UC SANTA CRUZ

CERN

indico://e/782953/contributions/3462560/

Likelihood Preservation

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

DPF2019

July 30th, 2019

giordonstark.com

pyhf ∂ifferentiable 𝓛ikelihoods

ATLAS EXPERIMENT

Run: 300800

Event: 2418777995

2016-06-04 03:47:03 C

… alternative title…

"I want it that way"

2000

… alternative title…

"I want it that way"

2000

2019

Overview of today's talk

multi-bin histogram-based statistical fits

and how to preserve them

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

THE MAIN DEVELOPERS


G. Stark

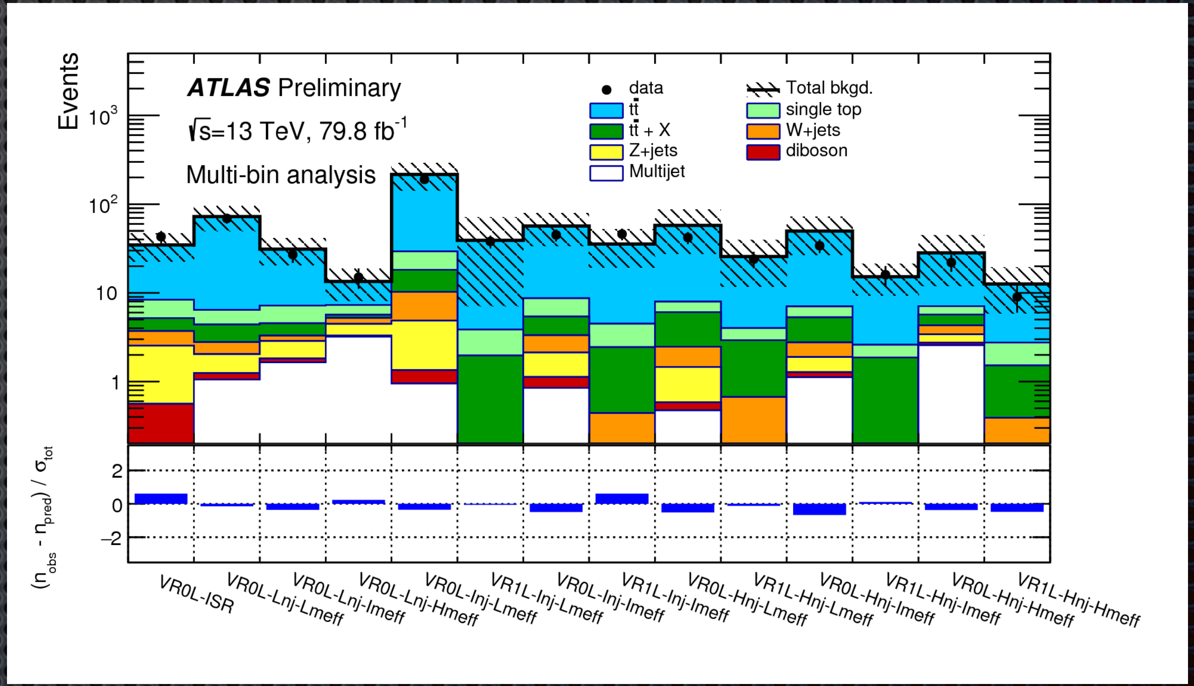

M. Feickert


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



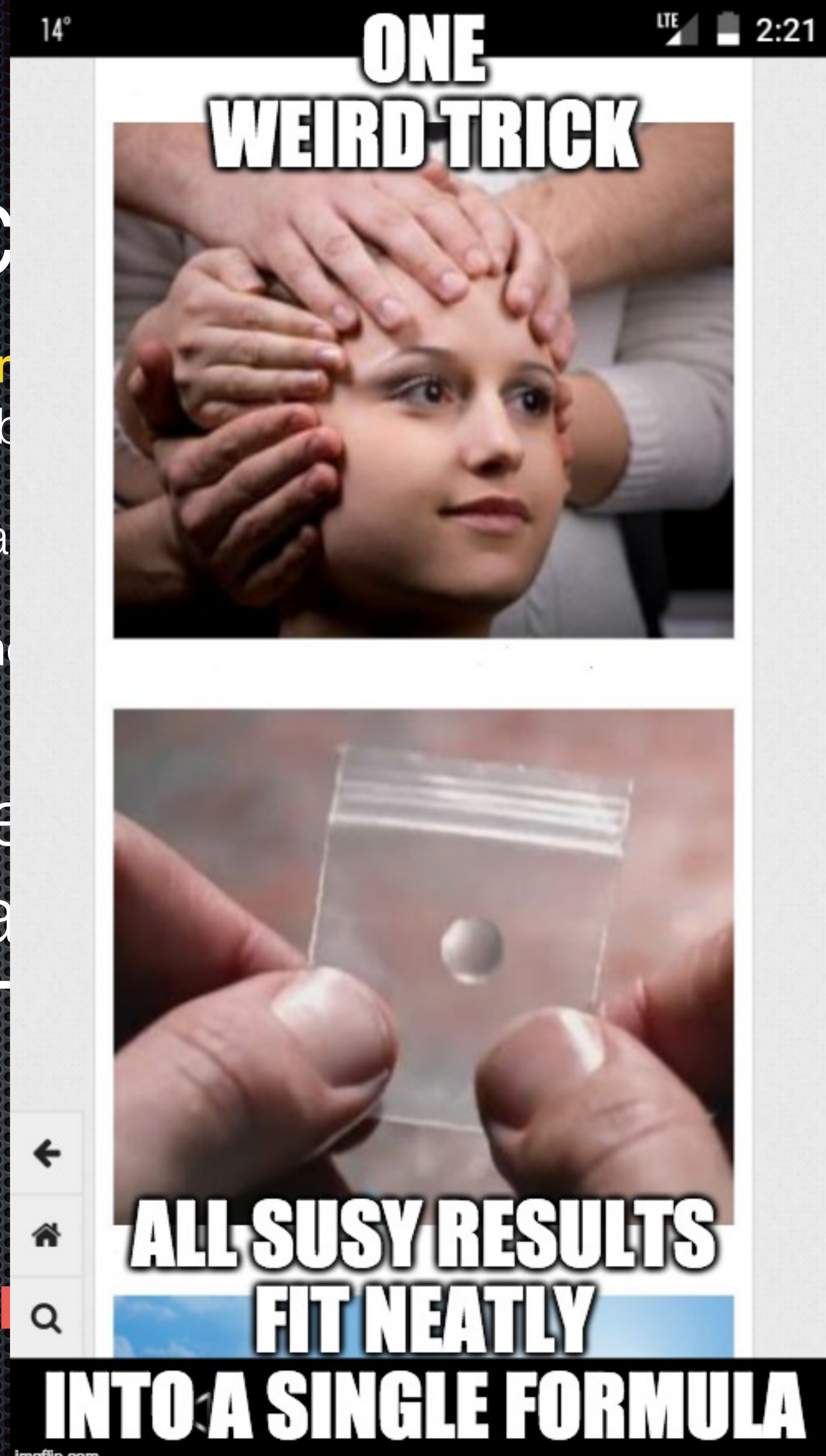
K. Cranmer

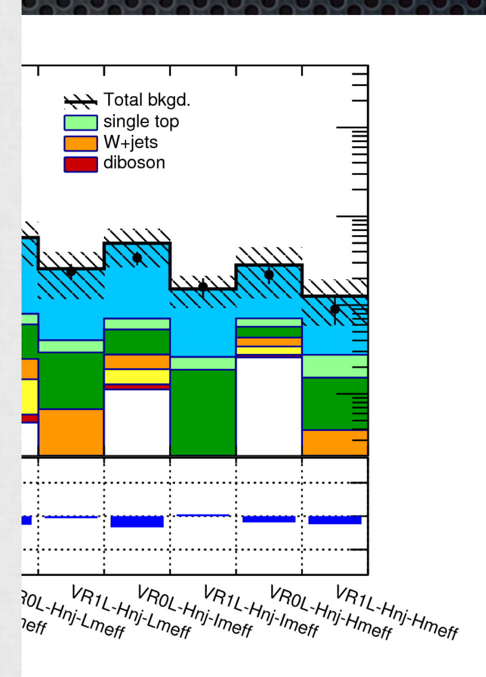
HistFactory is partially independent of its implementation in ROOT

HistFac EN-2012-016]

- A flexible **p.d.f ter** of statistical models from binned distrib
- Developed by Cra kerke
- Widely used by th measurements and BSM searches

Calculate
HistFa





ndent of its
OT

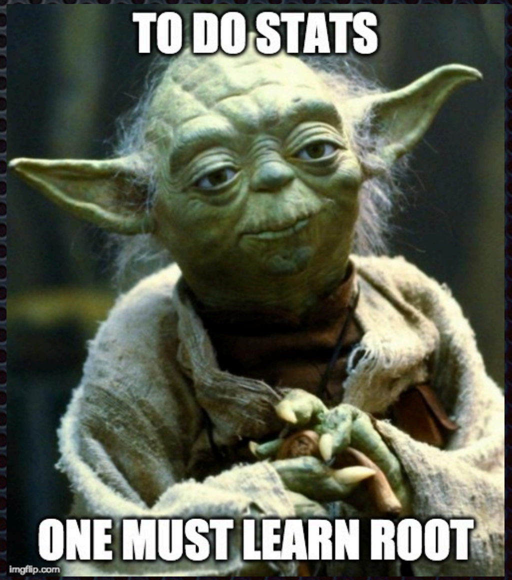
K. Cranmer

HistFactory? It's just math!

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

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

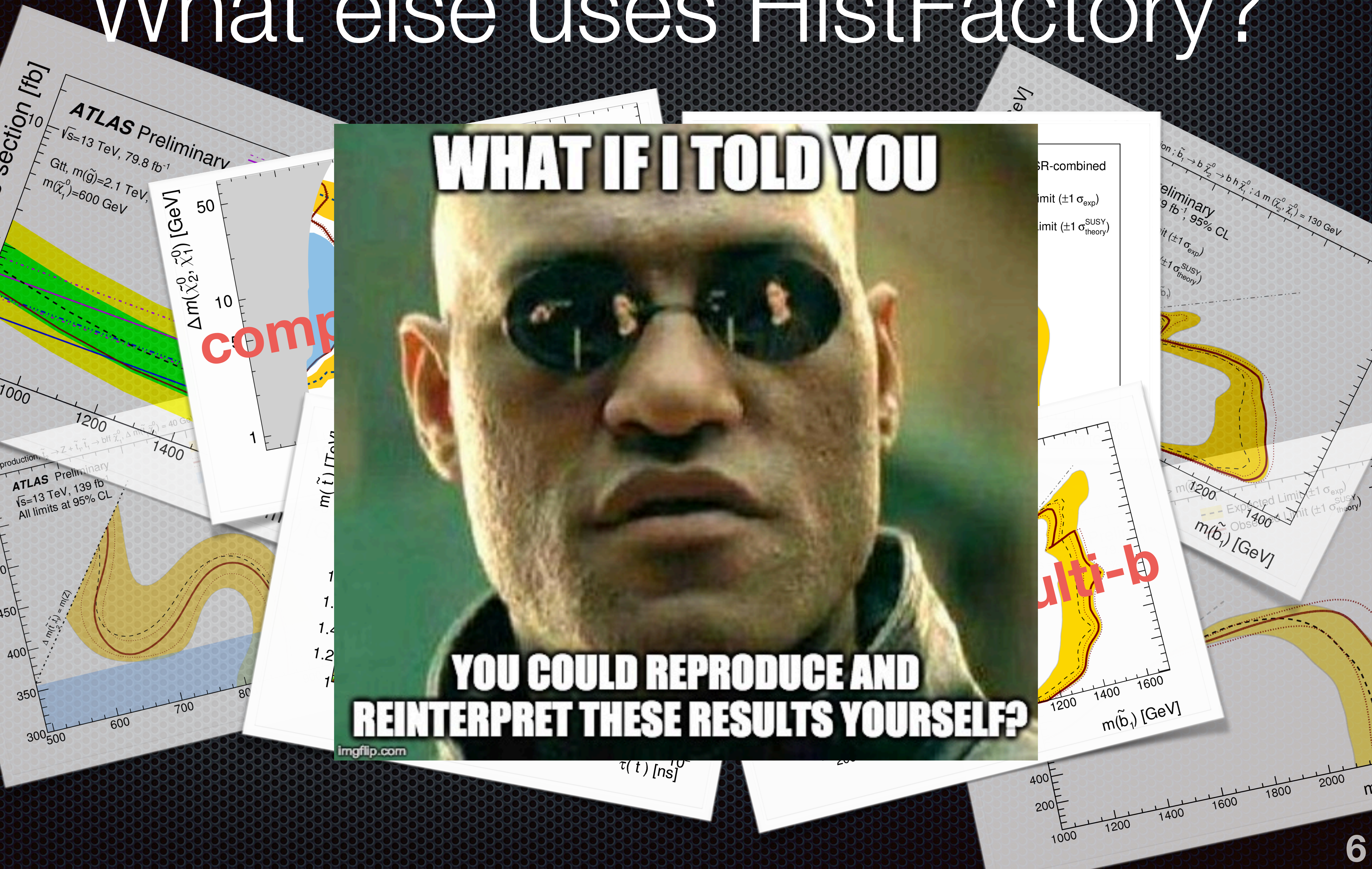
- An XML specification with data stored in ROOT files — it's been the ***only implementation*** of this calculation
  - **Poisson p.d.f.** for bins observed in all channels
  - **Constraint p.d.f.** (and data) for auxiliary measurements (systematics: normalization, shape, etc)
  - ⚠ Tied to ROOT ecosystem
  - ⚠ 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!



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

What else uses HistFactory?

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
```

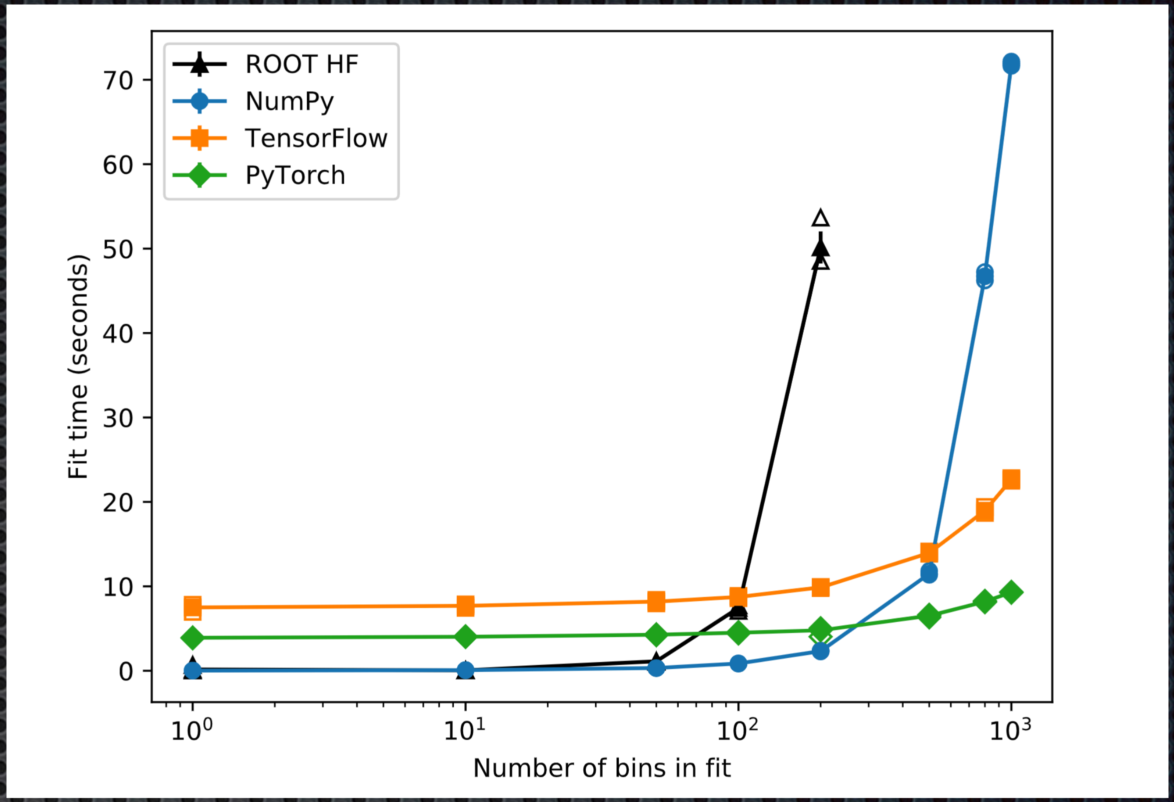
A **python-only** (scipy, numpy) implementation of the HistFactory model + profile likelihood hypothesis tests

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



NumPy

PyTorch TensorFlow

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
```

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

Demo (I) — 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" }]
            }
        ]
    }],
    "observations": [
        {"name": "singlechannel", "data": [51.0, 48.0]}
    ],
    "measurements": [{
            "config": { "poi": "mu", "parameters": [] },
            "name": "singlechannel"
    }],
    "version": "1.0.0"
}
```

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

```
$ curl -sL https://git.io/fj1yb | pyhf cls | jq .CLs_obs
0.052515541856109835
```

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

Demo (II) — 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" }]
            }
        ]
    }],
    "observations": [
        {"name": "singlechannel", "data": [51.0, 48.0]}
    ],
    "measurements": [{
            "config": { "poi": "mu", "parameters": [] },
            "name": "singlechannel"
    }],
    "version": "1.0.0"
}
```

- 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 (II) — Simple Re-use

```
$ curl -sL https://git.io/fj1yb | pyhf cls | jq .CLs_obs
0.052515541856109835

# reinterpretation time
$ curl -sL https://git.io/fj1yb | pyhf cls --patch <(curl -sL https://git.io/fj1yN)
| jq .CLs_obs
0.33650544273363076
```

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]
}]
```


ATLAS THE
REAL
WORLD
EXAMPLE

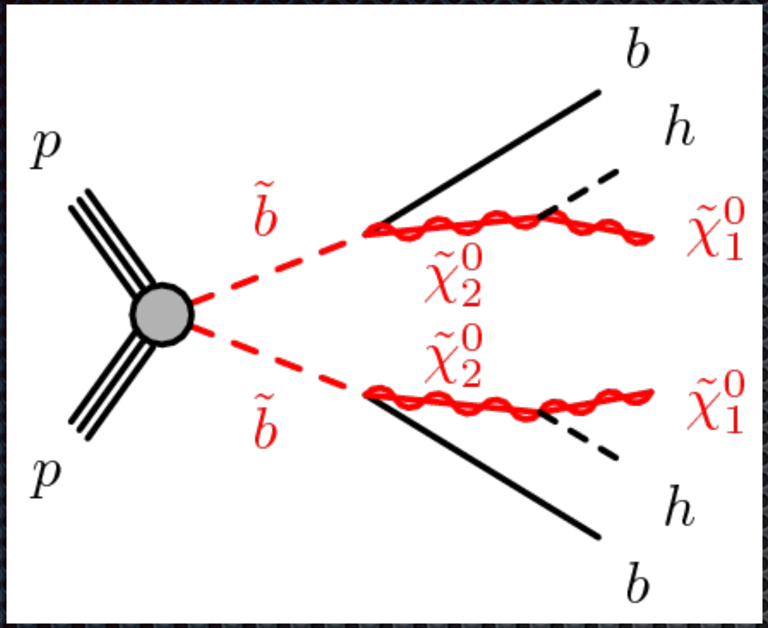
https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CONFNOTES/ATLAS-CONF-2019-011

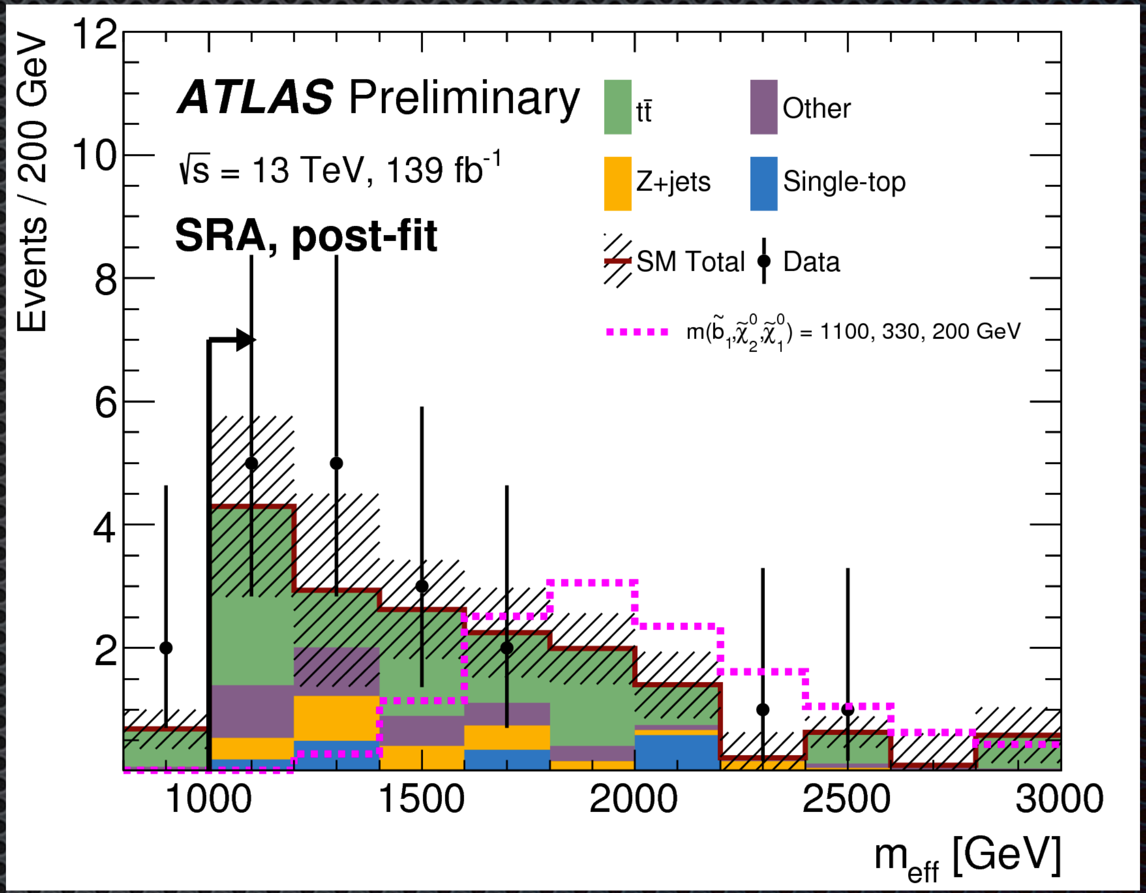
3G ($\tilde{t}$,$\tilde{b}$)

139 fb^{-1}

cut and count

sbottom multi-*b*-jets





- MODEL PARAMETERS: $\tilde{\chi}_2^0$, $\tilde{\chi}_1^0$, $\tilde{b}$
- FINAL STATE: up to 6 *b*-jets, MET, no leptons
- TWO SIGNAL MASS SCENARIOS:
  - $m(\tilde{\chi}_1^0)$ = 60 GeV
  - $\Delta m(\tilde{\chi}_2^0, \tilde{\chi}_1^0)$ = 130 GeV
- DOMINANT BACKGROUNDS: $t\bar{t}$, $Z \rightarrow \nu\bar{\nu}$
- CHALLENGE: reconstructing the Higgs bosons

- Analysis strategy:
  - 3 overlapping single-bin regions targeting (SRA) highly-boosted *b*-jets in "bulk" of both scenarios and (SRB,SRC) compressed with soft *b*-jets from $\tilde{b}$

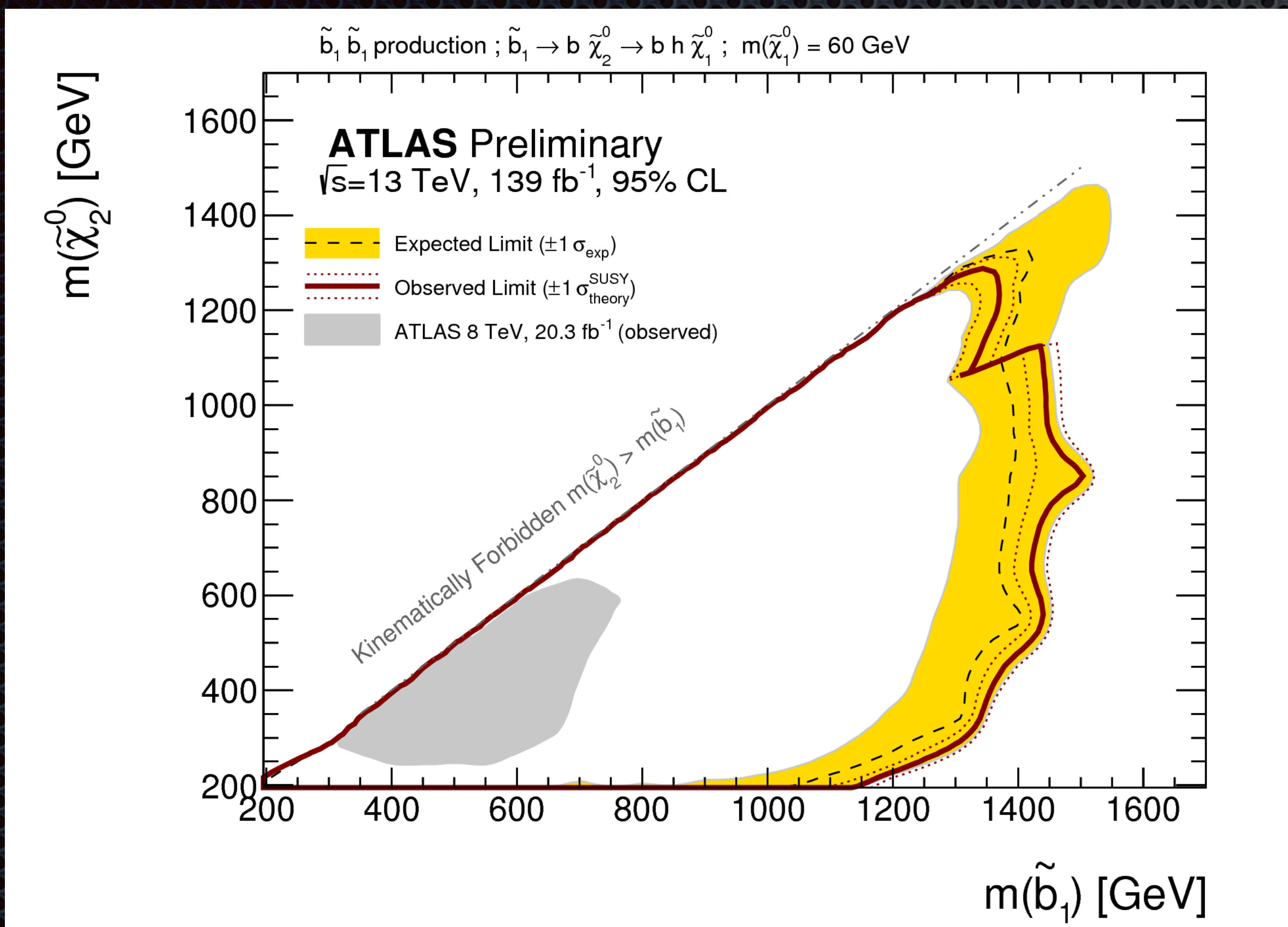
https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CONFNOTES/ATLAS-CONF-2019-011

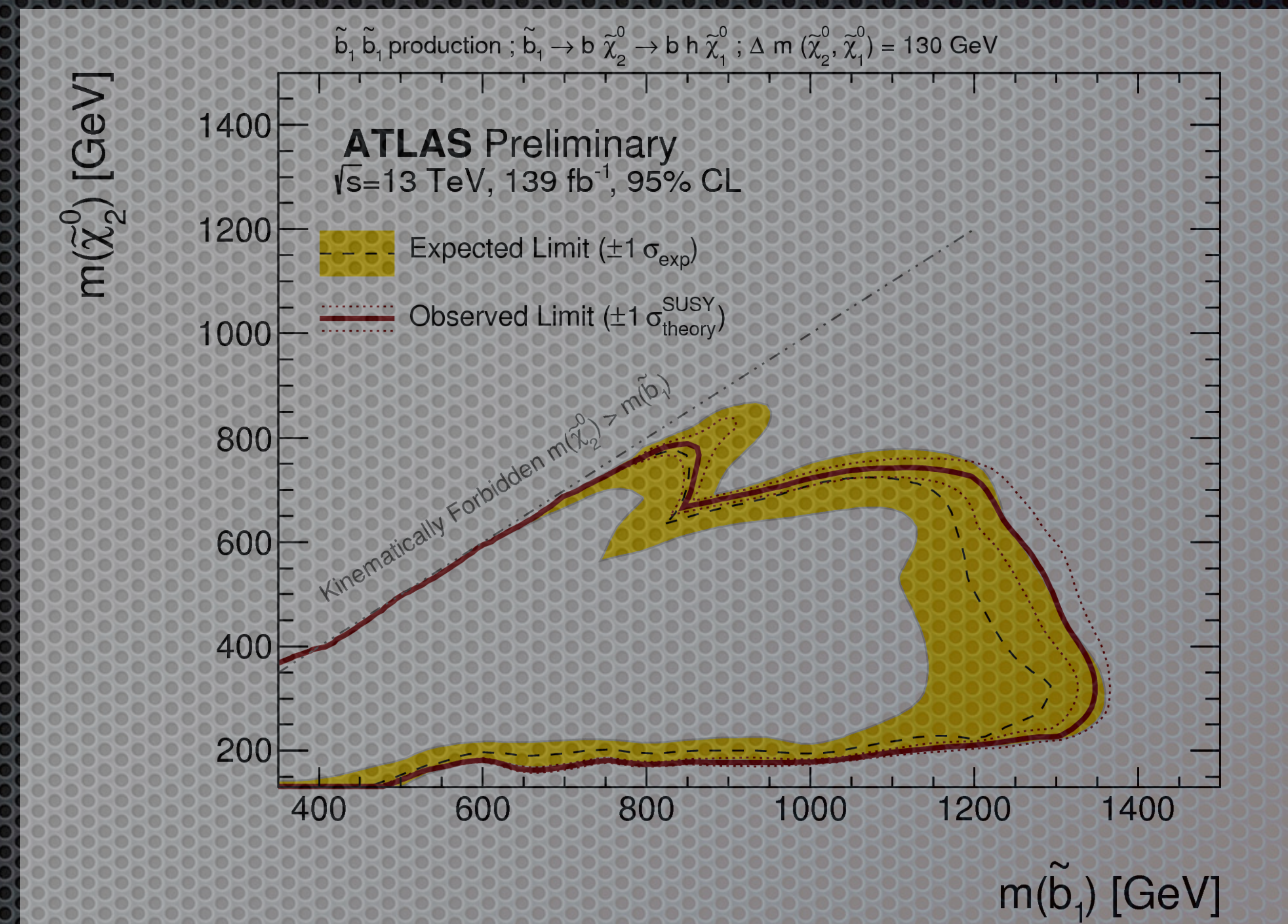
3G $(\tilde{t},\tilde{b})$

139 fb^{-1}

cut and count

sbottom multi-*b*-jets





$m(\tilde{\chi}_1{}^0) = 60$ GeV

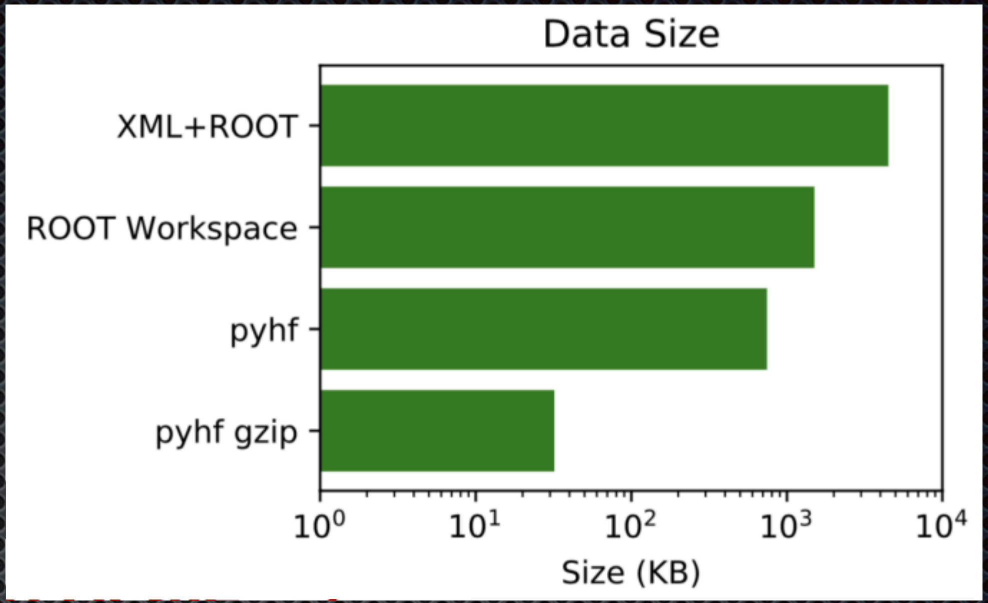
$\Delta m(\tilde{\chi}_2{}^0, \tilde{\chi}_1{}^0) = 130$ GeV

★ *new in Run 2*

Sensitivity increased to 1.45 TeV

HEPDATA DOWNLOAD LINK (NOT YET UPLOADED)

Preservation

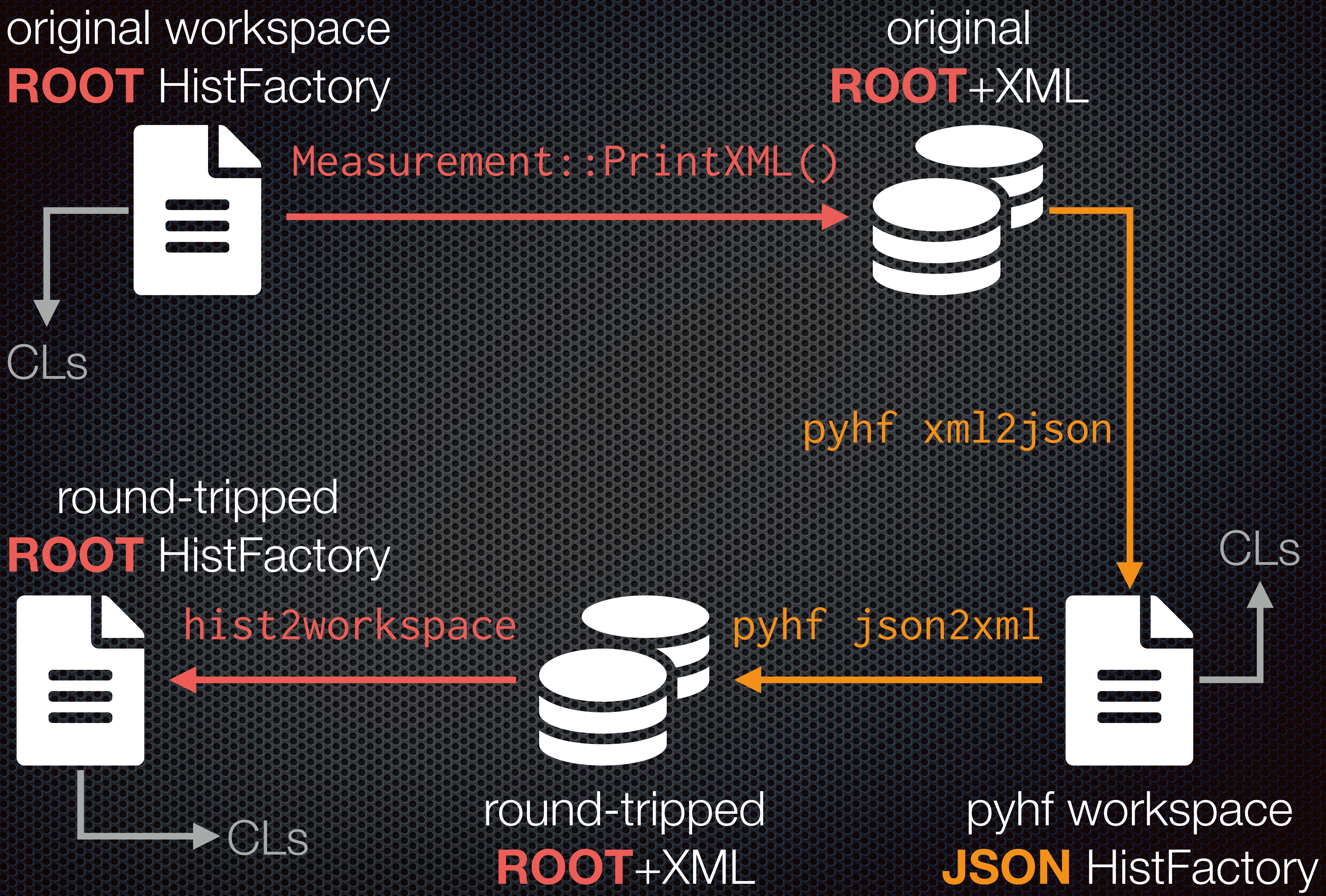


- tarball uploaded on https://www.hepdata.net/ **soon**!
- 3 background-only JSON workspaces: `RegionA`, `RegionB`, `RegionC`
- For each region, up to 218 jsonpatch files
  - reinterpret the background-only workspace in the context of a signal

```
-rw-r--r--   1 kratsg 862K Jul 16 11:48 BkgOnly.json
-rw-r--r--   1 kratsg  57K Jul 16 11:50 patch.sbottom_300_205_60.json
-rw-r--r--   1 kratsg  57K Jul 16 11:50 patch.sbottom_350_345_60.json
-rw-r--r--   1 kratsg  57K Jul 16 11:50 patch.sbottom_400_205_60.json
-rw-r--r--   1 kratsg  57K Jul 16 11:50 patch.sbottom_400_280_150.json
-rw-r--r--   1 kratsg  57K Jul 16 11:50 patch.sbottom_400_300_60.json
...
```

ATL-PHYS-PUB-2019-029

extremely automated process



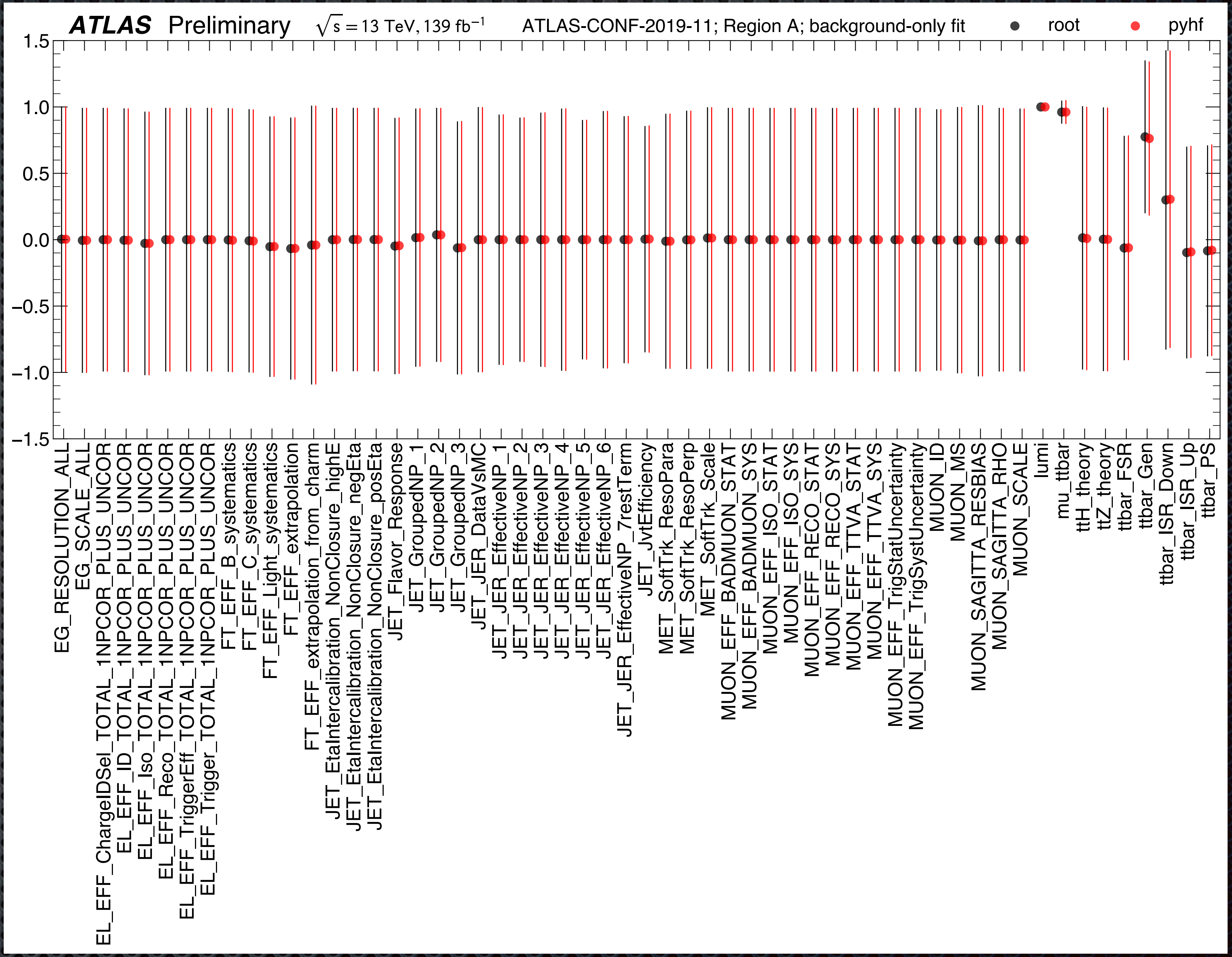
ATL-PHYS-PUB-2019-029

negate this

Background-only fit

L = ~~μ~~ S + B

systematic pulls →



Validate python implementation against ROOT

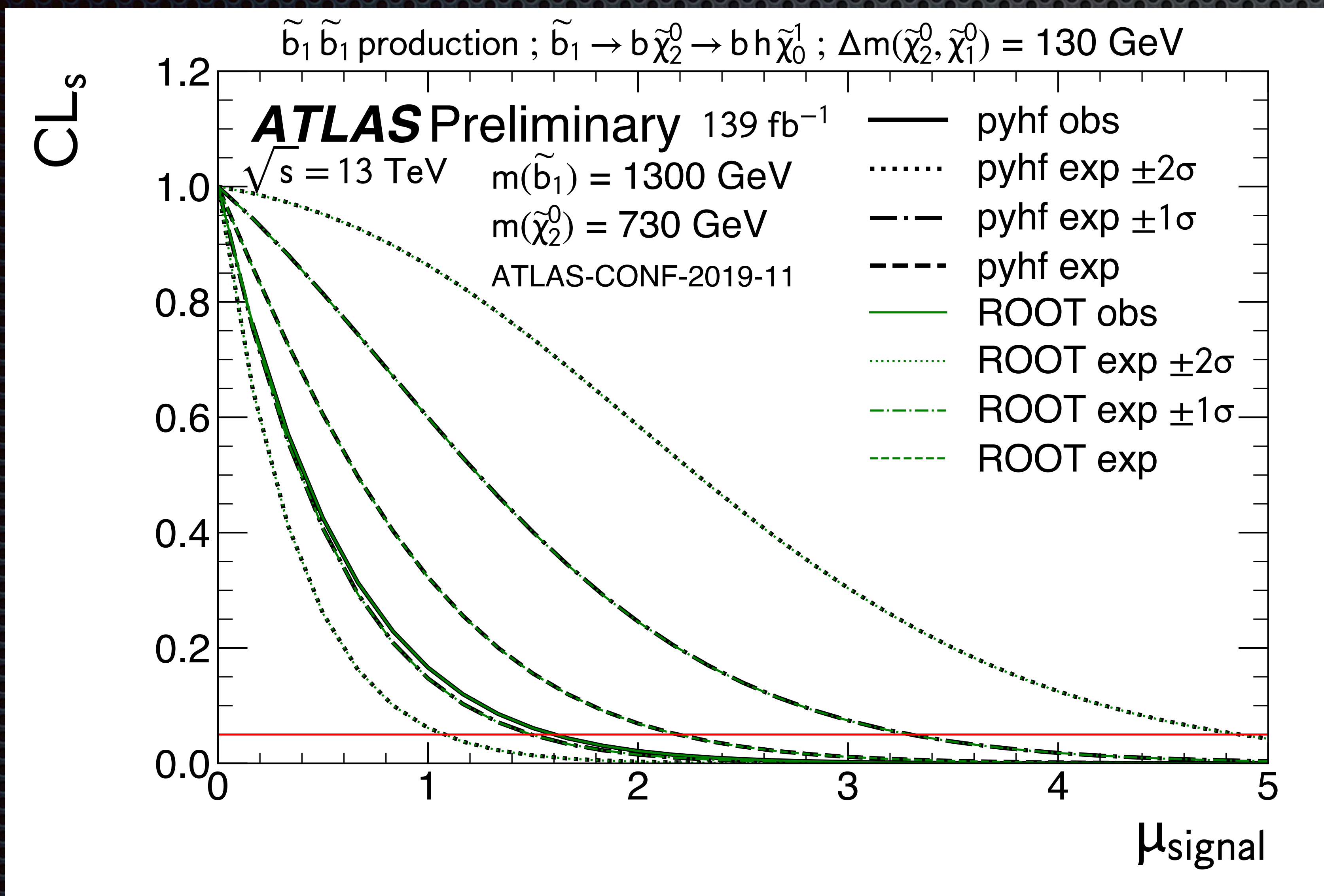
✔ **No large observed differences between ROOT and pyhf**

ATL-PHYS-PUB-2019-029

Signal strength scan

vary this

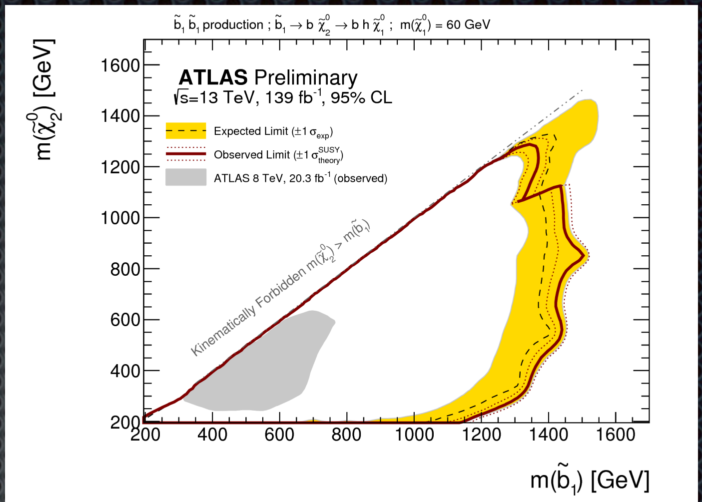
L = μ S + B

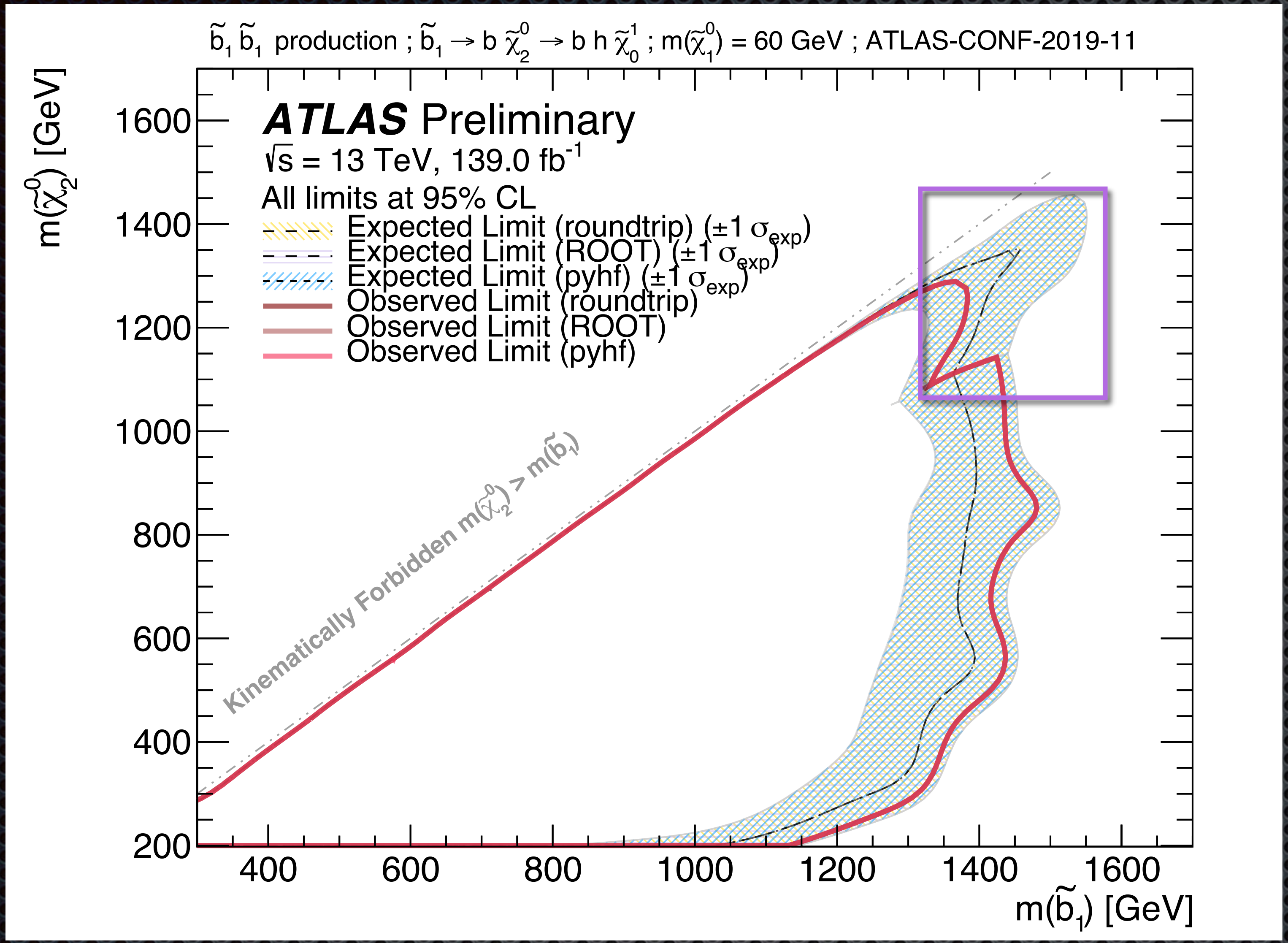


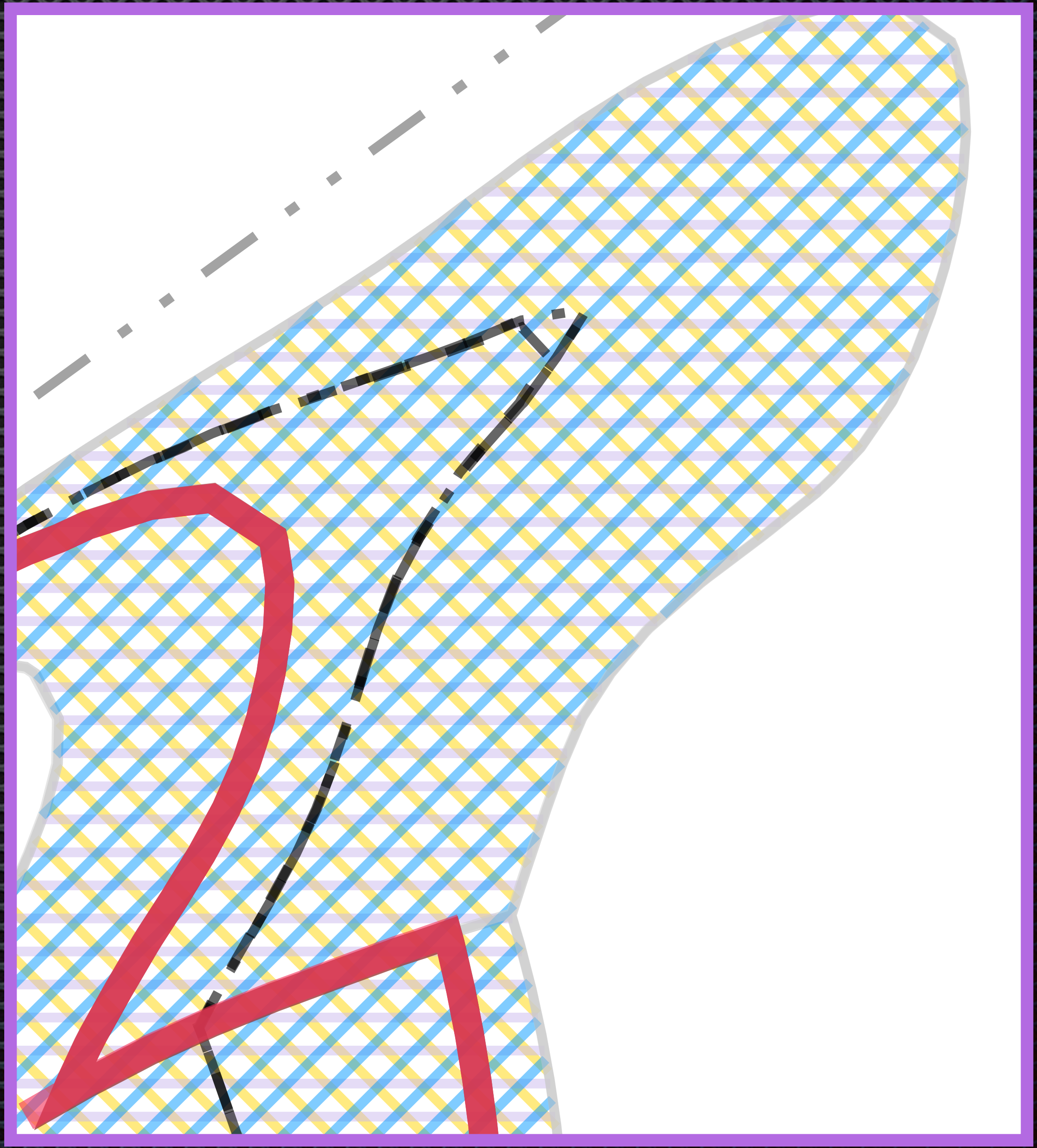
Compare likelihood minimization as a function of signal strength

✔ **No large observed differences between ROOT and pyhf**

Reproducing Results





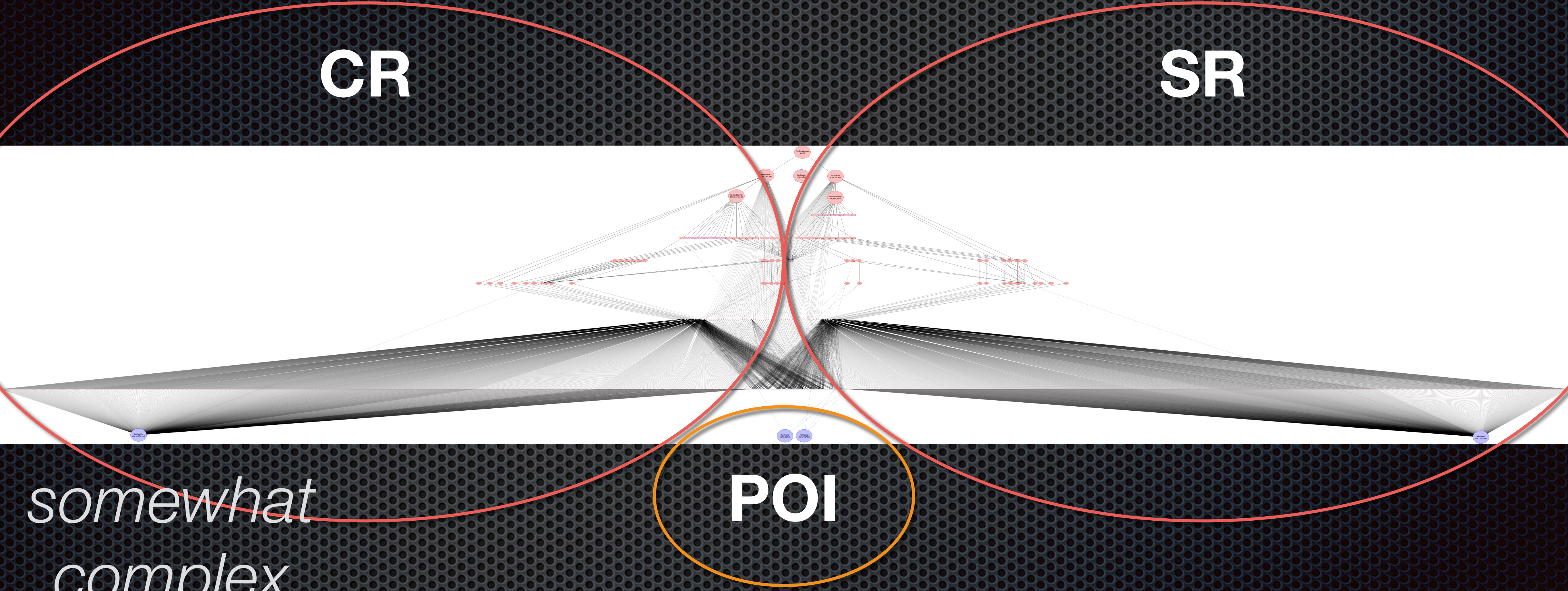


time to calculate every point in this plot

ROOT: 10+ hours
pyhf: 30 mins

✔ **No large observed differences between ROOT and pyhf**

Why the speed up? (I)

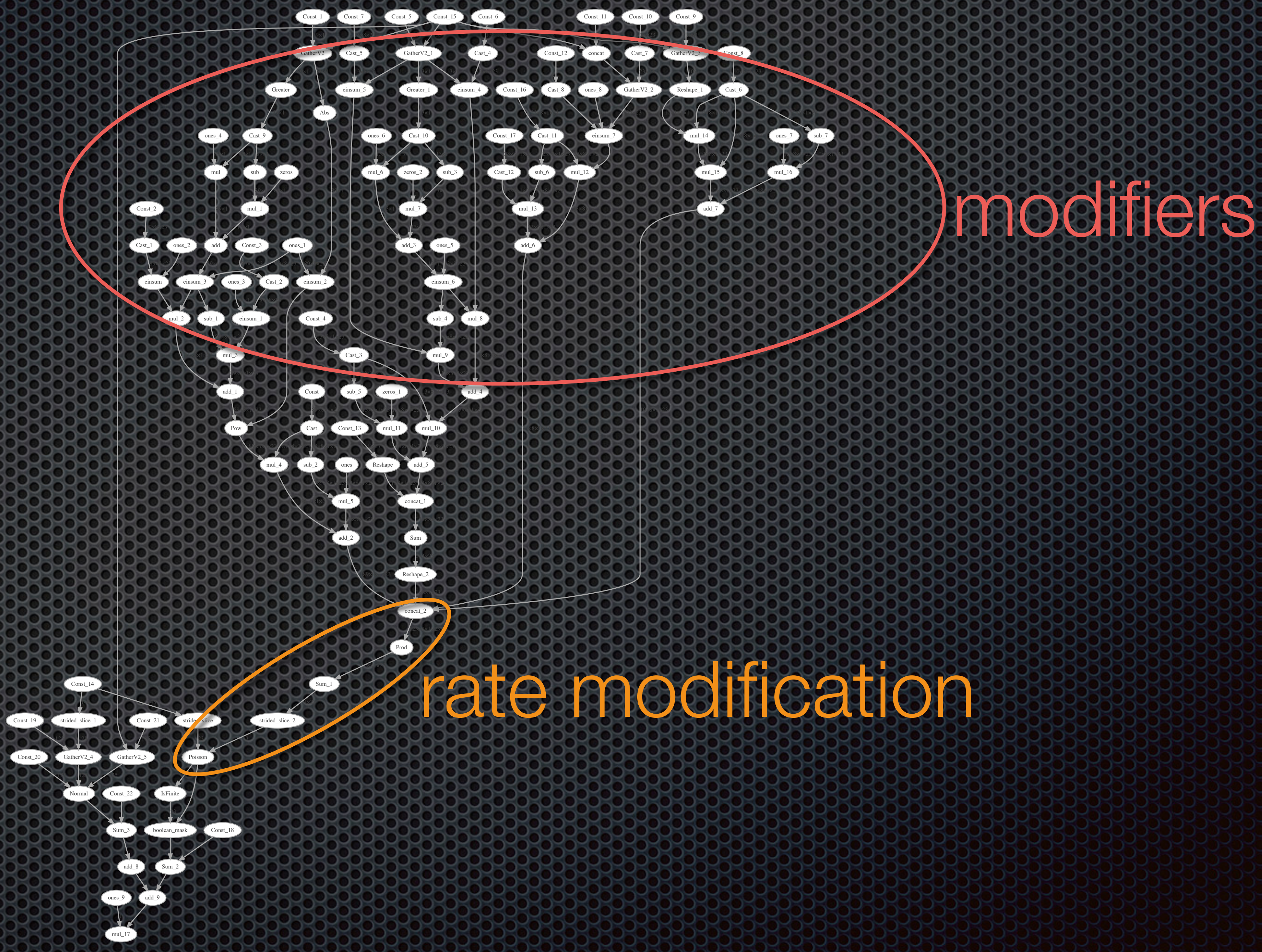


somewhat complex with lots of structure

ROOT-HistFactory computational tree

tensor dimensionality (rather than computational graph complexity) correlates with statistical model complexity

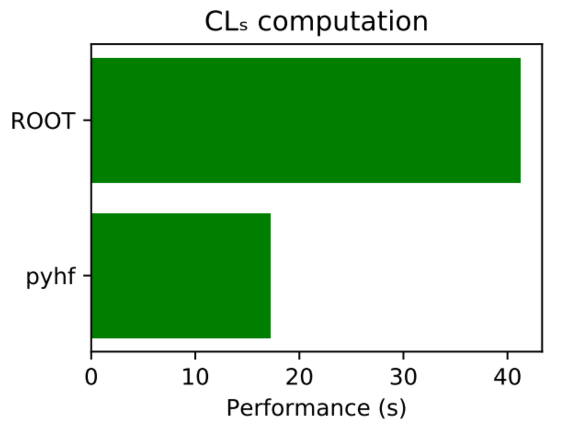
Why the speed up? (II)



somewhat simple but loss of structure

python-HistFactory computational tree

https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2019-029/
https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CONFNOTES/ATLAS-CONF-2019-011



Conclusion

ATLAS has made public the full likelihood of a search for new physics

- pyhf provides **JSON specification of likelihoods**
  - plain-text format is advantageous for archivability and reusability
  - "HEPData"-friendly
  - JSONPatch provides a clear target for analyses implementing RECAST
- 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!

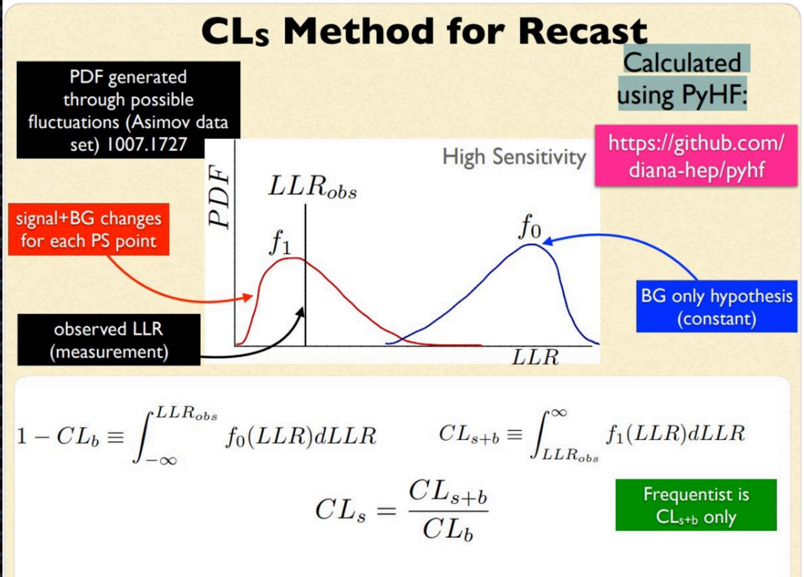
Other uses of pyhf in the wild

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/ …



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

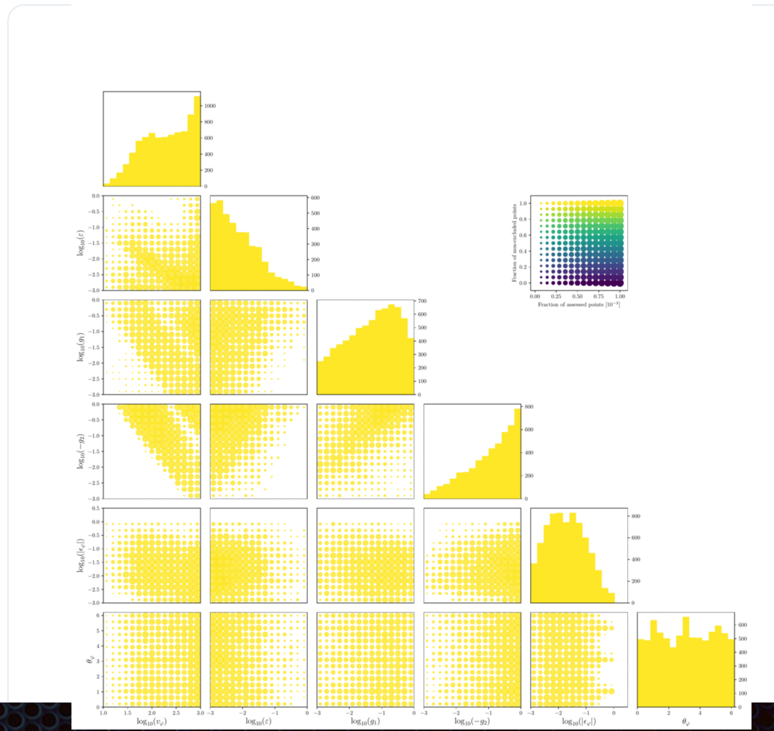
Lukas Heinrich @lukasheinrich_
Paper with Jessica, Ye-Ling and @Holger_Schulz. This is the first paper that uses pyhf for reinterpretation!

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…



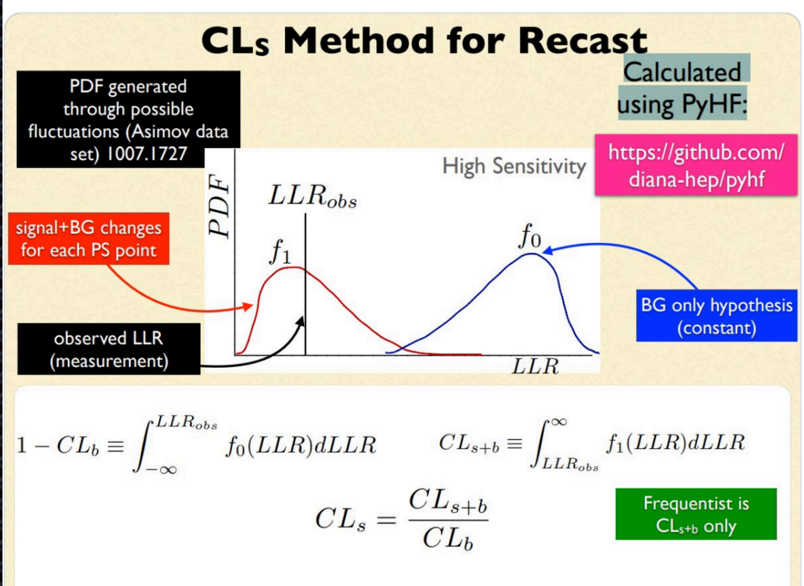
Other uses of pyhf in the wild

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/ ...



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

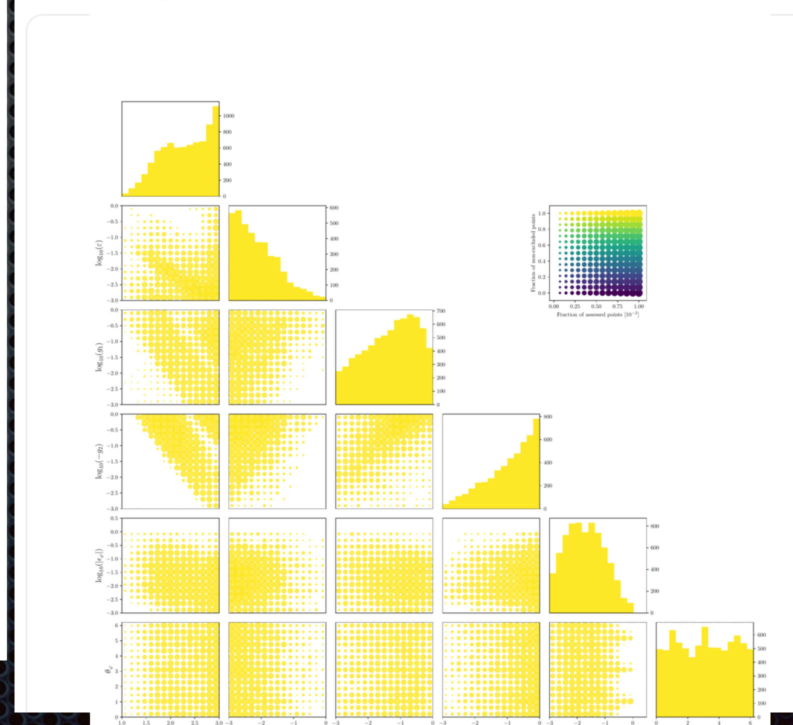
Lukas Heinrich @lukasheinrich_
Paper with Jessica, Ye-Ling and @Holger_Schulz. This is the first paper that uses pyhf for reinterpretation!

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…



notation/terminology

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

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

	Symbol	Name
	$f(\boldsymbol{x} \mid \boldsymbol{\phi})$	model
	$\mathcal{L}(\boldsymbol{\phi})$	likelihood
data	$\boldsymbol{x} = \{\boldsymbol{n}, \boldsymbol{a}\}$	full dataset (including auxiliary data)
	$\boldsymbol{n}$	channel data (or event counts)
	$\boldsymbol{a}$	auxiliary data
	$\nu(\boldsymbol{\phi})$	calculated event rates
parameters	$\boldsymbol{\phi} = \{\boldsymbol{\eta}, \boldsymbol{\chi}\} = \{\boldsymbol{\psi}, \boldsymbol{\theta}\}$	all parameters
	$\boldsymbol{\eta}$	free parameters
	$\boldsymbol{\chi}$	constrained parameters
	$\boldsymbol{\psi}$	parameters of interest
	$\boldsymbol{\theta}$	nuisance parameters
rate modifiers	$\boldsymbol{\kappa}(\boldsymbol{\phi})$	multiplicative rate modifier
	$\boldsymbol{\Delta}(\boldsymbol{\phi})$	additive rate modifiers
	$c_{\chi}(a_{\chi} \mid \chi)$	constraint term for constrained parameter χ
	σ_{χ}	relative uncertainty in the constrained parameter

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

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

	Description	Modification	Constraint Term c_χ	Input
constrained	Uncorrelated Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Pois}\left(r_b = \sigma_b^{-2} \middle| \rho_b = \sigma_b^{-2}\gamma_b\right)$	σ_b
	Correlated Shape	$\Delta_{scb}(\alpha) = f_p\left(\alpha \middle| \Delta_{scb,\alpha=-1}, \Delta_{scb,\alpha=1}\right)$	$\text{Gaus}\left(a = 0 \middle| \alpha, \sigma = 1\right)$	$\Delta_{scb,\alpha=\pm 1}$
	Normalisation Unc.	$\kappa_{scb}(\alpha) = g_p\left(\alpha \middle| \kappa_{scb,\alpha=-1}, \kappa_{scb,\alpha=1}\right)$	$\text{Gaus}\left(a = 0 \middle| \alpha, \sigma = 1\right)$	$\kappa_{scb,\alpha=\pm 1}$
	MC Stat. Uncertainty	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Gaus}\left(a_{\gamma_b} = 1 \middle| \gamma_b, \delta_b\right)$	$\delta_b^2 = \sum_s \delta_{sb}^2$
	Luminosity	$\kappa_{scb}(\lambda) = \lambda$	$\text{Gaus}\left(l = \lambda_0 \middle| \lambda, \sigma_\lambda\right)$	$\lambda_0, \sigma_\lambda$
free	Normalisation	$\kappa_{scb}(\mu_b) = \mu_b$		
	Data-driven Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$		

types of modifiers

Bkg-only yields

ROOT

	Model A			Model B	Model C			
	SRA-L	SRA-M	SRA-H	SRB	SRC22	SRC24	SRC26	SRC28
Observed events	12.00	3.00	2.00	3.00	28.00	12.00	4.00	3.00
Fitted SM bkg events	8.35	5.66	3.01	3.29	20.87	10.29	3.95	2.45
$t\bar{t}$	4.77	3.69	1.73	2.31	3.89	1.08	0.34	0.12
Z+jets	1.21	0.84	0.41	0.28	8.50	5.73	1.92	1.08
Single+top	0.43	0.33	0.59	0.48	2.70	1.21	0.68	0.44
$t\bar{t} + W/Z$	0.73	0.33	0.12	0.08	2.52	1.01	0.52	0.25
$t\bar{t} + h$	0.65	0.33	0.08	0.12	0.16	0.04	0.08	0.00
W+jets	0.22	0.13	0.04	0.02	2.16	0.63	0.24	0.42
Diboson	0.34	0.00	0.04	0.00	0.94	0.58	0.17	0.13

pyhf

	Model A			Model B	Model C			
	SRA-L	SRA-M	SRA-H	SRB	SRC22	SRC24	SRC26	SRC28
Observed events	12.00	3.00	2.00	3.00	28.00	12.00	4.00	3.00
Fitted SM bkg events	8.37	5.66	3.01	3.30	20.85	10.28	3.95	2.45
$t\bar{t}$	4.79	3.70	1.73	2.31	3.88	1.08	0.34	0.12
Z+jets	1.20	0.84	0.41	0.28	8.49	5.72	1.92	1.08
Single+top	0.43	0.33	0.58	0.48	2.71	1.22	0.68	0.44
$t\bar{t} + W/Z$	0.73	0.33	0.12	0.08	2.52	1.01	0.52	0.25
$t\bar{t} + h$	0.65	0.33	0.08	0.12	0.16	0.04	0.08	0.00
W+jets	0.22	0.13	0.04	0.02	2.16	0.63	0.24	0.42
Diboson	0.34	0.00	0.04	0.00	0.94	0.59	0.17	0.13