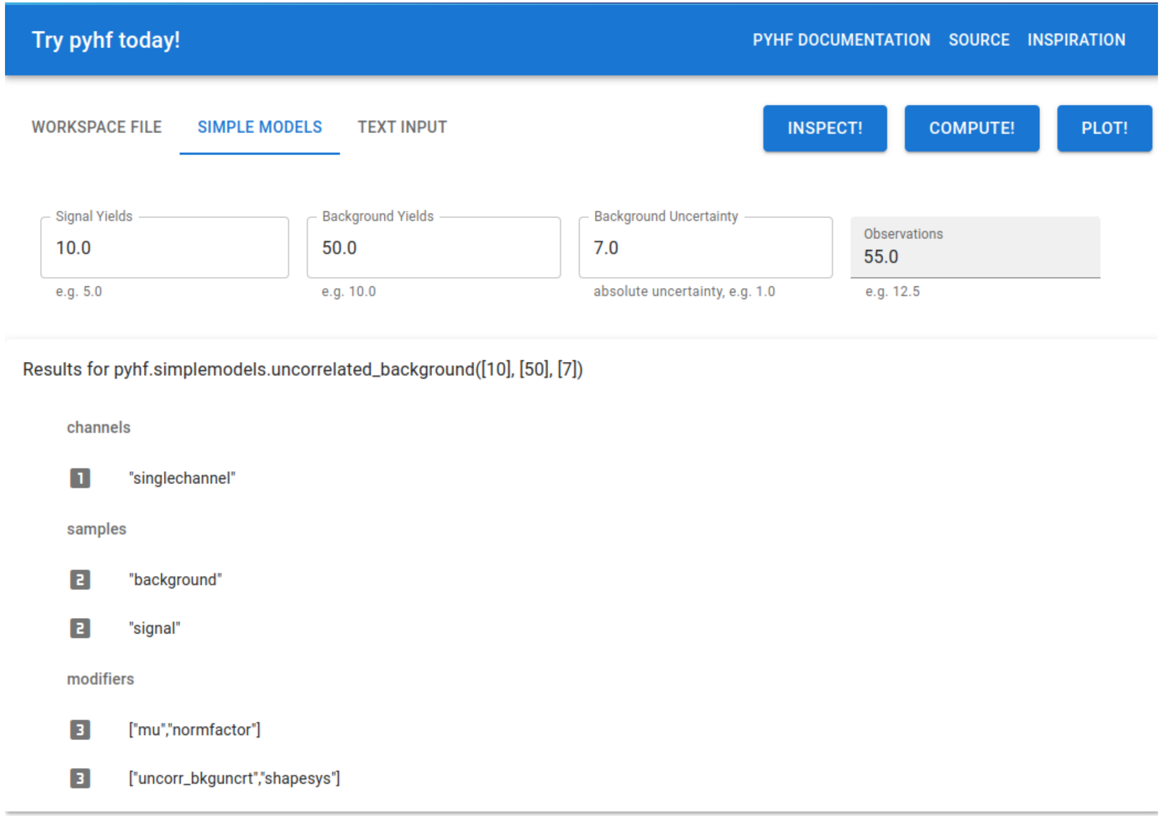
```python
]
fig, ax = plt.subplots()
fig.set_size_inches(7, 5)
brazil.plot_results(poi_vals, results, ax=ax);
```




```
[ ]:
```

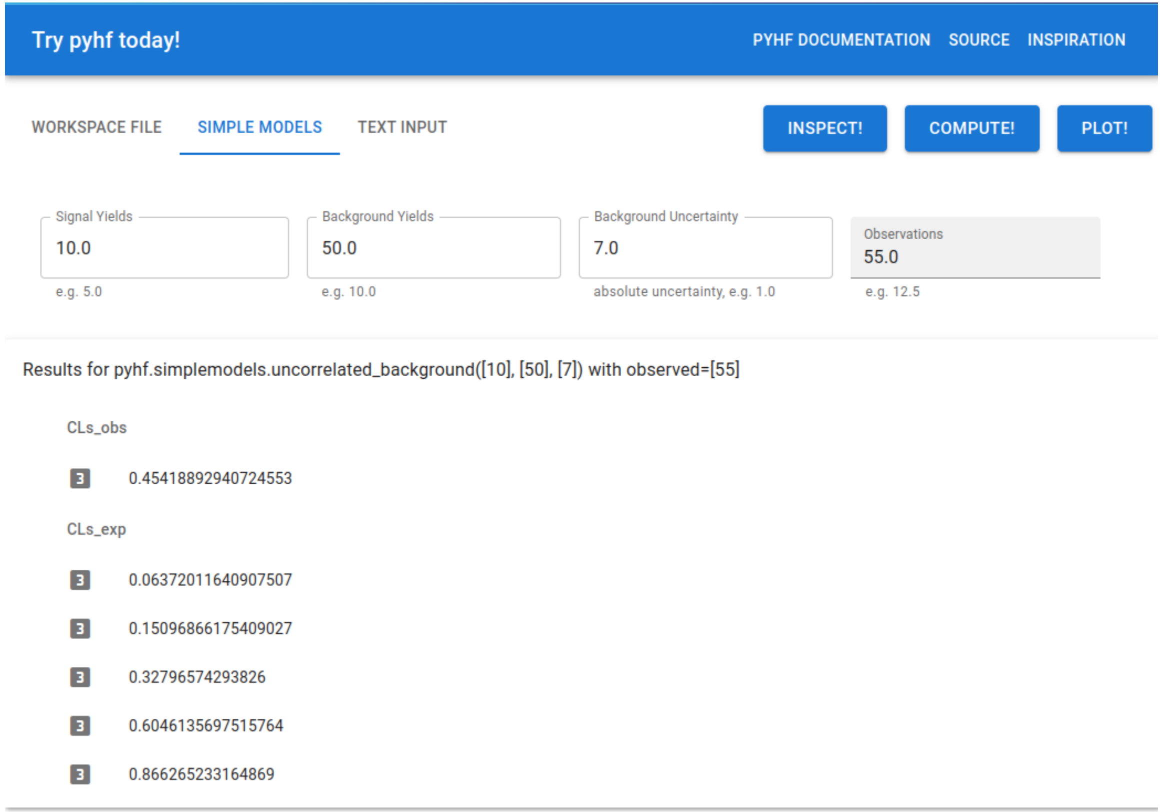
Pyodide CPython port to WebAssembly/Emscripten powering JupyterLite kernel

Enabling full web apps with PyScript



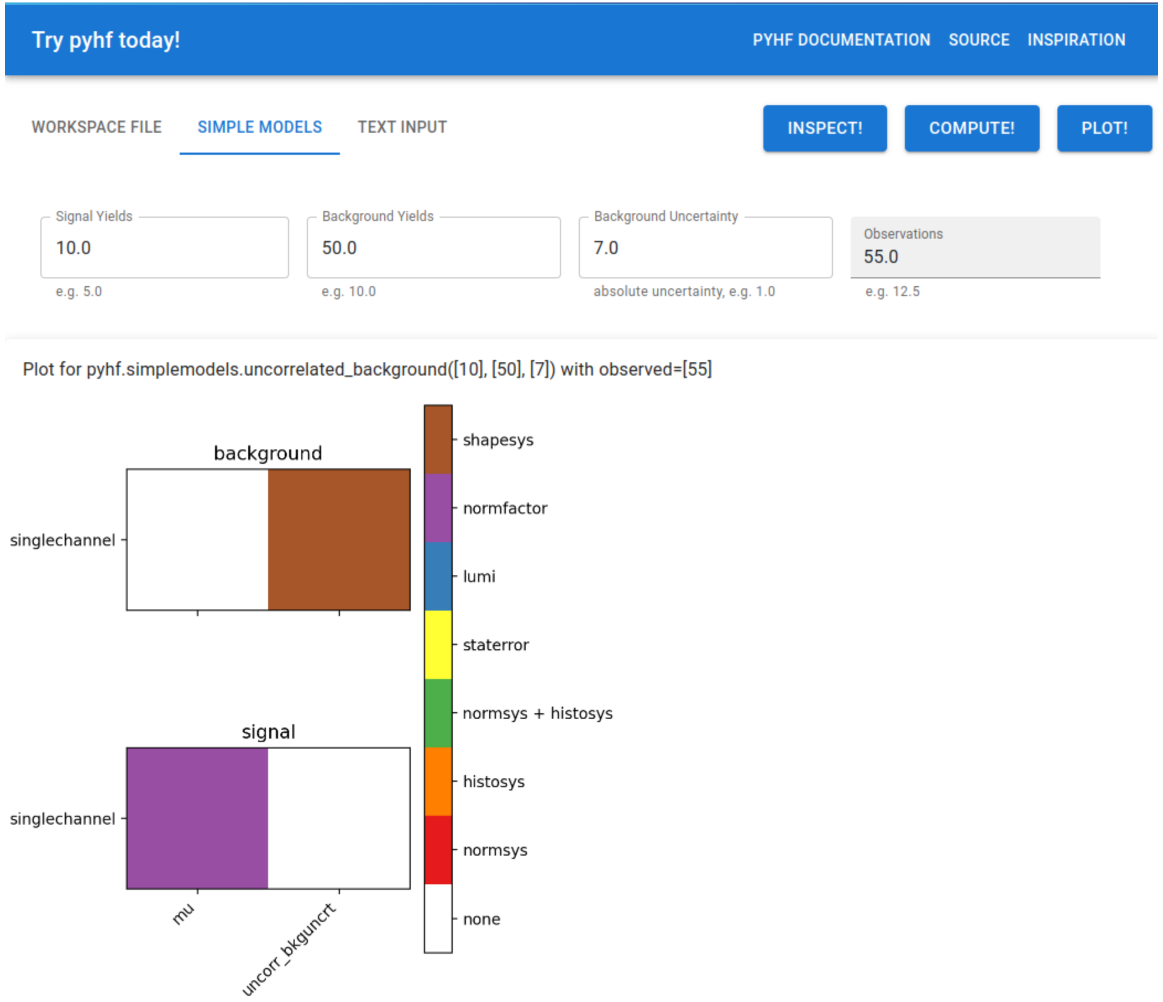
Future software/statistics training, web applications, schemea validation enabled with Pyodide and PyScript

Enabling full web apps with PyScript



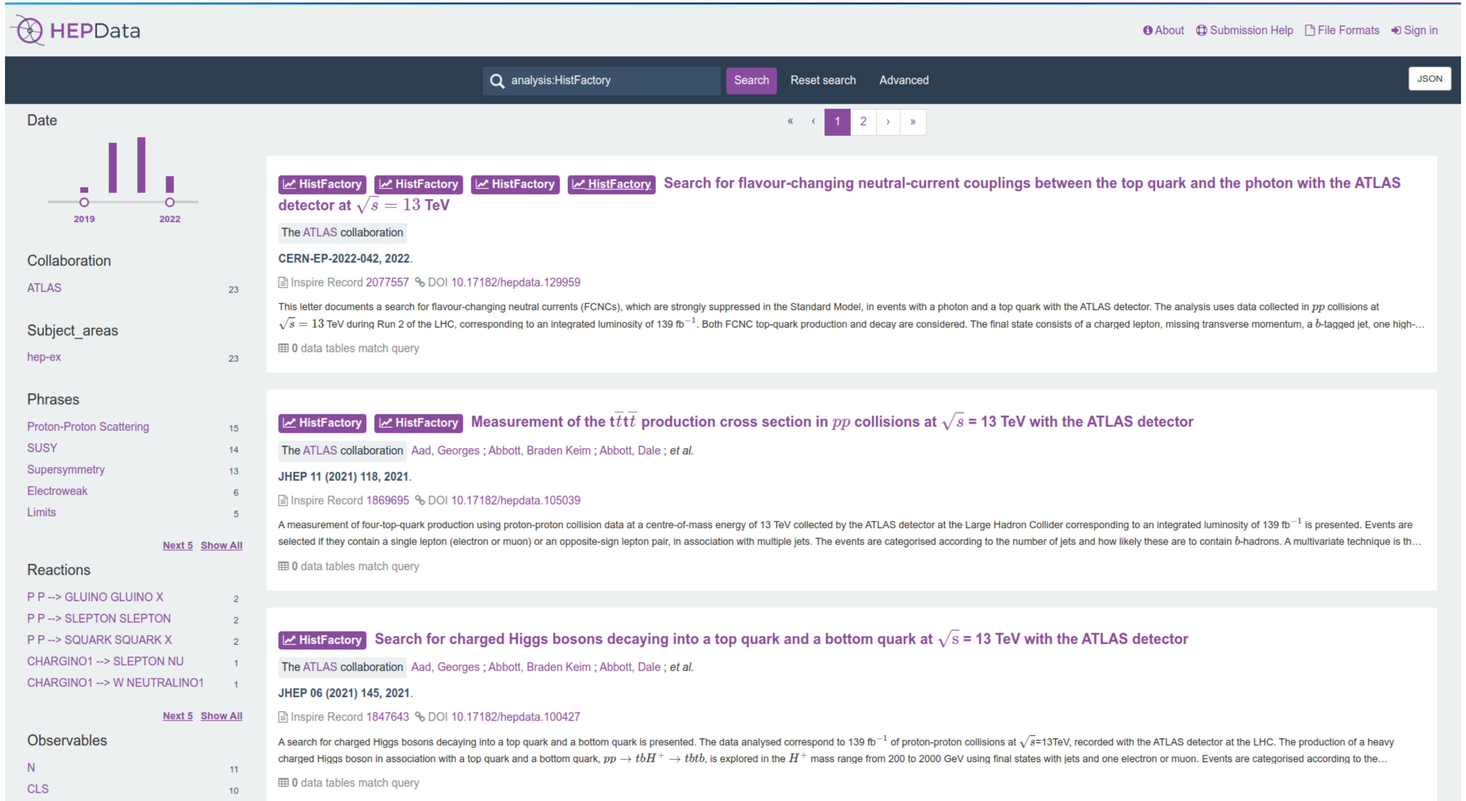
Future software/statistics training, web applications, schemea validation enabled with Pyodide and PyScript

Enabling full web apps with PyScript



Future software/statistics training, web applications, schemea validation enabled with Pyodide and PyScript

HEPData support for HistFactory JSON and more



Published HistFactory probability models get own DOI (future: model render, interactivity)

Summary

- Library for modeling and **accelerated** fitting
  - reducing time to insight/inference!
  - Hardware acceleration on GPUs and vectorized operations
  - Backend agnostic Python API and CLI
- Flexible **declarative** schema
  - JSON: ubiquitous, universal support, versionable
- Enabling technology for **reinterpretation**
  - JSON Patch files for efficient computation of new signal models
  - Unifying tool for theoretical and experimental physicists
- Growing use community across **all of HEP**
  - Theory and experiment
- Project in growing **Pythonic HEP ecosystem**
  - Openly developed on GitHub and welcome contributions
  - Comprehensive open tutorials

Thanks for listening!

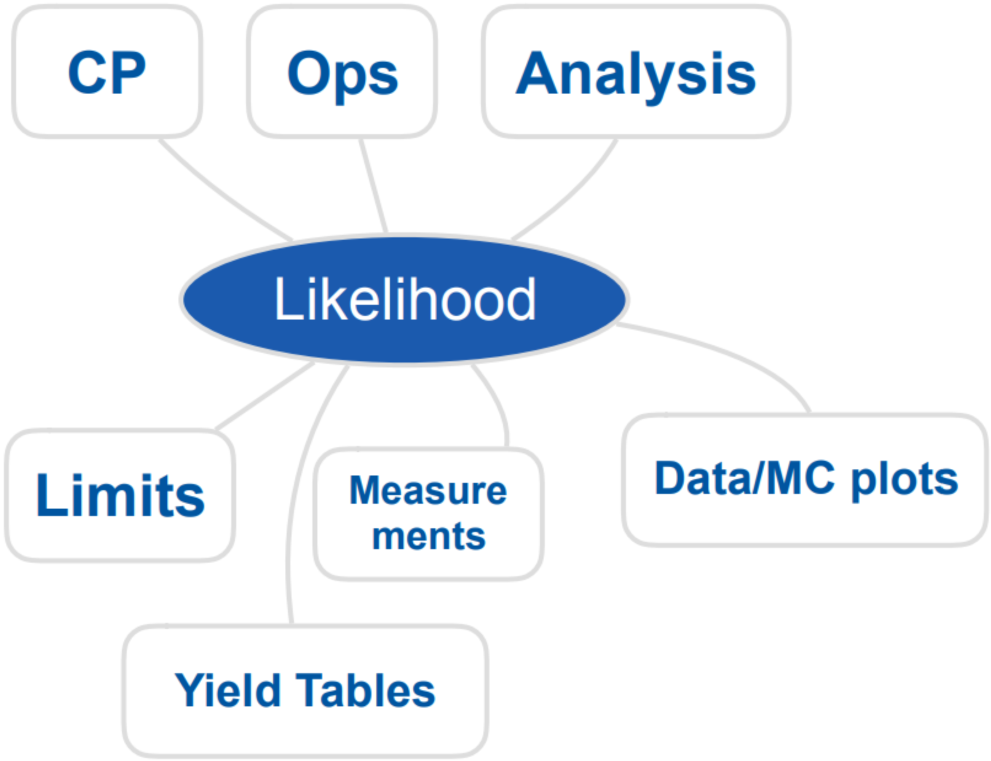
Come talk with us!

www.scikit-hep.org/pyhf

Backup

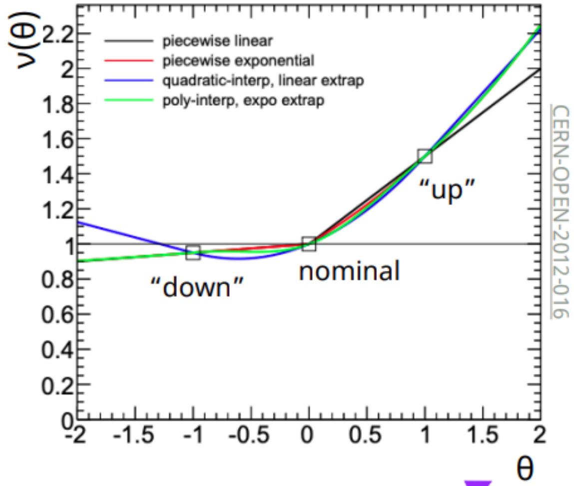
Why is the likelihood important?

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



HistFactory Template: systematic uncertainties

- In HEP common for systematic uncertainties to be specified with two template histograms: "up" and "down" variation for parameter $\theta \in \{\vec{\eta}, \vec{\chi}\}$
  - "up" variation: model prediction for $\theta = +1$
  - "down" variation: model prediction for $\theta = -1$
  - Interpolation and extrapolation choices provide **model predictions** $\nu(\vec{\theta}\,)$ **for any** $\vec{\theta}$
- Constraint terms $c_j\left(a_j|\theta_j\right)$ used to model auxiliary measurements. Example for Normal (most common case):
  - Mean of nuisance parameter θ_j with normalized width ($\sigma = 1$)
  - Normal: auxiliary data $a_j = 0$ (aux data function of modifier type)
  - Constraint term produces penalty in likelihood for pulling θ_j away from auxiliary measurement value
  - As $\nu(\vec{\theta}\,)$ constraint terms inform rate modifiers (**systematic uncertainties**) during simultaneous fit



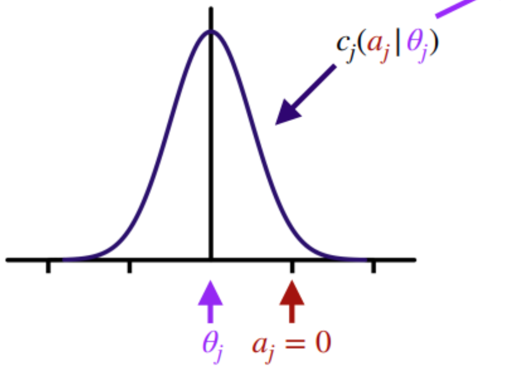


Image credit: Alexander Held

Full 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)

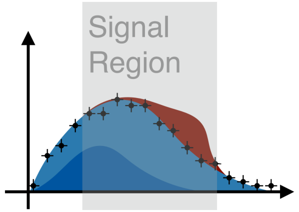
This hadn't been done in HEP until 2019

- 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

JSON Patch for signal model (reinterpretation)

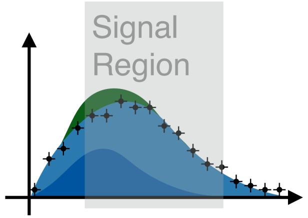
JSON Patch gives ability to **easily mutate model**
Think: test a **new theory** with a **new patch**!
(c.f. Lukas Heinrich's RECAST talk from Snowmass 2021 Computational Frontier Workshop)

Combined with RECAST gives powerful tool for **reinterpretation studies**



Signal model A

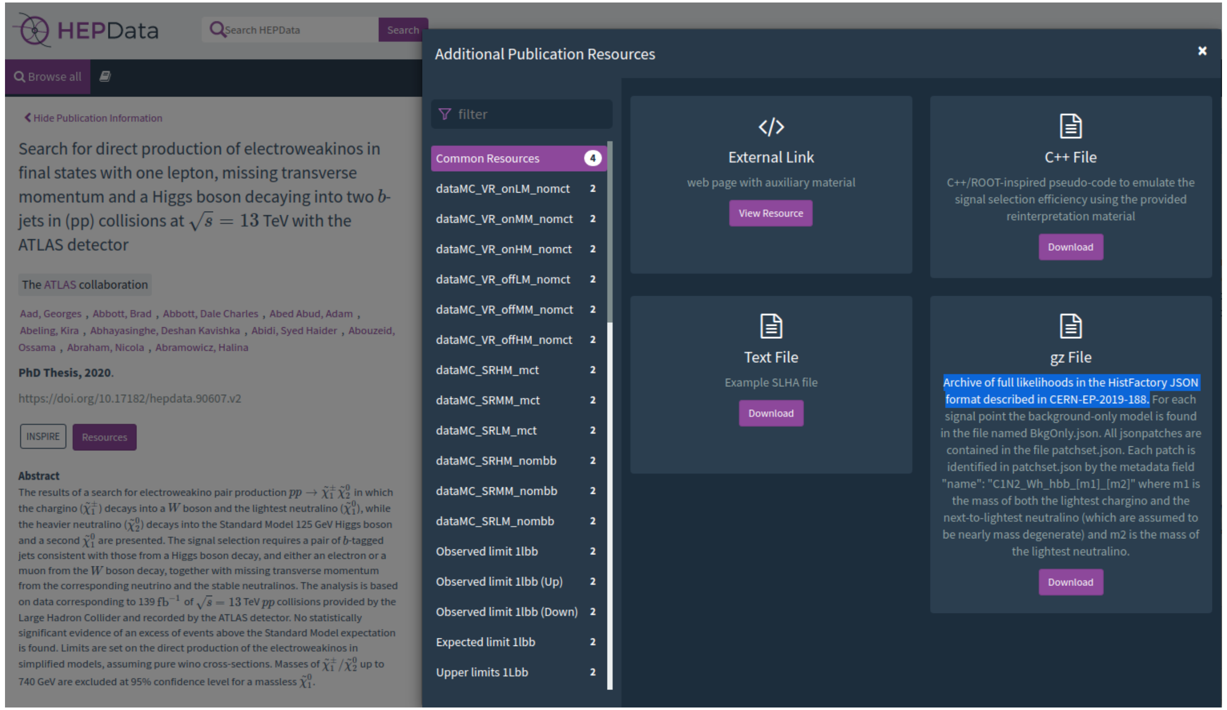
```
# Using CLI
$ pyhf cls example.json | jq .CLs_obs
0.053994246621274014

$ 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
```



Signal model B

Probability models reserved on HEPData

- `pyhf` pallet:
  - Background-only model JSON stored
  - Hundreds of signal model JSON Patches stored together as a `pyhf` "patch set" file
- Fully preserve and publish the full statistical model and observations to give likelihood
  - with own DOI! DOI 10.17182/hepdata.90607.v3/r3



...can be used from HEPData

- `pyhf` pallet:
  - Background-only model JSON stored
  - Hundreds of signal model JSON Patches stored together as a `pyhf` "patch set" file
- Fully preserve and publish the full statistical model and observations to give likelihood
  - with own DOI! DOI 10.17182/hepdata.90607.v3/r3

```
# pyhf pallet for the SUSY EWK 1Lbb analysis
$ pyhf contrib download https://doi.org/10.17182/hepdata.90607.v3/r3 1Lbb-pallet && cd 1Lbb-pallet

# verify patchset is valid
$ pyhf patchset verify BkgOnly.json patchset.json
All good.

# signal model: m1 = 900, m2 = 300 (chain CLI API output)
$ cat BkgOnly.json | \
  pyhf cls --patch <(pyhf patchset extract --name C1N2_Wh_hbb_900_300 patchset.json) | \
  jq .CLs_obs
0.5004165245329418

# new signal model: m1 = 900, m2 = 400 (use serialized CLI API output)
$ pyhf patchset extract --name C1N2_Wh_hbb_900_400 --output-file C1N2_Wh_hbb_900_400_patch.json patchset.json
$ pyhf cls --patch C1N2_Wh_hbb_900_400_patch.json BkgOnly.json | jq .CLs_obs
0.5735007268333779
```

API Example: Hypothesis test

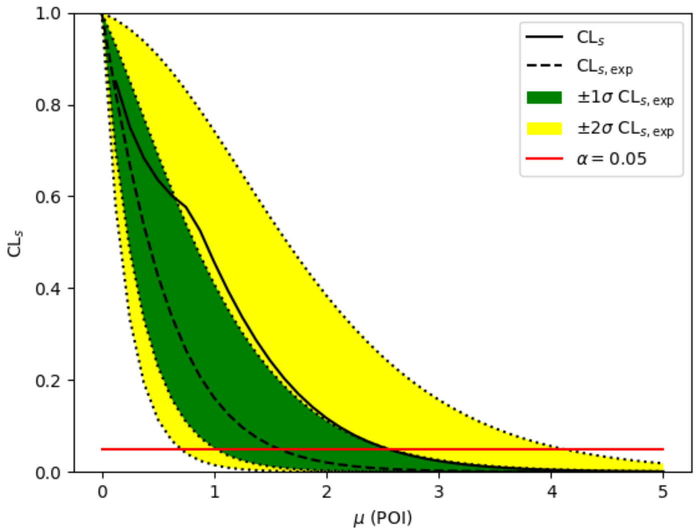
```
$ python -m pip install pyhf[jax,contrib]
$ pyhf contrib download https://doi.org/10.17182/hepdata.90607.v3/r3 1Lbb-pallet
```

```python
import json
import pyhf

pyhf.set_backend("jax")  # Optional for speed
spec = json.load(open("1Lbb-pallet/BkgOnly.json"))
patchset = pyhf.PatchSet(json.load(open("1Lbb-pallet/patchset.json")))

workspace = pyhf.Workspace(spec)
model = workspace.model(patches=[patchset["C1N2_Wh_hbb_900_250"]])

test_poi = 1.0
data = workspace.data(model)
cls_obs, cls_exp_band = pyhf.infer.hypotest(
    test_poi, data, model, test_stat="qtilde", return_expected_set=True
)
print(f"Observed CLs: {cls_obs}")
# Observed CLs: 0.4573416902360917
print(f"Expected CLs band: {[exp.tolist() for exp in cls_exp_band]}")
# Expected CLs band: [0.014838293214187472, 0.05174259485911152,
# 0.16166970886709053, 0.4097850957724176, 0.7428200727035176]
```

Python API Example: Upper limit

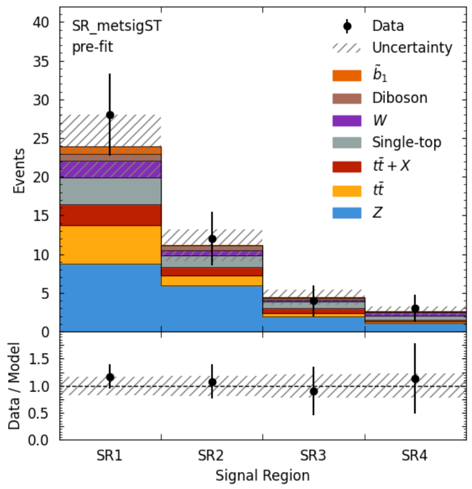
```
$ python -m pip install pyhf[jax,contrib]
$ pyhf contrib download https://doi.org/10.17182/hepdata.90607.v3/r3 1Lbb-pallet
```

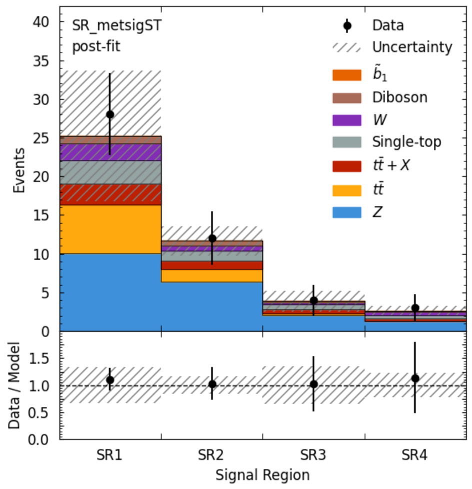
```python
import json
import matplotlib.pyplot as plt
import numpy as np
import pyhf
from pyhf.contrib.viz.brazil import plot_results

pyhf.set_backend("jax")  # Optional for speed

spec = json.load(open("1Lbb-pallet/BkgOnly.json"))
patchset = pyhf.PatchSet(json.load(open("1Lbb-pallet/patchset.json")))

workspace = pyhf.Workspace(spec)
model = workspace.model(patches=[patchset["C1N2_Wh_hbb_900_250"]])

test_pois = np.linspace(0, 5, 41)  # POI step of 0.125
data = workspace.data(model)
obs_limit, exp_limits, (test_pois, results) = pyhf.infer.intervals.upperlimit(
    data, model, test_pois, return_results=True
)

print(f"Observed limit: {obs_limit}")
# Observed limit: 2.547958147632675
print(f"Expected limits: {[limit.tolist() for limit in exp_limits]}")
# Expected limits: [0.7065311975182036, 1.0136453820160332,
# 1.5766626372587724, 2.558234487679955, 4.105381941514062]

fig, ax = plt.subplots()
artists = plot_results(test_pois, results, ax=ax)
fig.savefig("upper_limit.pdf")
```



API Example: Extend with cabinetry

```python
import json
import cabinetry
import pyhf
from cabinetry.model_utils import prediction
from pyhf.contrib.utils import download

# download the ATLAS bottom-squarks analysis probability models from HEPData
download("https://www.hepdata.net/record/resource/1935437?view=true", "bottom-squarks")

# construct a workspace from a background-only model and a signal hypothesis
bkg_only_workspace = pyhf.Workspace(
    json.load(open("bottom-squarks/RegionC/BkgOnly.json"))
)
patchset = pyhf.PatchSet(json.load(open("bottom-squarks/RegionC/patchset.json")))
workspace = patchset.apply(bkg_only_workspace, "sbottom_600_280_150")

# construct the probability model and observations
model, data = cabinetry.model_utils.model_and_data(workspace)

# produce visualizations of the pre-fit model and observed data
prefit_model = prediction(model)
cabinetry.visualize.data_mc(prefit_model, data)

# fit the model to the observed data
fit_results = cabinetry.fit.fit(model, data)

# produce visualizations of the post-fit model and observed data
postfit_model = prediction(model, fit_results=fit_results)
cabinetry.visualize.data_mc(postfit_model, data)
```

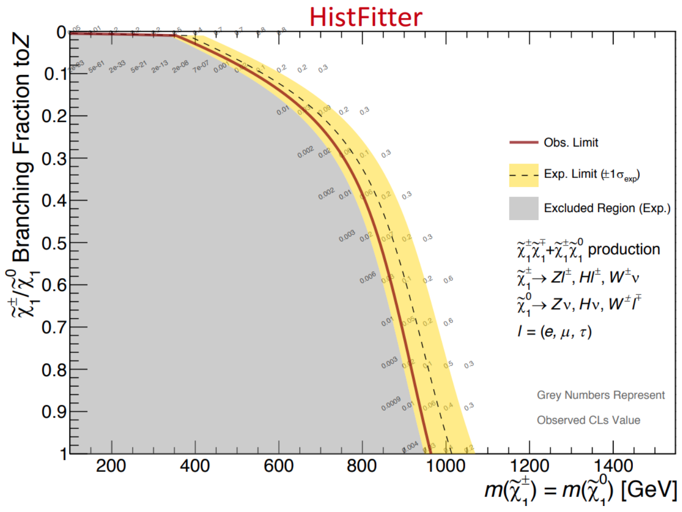


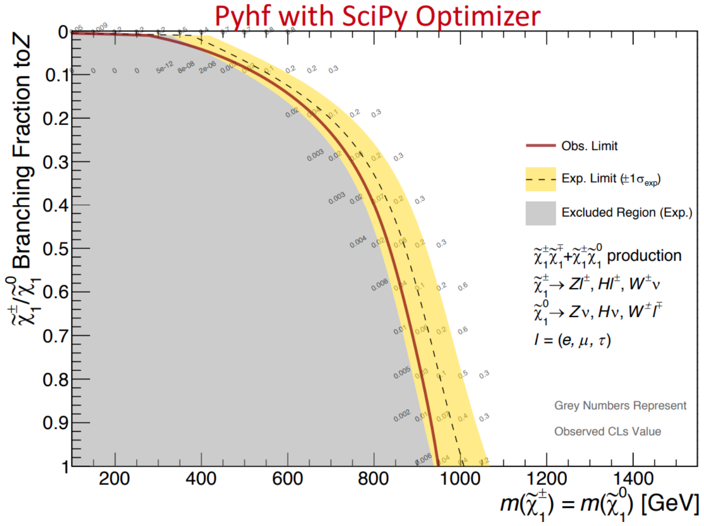


Rapid adoption in ATLAS...

- 22 ATLAS SUSY, Exotics, Top analyses with full probability models published to HEPData
- ATLAS SUSY will be continuing to publish full Run 2 likelihoods

- direct staus, doi:10.17182/hepdata.89408 (2019)
- sbottom multi-b, doi:10.17182/hepdata.91127 (2019)
- 1Lbb, doi:10.17182/hepdata.92006 (2019)
- 3L eRJR, doi:10.17182/hepdata.90607 (2020)
- ss3L search, doi:10.17182/hepdata.91214 (2020)





SUSY EWK 3L RPV analysis (ATLAS-CONF-2020-009): Exclusion curves as a function of mass and branching fraction to Z bosons

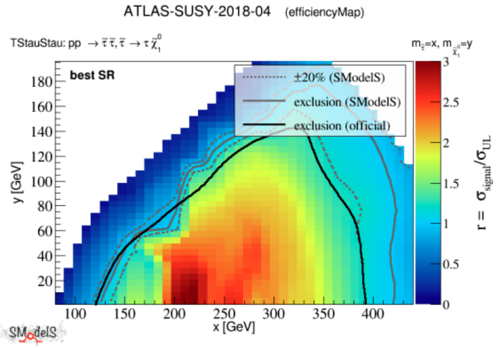
...and by theory

- `pyhf` likelihoods discussed in
  - Les Houches 2019 Physics at TeV Colliders: New Physics Working Group Report
  - Higgs boson potential at colliders: status and perspectives
- SModelS team has implemented a `SModelS`/`pyhf` interface [arXiv:2009.01809]
  - tool for interpreting simplified-model results from the LHC
  - designed to be used by theorists
  - `SModelS` authors giving tutorial later today!

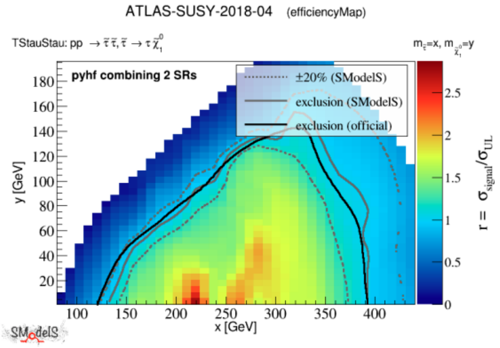
Validation & impact

Gaël Alguero, SK, Wolfgang Waltenberger, arXiv:2009.01809

- ATLAS-SUSY-2018-04: TStauStau



Best SR: over exclusion



Full likelihood: very good agreement with official ATLAS result

The remaining small difference is probably due to the (interpolated) A×ε values from the simplified model efficiency maps not exactly matching the "true"ones of the experimental analysis.

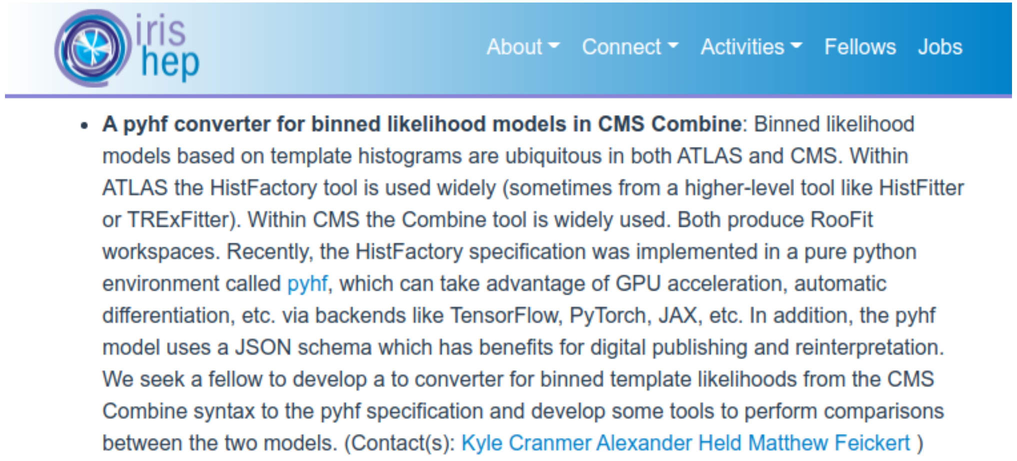
S. Kraml - Feedback on use of public likelihoods - 24 Sep 2020 16

Feedback on use of public probability models, Sabine Kraml (ATLAS Exotics + SUSY Reinterpretations Workshop)

- Have produced three comparisons to published ATLAS likelihoods: ATLAS-SUSY-2018-04, ATLAS-SUSY-2018-31, ATLAS-SUSY-2019-08
  - Compare simplified likelihood (bestSR) to full likelihood (`pyhf`) using `SModelS`

Ongoing work to interface CMS Combine

- `pyhf` users in 2022: ATLAS, Belle II, phenomenology community, IRIS-HEP
- Working to create a bridge for CMS to use and validate with a converter to CMS Combine
  - Difficult as HistFactory is "closed world" of models and CMS Combine is RooFit "open world"
- IRIS-HEP Fellow Summer 2022 project is ongoing with some promising preliminary results



- **A pyhf converter for binned likelihood models in CMS Combine**: Binned likelihood models based on template histograms are ubiquitous in both ATLAS and CMS. Within ATLAS the HistFactory tool is used widely (sometimes from a higher-level tool like HistFitter or TRExFitter). Within CMS the Combine tool is widely used. Both produce RooFit workspaces. Recently, the HistFactory specification was implemented in a pure python environment called pyhf, which can take advantage of GPU acceleration, automatic differentiation, etc. via backends like TensorFlow, PyTorch, JAX, etc. In addition, the pyhf model uses a JSON schema which has benefits for digital publishing and reinterpretation. We seek a fellow to develop a to converter for binned template likelihoods from the CMS Combine syntax to the pyhf specification and develop some tools to perform comparisons between the two models. (Contact(s): Kyle Cranmer Alexander Held Matthew Feickert)

A pyhf converter for binned likelihood models in CMS Combine

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

The end.