UC SANTA CRUZ

Likelihood Preservation

Dr. Giordon Stark (on behalf of the ATLAS Collaboration)

SUSY2019

May 23rd, 2019

giordonstark.com

pyhf

∂ifferentiable

ℒikelihoods

ATLAS EXPERIMENT

Run: 300800

Event: 2418777995

2016-06-04 03:47:03 G

if you can read this, you're too close

UC SANTA CRUZ

CERN

Likelihood Preservation

Dr. Giordon Stark (on behalf of the ATLAS Collaboration)

SUSY2019

May 23rd, 2019

giordonstark.com

ATLAS EXPERIMENT

pyhf

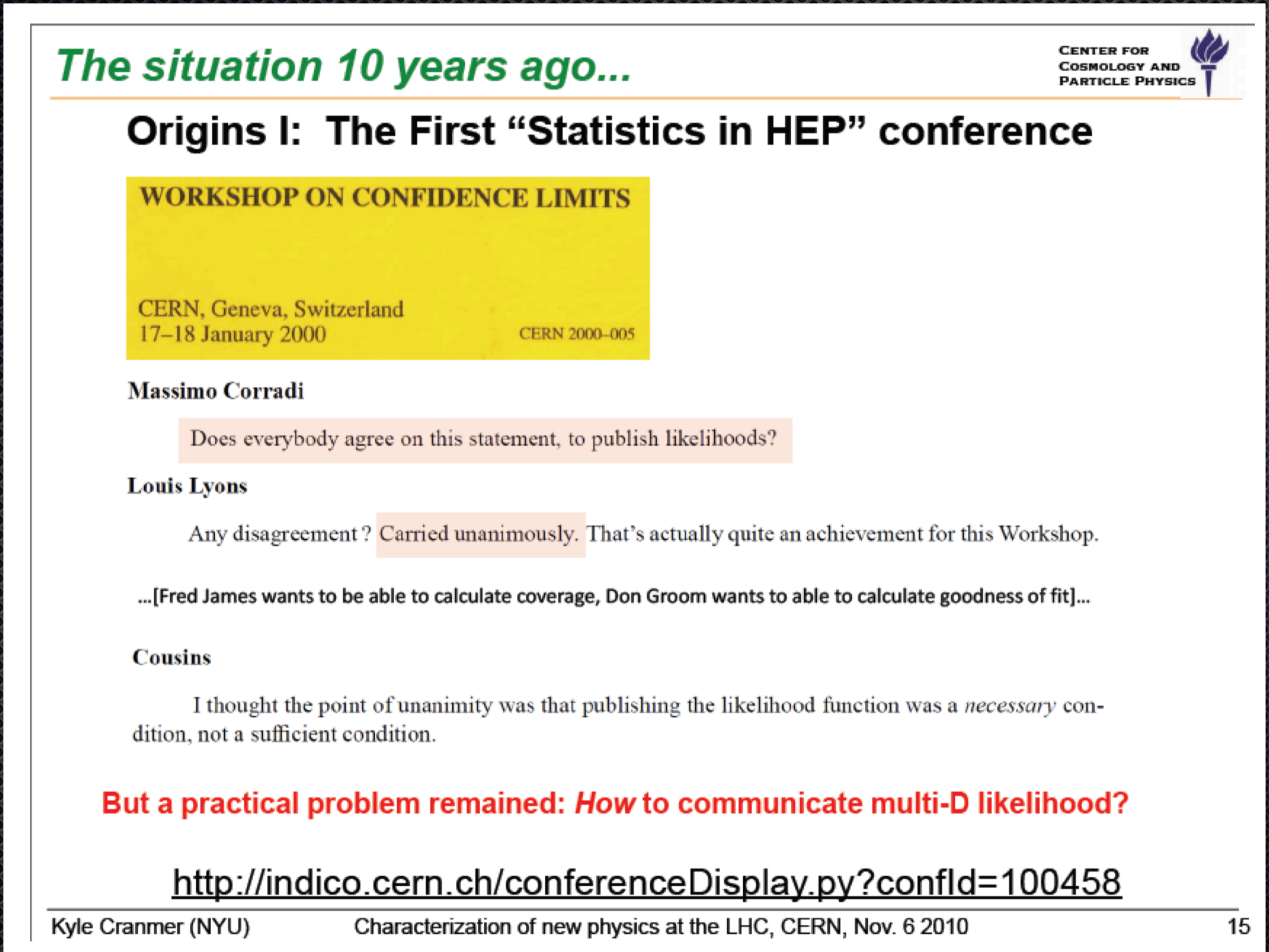
∂ifferentiable ℒikelihoods

“piff”?

if you can read this, you're too close

https://indico.cern.ch/event/100458/

9 years ago...



The situation 10 years ago...

CENTER FOR COSMOLOGY AND PARTICLE PHYSICS

Origins I: The First “Statistics in HEP” conference

WORKSHOP ON CONFIDENCE LIMITS

CERN, Geneva, Switzerland
17–18 January 2000

CERN 2000–005

Massimo Corradi

Does everybody agree on this statement, to publish likelihoods?

Louis Lyons

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

...[Fred James wants to be able to calculate coverage, Don Groom wants to able to calculate goodness of fit]...

Cousins

I thought the point of unanimity was that publishing the likelihood function was a *necessary* condition, not a sufficient condition.

But a practical problem remained: *How* to communicate multi-D likelihood?

http://indico.cern.ch/conferenceDisplay.py?confId=100458

Kyle Cranmer (NYU) Characterization of new physics at the LHC, CERN, Nov. 6 2010 15

⚠ ATLAS reminded everyone that we all agreed in 2000 to publish likelihoods!

https://indico.cern.ch/event/100458/

⚠ **ATLAS reminded everyone that we all agreed in 2000 to publish likelihoods!**

Overview of today’s talk

multi-bin histogram-based statistical fits

and how to preserve them

- HistFactory: ROOT+XML
- pyhf: Python+JSON

THE DEVELOPERS

G. Stark

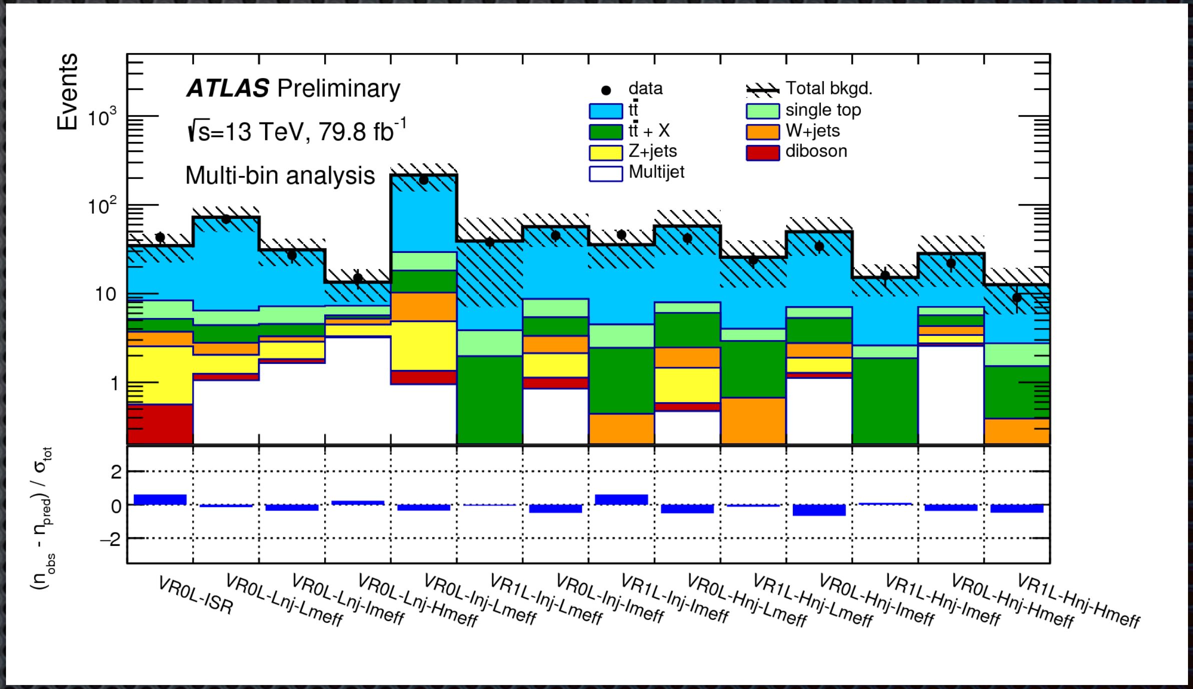
M. Feickert

L. Heinrich

HistFactory [CERN-OPEN-2012-016]

- A flexible **p.d.f template specification** for the building of statistical models from binned distributions and data
- Developed by Cranmer, Lewis, Moneta, Shibata, and Verkerke
- Widely used by the HEP community for standard model measurements and BSM searches

Calculated using HistFactory →



K. Cranmer

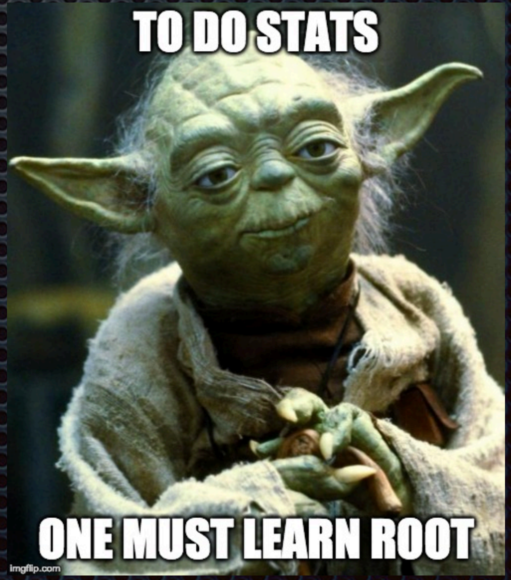
HistFactory is partially independent of its implementation in ROOT

HistFactory? It’s just math!

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

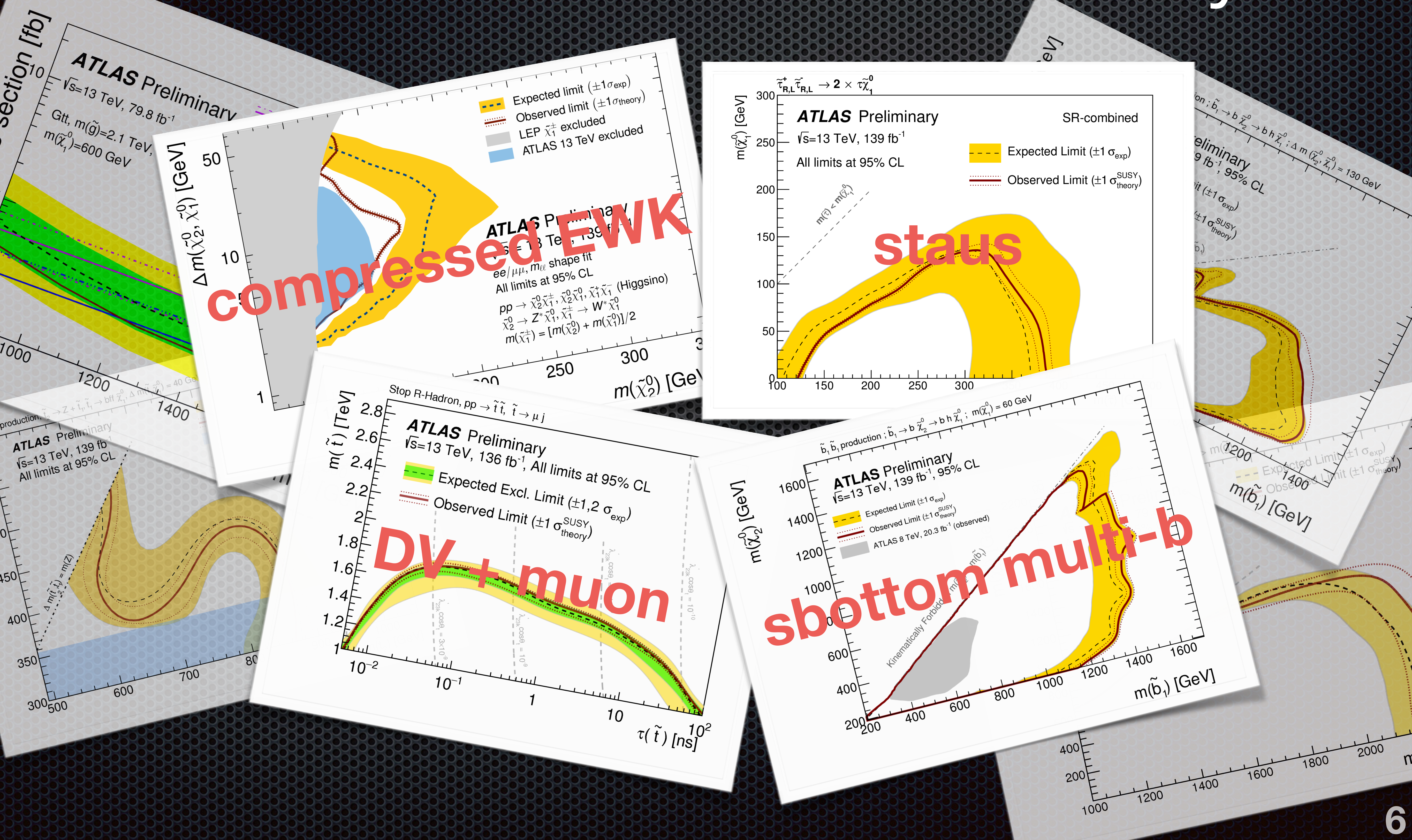
Multiple, disjoint channels of binned distributions with multiple samples contributing to each with additional (shared[?]) systematics between sample estimates

- An XML specification with data stored in ROOT files — it’s been the ***only implementation*** of this calculation
  - **Poisson p.d.f.** for bins observed in all channels
  - **Constraint p.d.f.** (and data) for auxiliary measurements (systematics: normalization, shape, etc)
  - ⚠ Tied to ROOT ecosystem
  - ⚠ How do we scale? (No multi-threading for larger workspaces e.g. combinations)
  - ⚠ How do we preserve?
  - ⚠ What if there’s a bug in ROOT’s HistFactory implementation? No cross-check!



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

What else uses HistFactory?



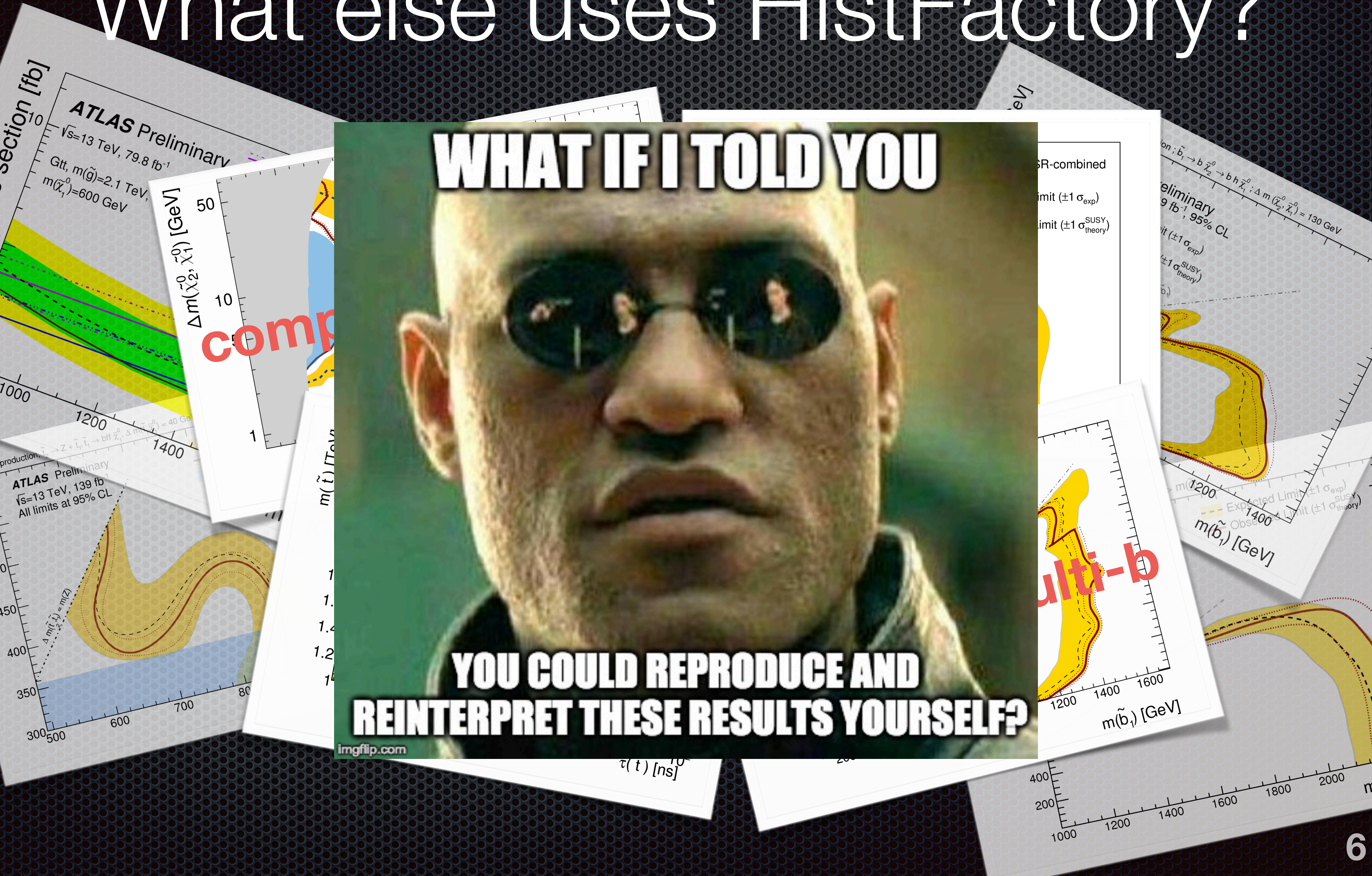
compressed EWK

staus

DV + muon

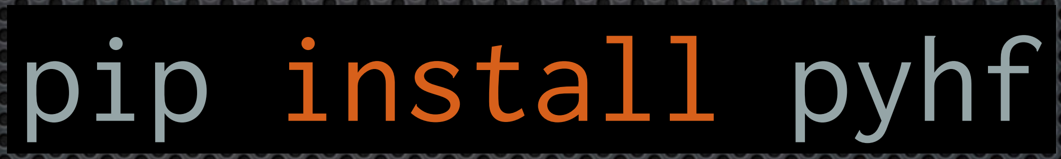
sbottom multi-b

What else uses HistFactory?



What is pyhf? (I)

it would be useful to **run statistical analysis outside of ROOT**, RooFit, RooStats framework



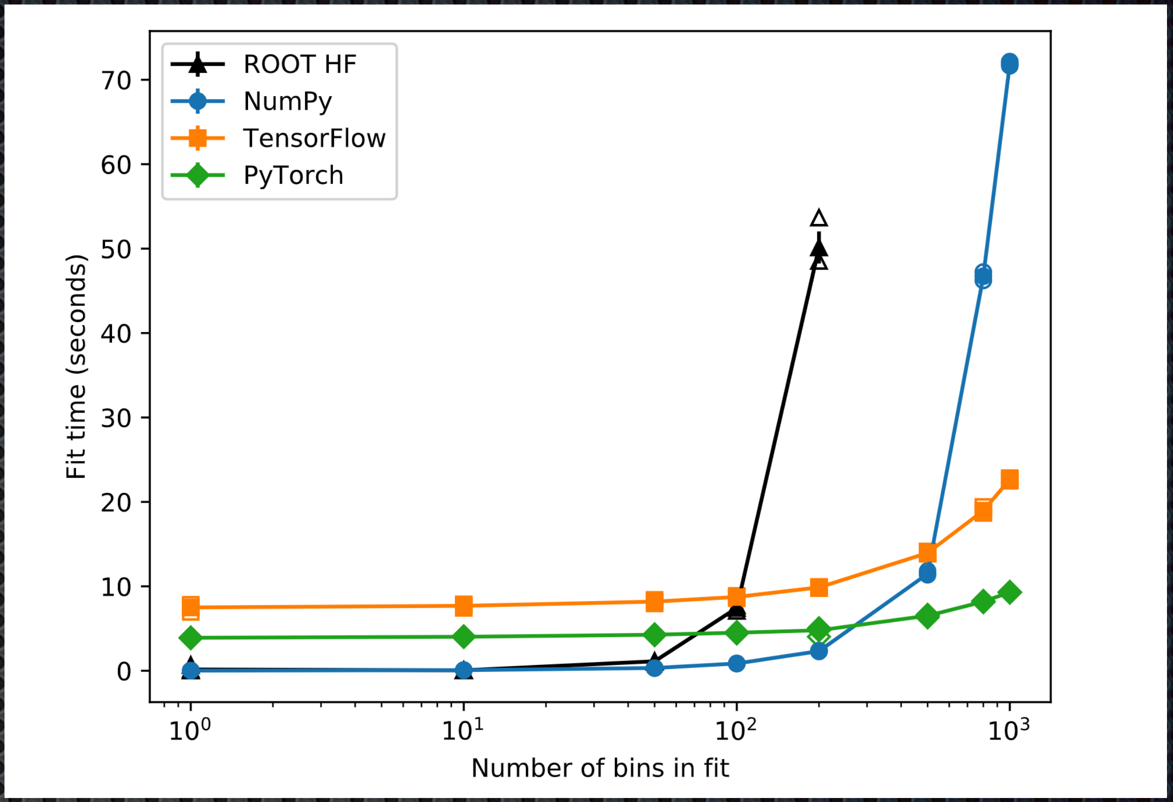
```
pip install pyhf
```

A **python-only** (scipy, numpy) implementation of the HistFactory model + profile likelihood hypothesis tests

For free: a single plain-text file (JSON) specifies the entire workspace

https://diana-hep.org/pyhf/

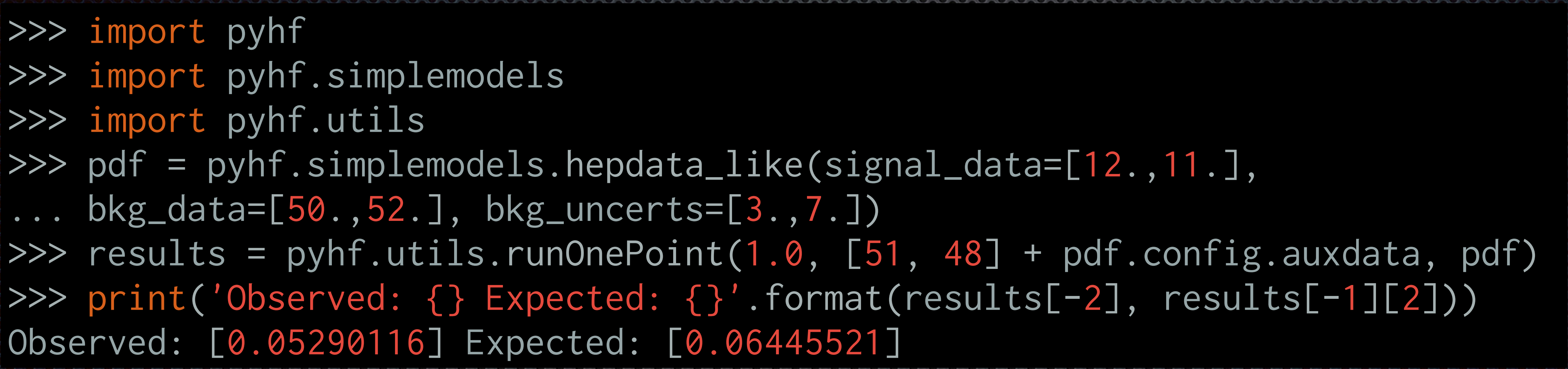
What is pyhf? (II)



- pyhf implements all numeric operations through a thin layer of of abstract n-D array operations to various **tensor algebra backends**
    - Rely on industry-standard open-source libraries to gain (instantaneous) benefits in speed ups and calculations as they come out

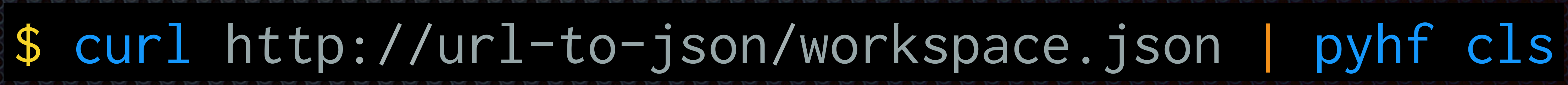
PyTorch TensorFlow NumPy mxnet

Hello World



```
>>> import pyhf
>>> import pyhf.simplemodels
>>> import pyhf.utils
>>> pdf = pyhf.simplemodels.hepdata_like(signal_data=[12.,11.],
... bkg_data=[50.,52.], bkg_uncerts=[3.,7.])
>>> results = pyhf.utils.runOnePoint(1.0, [51, 48] + pdf.config.auxdata, pdf)
>>> print('Observed: {} Expected: {}'.format(results[-2], results[-1][2]))
Observed: [0.05290116] Expected: [0.06445521]
```

- Want to use…
  - tensorflow? `pip install pyhf[tensorflow]`
  - pytorch? `pip install pyhf[pytorch]`
  - mxnet? `pip install pyhf[mxnet]`
- If the JSON workspace is online, can pipe and calculate CLs instantly



```
$ curl http://url-to-json/workspace.json | pyhf cls
```

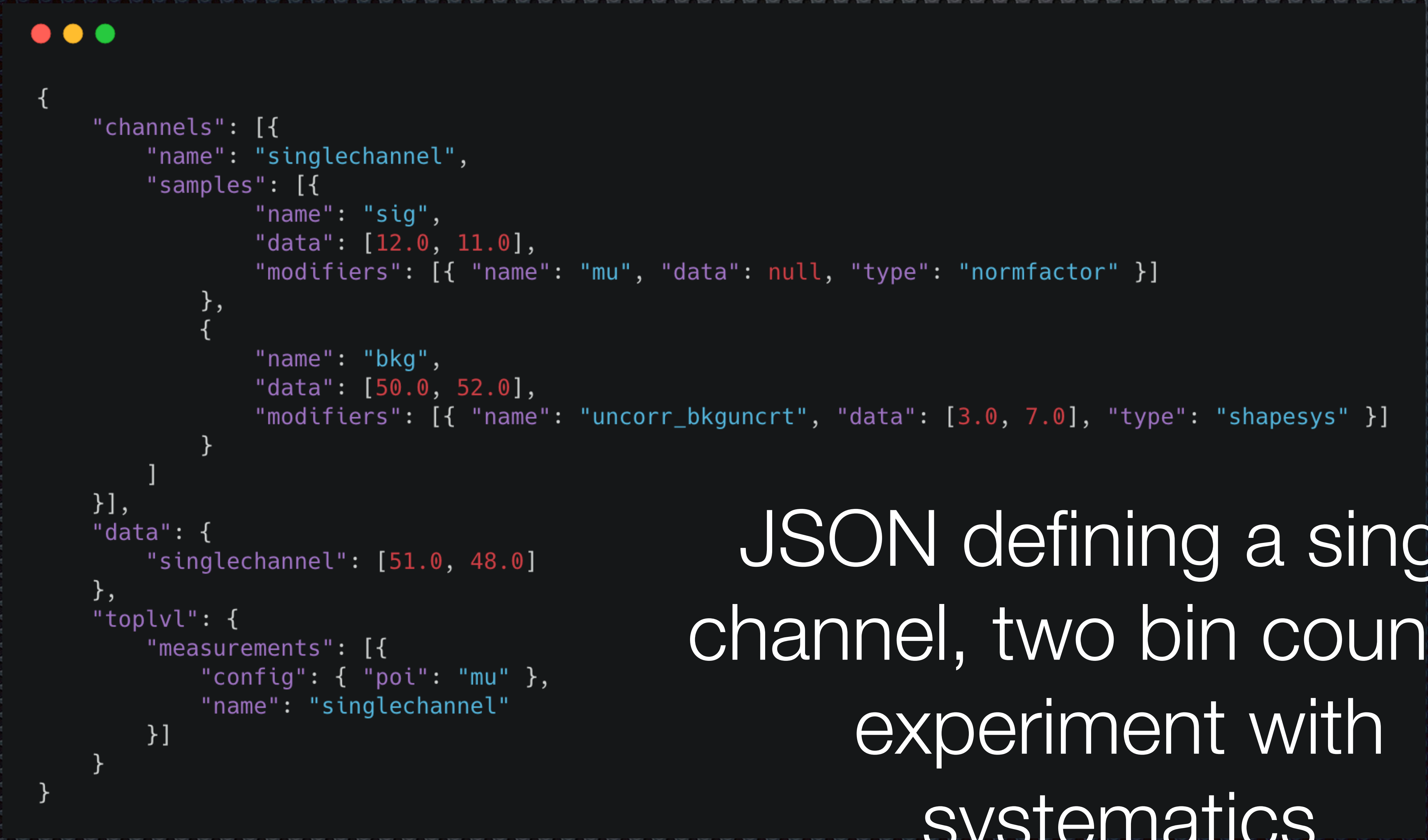
Demo (I)

- Interactive / real-time likelihood calculation and visualization with pyhf

launch binder

```
$ curl pdf.json | pyhf cls
```

Demo (II) — Simple CLs



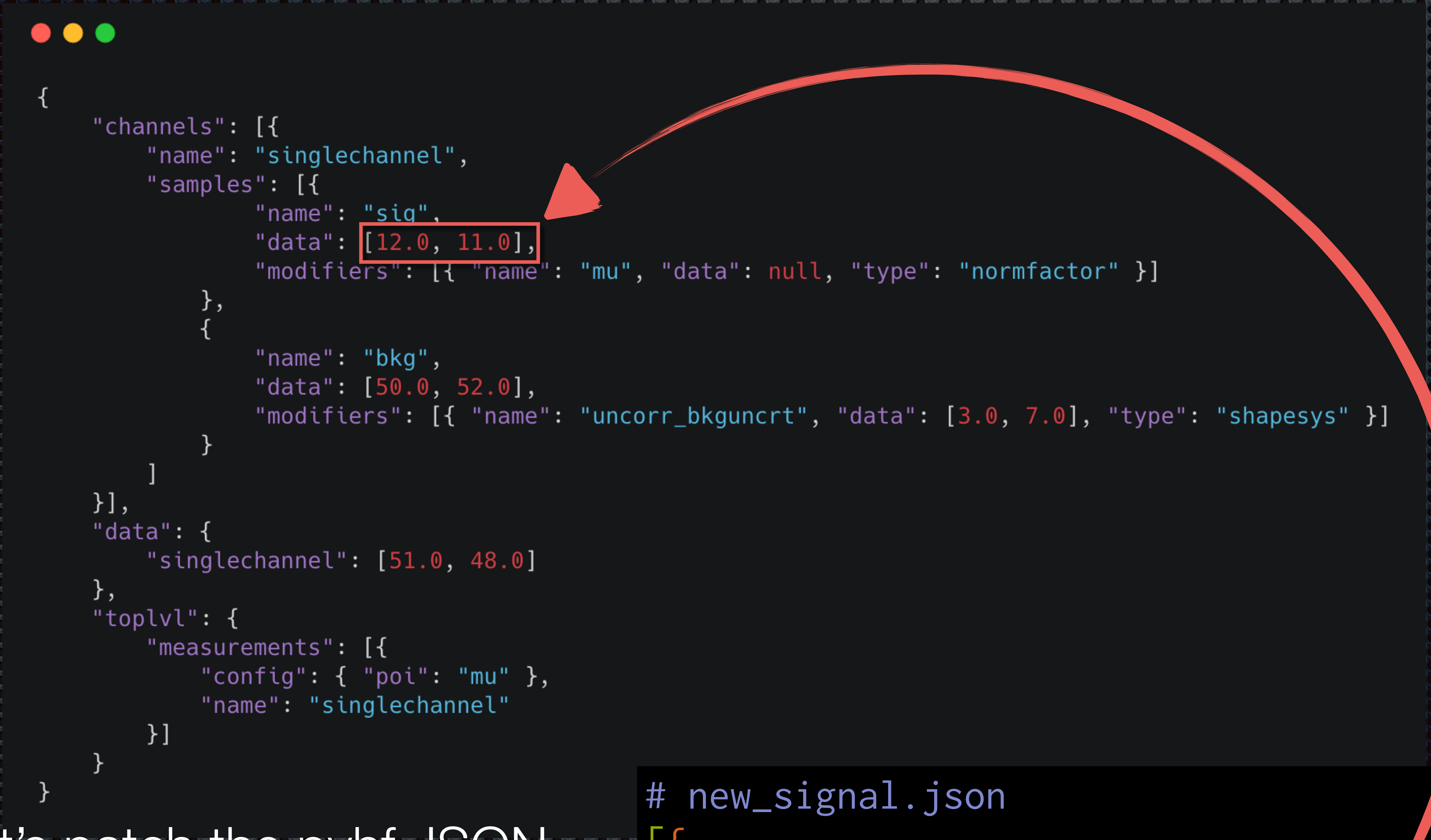
```
{
    "channels": [{
        "name": "singlechannel",
        "samples": [{
                "name": "sig",
                "data": [12.0, 11.0],
                "modifiers": [{ "name": "mu", "data": null, "type": "normfactor" }]
            },
            {
                "name": "bkg",
                "data": [50.0, 52.0],
                "modifiers": [{ "name": "uncorr_bkguncrt", "data": [3.0, 7.0], "type": "shapesys" }]
            }
        ]
    }],
    "data": {
        "singlechannel": [51.0, 48.0]
    },
    "toplvl": {
        "measurements": [{
            "config": { "poi": "mu" },
            "name": "singlechannel"
        }]
    }
}
```

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

```
$ curl -sL https://git.io/fpuyB | pyhf cls | jq .CLs_obs
0.053404965240922135
```

```
$ curl pdf.json | pyhf cls --patch patch.json
```

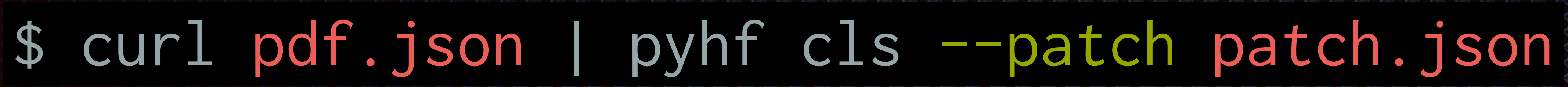
Demo (III) — Simple Re-use



```
{
    "channels": [{
        "name": "singlechannel",
        "samples": [{
                "name": "sig",
                "data": [12.0, 11.0],
                "modifiers": [{ "name": "mu", "data": null, "type": "normfactor" }]
            },
            {
                "name": "bkg",
                "data": [50.0, 52.0],
                "modifiers": [{ "name": "uncorr_bkguncrt", "data": [3.0, 7.0], "type": "shapesys" }]
            }
        ]
    }],
    "data": {
        "singlechannel": [51.0, 48.0]
    },
    "toplvl": {
        "measurements": [{
            "config": { "poi": "mu" },
            "name": "singlechannel"
        }]
    }
}
```

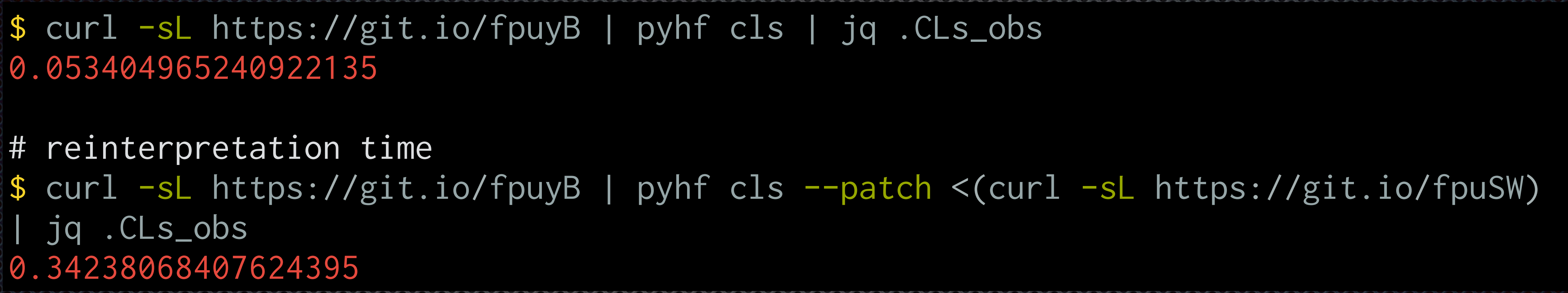
- Let’s patch the pyhf JSON spec provided with a different signal and recalculate!

```
# new_signal.json
[{
    "op": "replace",
    "path": "/channels/0/samples/0/data",
    "value": [5.0, 6.0]
}]
```



```
$ curl pdf.json | pyhf cls --patch patch.json
```

Demo (III) — Simple Re-use

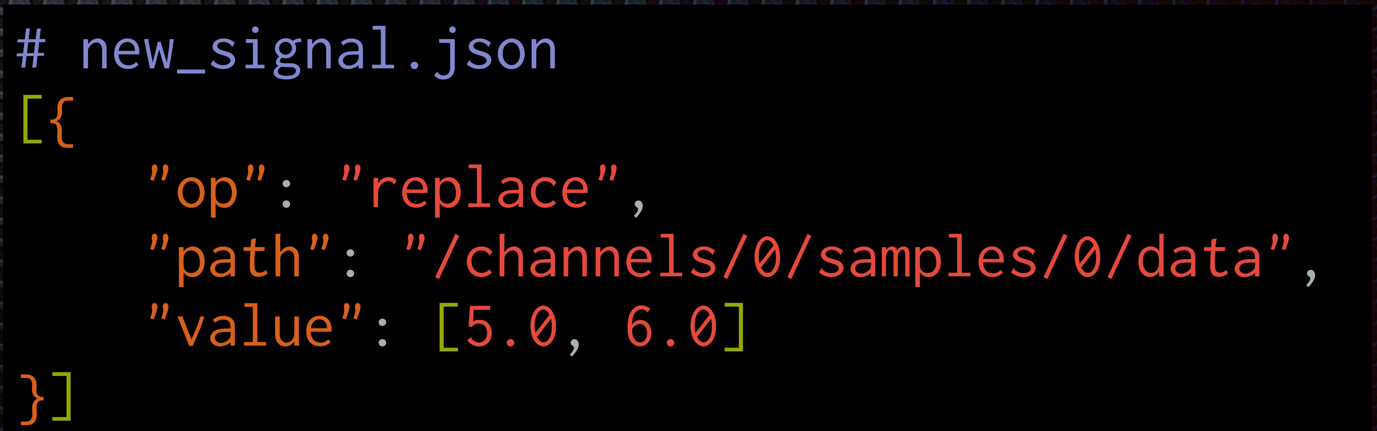


```
$ curl -sL https://git.io/fpuyB | pyhf cls | jq .CLs_obs
0.0534049652409221 35

# reinterpretation time
$ curl -sL https://git.io/fpuyB | pyhf cls --patch <(curl -sL https://git.io/fpuSW)
| jq .CLs_obs
0.3423806840762439 5
```

Patch with JSONPatch (http://jsonpatch.com/)

- Let’s patch the pyhf JSON spec provided with a different signal and recalculate!

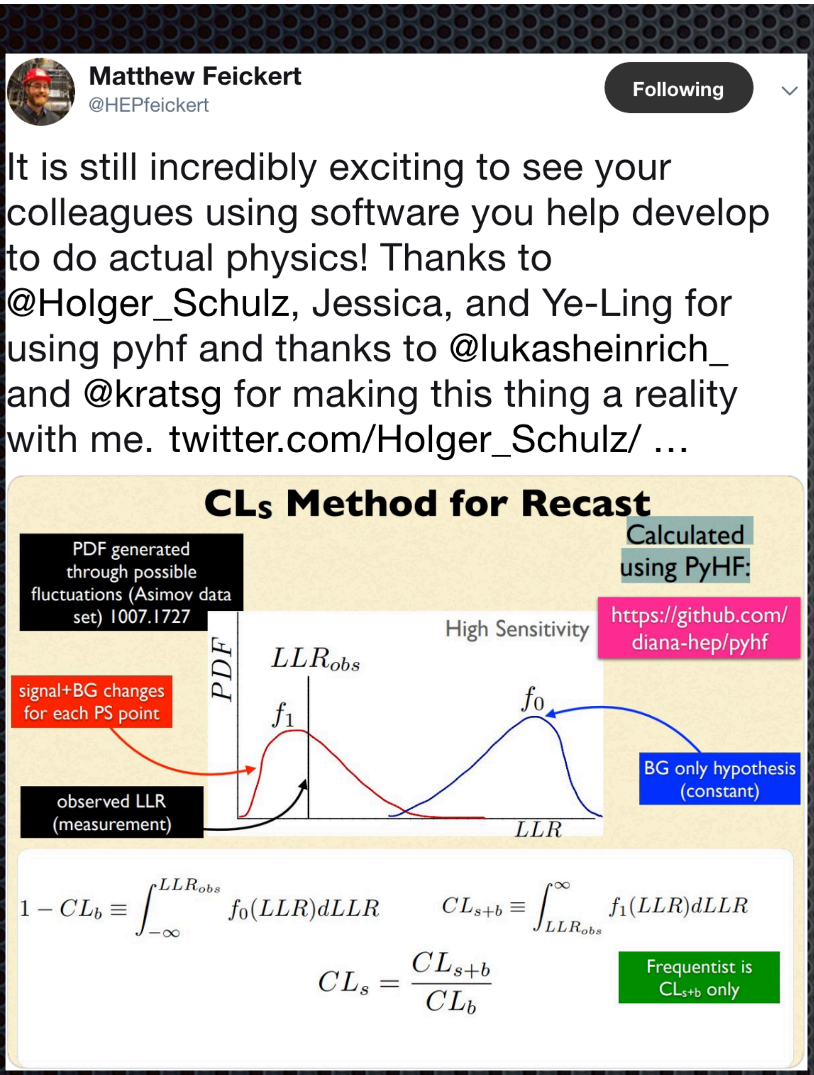


```
# new_signal.json
[{
    "op": "replace",
    "path": "/channels/0/samples/0/data",
    "value": [5.0, 6.0]
}]
```

pyhf in the wild

NuTheories 2018

[1810.05648]



Matthew Feickert
@HEPfeickert
Following

It is still incredibly exciting to see your colleagues using software you help develop to do actual physics! Thanks to @Holger_Schulz, Jessica, and Ye-Ling for using pyhf and thanks to @lukasheinrich_ and @kratsg for making this thing a reality with me. twitter.com/Holger_Schulz/ …

CLs Method for Recast

PDF generated through possible fluctuations (Asimov data set) 1007.1727

Calculated using PyHF:

https://github.com/diana-hep/pyhf

High Sensitivity

signal+BG changes for each PS point

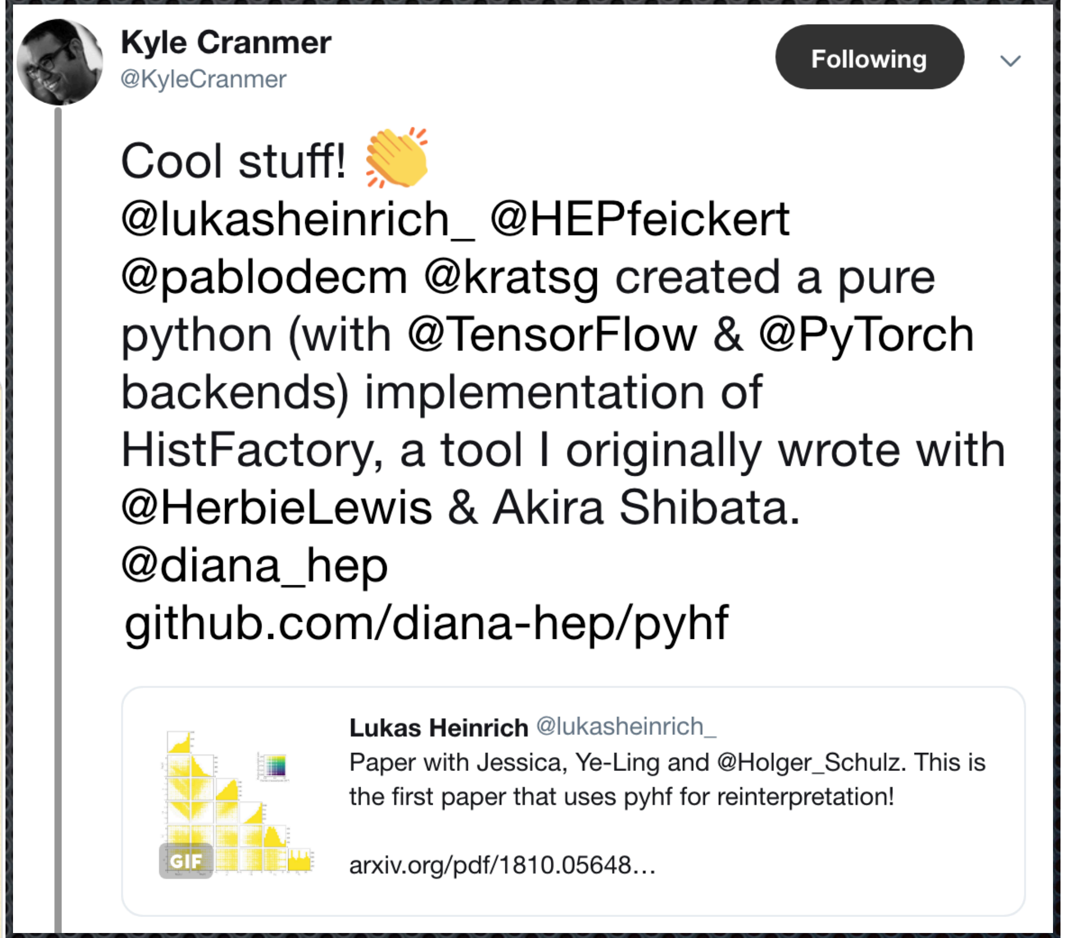
BG only hypothesis (constant)

observed LLR (measurement)

$$1 - CL_b \equiv \int_{-\infty}^{LLR_{obs}} f_0(LLR)dLLR \qquad CL_{s+b} \equiv \int_{LLR_{obs}}^{\infty} f_1(LLR)dLLR$$

$$CL_s = \frac{CL_{s+b}}{CL_b}$$

Frequentist is CL_{s+b} only

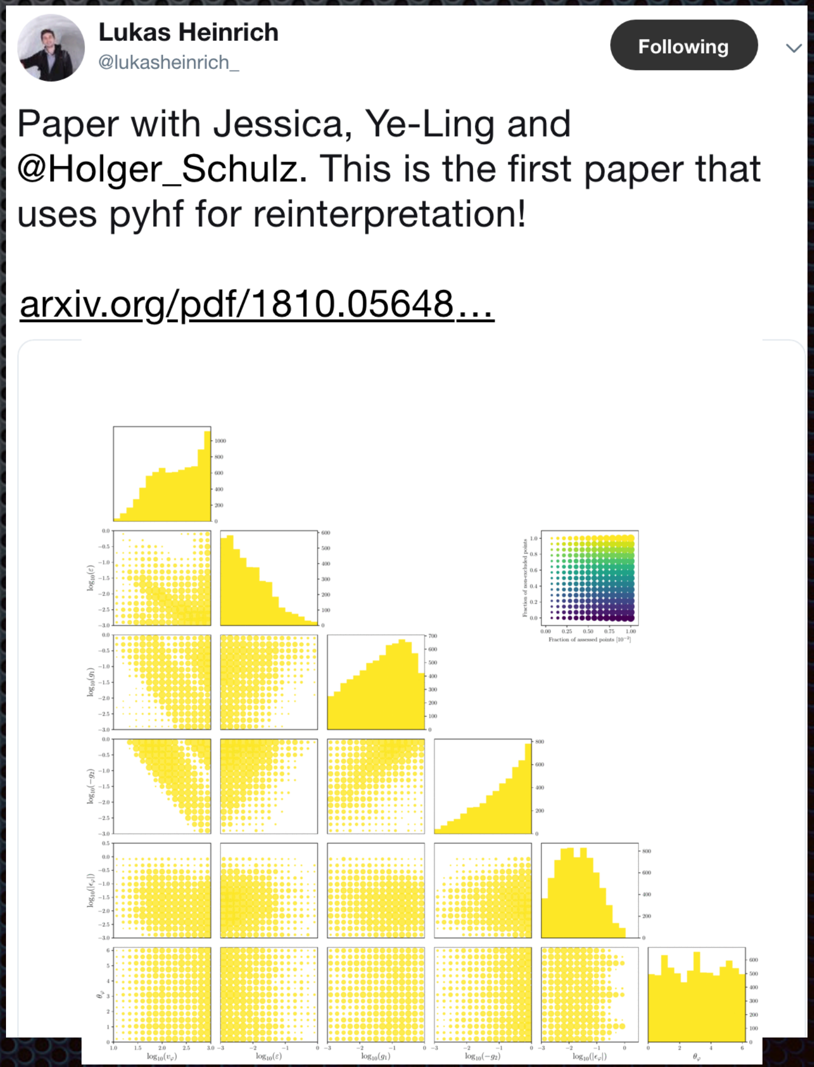


Kyle Cranmer
@KyleCranmer
Following

Cool stuff! 👏 @lukasheinrich_ @HEPfeickert @pablodecm @kratsg created a pure python (with @TensorFlow & @PyTorch backends) implementation of HistFactory, a tool I originally wrote with @HerbieLewis & Akira Shibata. @diana_hep github.com/diana-hep/pyhf

Lukas Heinrich @lukasheinrich_
Paper with Jessica, Ye-Ling and @Holger_Schulz. This is the first paper that uses pyhf for reinterpretation!

arxiv.org/pdf/1810.05648…



Lukas Heinrich
@lukasheinrich_
Following

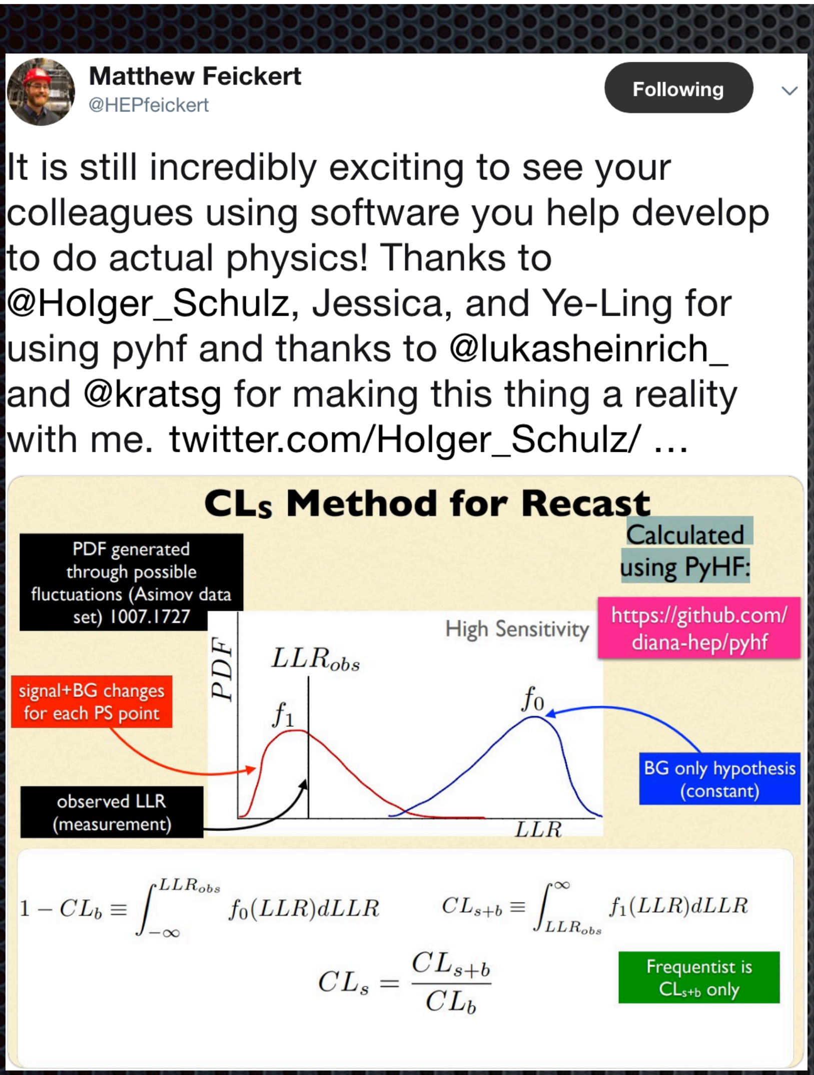
Paper with Jessica, Ye-Ling and @Holger_Schulz. This is the first paper that uses pyhf for reinterpretation!

arxiv.org/pdf/1810.05648…

pyhf in the wild

NuTheories 2018

[1810.05648]



Matthew Feickert @HEPfeickert Following

It is still incredibly exciting to see your colleagues using software you help develop to do actual physics! Thanks to @Holger_Schulz, Jessica, and Ye-Ling for using pyhf and thanks to @lukasheinrich_ and @kratsg for making this thing a reality with me. twitter.com/Holger_Schulz/ …

CLs Method for Recast

PDF generated through possible fluctuations (Asimov data set) 1007.1727

Calculated using PyHF:

https://github.com/diana-hep/pyhf

signal+BG changes for each PS point

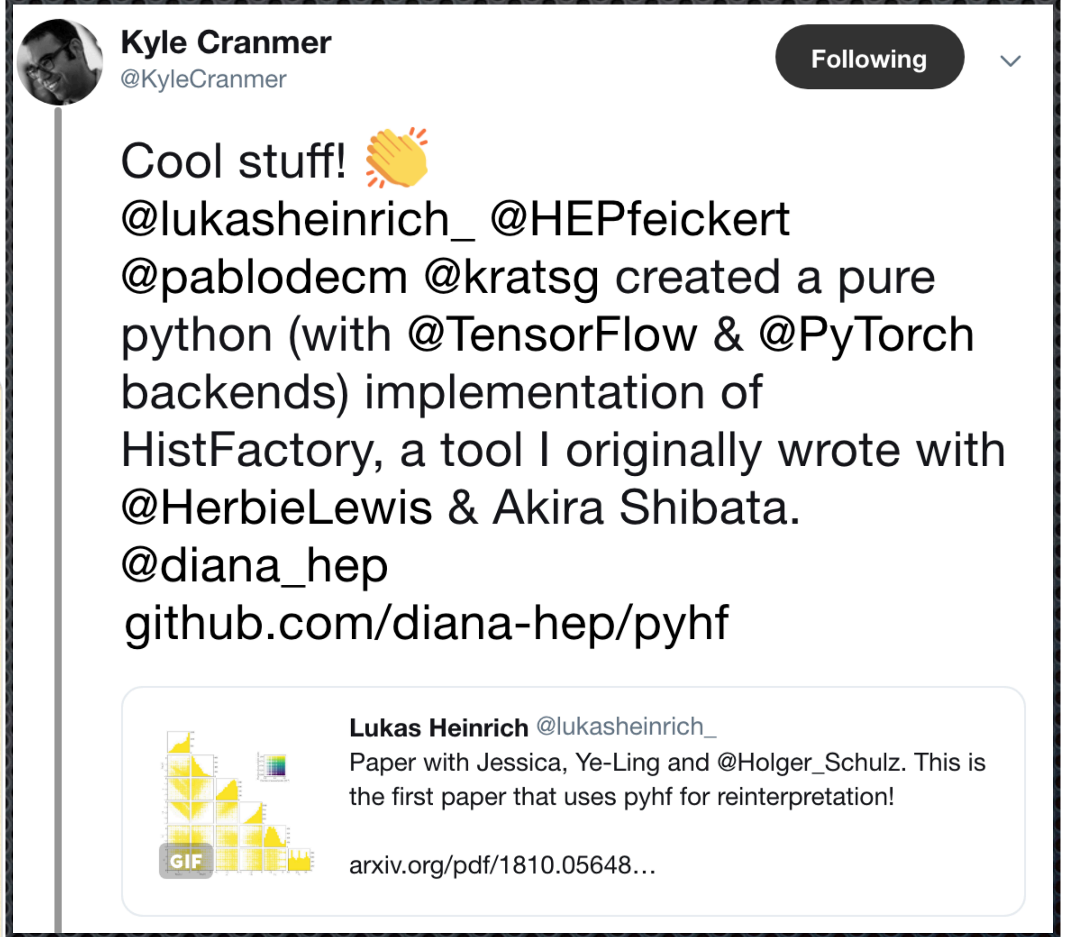
observed LLR (measurement)

BG only hypothesis (constant)

$$1 - CL_b \equiv \int_{-\infty}^{LLR_{obs}} f_0(LLR)dLLR \qquad CL_{s+b} \equiv \int_{LLR_{obs}}^{\infty} f_1(LLR)dLLR$$

$$CL_s = \frac{CL_{s+b}}{CL_b}$$

Frequentist is CL_{s+b} only



Kyle Cranmer @KyleCranmer Following

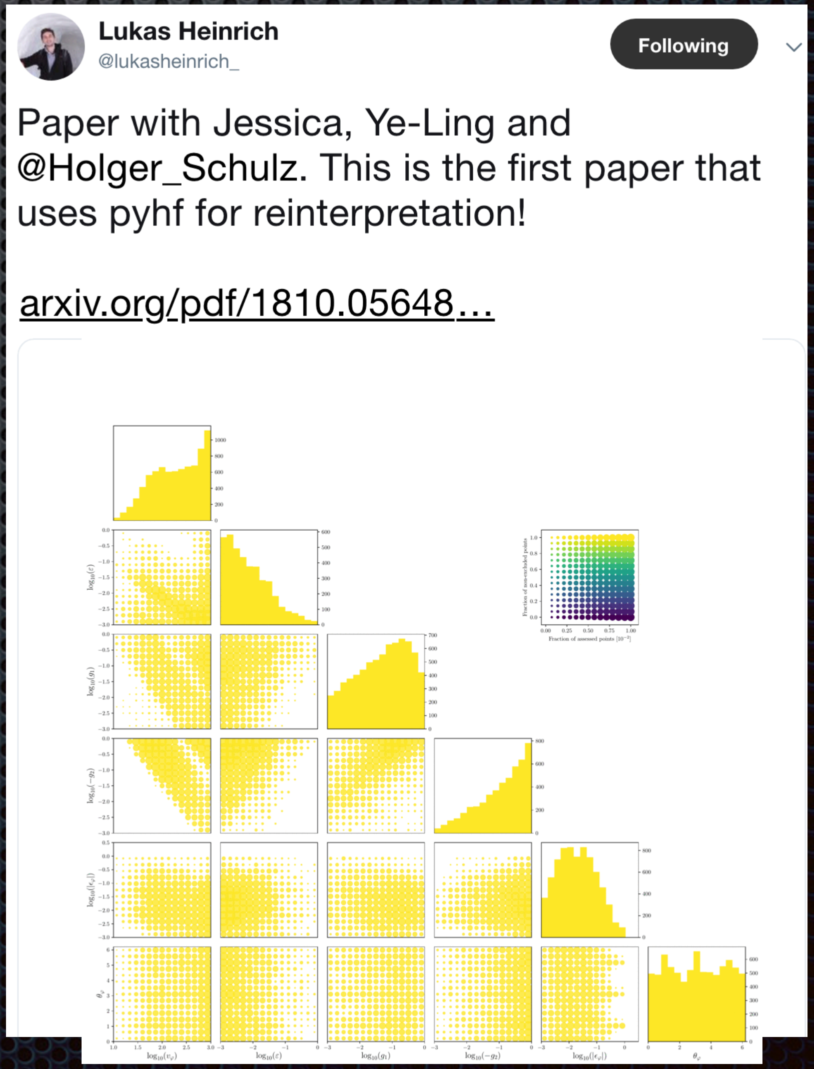
Cool stuff! 👏 @lukasheinrich_ @HEPfeickert @pablodecm @kratsg created a pure python (with @TensorFlow & @PyTorch backends) implementation of HistFactory, a tool I originally wrote with @HerbieLewis & Akira Shibata. @diana_hep github.com/diana-hep/pyhf

GIF

Lukas Heinrich @lukasheinrich_

Paper with Jessica, Ye-Ling and @Holger_Schulz. This is the first paper that uses pyhf for reinterpretation!

arxiv.org/pdf/1810.05648…

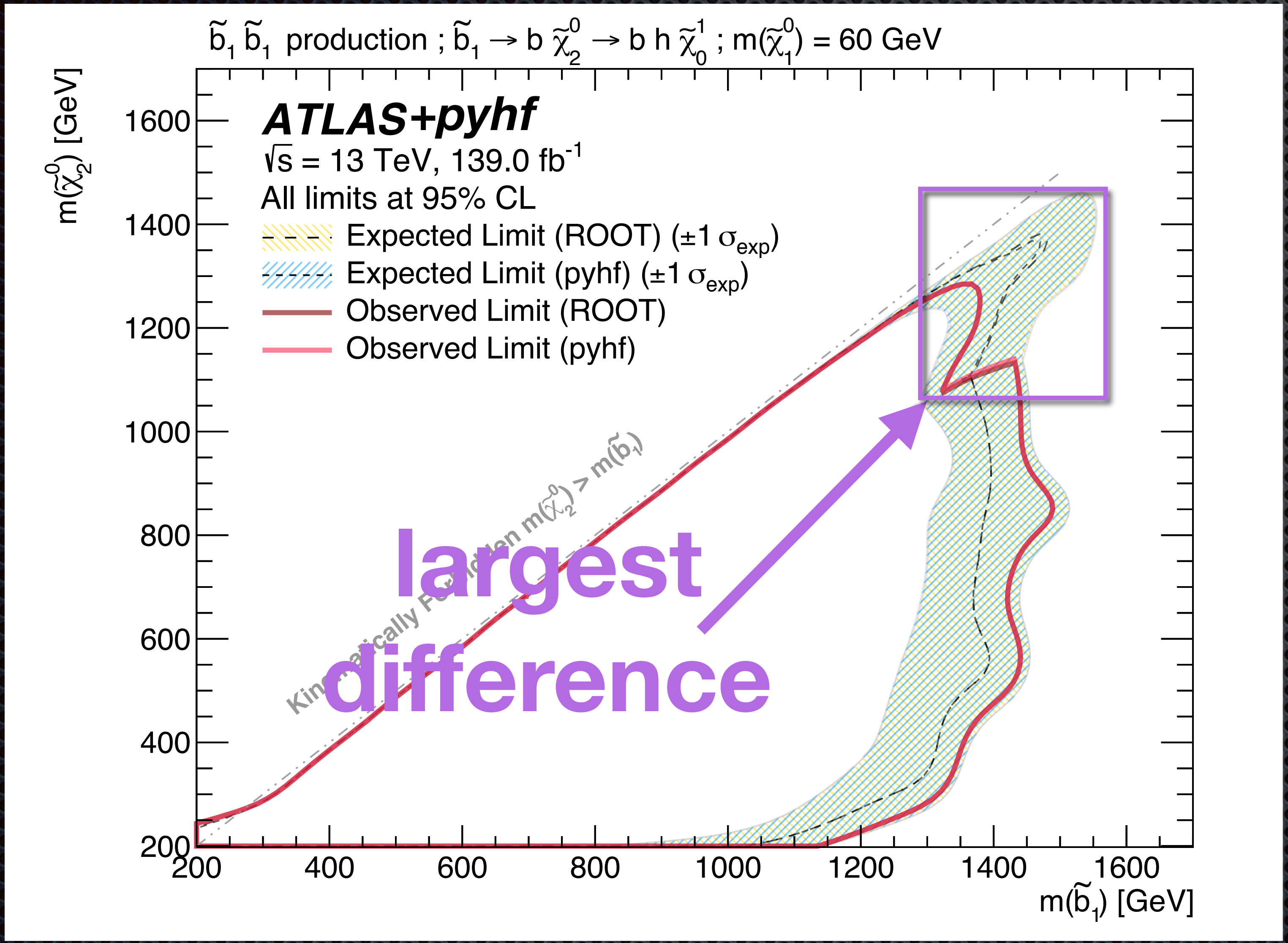


Lukas Heinrich @lukasheinrich_ Following

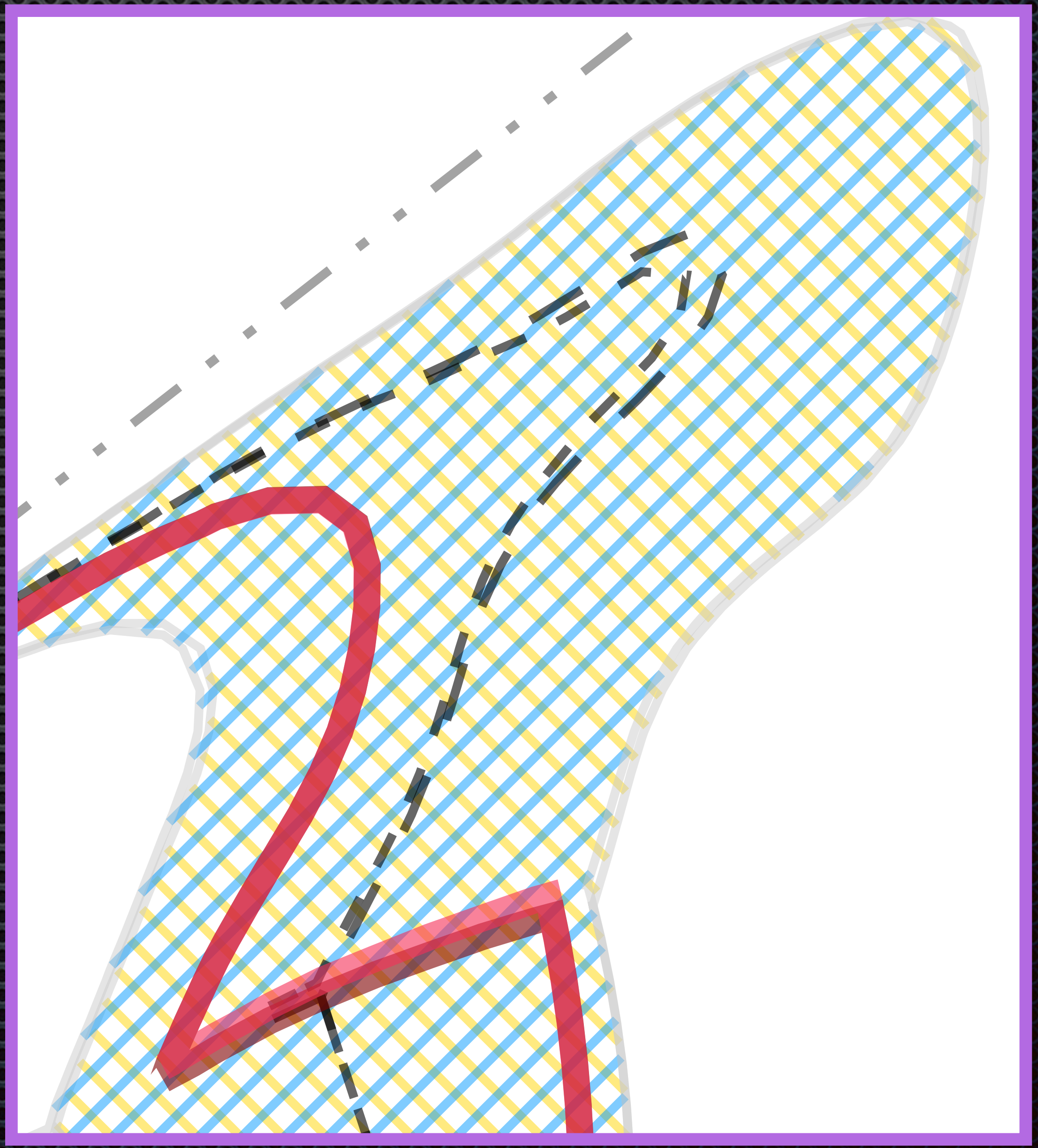
Paper with Jessica, Ye-Ling and @Holger_Schulz. This is the first paper that uses pyhf for reinterpretation!

arxiv.org/pdf/1810.05648…

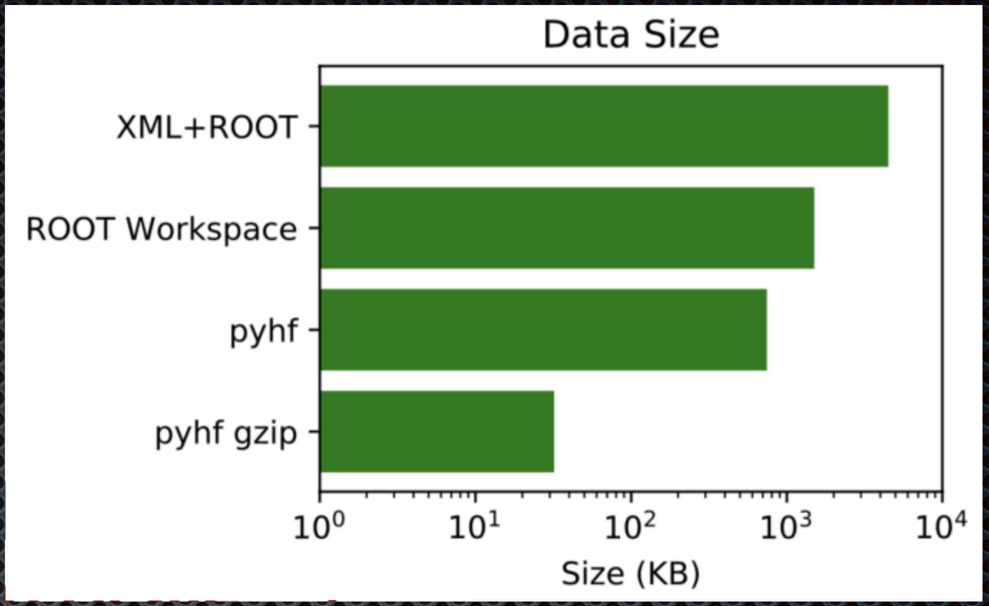
Preserving sbottom multi-b



largest difference



Conclusion



- pyhf provides **JSON specification of likelihoods**
  - plain-text format is advantageous for archivability and reusability
  - “HEPData”-friendly
- pyhf provides **bidirectional translation of likelihood specifications**
  - from ROOT workspaces to JSON: `xml2json`
  - from JSON to ROOT workspace: `json2xml + hist2workspace`
- pyhf provides **independent python-only implementation of HistFactory** + hypothesis testing
  - take advantage of industry-developed tools such as numpy and tensorflow

Connect with us on GitHub!