pyhf

Matthew Feickert, Lukas Heinrich, Giordon Stark

pyhf

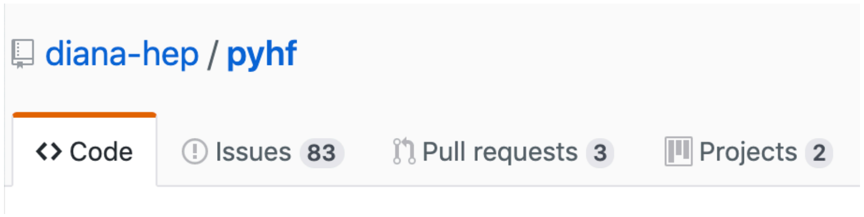
- Simultaneous binned fit to multiple channels, each with multiple samples.
- Sample yields estimated function of nominal rate, scale factors and systematics.
- Systematics imply constraint term on the pdf.

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

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

- but so far only implementation in ROOT

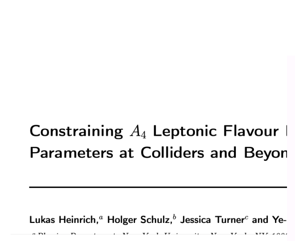
pyhf


diana-hep / pyhf

∂ifferentiable
ℒikelihoods

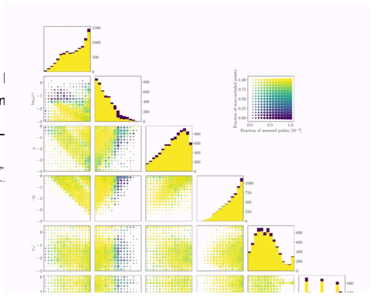
High Level Goals of pyhf:

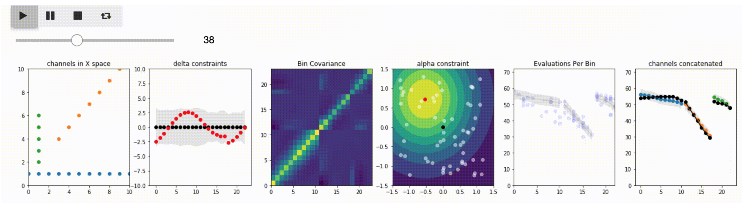
- Python-based implementation
  - unlock python / data science eco-system
    - (new systematic types Gaussian Processes)
    - differentiable formulation
  - performance
  - lower barrier of entry to use HistFactory (e.g. phenomenologists..)


FERMILAB-PUB-18-549-T
NSF ACI-1450310
PHY-1505463

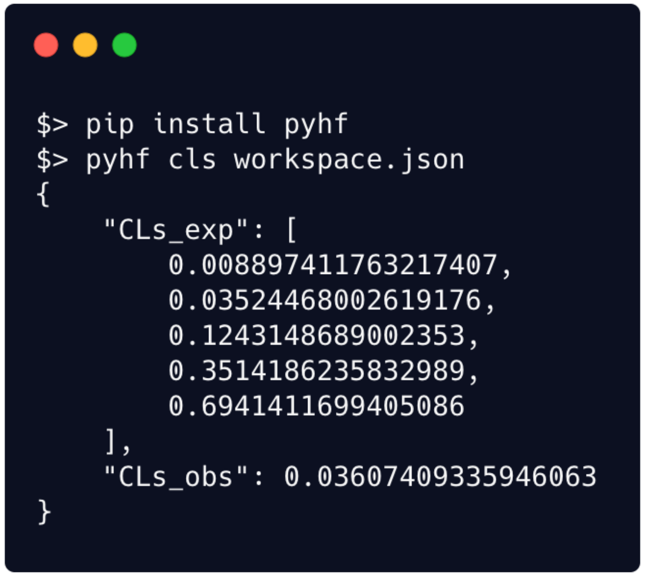
Constraining A_4 Leptonic Flavour
Parameters at Colliders and Beyon

Lukas Heinrich,[a] Holger Schulz,[b] Jessica Turner[c] and Ye-

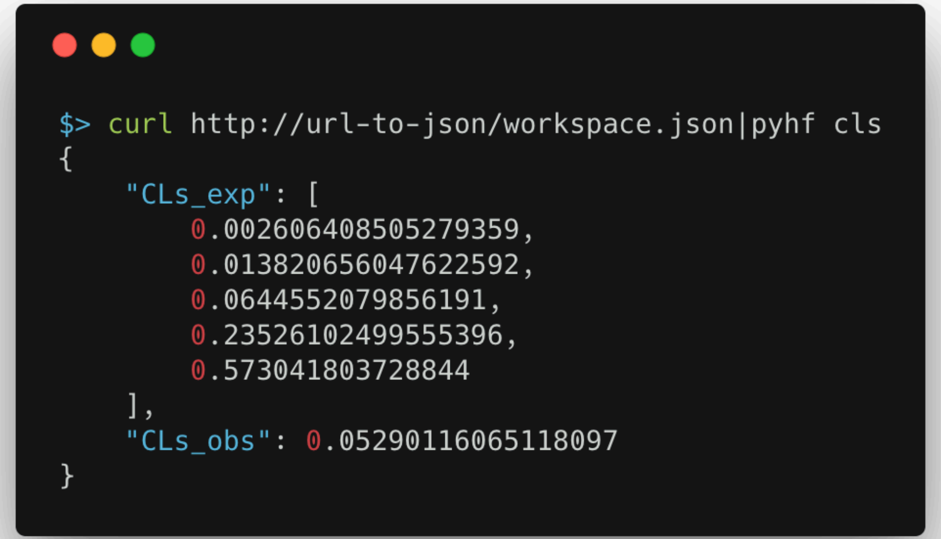




- Likelihood Preservation
  - side benefit: find language-independent spec
  - **likelihoods more important data product of an analysis**



```
$> pip install pyhf
$> pyhf cls workspace.json
{
    "CLs_exp": [
        0.008897411763217407,
        0.03524468002619176,
        0.1243148689002353,
        0.3514186235832989,
        0.6941411699405086
    ],
    "CLs_obs": 0.03607409335946063
}
```

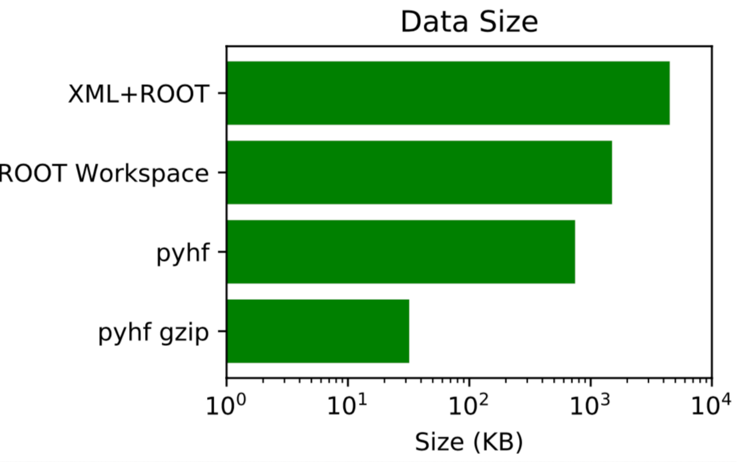


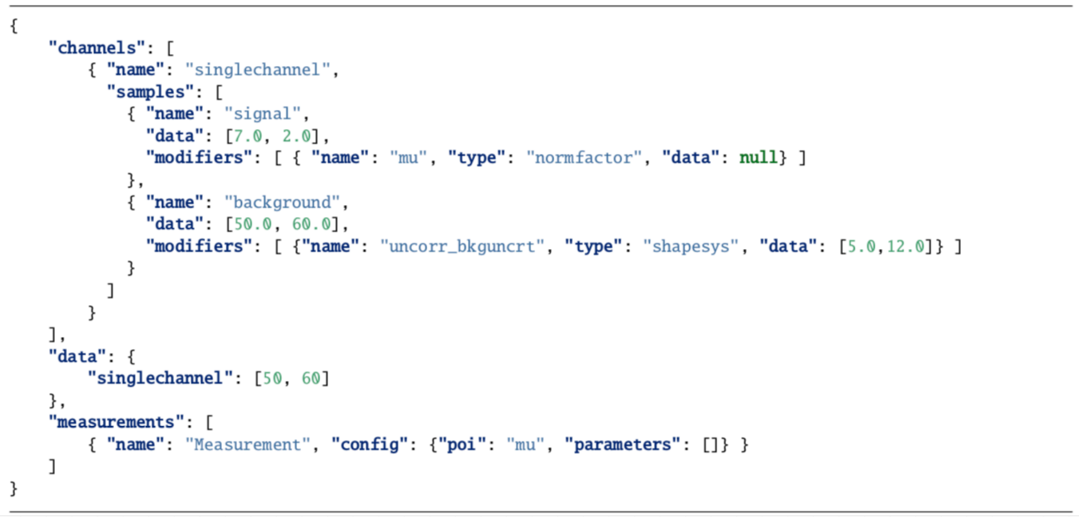
```
$> curl http://url-to-json/workspace.json|pyhf cls
{
    "CLs_exp": [
        0.002606408505279359,
        0.013820656047622592,
        0.0644552079856191,
        0.23526102499555396,
        0.573041803728844
    ],
    "CLs_obs": 0.05290116065118097
}
```

JSON Format:

Idea: remove "split brain" from XML + ROOT and inline all data into a single JSON document. For binned data, this should be fine.

(Should be find for very large binned likelihoods, but can use pointers into external storage if needed)



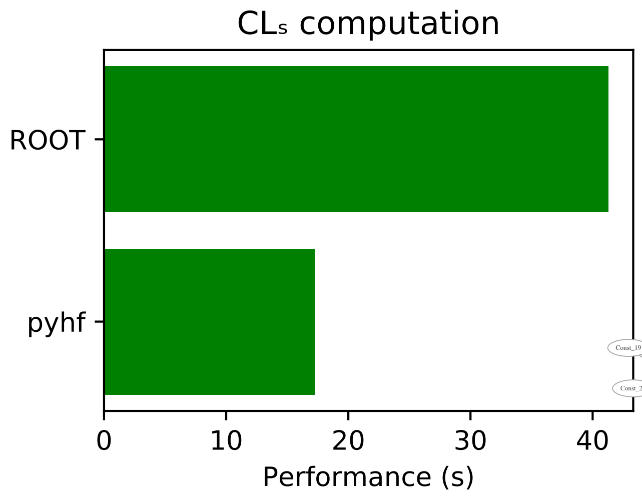


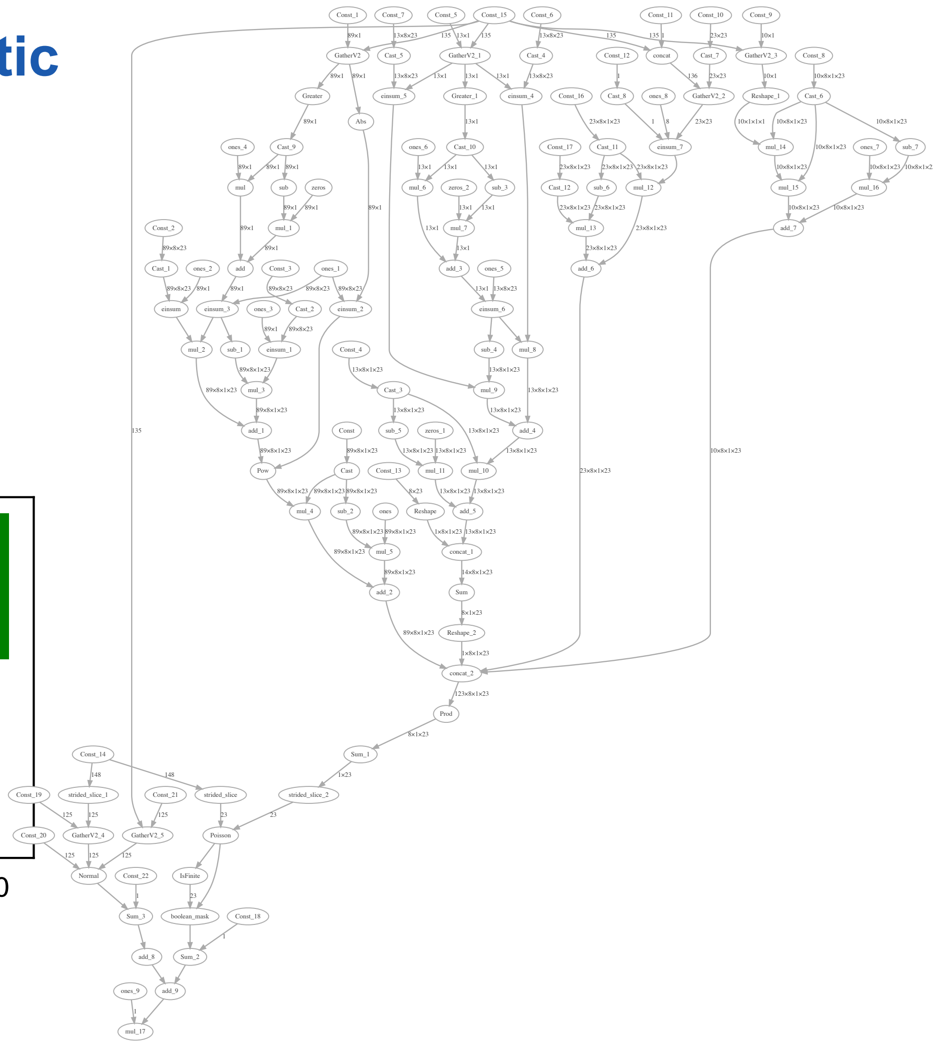
```
{
    "channels": [
        { "name": "singlechannel",
          "samples": [
            { "name": "signal",
              "data": [7.0, 2.0],
              "modifiers": [ { "name": "mu", "type": "normfactor", "data": null} ]
            },
            { "name": "background",
              "data": [50.0, 60.0],
              "modifiers": [ {"name": "uncorr_bkguncrt", "type": "shapesys", "data": [5.0,12.0]} ]
            }
          ]
        }
    ],
    "data": {
        "singlechannel": [50, 60]
    },
    "measurements": [
        { "name": "Measurement", "config": {"poi": "mu", "parameters": []} }
    ]
}
```

Fully vectorized computation

Use shim to make backend agnostic

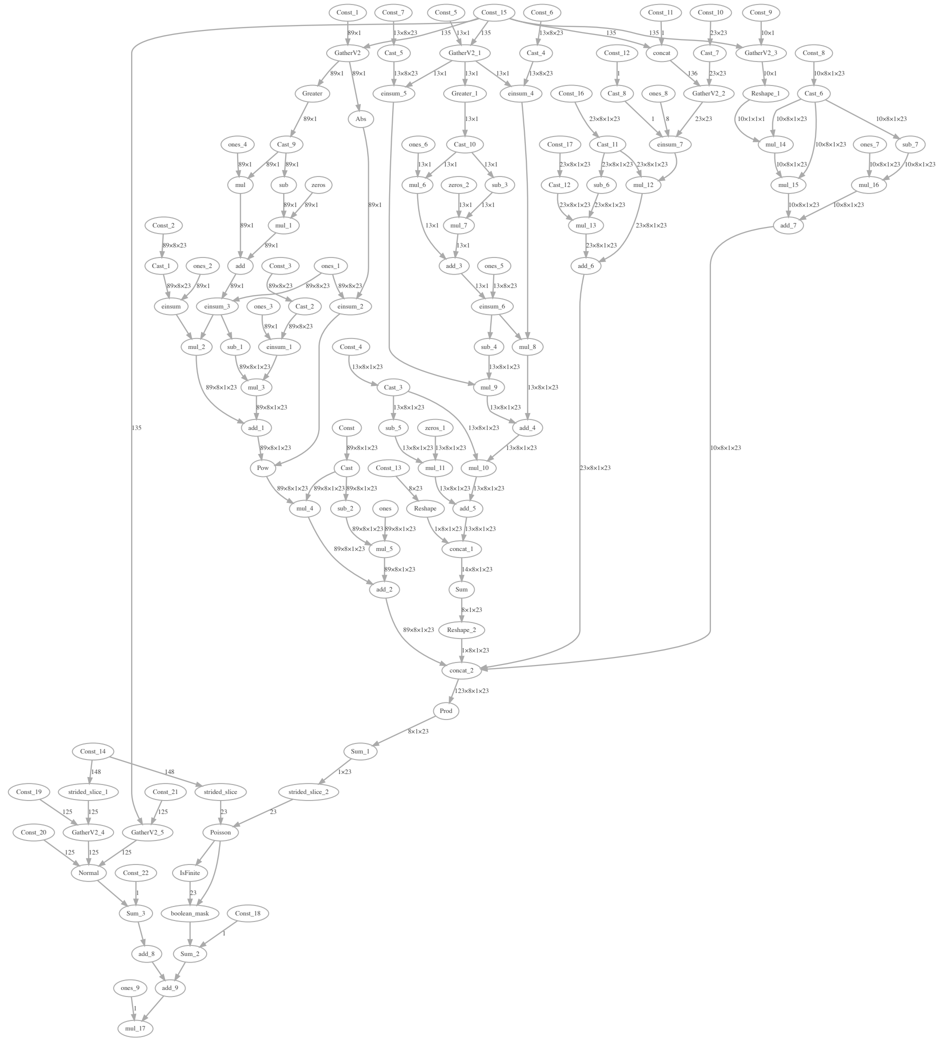
- NumPy (default)
- Tensorflow
- PyTorch
- (MXnet)
- (jax)
- (Dask)



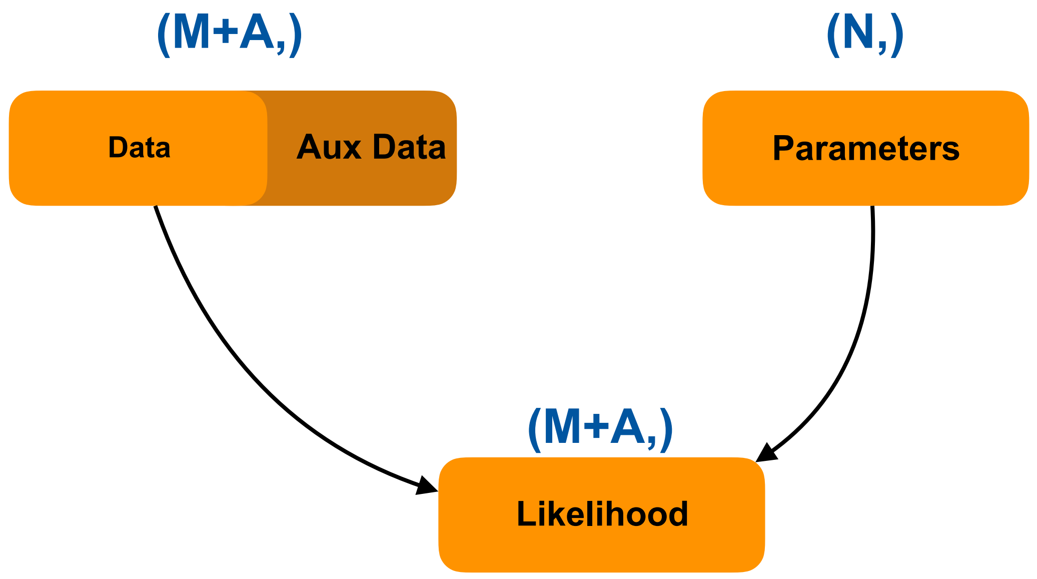


Advantage of non-Numpy backends

- Distribution across multiple machines (Dask)
- Hardware Acceleration (ML backends)
- Improved Fitting through Automatic Differentiation (ML Backends)



Tensor Structure



rather bare-bones, optimized for vectorized computation,

easy to extend to more batch dimensions (batched data, batched parameters)

Constraint "data" treated the same as observed data

Integration with other stats packages.

- See multiple projects aiming at doing out-of-ROOT stats. A lot of potential but also need to be careful not to fragment too much
  - RooFit/RooStats provided useful common language

- happy to have pyhf be only responsible for the pdf / likelihood function implemented in various ML backends
- Other packages for
  - minimization
  - hypothesis testing / limit setting
  - etc..

- would like to keep independent from any one ML framework
  - easier for closed world of HistFactory, maybe more challenging for e.g. RooFit/zfit type open world (see next)

RooStats:

- aims to separate modeling (i.e. p(x)) from inference / testing methods. (e.g. frequentist v bayesian
- adds some semantics on the model (RooStats::ModelConfig)
  - what are POI vs NPs
  - links to data (and aux data)
  - link to pdf
  - set of well-defined parameter points (S+B vs B vs best Fit) etc

- Operates mostly on abstract notion of pdf, which we could as a community try to agree on outside of ROOT. ABC which can
  - generate toys
  - evaluate nll
  - be composed