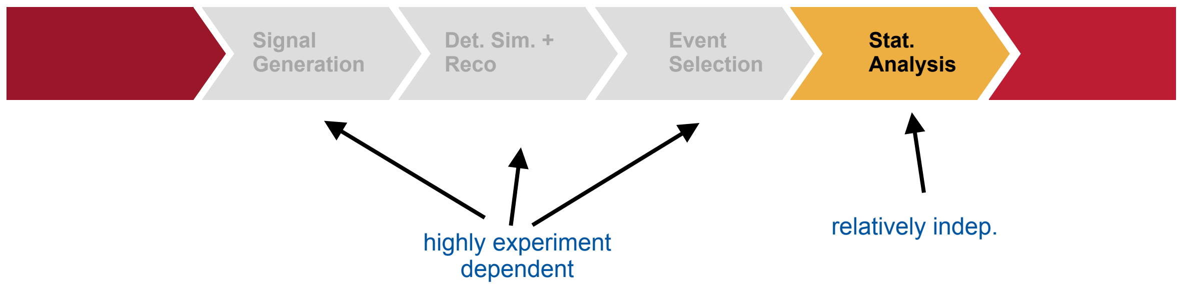
pyhf

Matthew Feickert, Lukas Heinrich, Giordon Stark

IRIS-HEP

pyhf in the HEP analysis workflow:

Overall Goal of an analysis: inference on some parameters of nature (masses, couplings, ...)...



Statistical Analysis:

- final step of analysis
- you've collected all events, what do they tell you

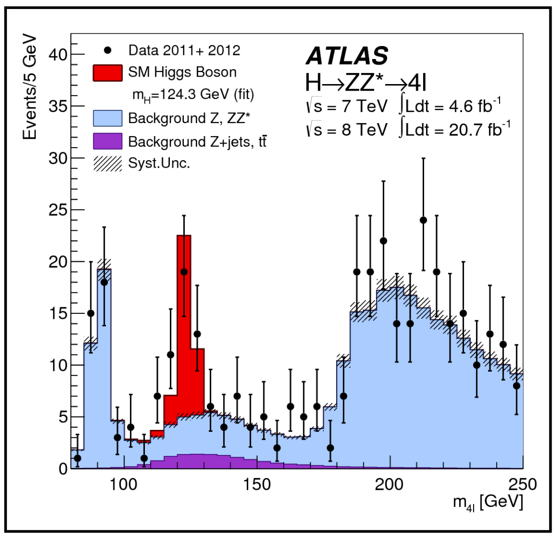
Two broad categories

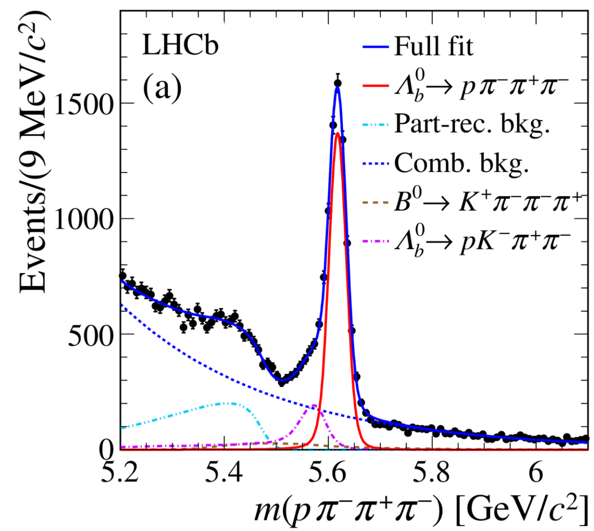
Decision: aggregate in the events before statistical analysis or not?

Statistical Analysis

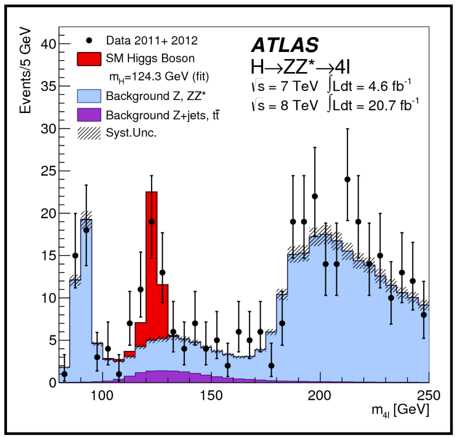
yes → Binned

no → Unbinned

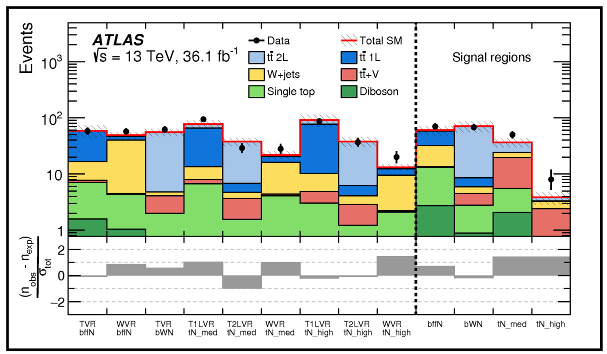




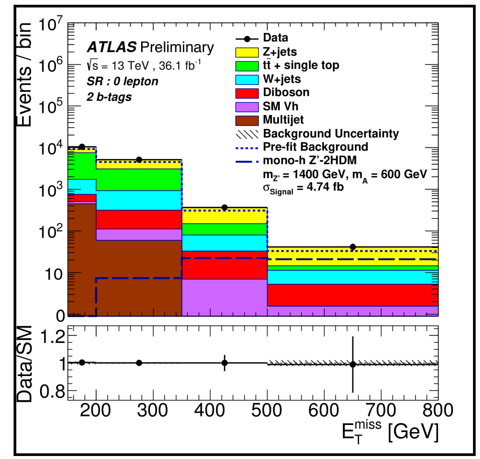
HistFactory is one of the most widely used families of stat. models used in HEP, esp. ATLAS: Majority binned template analyses formulated within this **one model.**



SM



SUSY



Exotics

pyhf

- Simultaneous 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.

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

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

Straight forward math:

but sofar only implementation in ROOT/RooStats/RooFit

HistFactory: A tool for creating statistical models for use with RooFit and RooStats

Kyle Cranmer, George Lewis, Lorenzo Moneta, Akira Shibata, Wouter Verkerke

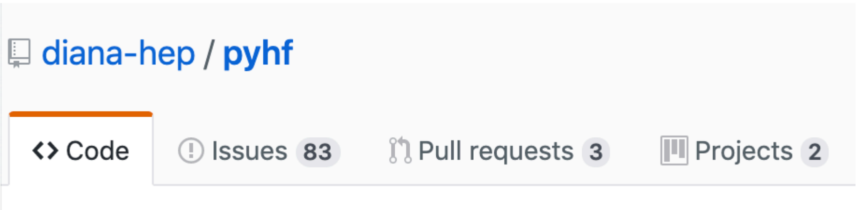
June 20, 2012

Contents

1 Introduction **2**

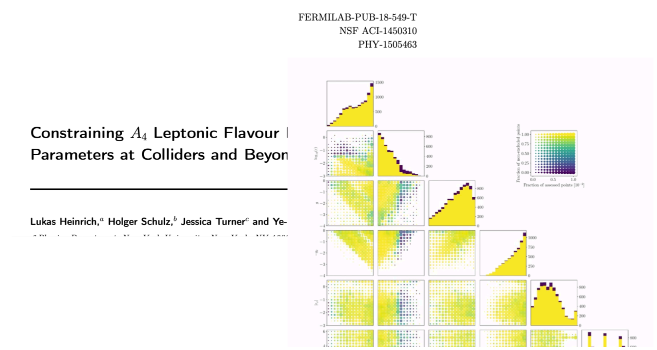
... but in principle implementable in other languages (of course)

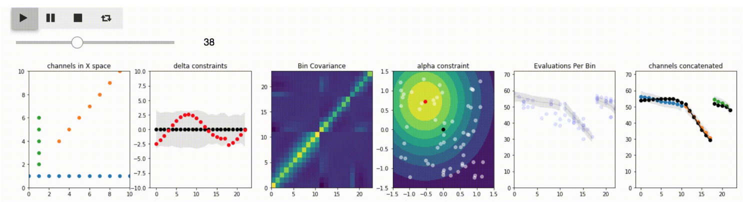
pyhf



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..)





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

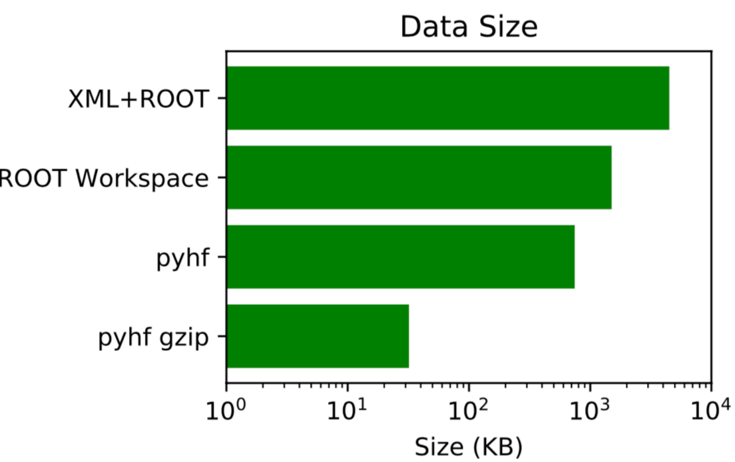
```
$> pip install pyhf
$> pyhf cls workspace.json
{
    "CLs_exp": [
        0.008897411763217407,
        0.03524468002619176,
        0.1243148689002353,
        0.3514186235832989,
        0.6941411699405086
    ],
    "CLs_obs": 0.036074093359460063
}
```

```
$> curl http://url-to-json/workspace.json|pyhf cls
{
    "CLs_exp": [
        0.0026064085052793559,
        0.013820656047622592,
        0.06445520798561910,
        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)

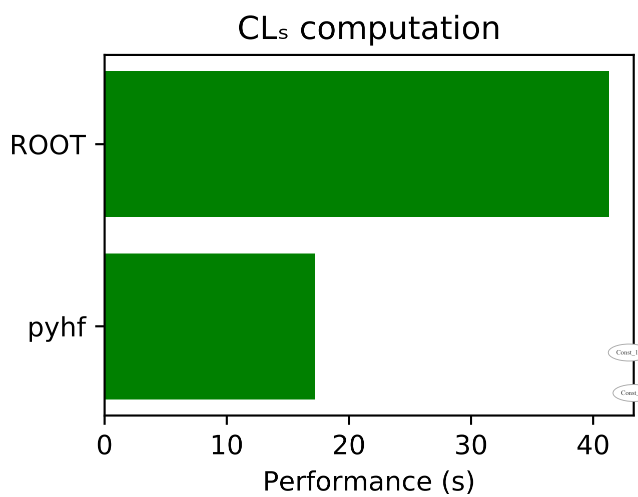


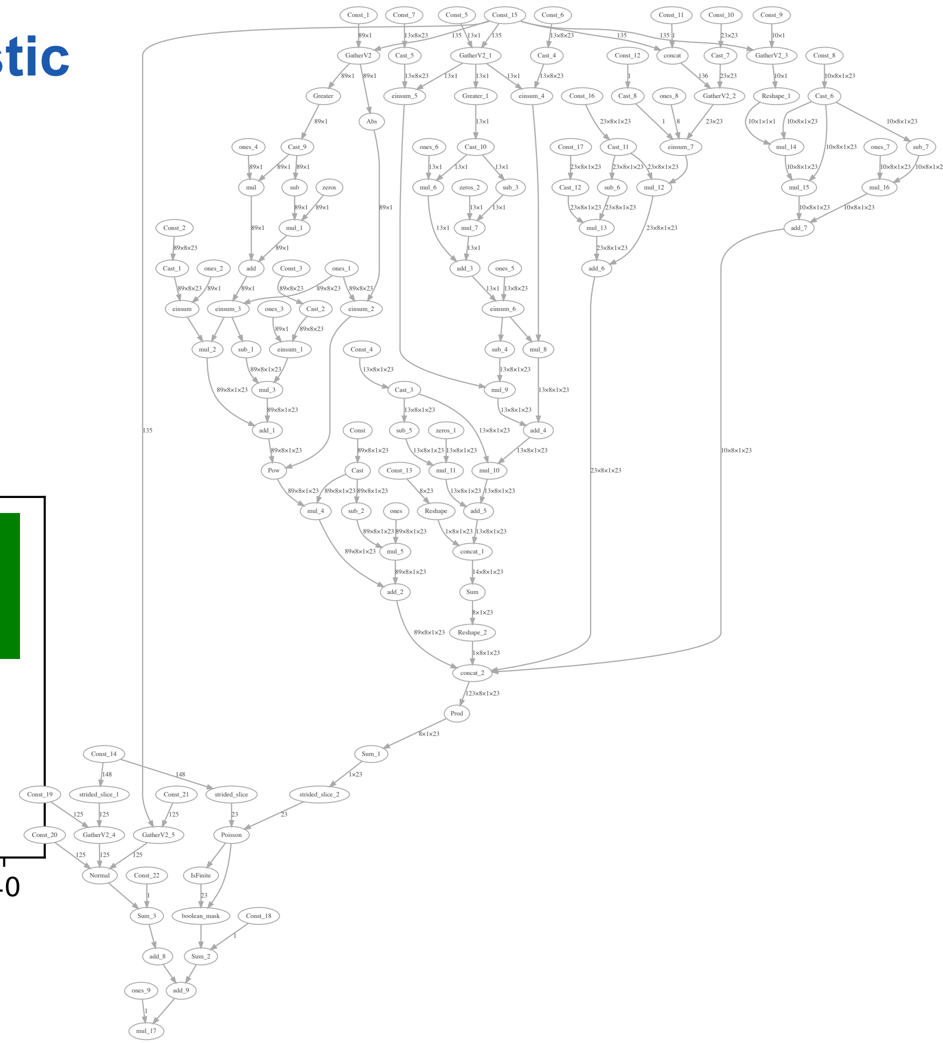

```
{
    "channels": [
        { "name": "singlechannel",
          "samples": [
            { "name": "signal",
              "data": [7.0, 2.0],
              "modifiers": [ { "name": "mu", "type": "normfactor", "data": null} ]
            },
            { "name": "background",
              "data": [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

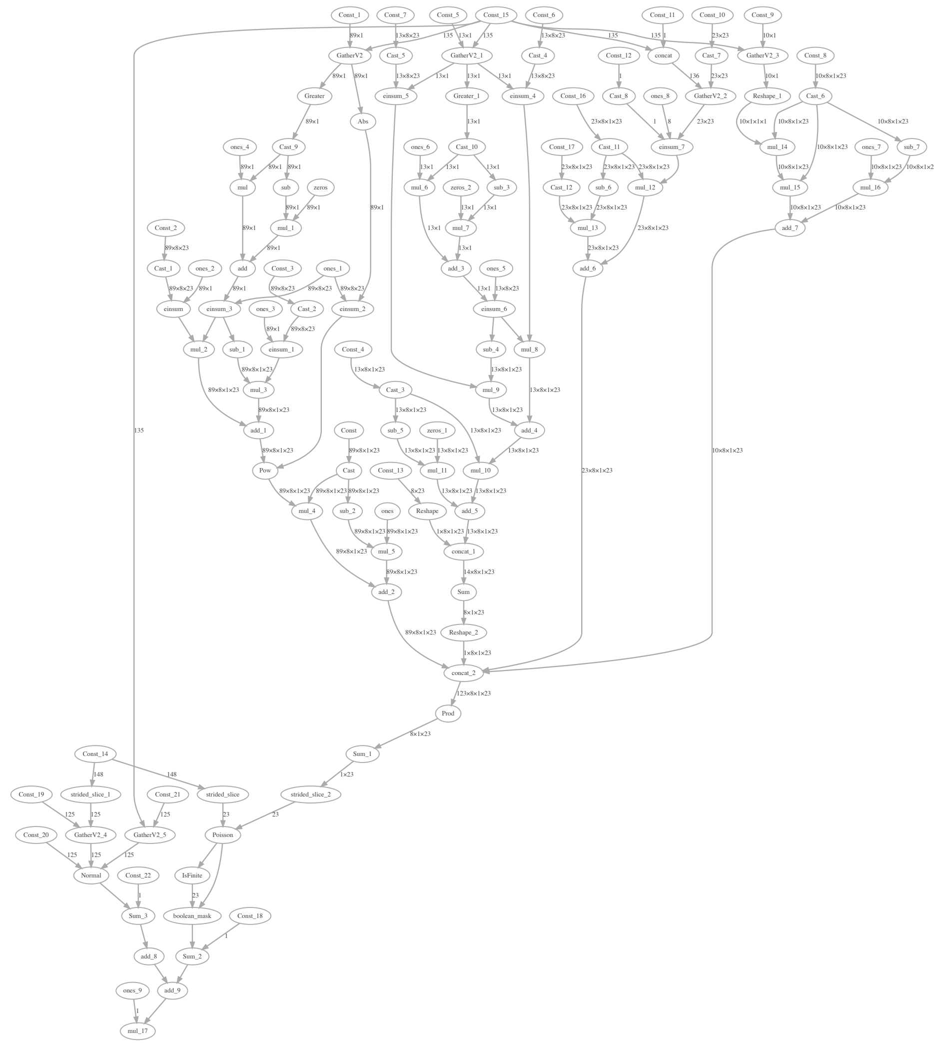
- NumPy (default)
- Tensorflow
- PyTorch
- (MXnet)
- (jax)
- (Dask)



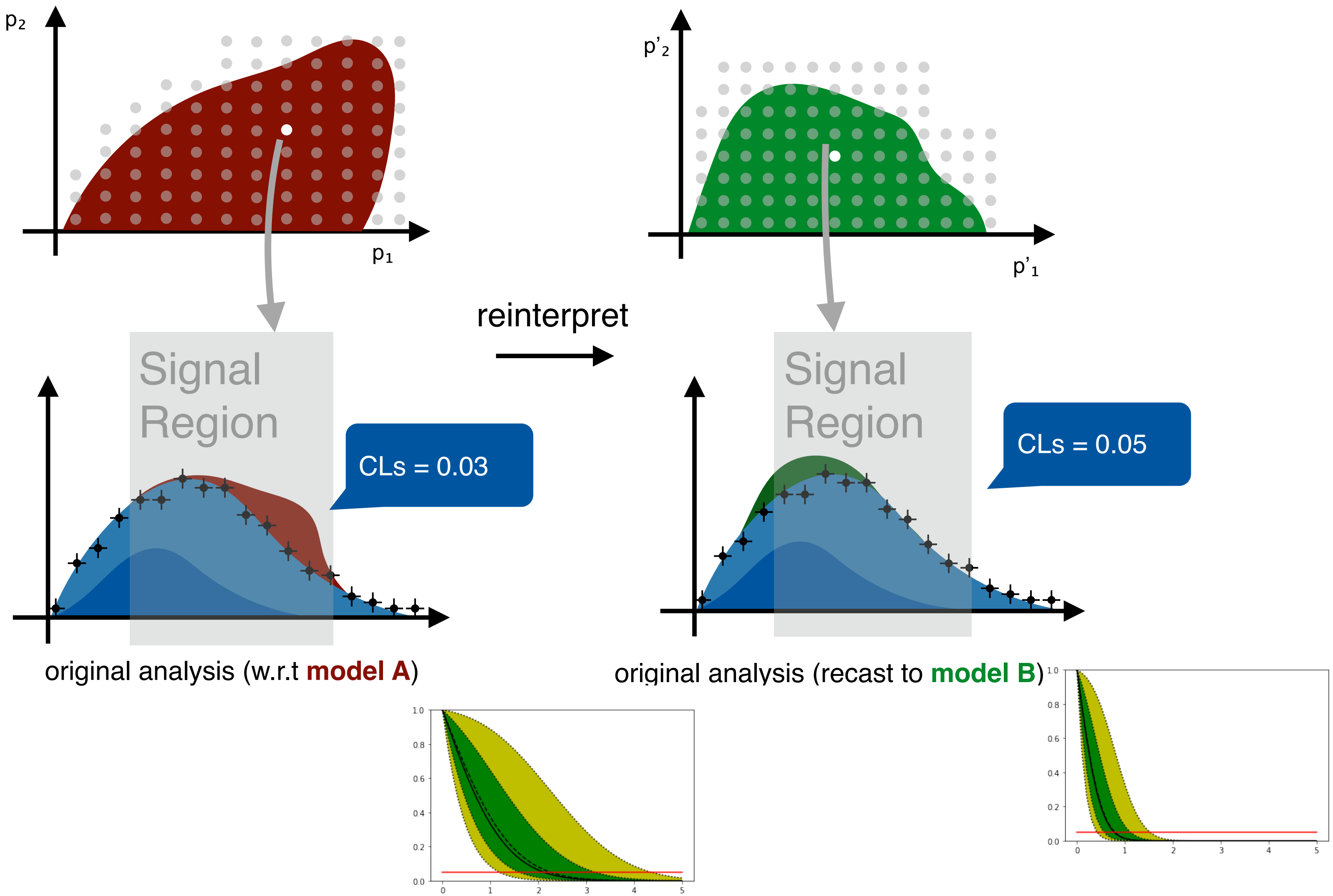


Advantage of non-Numpy backends

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



Easy RECAST



p_2
p_1
p'_2
p'_1
reinterpret
Signal Region
CLs = 0.03
Signal Region
CLs = 0.05
original analysis (w.r.t **model A**)
original analysis (recast to **model B**)

Easy RECAST

Advantages: very easy to modify likelihoods using `JSONPatch`
Important for e.g. Reinterpretations (think: pMSSM scan)

```
{
    "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": []} }
    ]
}
```

+

```
[{
  "op": "replace",
  "path": "/channels/0/samples/0/data",
  "value": [7.0, 2.0]
}]
```

=

```
jsonpatch likelihood.json patch.json > new.json
pyhf cls new.json
```

```
{
    "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": []} }
    ]
}
```

new yields injected

Connecting upstream

Input to HistFactory is (structured) set of Histograms (.. what used to be histbook?)

Histogram = result of a functional DL operating on event-wise data

Example:

- simple TTree::Draw-like syntax
- coffea / uproot / awkward ...

```
samples:
- name: 'data'
  variations:
  - name: nominal
    tree: data16
    data: true
    glob: 'exported/data16/*.root'

- name: 'wjets_mc16a'
  variations:
  - name: nominal
    tree: wjets_mc16a_Nom
    glob: 'exported/wjets_mc16a/*.root'
    weight: 'xs_weight * weight'
    lumi: 36207.66
- name: 'zjets_mc16a'
  variations:
  - name: nominal
    tree: zjets_mc16a_Nom
    glob: 'exported/zjets_mc16a/*.root'
    weight: 'xs_weight * weight'
    lumi: 36207.66
- name: 'ttbar_mc16a'
  variations:
  - name: nominal
    tree: ttbar_mc16a_Nom
    glob: 'exported/ttbar_mc16a/*.root'
    weight: 'xs_weight * weight'
    lumi: 36207.66

regions:
- name: regionA
  binning: [-0.5,15.5,17]
  filter: 'n_jet < 7 && n_mu > 0'
  observable: 'n_jet'
- name: regionB
  binning: [-0.5,15.5,17]
  filter: 'n_jet > 8 && n_mu > 0'
  observable: 'n_jet'
```


```
regions:
- name: regionA
  binning: [-0.5,15.5,17]
  filter: 'n_jet < 7 && n_mu > 0'
  observable: 'n_jet'
- name: regionB
  binning: [-0.5,15.5,17]
  filter: 'n_jet > 8 && n_mu > 0'
  observable: 'n_jet'
```

Likelihood Preservation (tomorrow)

ATLAS CONF Note

ATLAS-CONF-2019-011

March 27, 2019

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