# Differentiable Programming and the IRIS-HEP Analysis Grand Challenge

G. Watts (for IRIS-HEP)

2022-09-13

$$\frac{\partial Analysis}{\partial \Theta}$$

Full analysis

- including likelihood (loss function)
- Systematic errors
- etc.

$$\frac{\partial Analysis}{\partial \Theta}$$

- Cuts
- ML Parameters
- etc.

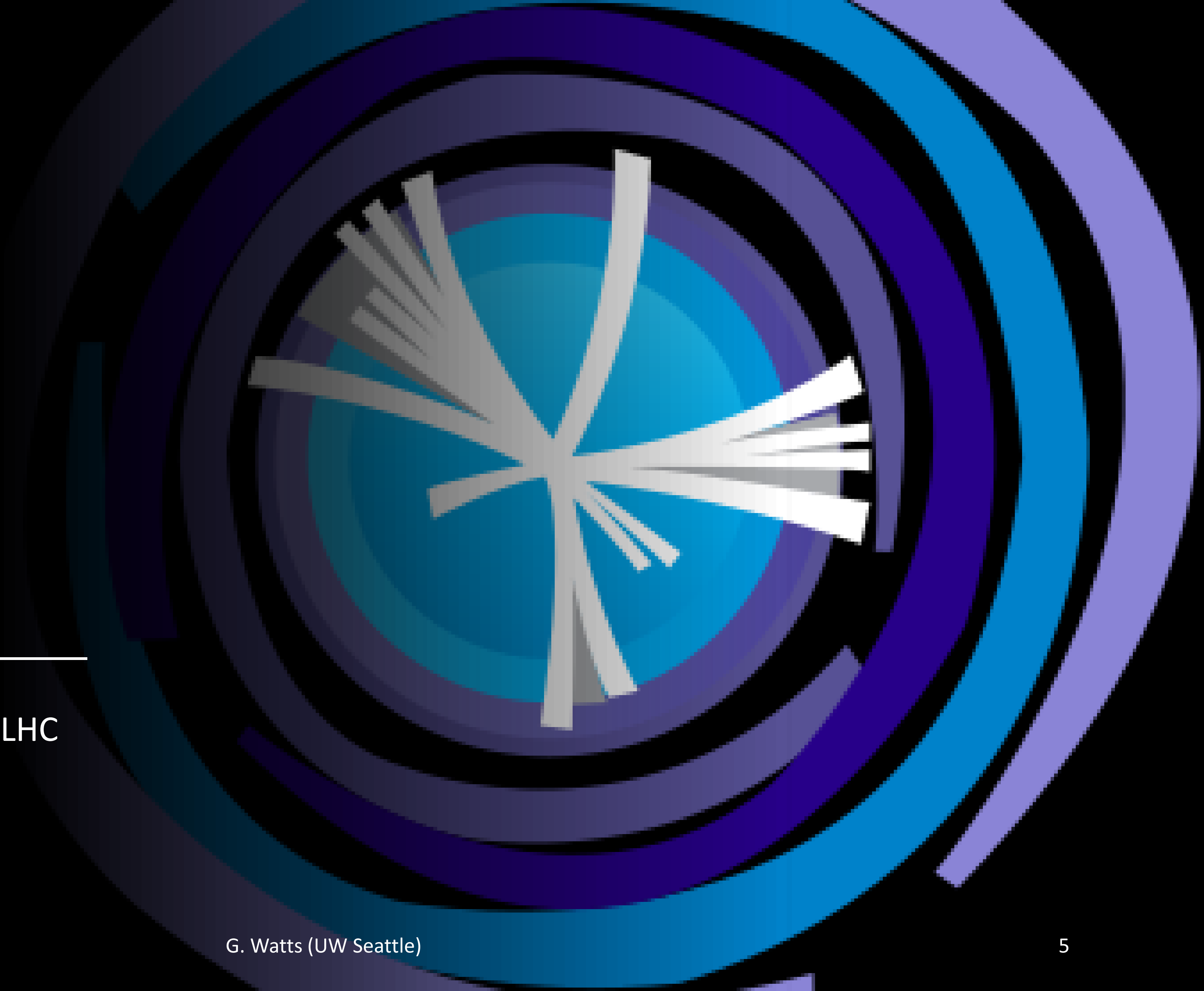
$$\frac{\partial Analysis}{\partial \Theta}$$

+ Take Full Advantage of Systematic Errors and optimize for actual sensitvity
+ Interpretability – encode and optimize physics-based cuts, along side more complex ML
+ NN's are trained in-situ rather than on separate datasets
+ Don't burn grad student time optimizing individual straight cuts!

– Potential Impact on Analysis is not fully understood given the amount of work required
– Touches almost every single tool in our tool chain – huge amount of work
– NN's are trained in-situ rather than on separate datasets (small datasets)
– Burn graduate student time setting up a complex and interconnected tool chain

# The Analysis Grand Challenge

An IRIS-HEP and community HL-LHC Challenge



# IRIS-HEP

Institute for Research & Innovation in Software for High Energy Physics

## “Institute for Research and Innovation in Software for High Energy Physics”

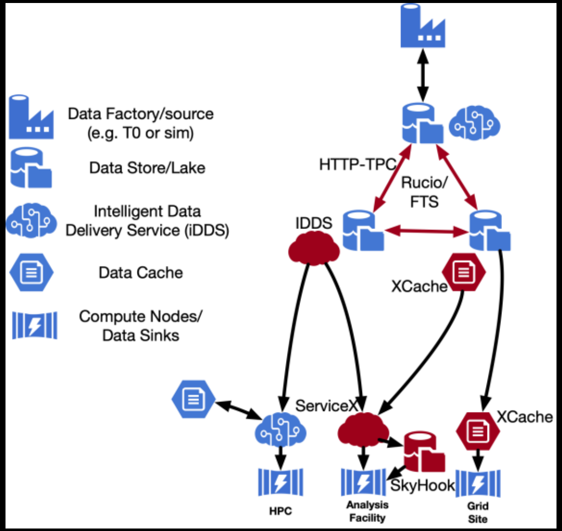
- Large software institute funded by the US National Science Foundation (NSF)
  - 19 universities, ~30 FTE’s, spread over ~60 people
- Research and development for the HL-LHC
  - Innovative Algorithms for Data Reconstruction and Triggering
  - Data Organization, Management, and Access (DOMA)
  - Analysis Systems, to reduce time-to-insigt and maximize physics potential
  - Facilities, integration to production (scalable systems laboratory and OSG)

See our website for further information

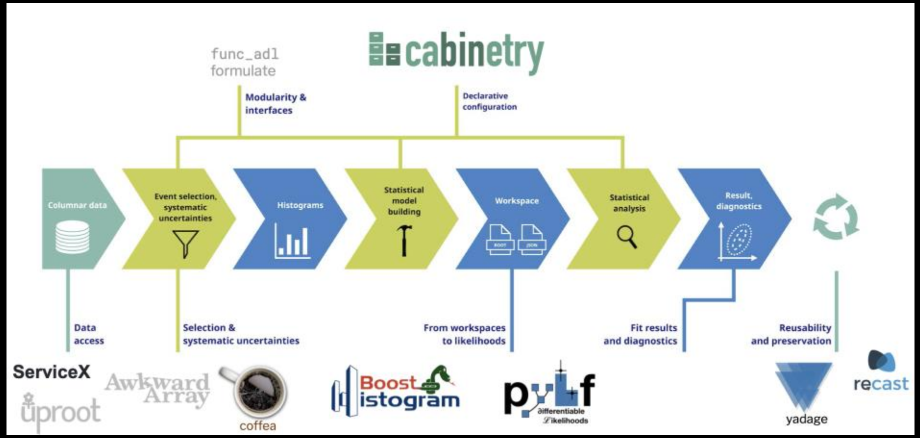
# Analysis Grand Challenge

- Started as an integration exercise, now a milestone for HL-LHC
  - Test a realistic analysis pipeline aimed at HL-LHC datasets and analyses
  - Combine technologies being developed in various areas of IRIS-HEP and the ecosystem
  - Identify and address performance bottlenecks and usability issues
- Organized jointly with the US ATLAS and US CMS operations programs
  - Operations programs maintain analysis facilities where we expect these analyses to be performed

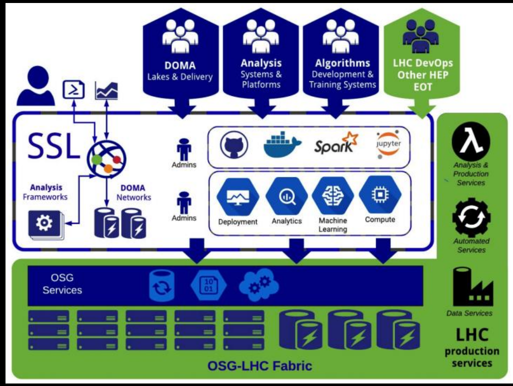
## Data Delivery




## Analysis Tools

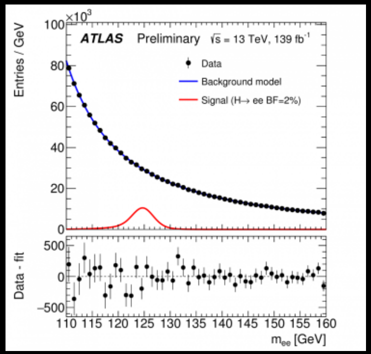



## Facilities



# The Analysis Pipeline in the AGC

ServiceX → Awkward & coffea → cabinetry → pyhf →




Data Delivery

Analysis Tools

(thanks to A. Held for graphic)

uproot · Awkward Array · func_adl · VECTOR · Boost-histogram · mplhep · cabinetry · pyhf (differentiable Likelihoods) · iminuit

HEP-specific libraries used for data analysis

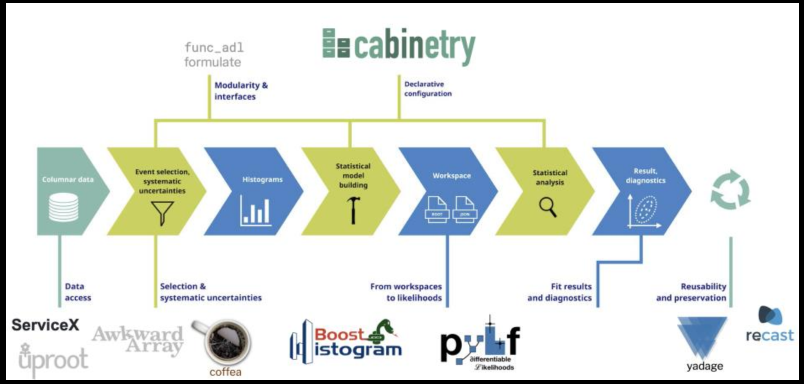
ServiceX

data delivery services

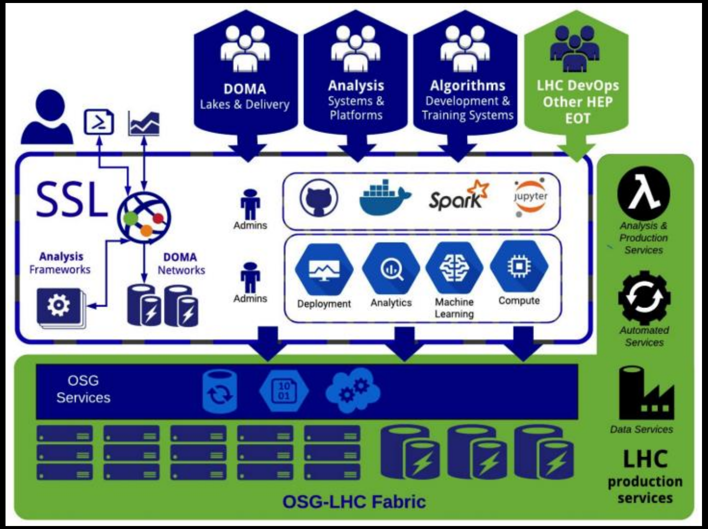
Coffea-Casa · XCache · funcX

optional services

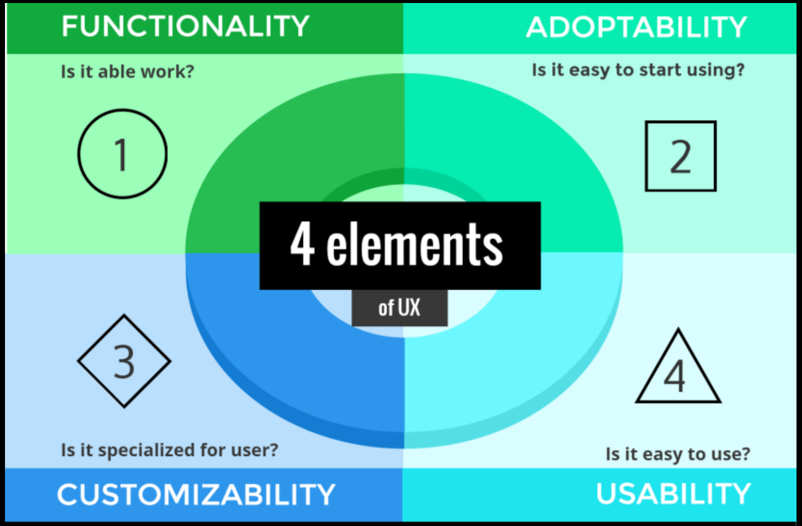
# Integration and Time-To-Insight and UX




Tools must easily talk to each other!




Scale-out by default

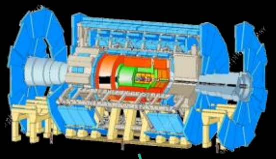



Reasonable User Interface and Experience

# Differentiable Analysis

Adding differentiable capabilities to the analysis pipeline

# Not A Single System

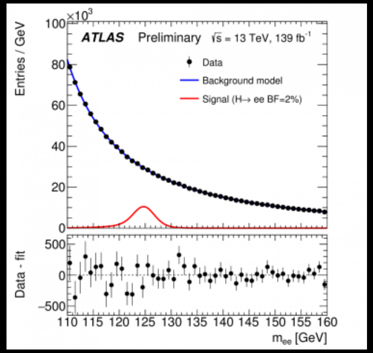


Data

Monte Carlo

All running on a Analysis Facility

PyTorch

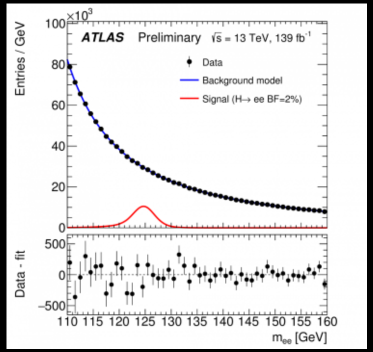


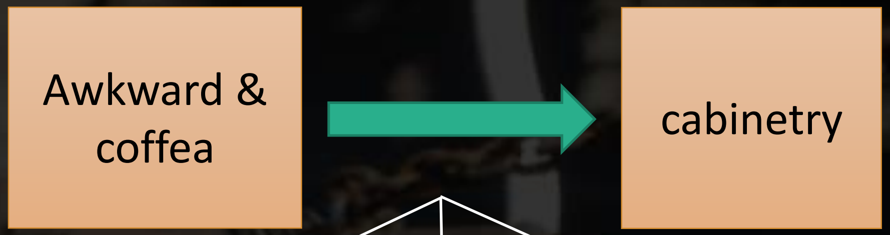

ServiceX

Awkward & coffea

cabinetry

pyhf



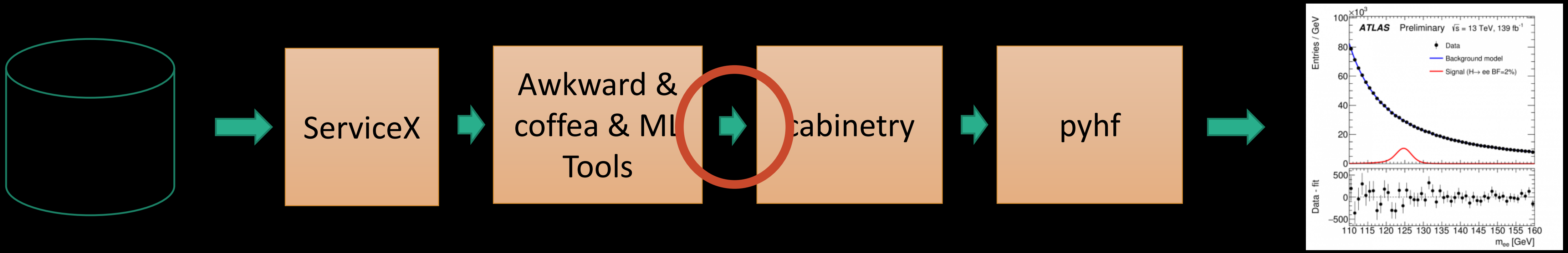



Analysis
- Forward pass only
- Only data needs to be passed from one step to the next
- Awkward arrays, parquet files, ROOT files

Autodiff – forward pass
- Updated parameters for all earlier layers
- Derivatives and primal values
- Per parameter running

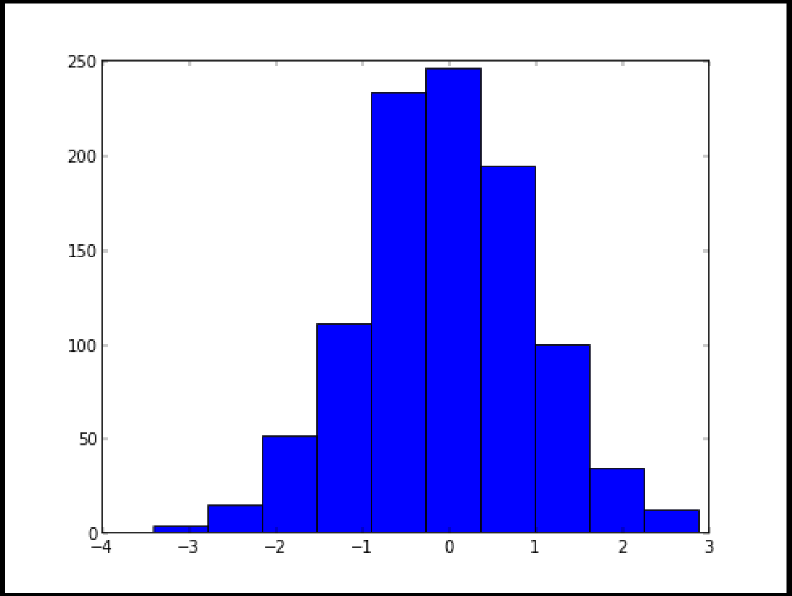
Autodiff – backward pass
- Updated parameters for all earlier layers
- Tape of operations
  - Does this have to be passed up to the top?
- Intermediate values have to be stored

# Infrastructure




- Need a standardized way to communicate between layers and systems
  - Language agnostic?
- Need a library that implements this standard
- Will need to be 2-way communication as data and Jacobian's move up and down the chain!
- And will need to run in real-time as training is iterative!

# Common Libraries - histograms




We have powerful libraries for histogramming that are packed with features

- python's hist
- ROOT

These are carefully designed for speed and distributed filling! Years of engineering work!

What does it mean to make a histogram differentiable?

1. Assume binning is fixed
   - Each bin is a scalar, $n_i$, and you must now calculate $\frac{dn_i}{d\theta}$
   - 100 bins is very expensive!
   - What happens to all the nifty histogram manipulation utilities we already have?

Relaxed Library

2. Binning is a function of the data (width, number)

# awkward Array

The numpy analog for jagged data structures

numpy – rectilinear data (square table, each row has same number of columns)
awkward – jagged data (rows can have variable numbers of columns – 3 jet pt's in one event, 10 in another)

Making awkward differentiable is under way as part of the awkward-dask project.

Gluing awkward array together with a NN training

- PyTorch, TensorFlow and JAX all have the ability to add differentiable function to the language
- Glue code will have to be written if a NN is expected to participate in the differentiable analysis pipeline

# Components

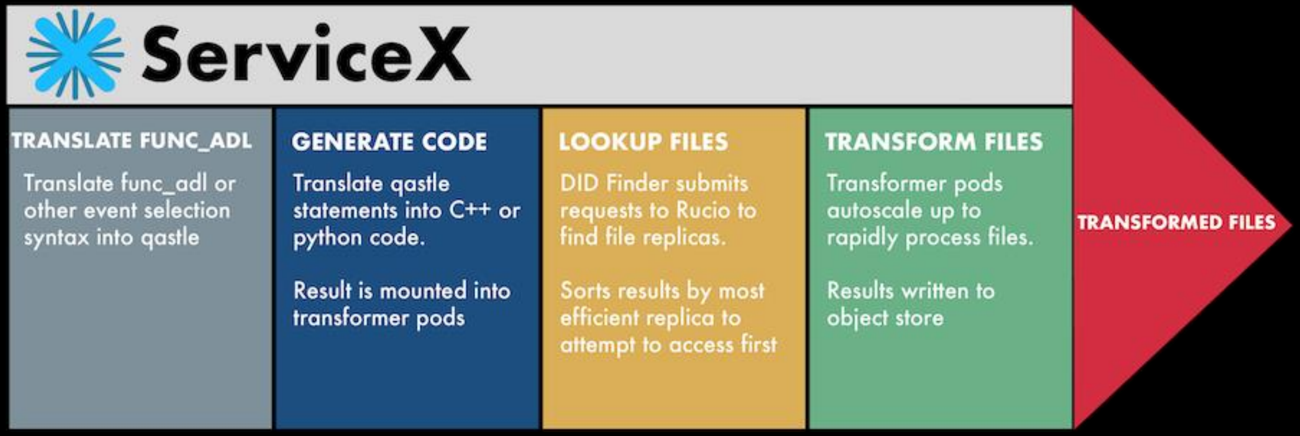
pyhf is a pure-python implementation of that statistical model for multi-bin histogram-based analysis and its interval estimation is based on the asymptotic formulas of "Asymptotic formulae for likelihood-based tests of new physics"

- Likelihood is calculated using JAX, PyTorch, or TensorFlow (and numpy)
- As a result, already differentiable
- Has already been used to drive minimizations

cabinetry is a Python package to build and steer (profile likelihood) template fits.

- Uses hist, pyhf and awkward array
- Glue code will have to be written, but will rely on those tools' differentiability for the most part

# Components




Data-delivery: transforms data from "arbitrary" formats into columnar data, as well as enabling predicate push-down.

Does need to be differentiable

- Implements selection cuts in its code to reduce data
- Can do object aggregation (# of good jets with $p_T > 30$ GeV)

But this is a big ask

- Every transformer backend would have to be differentiable
- Some transformers are running in legacy software (e.g. CMS Run 1 Transformer)

Possible Solutions

1. Implement a "wrapper" that can differentiate the cut language
2. Separate training from inference:
    - Implement simple cuts in differentiable training code
    - Move cuts to SX once training is complete

# Systematic Errors

Systematic Errors steer the training away from problematic areas of phase space and variables that aren't well modeled → Very powerful in the training

1 Include systematic errors as (differentiable) functions

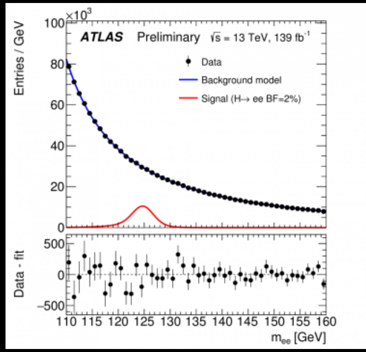
- Would allow one to stay away from poorly modeled areas of phase space
- Most systematic errors are applied as functions already
- Some experiments have hidden systematic errors under many layers of C++
- Handling operating points (make tight → medium → loose continuous)

2 Calibrations fully differentiable

- Include datasets and calibration as part of training
- Calibration will be best for the part of phase space analyzer is most interested in
- Akin to in-situ jet calibration in top mass measurements
- Lots of data, complex calibrations, and potential approval nightmare in large experiments

# Conclusion

ServiceX → Awkward & coffea & ML Tools → cabinetry → pyhf →




- **ServiceX**: Approach and implementation must be decided and written
- **Awkward & coffea & ML Tools**: Awkward is in progress, histogram library prototype exists
- **cabinetry**: Integration required but depends on other libraries.
- **pyhf**: Ready to go!

Protocol for passing Jacobian's and parameters between layers must move past proof-of-principle.

This is a tremendous amount of work: we are looking forward to more items turning green!