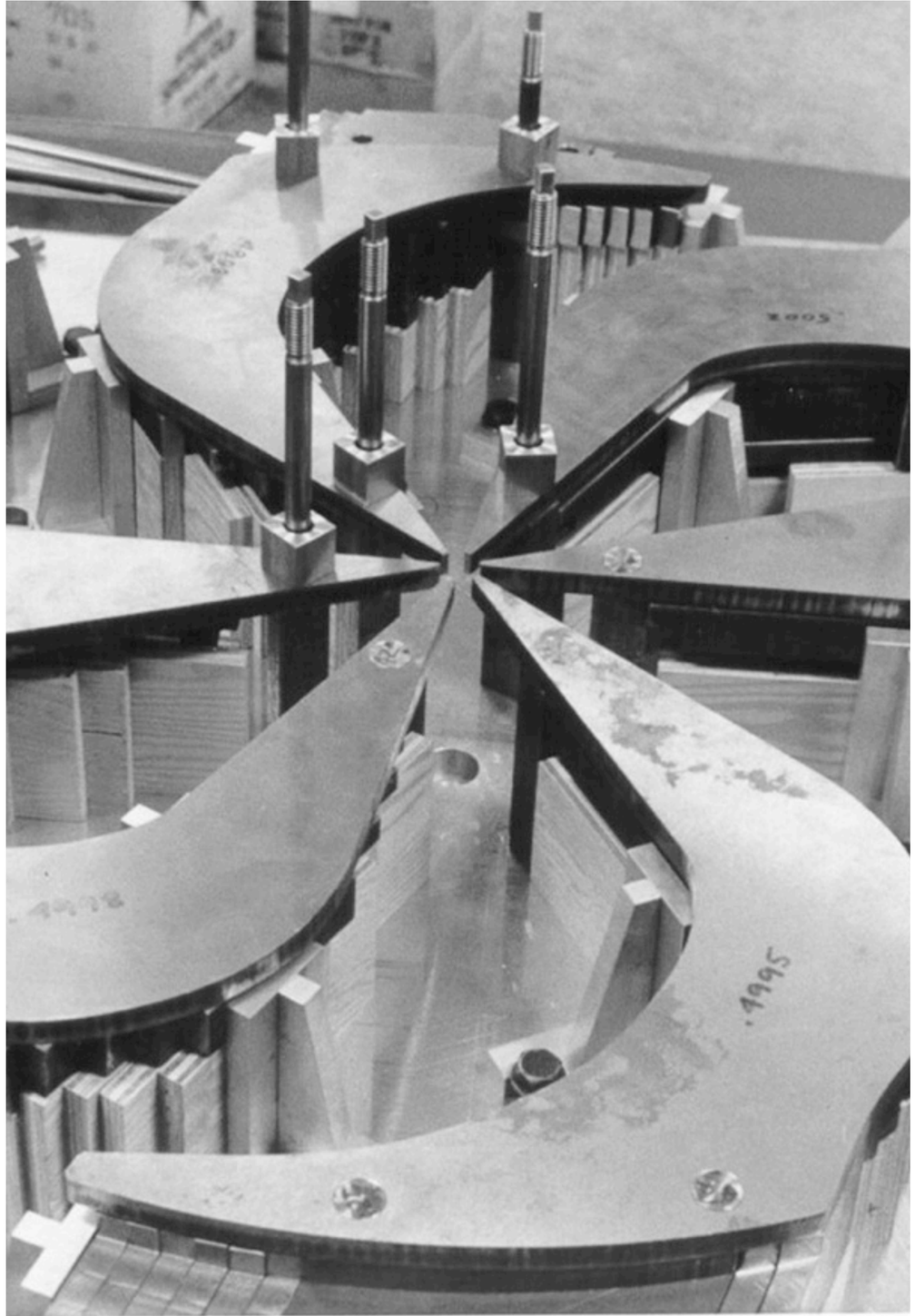
TRIUMF

An unbinned 24 dimensional measurement of Z+jets events in the ATLAS detector

Laura Miller

CFNS AI/ML Meeting
March 13th, 2025

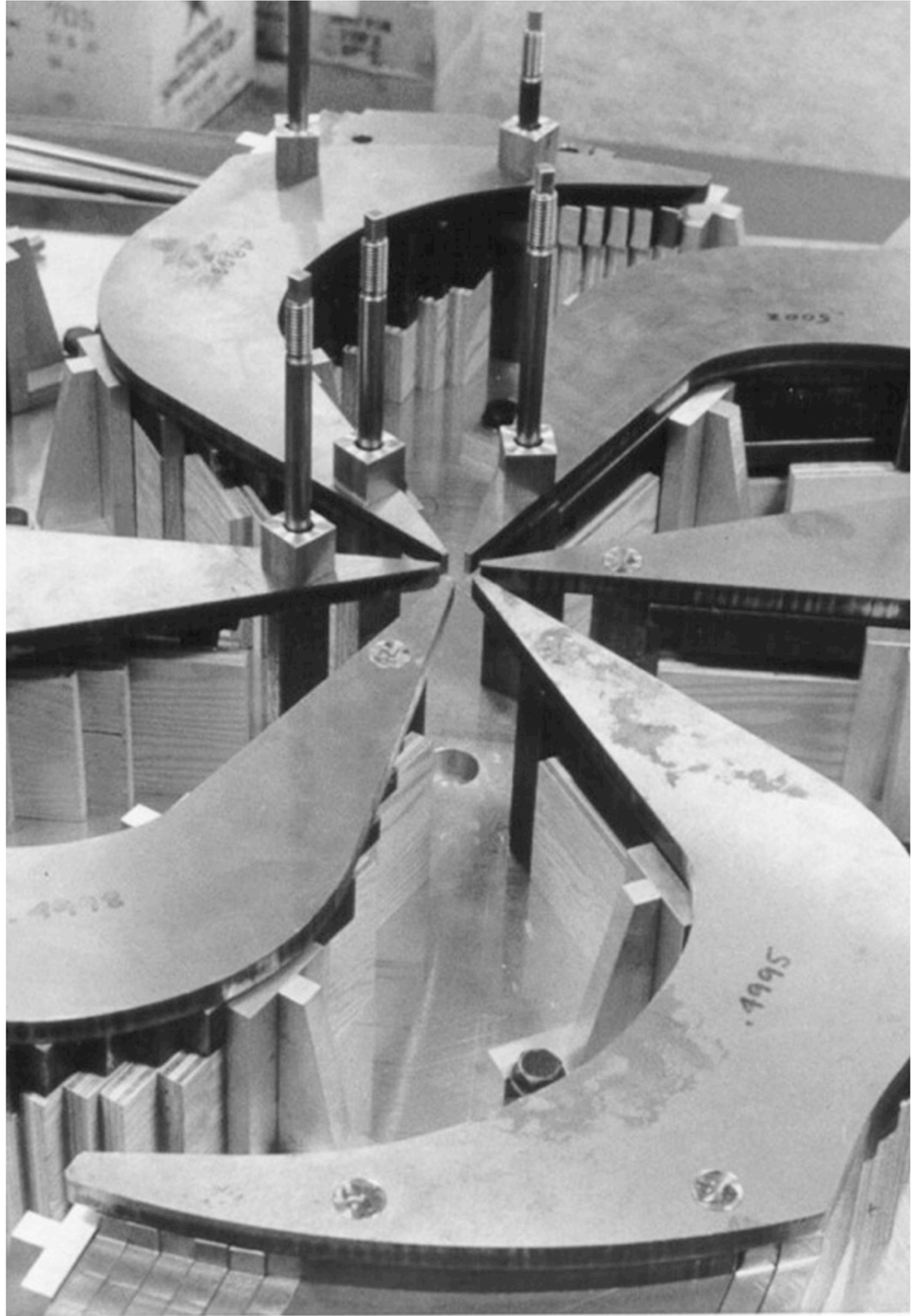


Discovery, accelerated

An unbinned 24 dimensional measurement of Z+jets events in the ATLAS detector

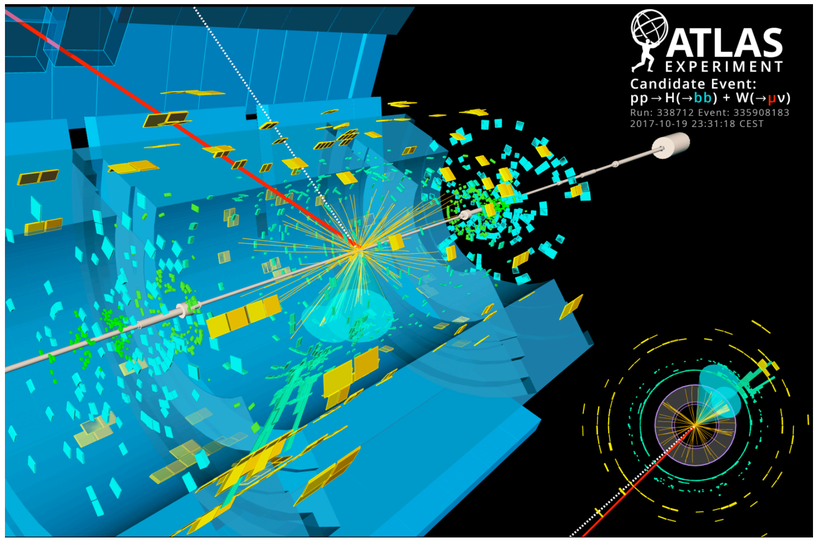
Laura Miller

CFNS AI/ML Meeting
March 13th, 2025

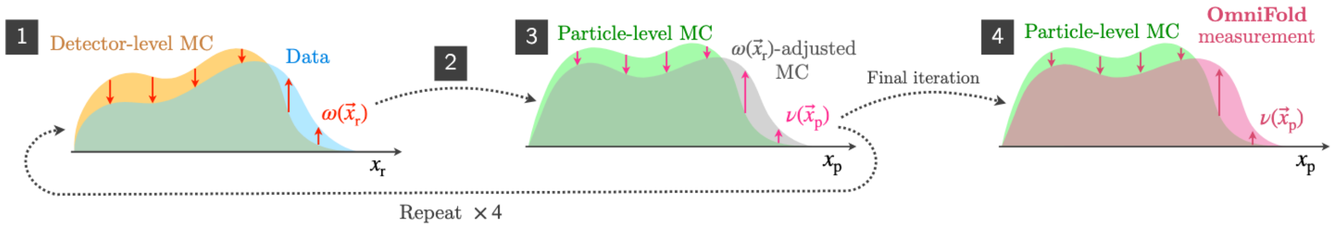


Discovery, accelerated

Outline



- Part 1: Introduction
  - CERN and ATLAS
- Part 2: Unfolding
  - Samples at ATLAS
  - Standard measurement techniques
  - The MultiFold algorithm
- Part 3: Analysis
  - Z+jets events in the ATLAS detector
  - Results
  - Demo!



Talk is based on Phys. Rev. Lett. 133, 261803 (or arXiv:2405.20041 if you prefer)

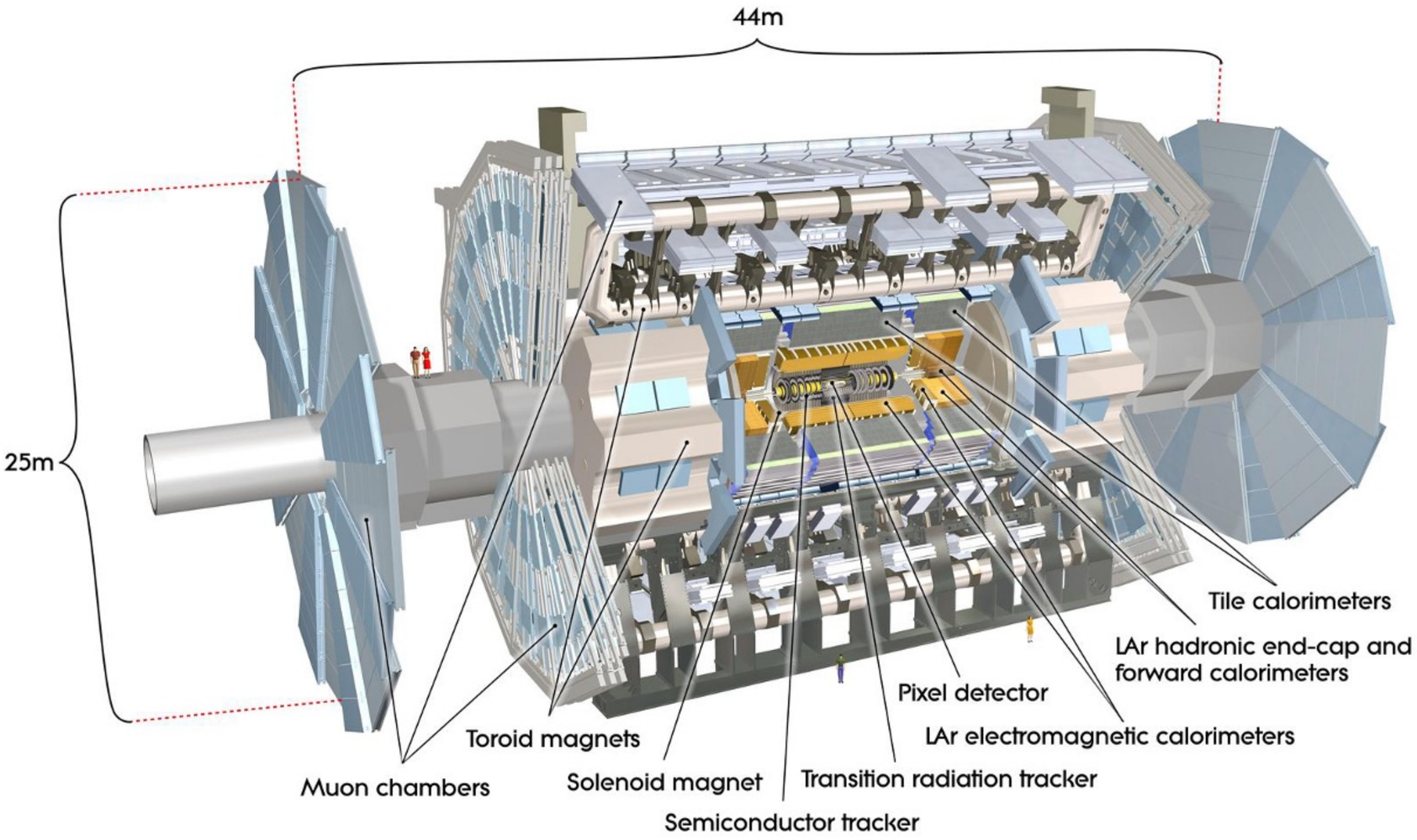
And my PhD thesis: CERN-THESIS-2023-169

Part 1: Introduction

The Large Hadron Collider and CERN

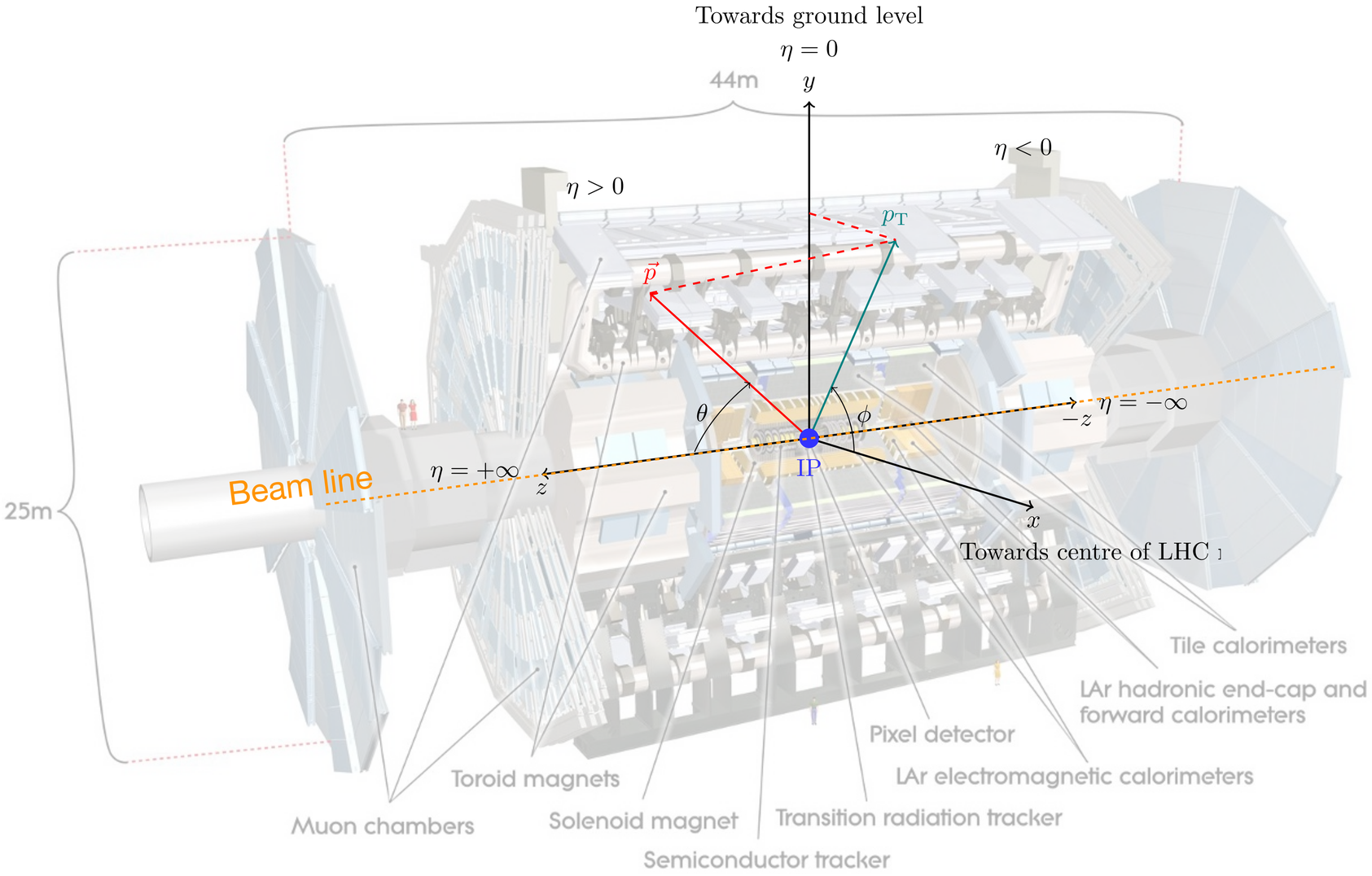
- CERN: largest particle accelerator complex in the world
- LHC: circular accelerator with a 27 km circumference, collides bunches of protons at very high energy
- Four major experiments, including ATLAS, located at interaction points around the LHC ring

The ATLAS detector



- General purpose detector located at point 1 on the LHC ring

The ATLAS detector



- General purpose detector located at point 1 on the LHC ring
- Right-handed coordinate system

Transverse momentum: $p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2}$

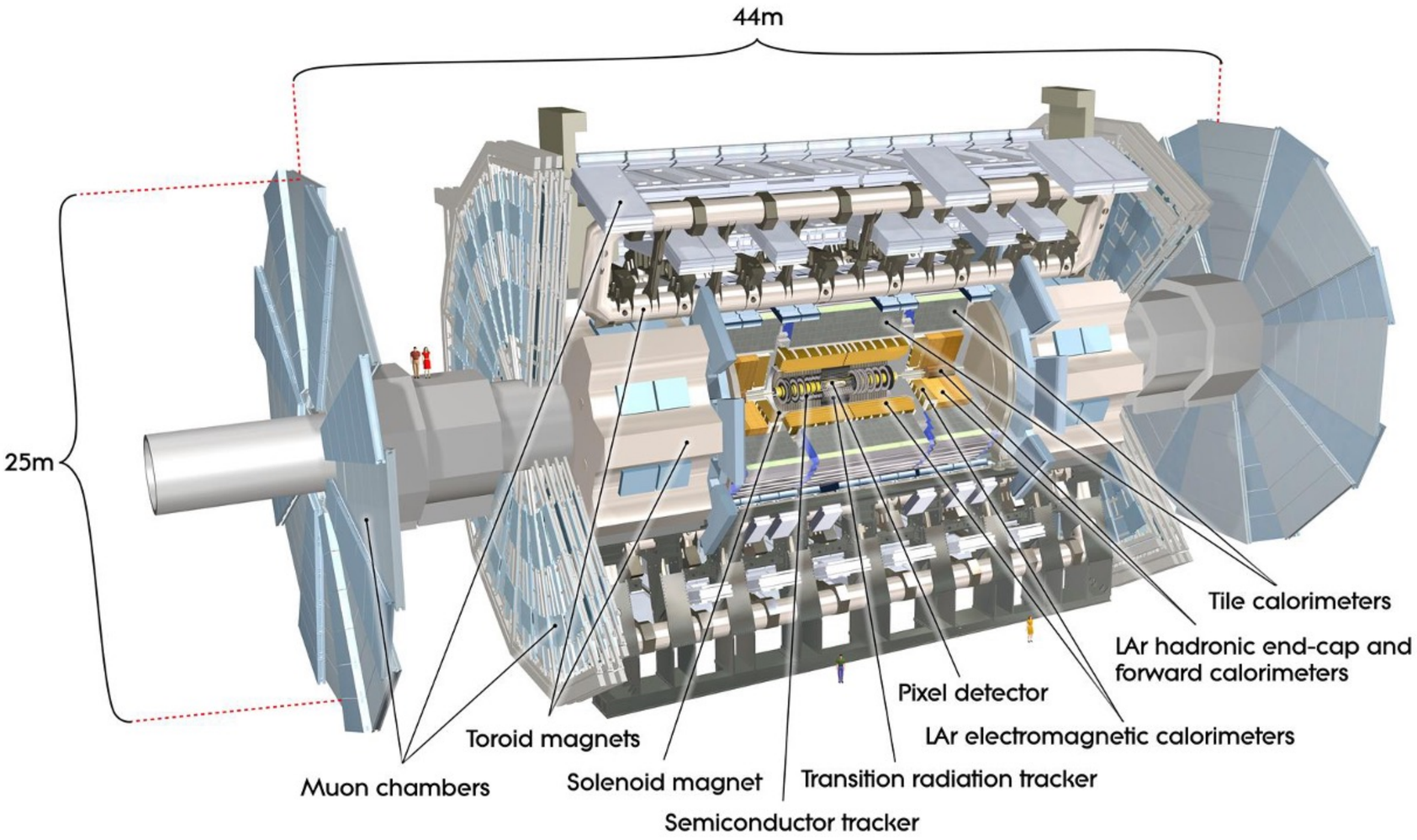
Pseudorapidity: $\eta = -\ln \tan\left(\frac{\theta}{2}\right)$

Rapidity: $y = \frac{1}{2} \ln\left(\frac{E + p_z}{E - p_z}\right)$

n.b. $y = \eta$ when $m = 0\,\mathrm{GeV}$

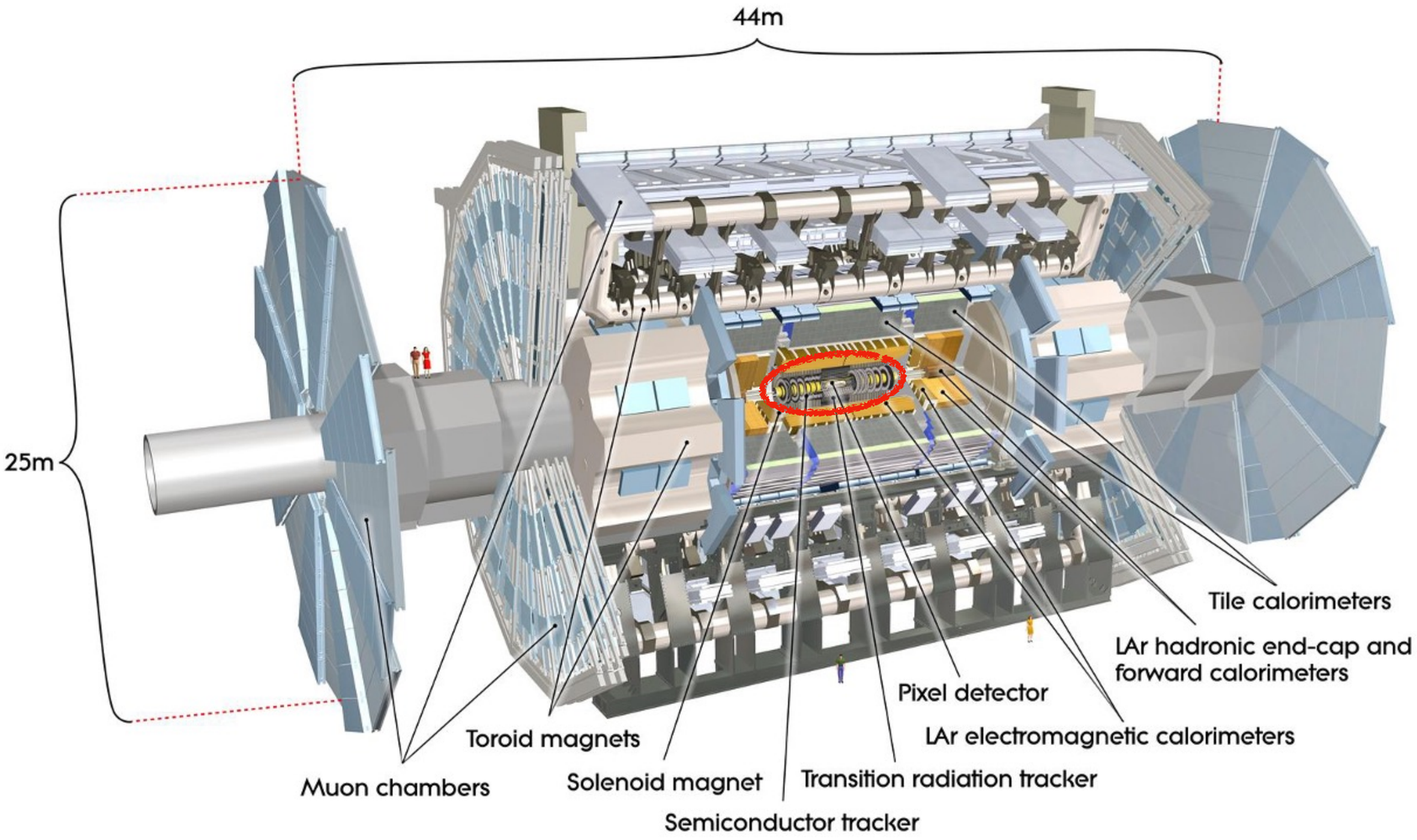
Distance: $\Delta R = \sqrt{\Delta y^2 + \Delta \phi^2}$

The ATLAS detector

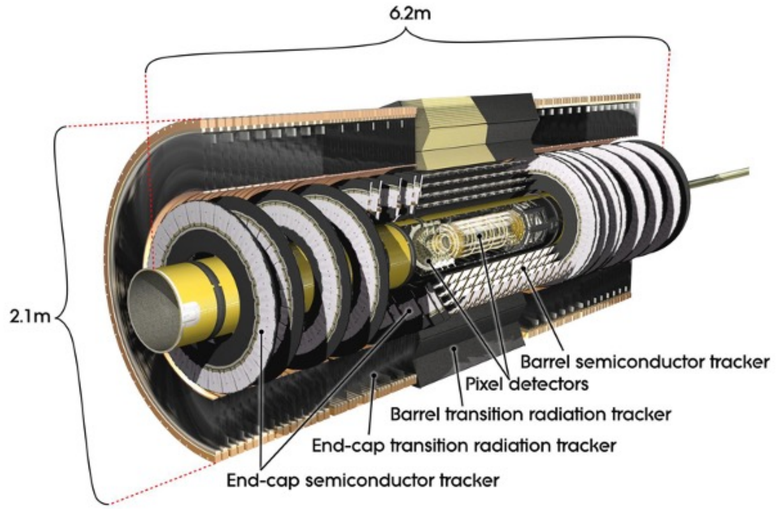


- General purpose detector located at point 1 on the LHC ring
- Right-handed coordinate system
- Components:

The ATLAS detector

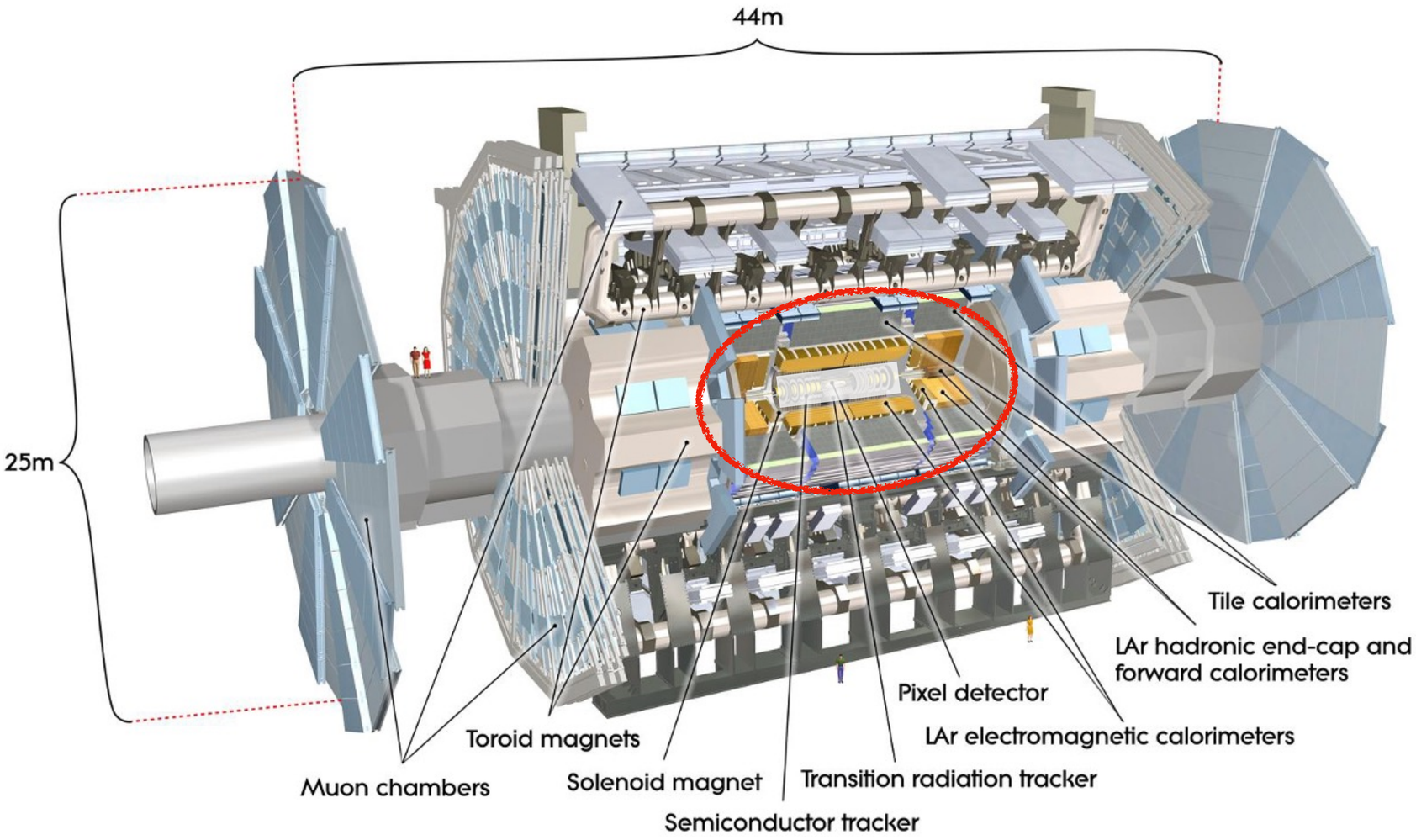


- General purpose detector located at point 1 on the LHC ring
- Right-handed coordinate system
- Components:
  - Inner detector

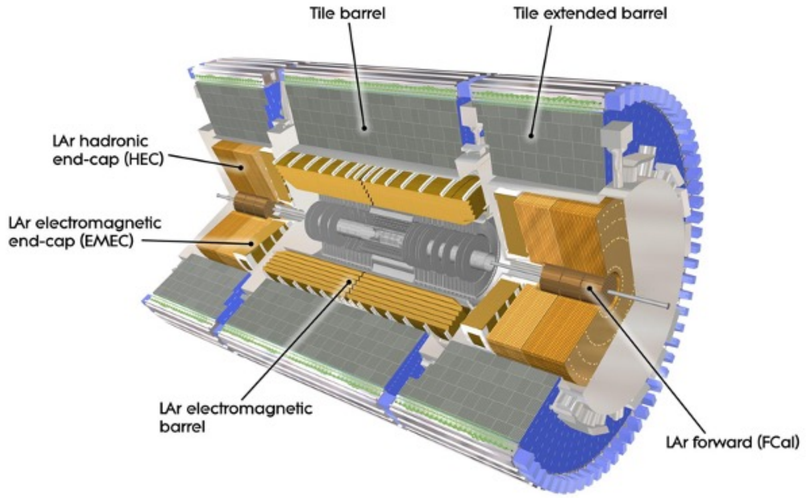


- Combination of silicon and gas detectors
- Measures charged particles

The ATLAS detector

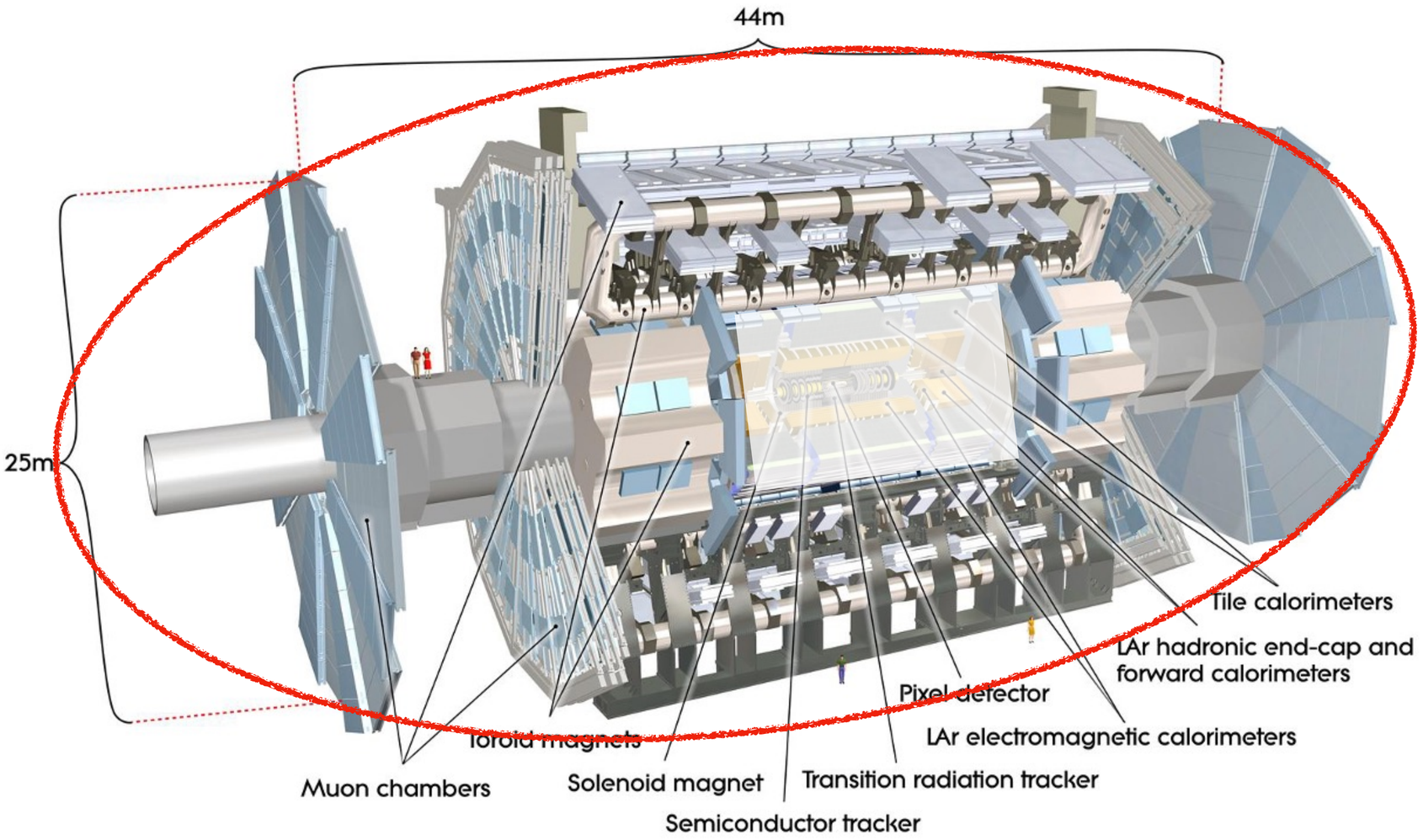


- General purpose detector located at point 1 on the LHC ring
- Right-handed coordinate system
- Components:
  - Inner detector
  - Calorimeters

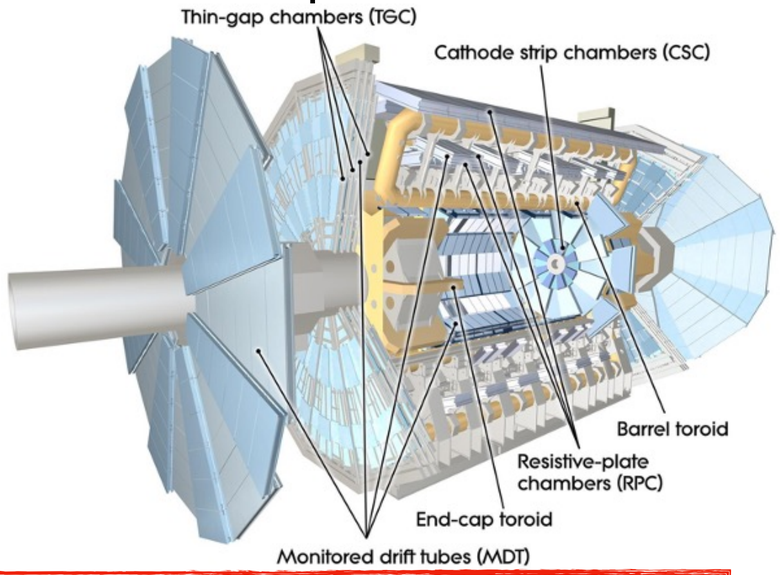


- Electromagnetic calorimeter: electrons and photons
- Hadronic calorimeter: all other particles except for muons

The ATLAS detector



- General purpose detector located at point 1 on the LHC ring
- Right-handed coordinate system
- Components:
  - Inner detector
  - Calorimeters
  - Muon spectrometer

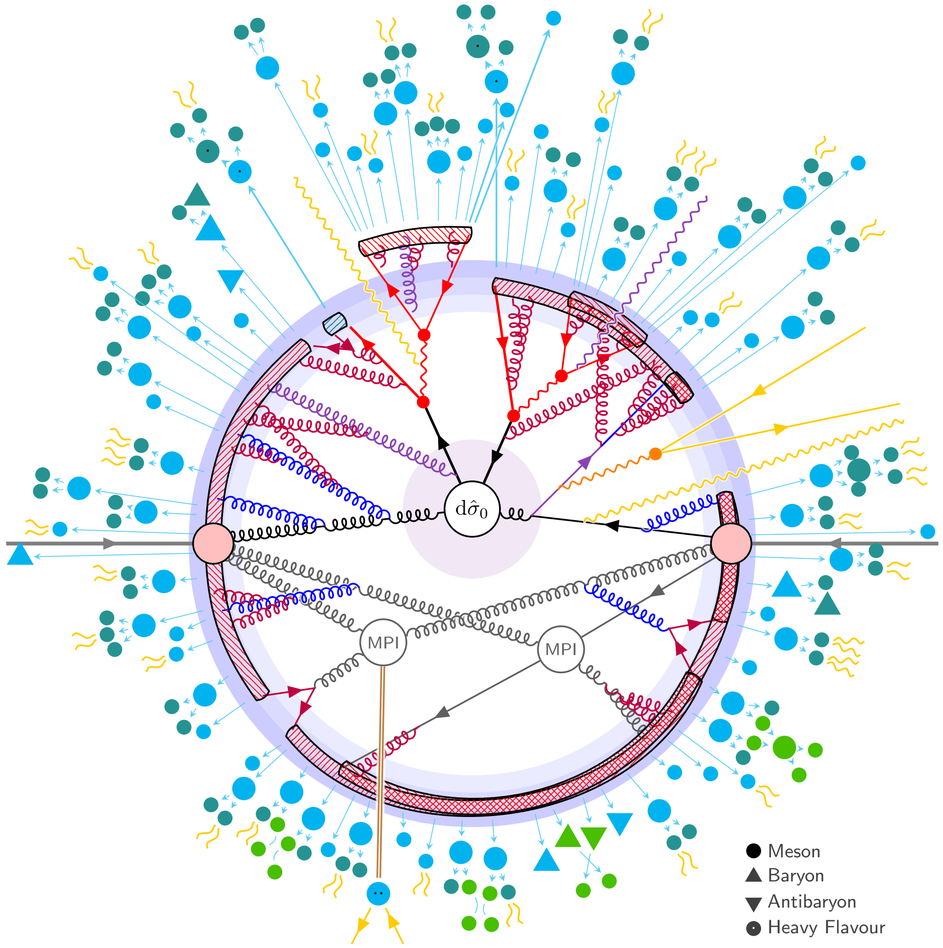


- Dedicated system to perform precision muon measurements

Part 2: Unfolding

Types of samples

Types of samples



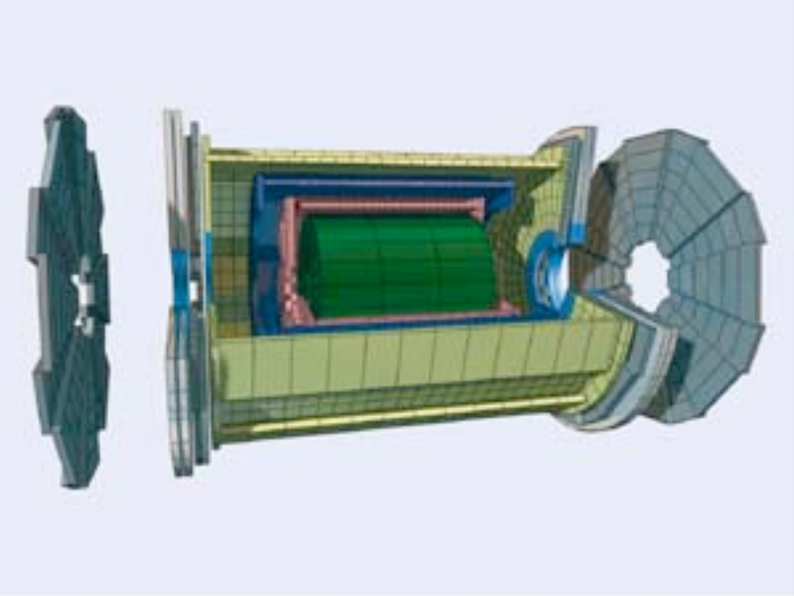
- Hard Interaction
- Resonance Decays
- MECs, Matching & Merging
- FSR
- ISR*
- QED
- Weak Showers
- Hard Onium
- Multiparton Interactions
- Beam Remnants*
- Strings
- Ministrings / Clusters
- Colour Reconnections
- String Interactions
- Bose-Einstein & Fermi-Dirac
- Primary Hadrons
- Secondary Hadrons
- Hadronic Reinteractions

(: incoming lines are crossed)*

Truth MC

- Simulates the stable particles produced from the proton-proton collision
- An event is composed of an event weight and a list of particle 4-vectors
- Many different Monte Carlo generators used within ATLAS (e.g. Powheg+Pythia, Sherpa, Madgraph)

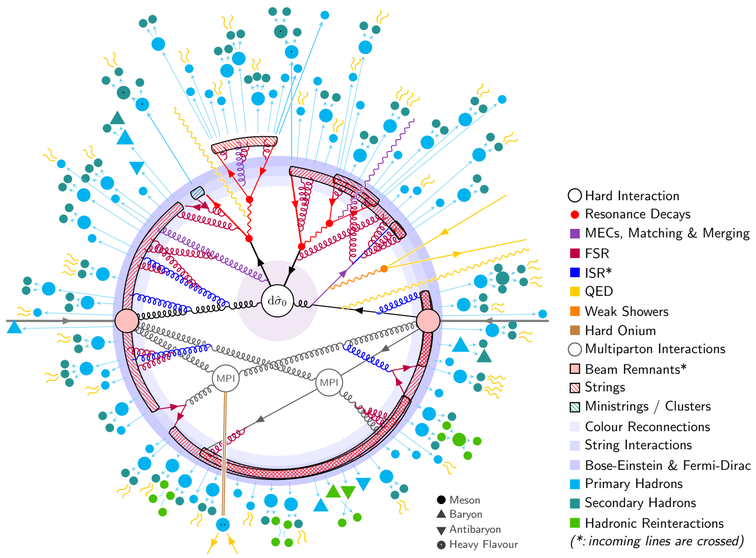
Types of samples



Reconstructed MC

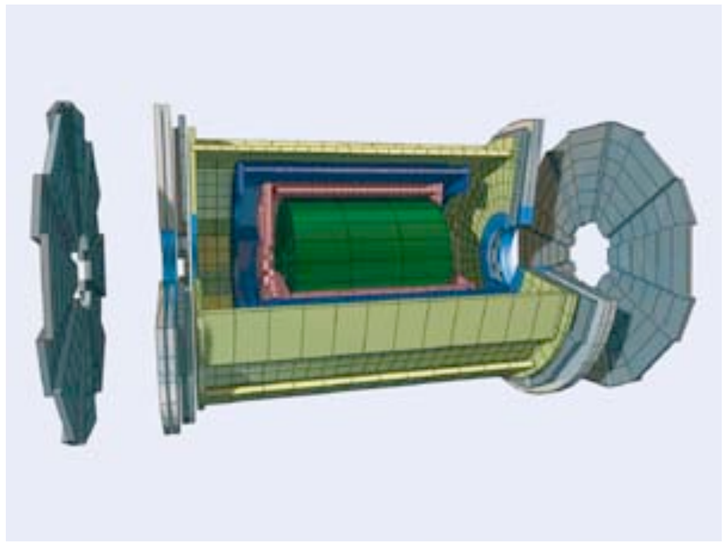
- Truth MC events are run through the Geant4 simulation of the ATLAS detector to get hits for digitization
- Digitized data then run through the ATLAS reconstruction software

Truth MC

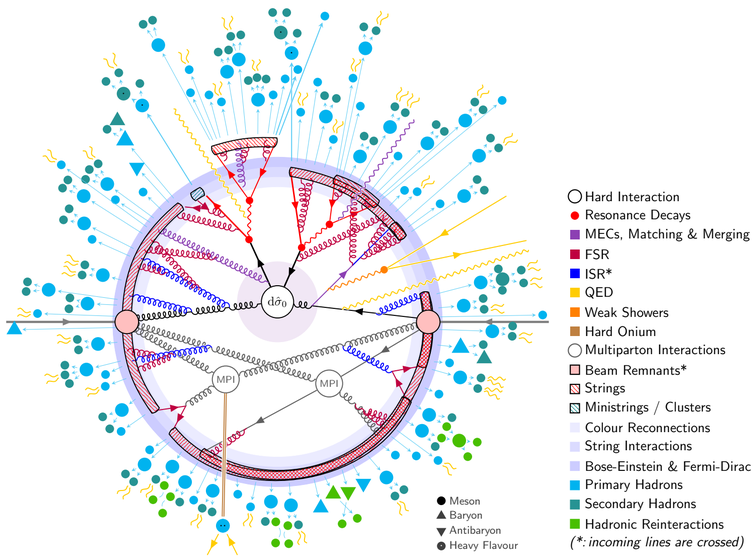


Types of samples

Reconstructed MC



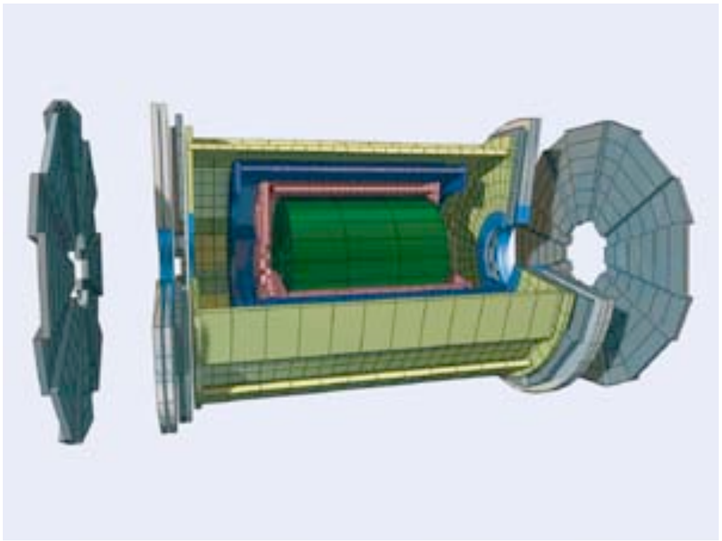
Truth MC



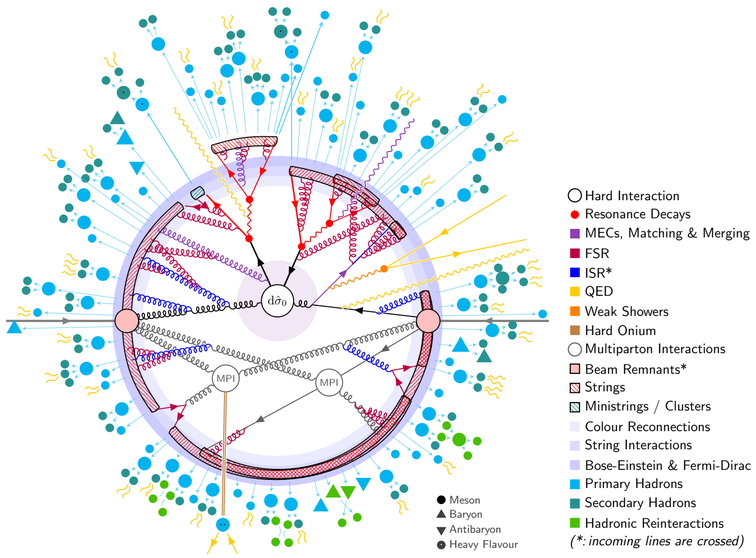
Types of samples

Monte Carlo sample

Reconstructed MC



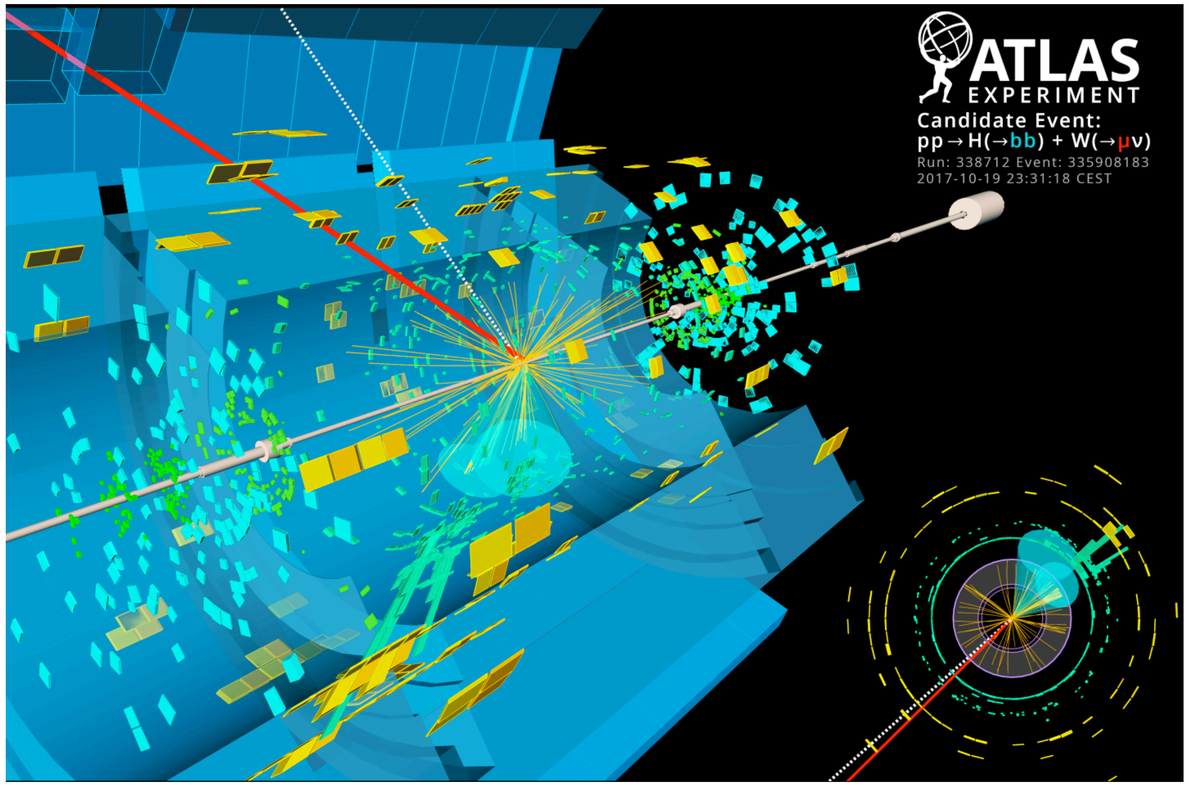
Truth MC



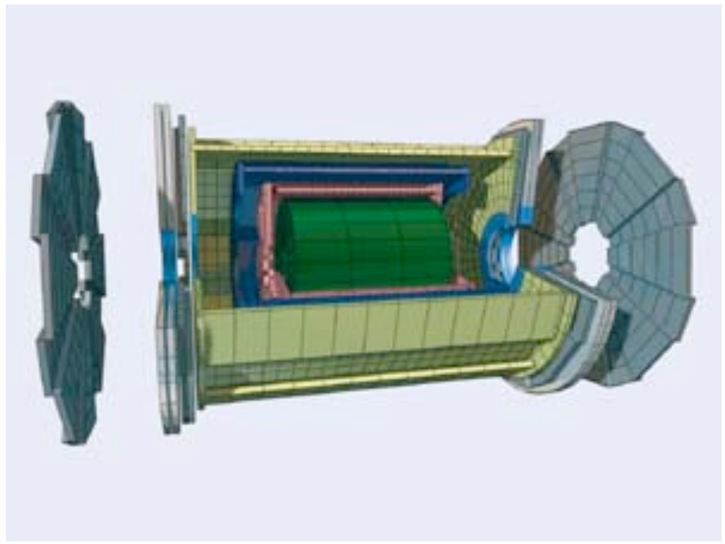
Types of samples

Data

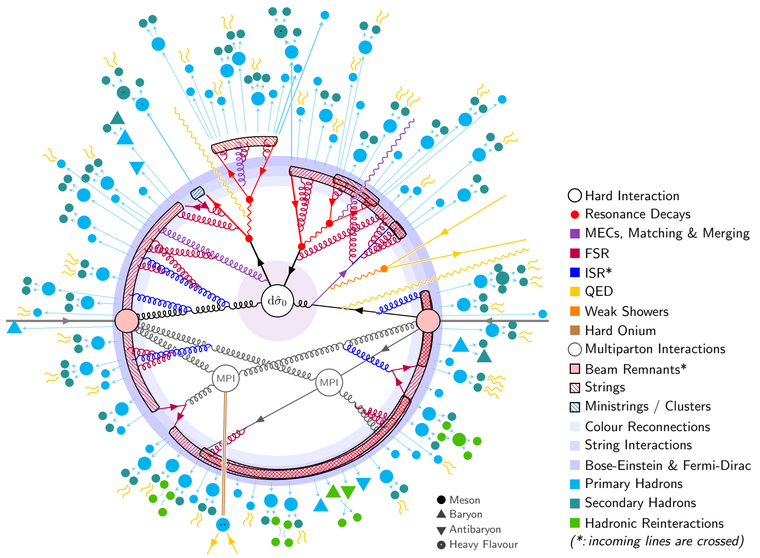
- What we actually measure in the ATLAS detector
- ATLAS reconstruction algorithms applied to the detector readout
- Gives us the 4-vectors of each particle in the event



Reconstructed MC

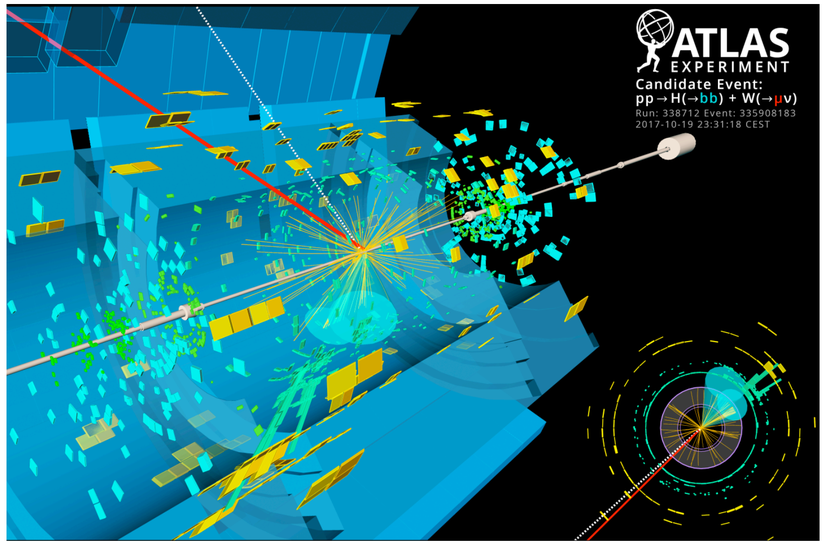


Truth MC

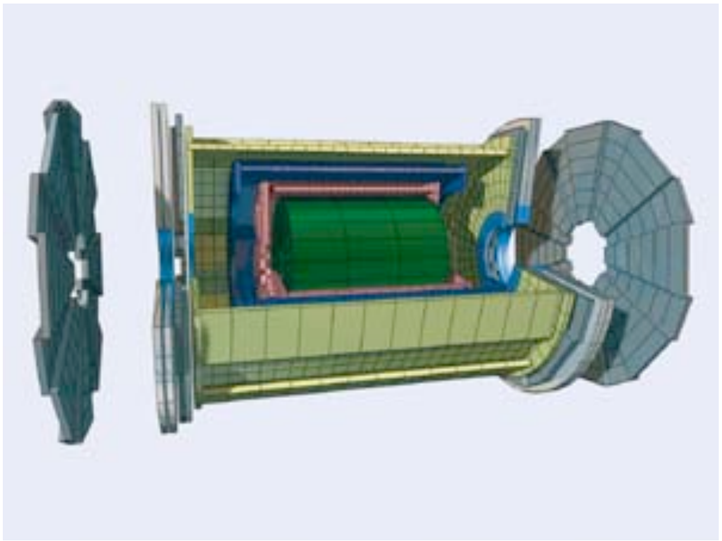


Types of samples

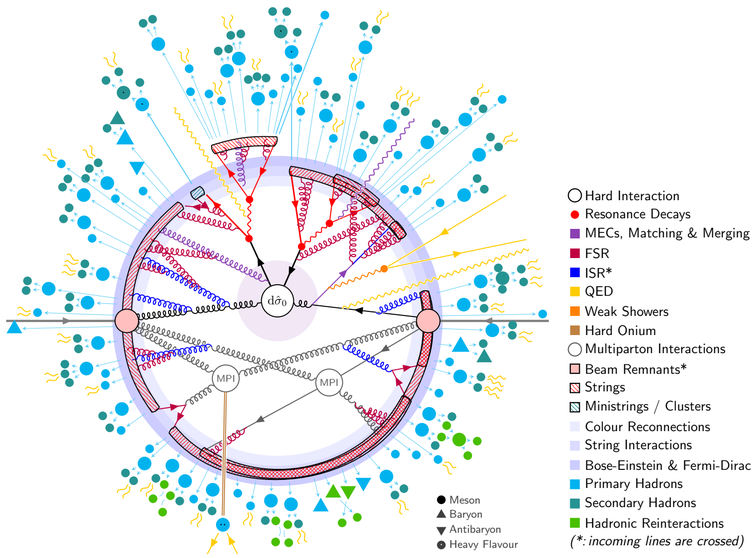
Data



Reconstructed MC

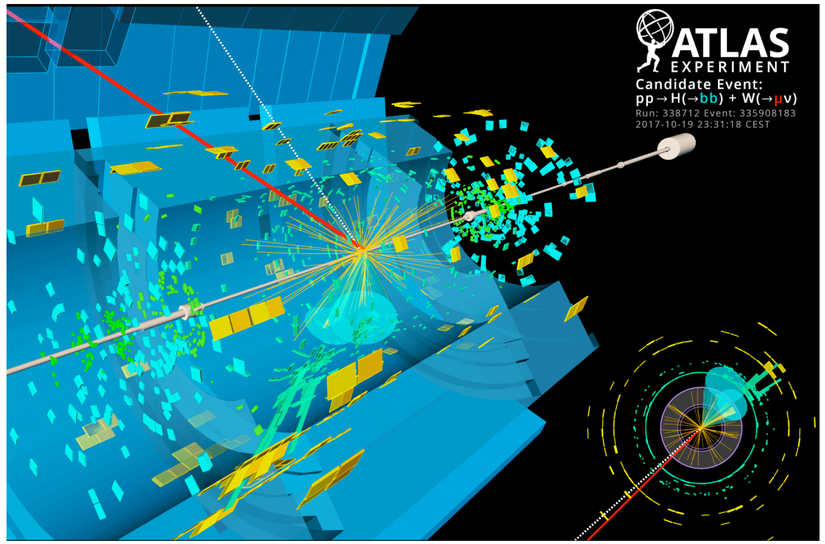


Truth MC



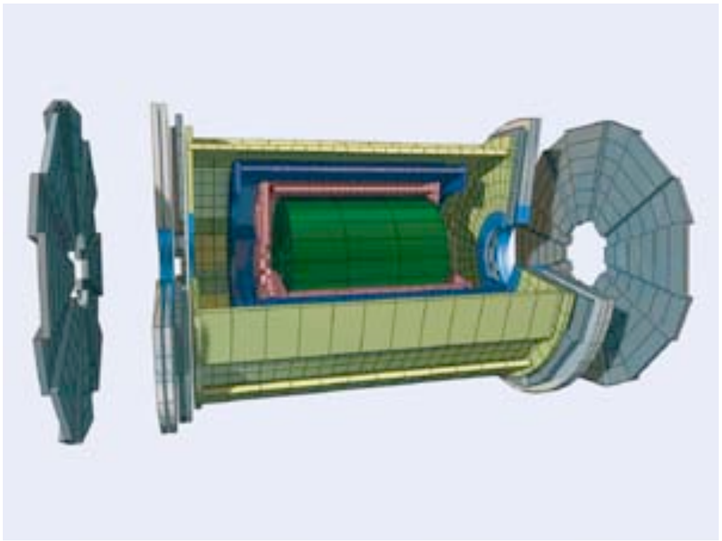
Types of samples

Data

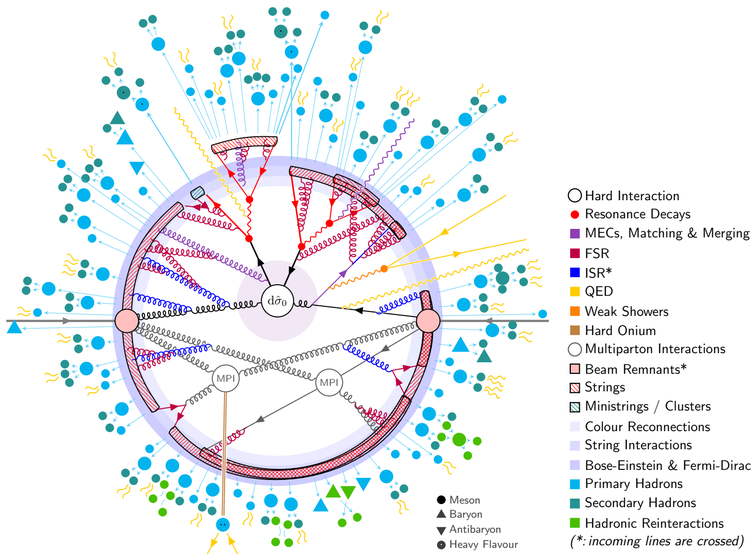


Truth

Reconstructed MC

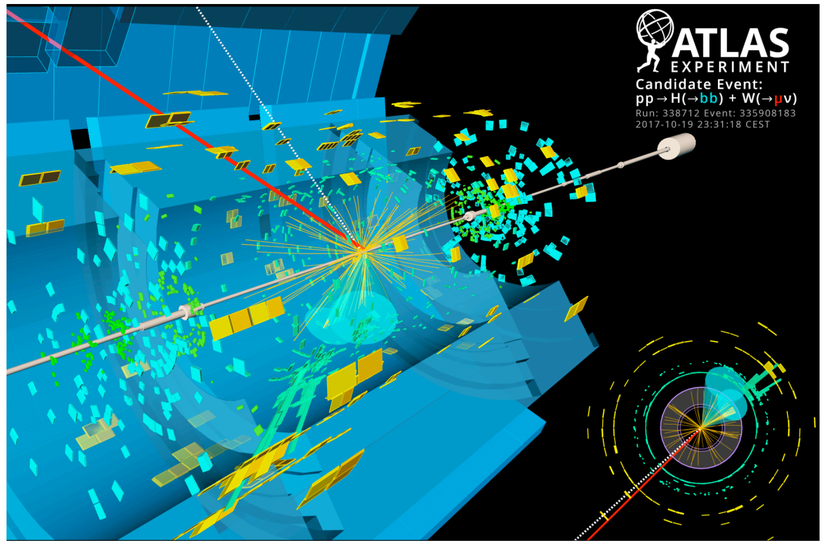


Truth MC



Types of samples

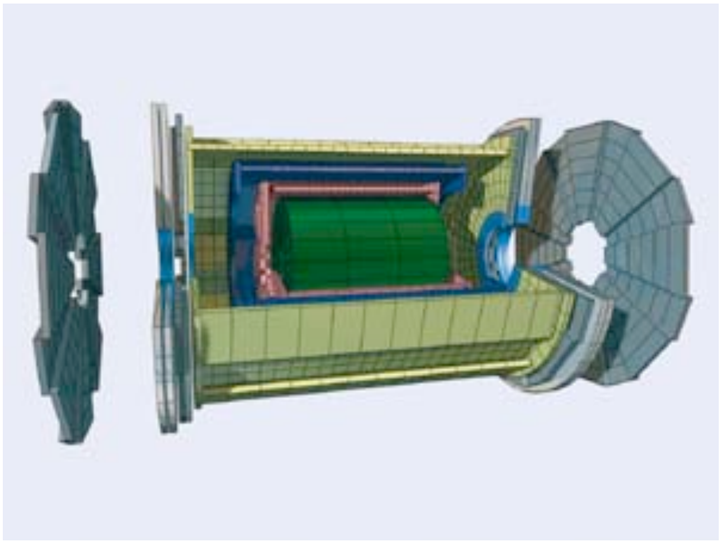
Data



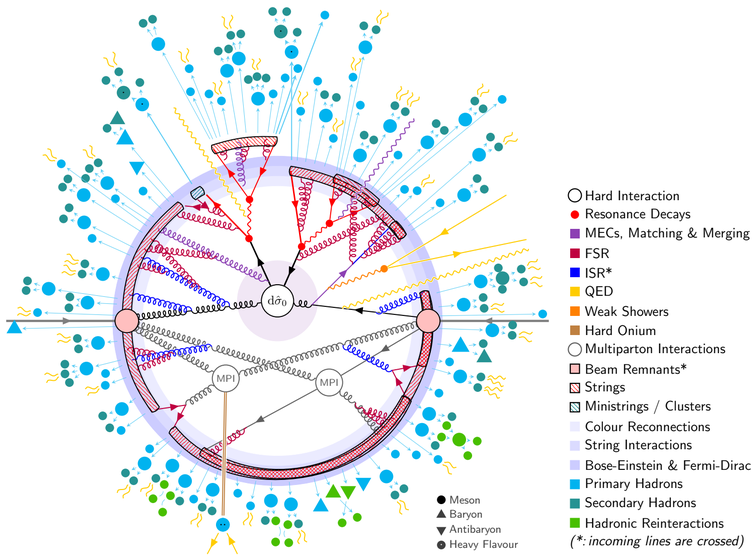
Truth

???

Reconstructed MC

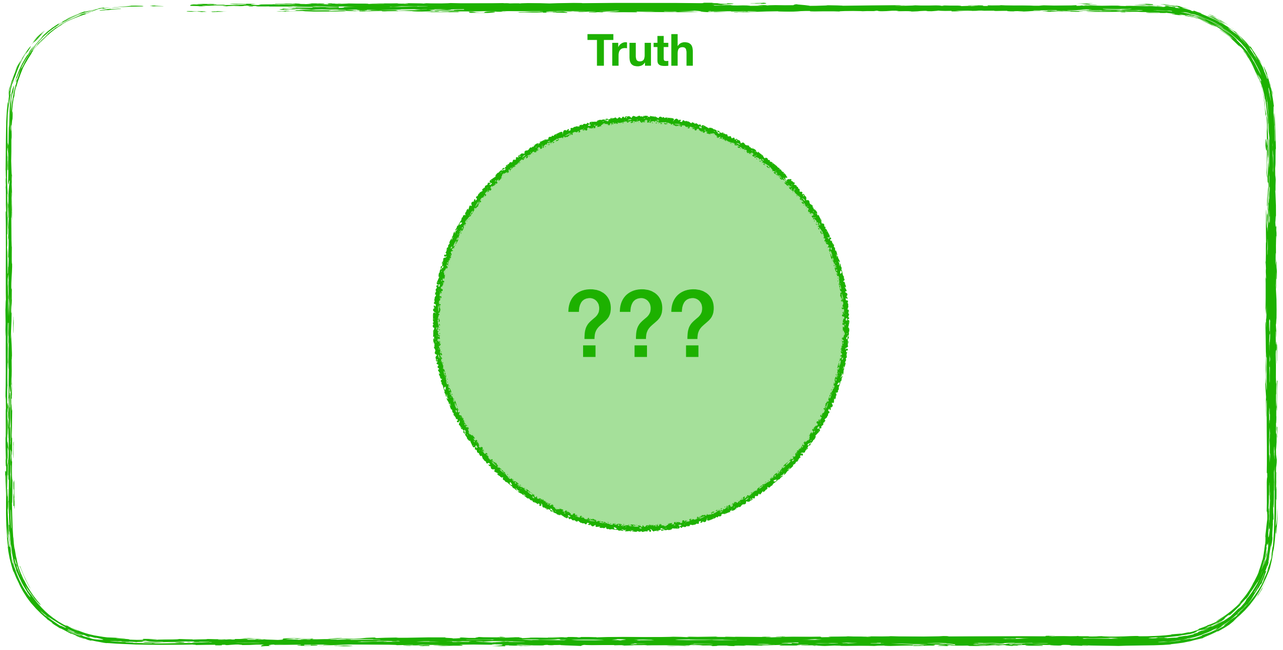


Truth MC



Measurements

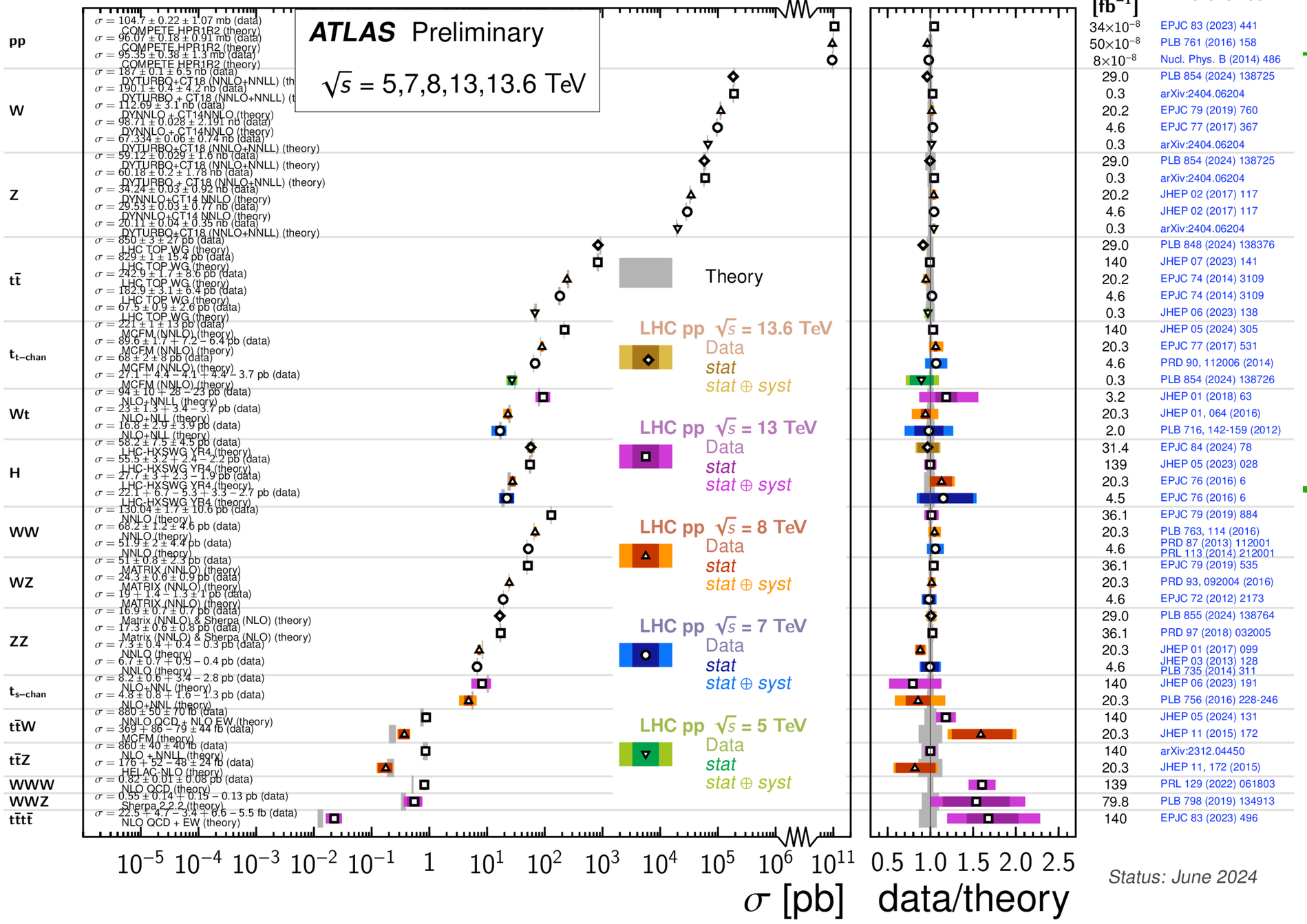
- When ATLAS presents results, they are presented at the truth level
- Needed to:
  - Compare with theory
  - Compare with other experimental results



Me

- Wh
  pre
- Ne
  - (
  - (

Standard Model Total Production Cross Section Measurements

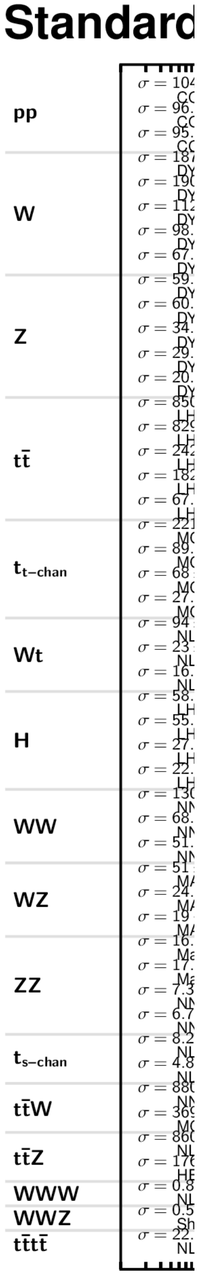


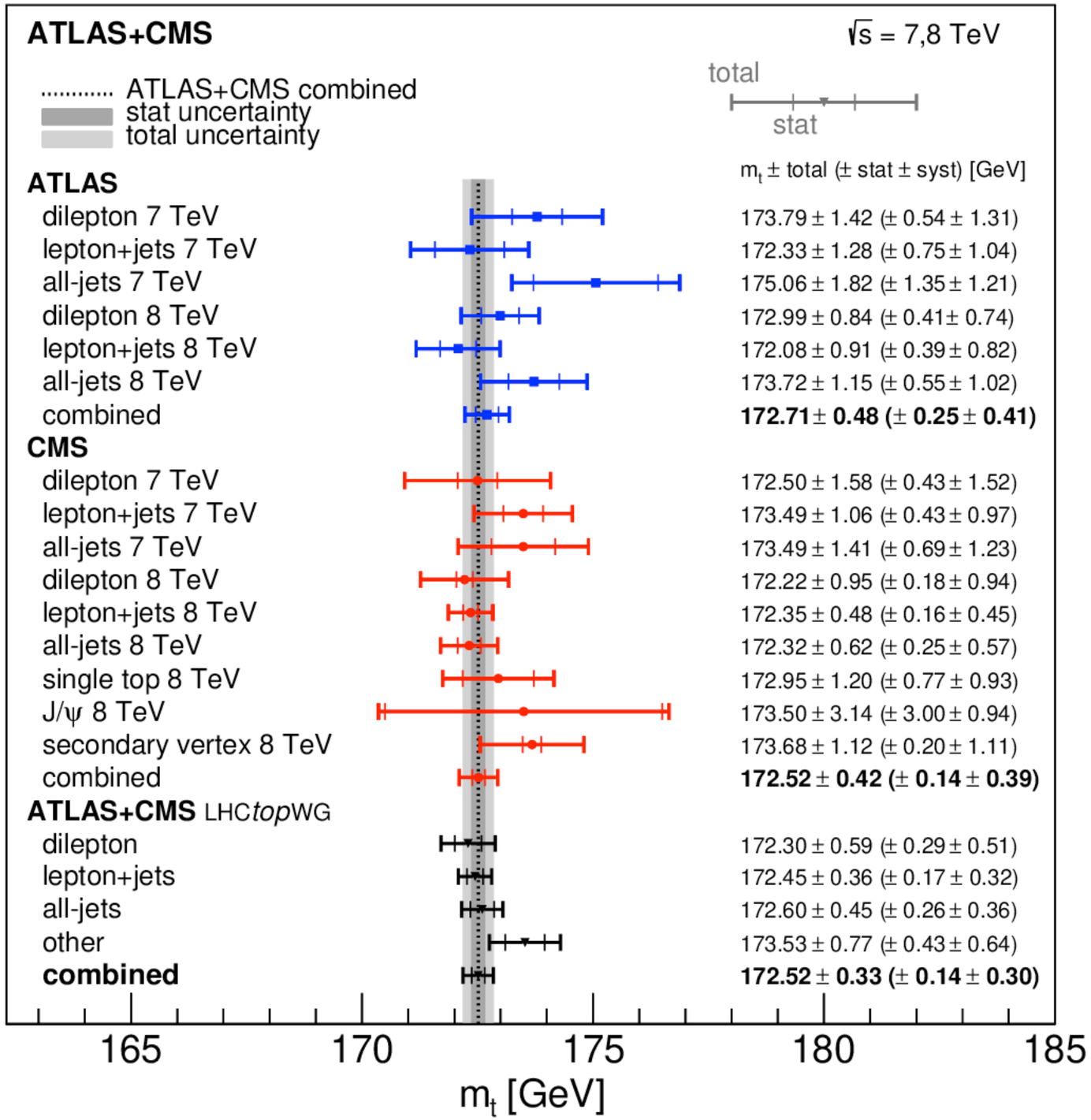
$\int \mathcal{L}\,dt$ [fb^{-1}]	Reference
34×10^{-8}	EPJC 83 (2023) 441
50×10^{-8}	PLB 761 (2016) 158
8×10^{-8}	Nucl. Phys. B (2014) 486
29.0	PLB 854 (2024) 138725
0.3	arXiv:2404.06204
20.2	EPJC 79 (2019) 760
4.6	EPJC 77 (2017) 367
0.3	arXiv:2404.06204
29.0	PLB 854 (2024) 138725
0.3	arXiv:2404.06204
20.2	JHEP 02 (2017) 117
4.6	JHEP 02 (2017) 117
0.3	arXiv:2404.06204
29.0	PLB 848 (2024) 138376
140	JHEP 07 (2023) 141
20.2	EPJC 74 (2014) 3109
4.6	EPJC 74 (2014) 3109
0.3	JHEP 06 (2023) 138
140	JHEP 05 (2024) 305
20.3	EPJC 77 (2017) 531
4.6	PRD 90, 112006 (2014)
0.3	PLB 854 (2024) 138726
3.2	JHEP 01 (2018) 63
20.3	JHEP 01, 064 (2016)
2.0	PLB 716, 142-159 (2012)
31.4	EPJC 84 (2024) 78
139	JHEP 05 (2023) 028
20.3	EPJC 76 (2016) 6
4.5	EPJC 76 (2016) 6
36.1	EPJC 79 (2019) 884
20.3	PLB 763, 114 (2016)
4.6	PRD 87 (2013) 112001 PRL 113 (2014) 212001
36.1	EPJC 79 (2019) 535
20.3	PRD 93, 092004 (2016)
4.6	EPJC 72 (2012) 2173
29.0	PLB 855 (2024) 138764
36.1	PRD 97 (2018) 032005
20.3	JHEP 01 (2017) 099
4.6	JHEP 03 (2013) 128 PLB 735 (2014) 311
140	JHEP 06 (2023) 191
20.3	PLB 756 (2016) 228-246
140	JHEP 05 (2024) 131
20.3	JHEP 11 (2015) 172
140	arXiv:2312.04450
20.3	JHEP 11, 172 (2015)
139	PRL 129 (2022) 061803
79.8	PLB 798 (2019) 134913
140	EPJC 83 (2023) 496

Status: June 2024

Me

- Wh
  pre
- Ne
  - (
  - (





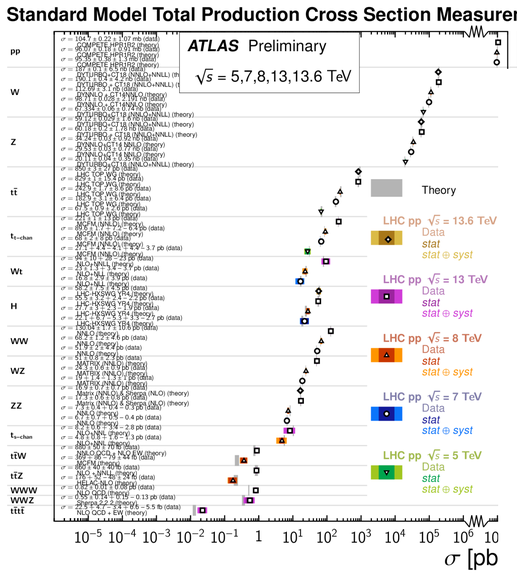
Reference

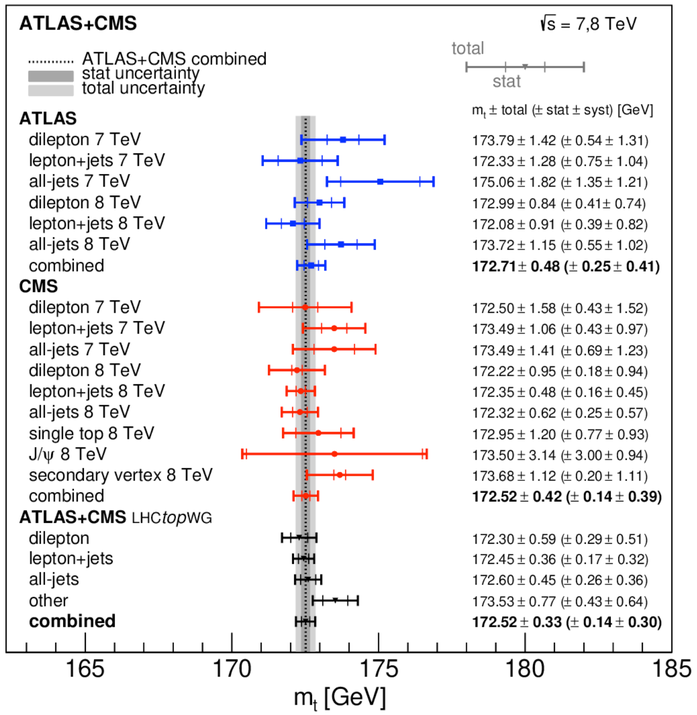
EPJC 83 (2023) 441
PLB 761 (2016) 158
Nucl. Phys. B (2014) 486
PLB 854 (2024) 138725
arXiv:2404.06204
EPJC 79 (2019) 760
EPJC 77 (2017) 367
arXiv:2404.06204
PLB 854 (2024) 138725
arXiv:2404.06204
JHEP 02 (2017) 117
JHEP 02 (2017) 117
arXiv:2404.06204
PLB 848 (2024) 138376
JHEP 07 (2023) 141
EPJC 74 (2014) 3109
EPJC 74 (2014) 3109
JHEP 06 (2023) 138
JHEP 05 (2024) 305
EPJC 77 (2017) 531
PRD 90, 112006 (2014)
PLB 854 (2024) 138726
JHEP 01 (2018) 63
JHEP 01, 064 (2016)
PLB 716, 142-159 (2012)
EPJC 84 (2024) 78
JHEP 05 (2023) 028
EPJC 76 (2016) 6
EPJC 76 (2016) 6
EPJC 79 (2019) 884
PLB 763, 114 (2016)
PRD 87 (2013) 112001
PRL 113 (2014) 212001
EPJC 79 (2019) 535
PRD 93, 092004 (2016)
EPJC 72 (2012) 2173
PLB 855 (2024) 138764
PRD 97 (2018) 032005
JHEP 01 (2017) 099
JHEP 03 (2013) 128
PLB 735 (2014) 311
JHEP 06 (2023) 191
PLB 756 (2016) 228-246
JHEP 05 (2024) 131
JHEP 11 (2015) 172
arXiv:2312.04450
JHEP 11, 172 (2015)
PRL 129 (2022) 061803
PLB 798 (2019) 134913
EPJC 83 (2023) 496

: June 2024

Measurements

- When ATLAS presents results, they are presented at the truth level
- Needed to:
  - Compare with theory
  - Compare with other experimental results





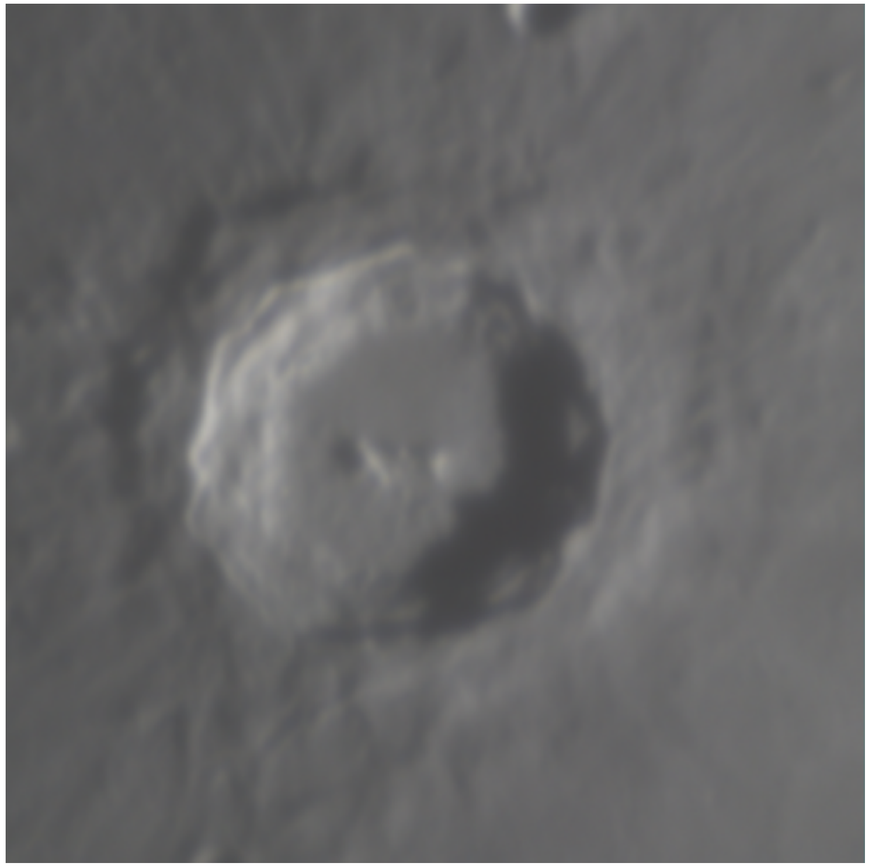
Truth

???

But how do we actually get a measurement at truth level?

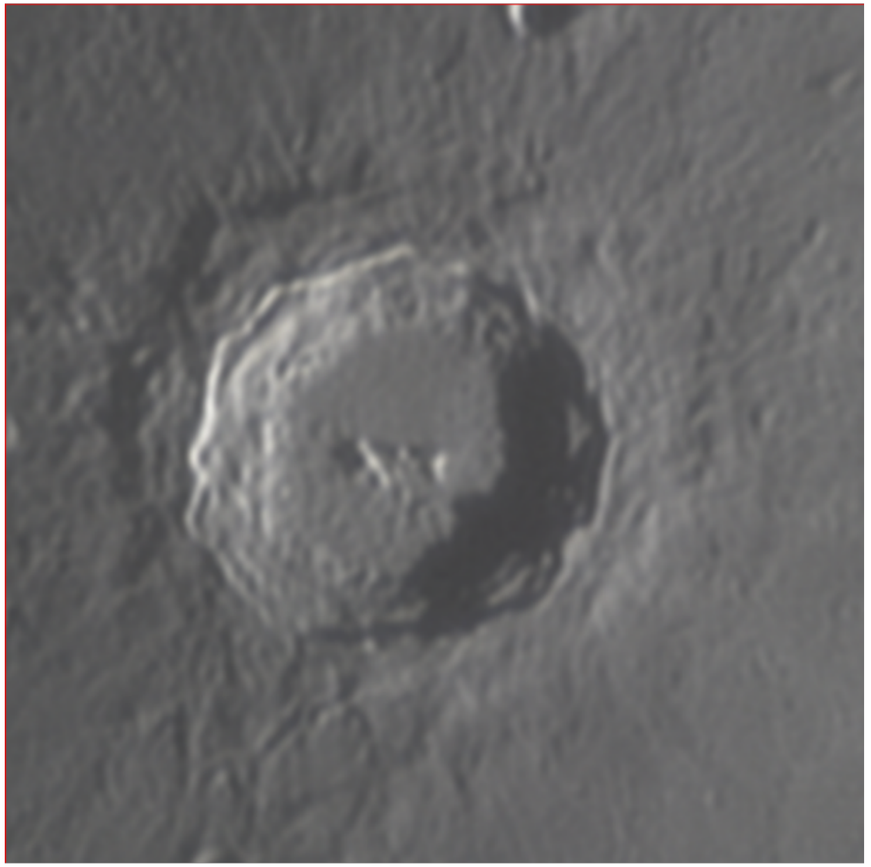
Unfolding

- Synonymous with deconvolution or unsmearing
- For example, deconvoluting images:



Deconvolution

Benefit: reduced smearing
Price: larger uncertainties



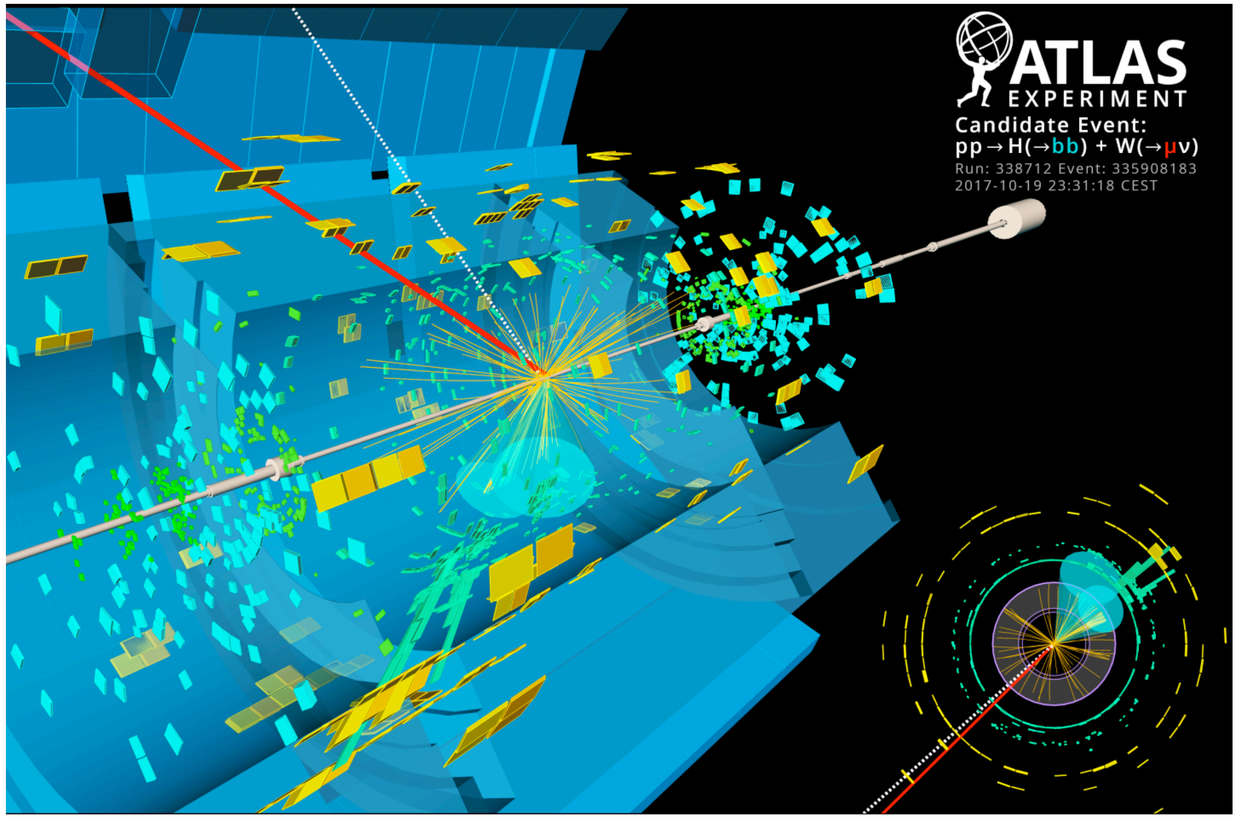
Unfolding

- At ATLAS, have to do something similar, but instead of unblurring an image we need to correct our measurement for the effects of the detector

Unfolding

- At ATLAS, have to do something similar, but instead of unblurring an image we need to correct our measurement for the effects of the detector

What we measure (detector level)

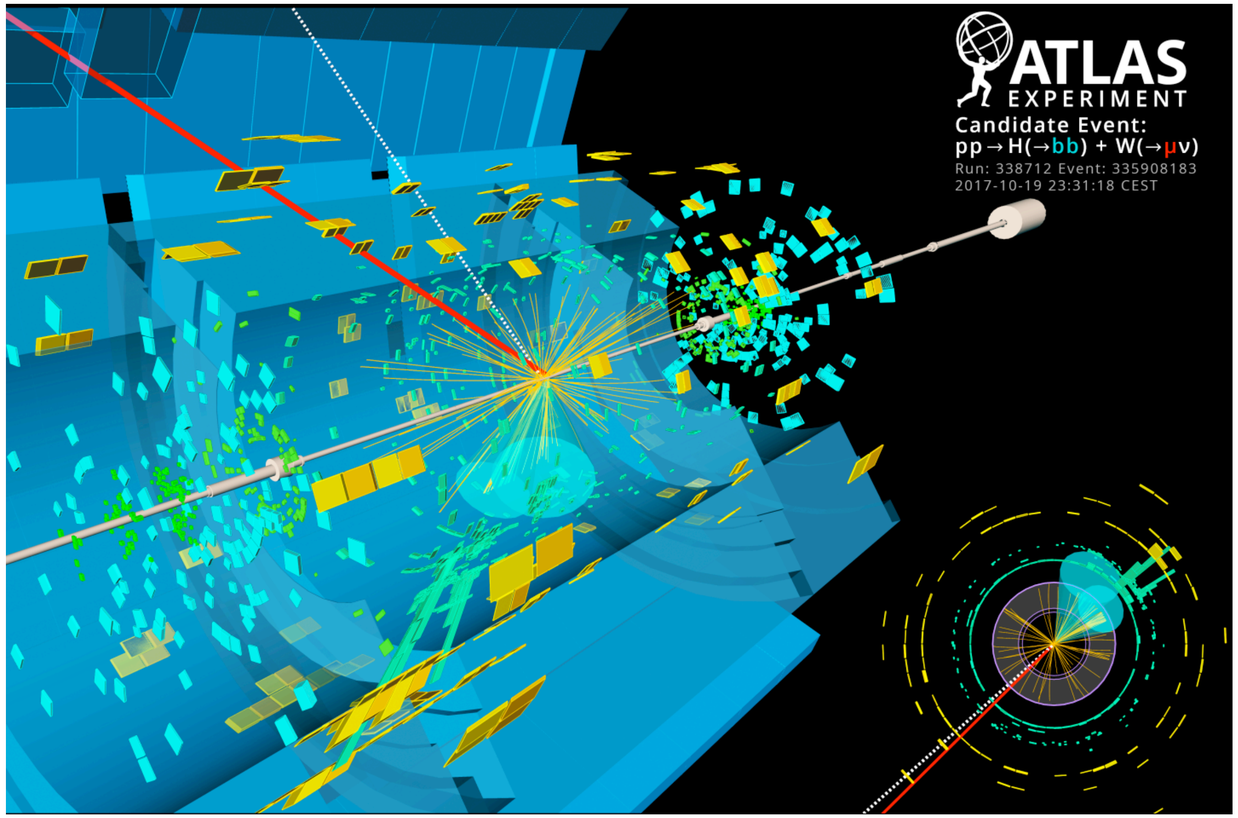


- Basic detector information (charged particle tracks, calorimeter energy deposits, hits in the muon detectors) used to reconstruct particles in an event
- Subject to resolution and efficiency effects

Unfolding

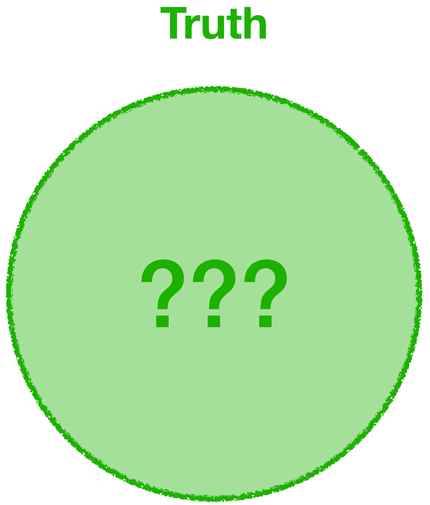
- At ATLAS, have to do something similar, but instead of unblurring an image we need to correct our measurement for the effects of the detector

What we measure (detector level)



- Basic detector information (charged particle tracks, calorimeter energy deposits, hits in the muon detectors) used to reconstruct particles in an event
- Subject to resolution and efficiency effects

What we want (particle level)

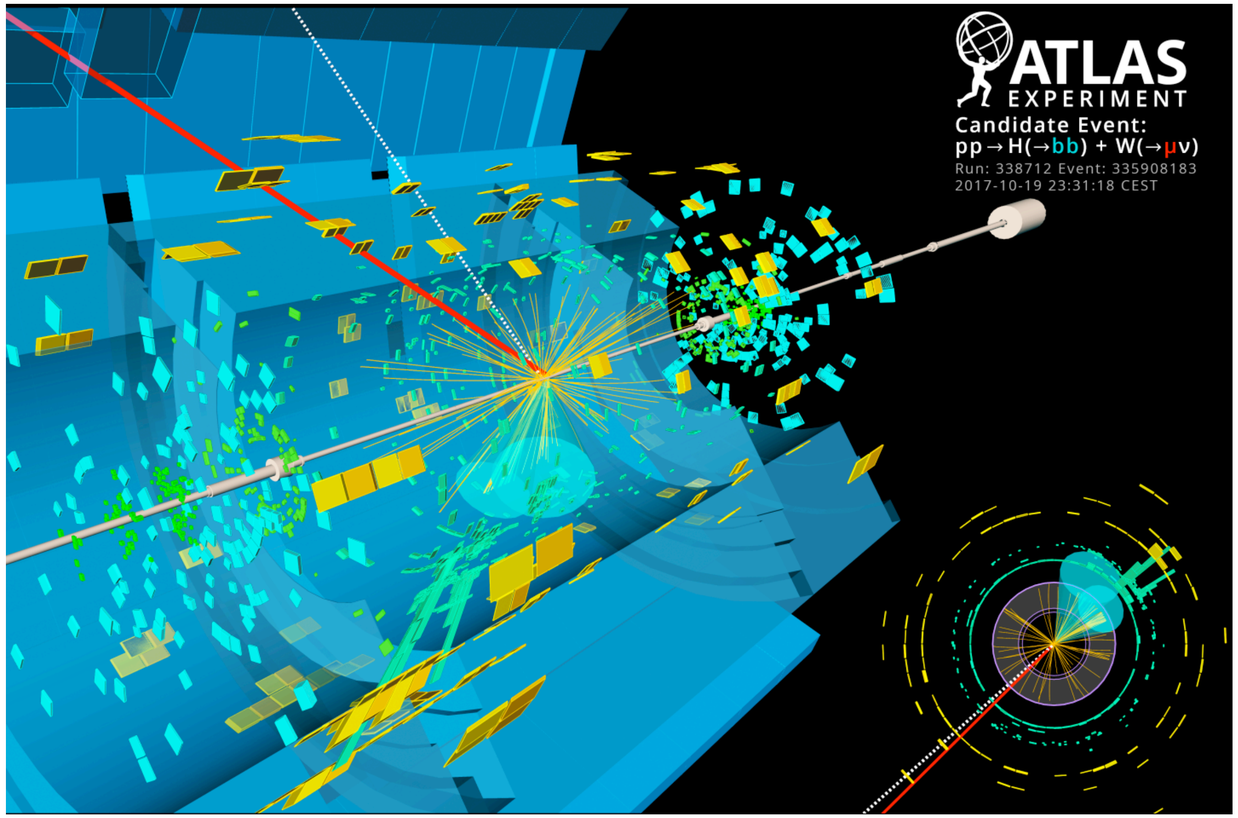


- Four vectors of all stable particles in the event
- If we had a perfect detector, this is what we would measure

Unfolding

- At ATLAS, have to do something similar, but instead of unblurring an image we need to correct our measurement for the effects of the detector

What we measure (detector level)



- Basic detector information (charged particle tracks, calorimeter energy deposits, hits in the muon detectors) used to reconstruct particles in an event
- Subject to resolution and efficiency effects

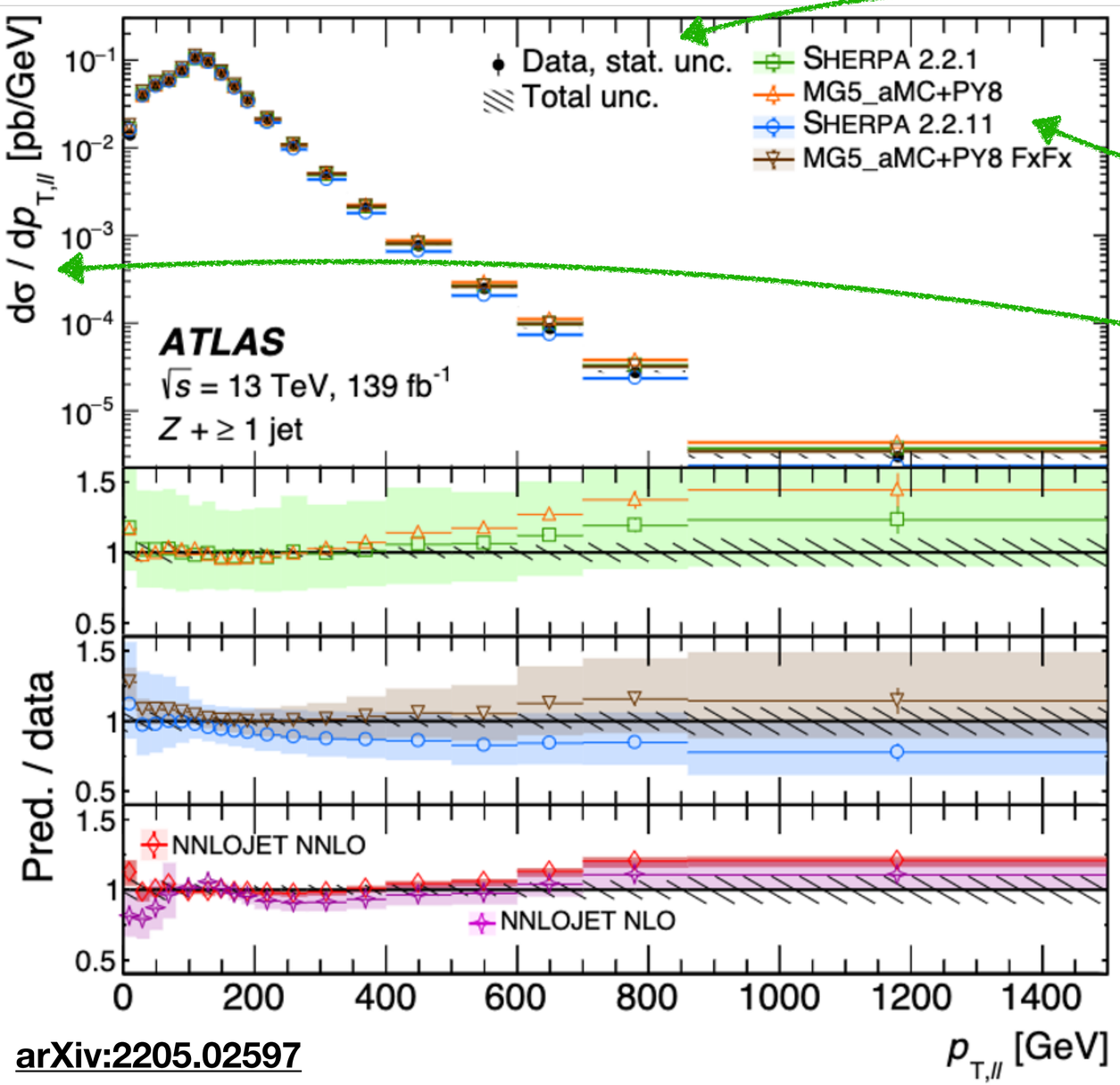
Unfolding

What we want (particle level)

Truth

- Four vectors of all stable particles in the event
- If we had a perfect detector, this is what we would measure

A traditional ATLAS measurement



Unfolded data

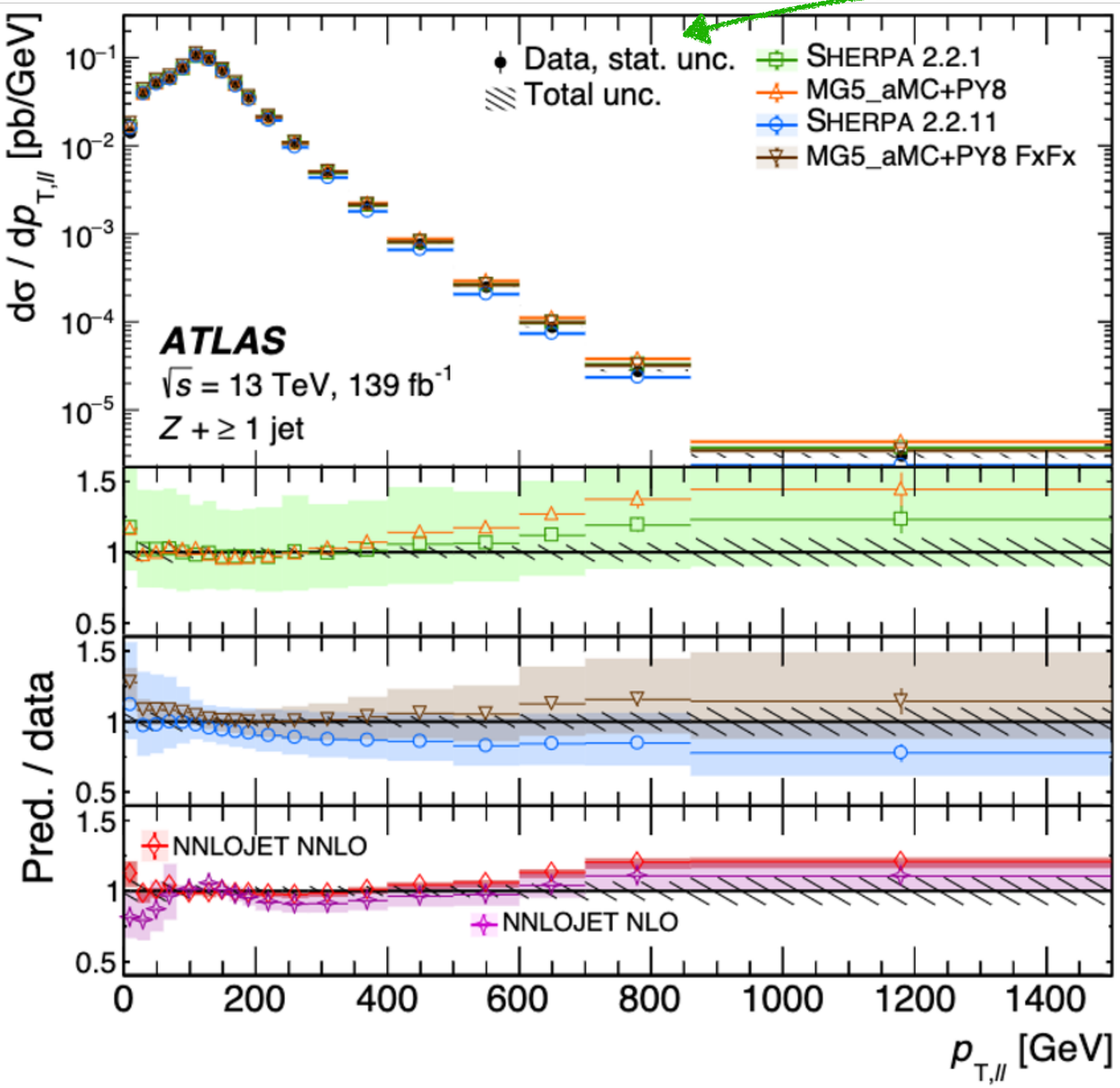
Particle-level predictions from various Monte Carlo generators

Differential cross section: related to the rate of events with respect to an observable

Observable

arXiv:2205.02597

A traditional ATLAS measurement



Unfolded data

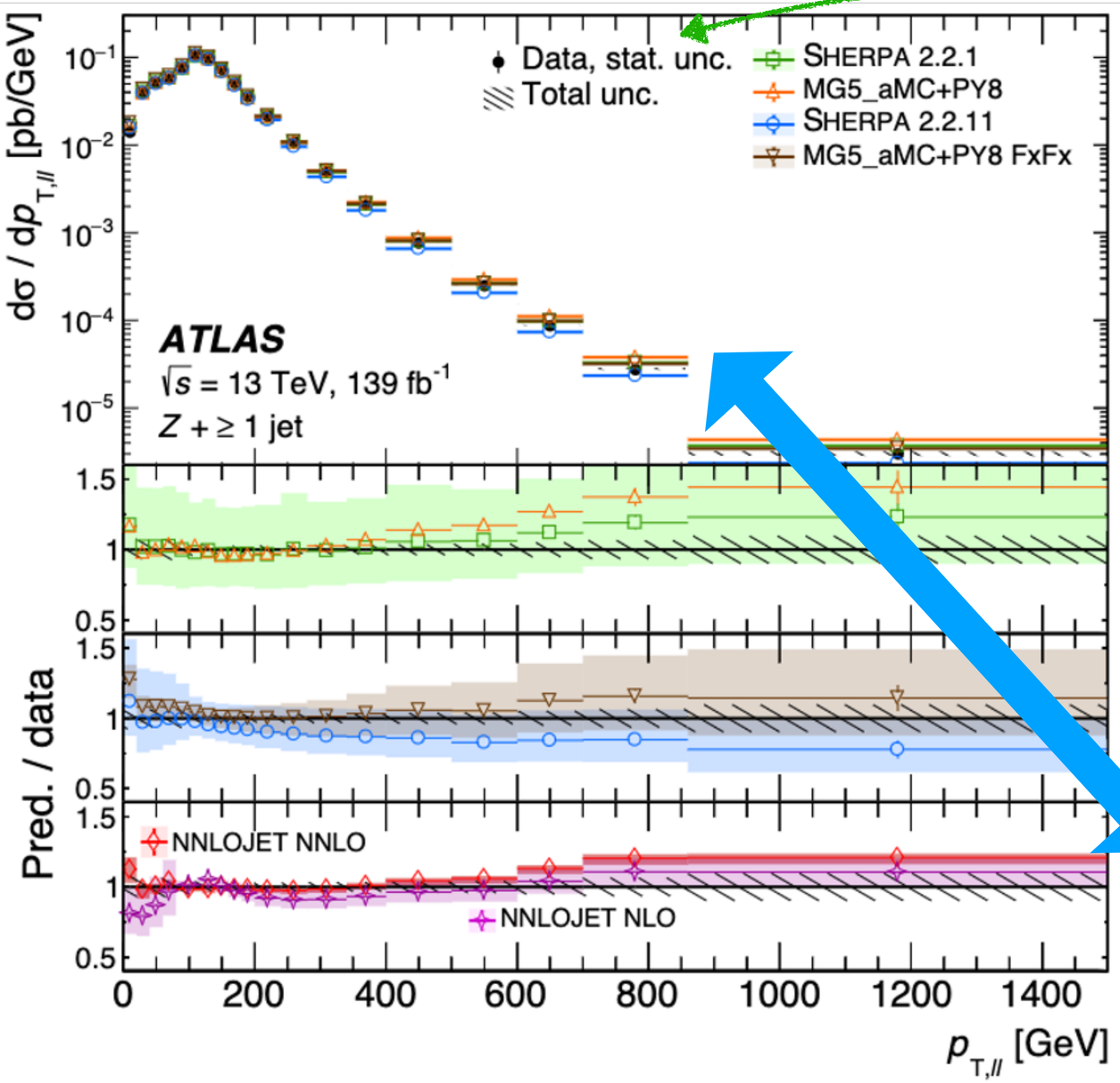
- Traditional unfolding methods: work with 1D binned data
  - Iterative Bayesian unfolding is one typical method in ATLAS

Create 1D histogram of the observable with the data events

Use information about the detector response from the MC samples to correct the histogram for detector effects in each bin

1D observable histogram with number of unfolded events per bin as output

A traditional ATLAS measurement



Unfolded data

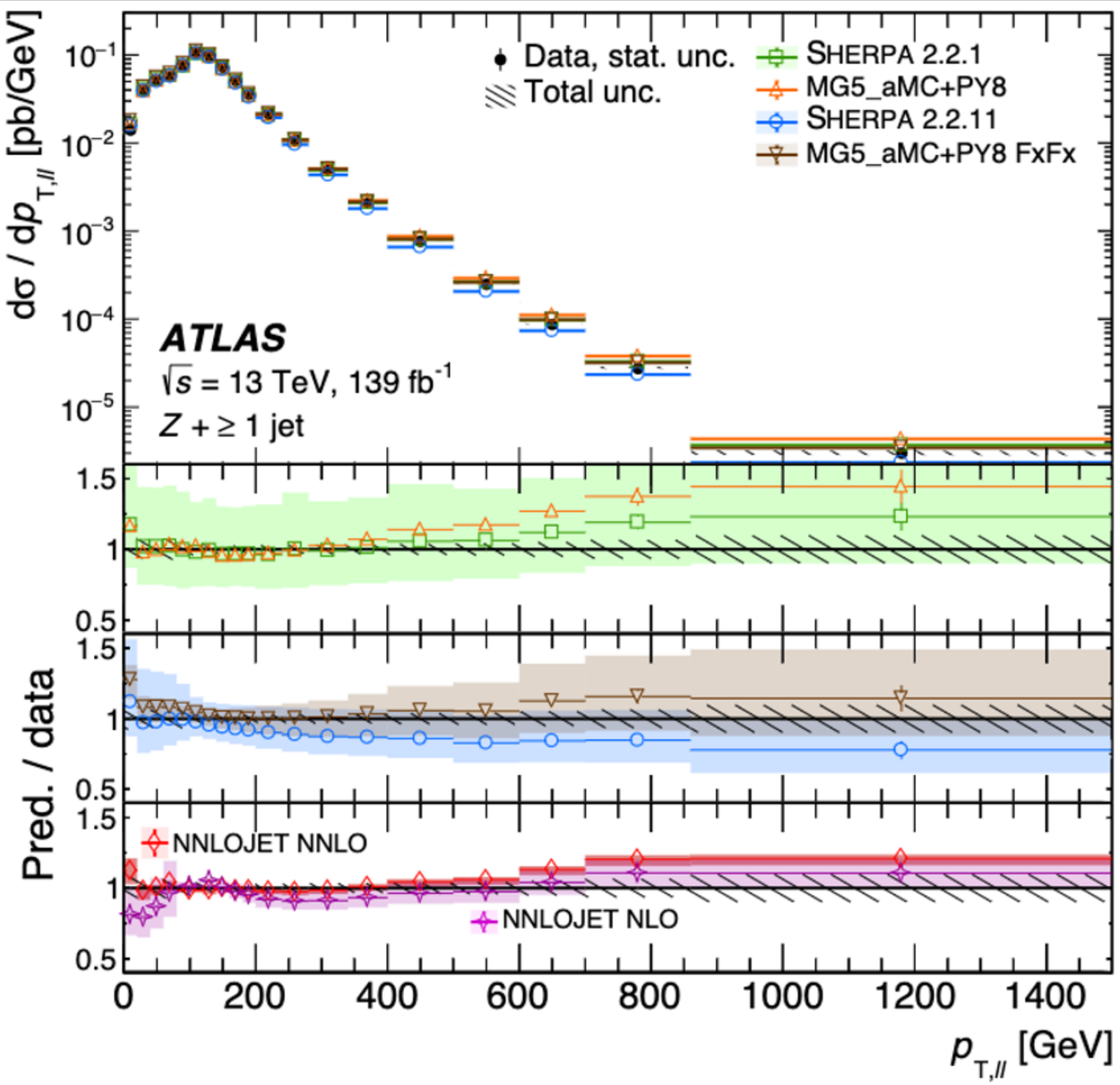
- Traditional unfolding methods: work with 1D binned data
  - Iterative Bayesian unfolding is one typical method in ATLAS

Create 1D histogram of the observable with the data events

Use information about the detector response from the MC samples to correct the histogram for detector effects in each bin

1D observable histogram with number of unfolded events per bin as output

A traditional ATLAS measurement



- Traditional unfolding methods present some challenges:
  1. The data must be binned
  2. Can only unfold a small number of observables simultaneously
  3. Do not consider the full phase space and so may miss hidden dependencies
- Can we leverage machine learning to perform unfolding in an **unbinned** and **highly-dimensional** way?

At its core, can consider unfolding a reweighting problem

Density ratio estimation with NNs

- Neural networks are well suited to carrying out reweighting tasks
- Basic principle behind density ratio estimation:
  - Imagine we have:
    - Dataset A with probability density function $p_A(\vec{x})$
    - Dataset B with probability density function $p_B(\vec{x})$

Both datasets cover the same phase space Ω

$\vec{x} \in \Omega$ represents an event

Density ratio estimation with NNs

- Neural networks are well suited to carrying out reweighting tasks
- Basic principle behind density ratio estimation:
  - Imagine we have:
    - Dataset A with probability density function $p_A(\vec{x})$
    - Dataset B with probability density function $p_B(\vec{x})$

    Both datasets cover the same phase space Ω

    $\vec{x} \in \Omega$ represents an event
  - If we wanted to reweight Dataset A to better match Dataset B, the per-event weight, $r(\vec{x})$, that we would need to apply to each event in Dataset A would be:

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

Density ratio estimation with NNs

- Neural networks are well suited to carrying out reweighting tasks
- Basic principle behind density ratio estimation:
  - Imagine we have:
    - Dataset A with probability density function $p_A(\vec{x})$
    - Dataset B with probability density function $p_B(\vec{x})$

    Both datasets cover the same phase space Ω

    $\vec{x} \in \Omega$ represents an event
  - If we wanted to reweight Dataset A to better match Dataset B, the per-event weight, $r(\vec{x})$, that we would need to apply to each event in Dataset A would be:

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

Estimating these directly turns out to be complicated

Density ratio estimation with NNs

- Neural networks are well suited to carrying out reweighting tasks
- Basic principle behind density ratio estimation:
  - Imagine we have:
    - Dataset A with probability density function $p_A(\vec{x})$
    - Dataset B with probability density function $p_B(\vec{x})$

    Both datasets cover the same phase space Ω

    $\vec{x} \in \Omega$ represents an event
  - If we wanted to reweight Dataset A to better match Dataset B, the per-event weight, $r(\vec{x})$, that we would need to apply to each event in Dataset A would be:

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

Actually much easier to estimate this directly!

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$r(\vec{x}) = \frac{P(A)}{P(B)} \frac{p(B\,|\,\vec{x})}{p(A\,|\,\vec{x})}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$r(\vec{x}) = \frac{P(A)}{P(B)} \frac{p(B\,|\,\vec{x})}{p(A\,|\,\vec{x})}$$

Normalization factor dependent on number of events in each sample, let's ignore this for now

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

$\boldsymbol{X}$: **random set of input features**

$Y \in \{0,1\}$: **random variable output labelling one of the two datasets**

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

Expectation value over the joint probability distribution

Chosen loss function

Specific trained classifier

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)}\frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)}\frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$\Rightarrow$ Optimal solution: $f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$\Rightarrow$ Optimal solution: $f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)}\frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$$\Rightarrow \text{Optimal solution: } f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y\,|\,\boldsymbol{X} = \vec{x}],\ 1 - f(\vec{x}) \to E[1-Y\,|\,\boldsymbol{X} = \vec{x}]$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$$\Rightarrow \text{Optimal solution: } f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y\,|\,\boldsymbol{X} = \vec{x}],\ 1 - f(\vec{x}) \to E[1-Y\,|\,\boldsymbol{X} = \vec{x}]$$

$$\frac{f(\vec{x})}{1-f(\vec{x})} \approx \frac{E[Y\,|\,\boldsymbol{X} = \vec{x}]}{E[1-Y\,|\,\boldsymbol{X} = \vec{x}]}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$$\Rightarrow \text{Optimal solution: } f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y\,|\,\boldsymbol{X} = \vec{x}],\ 1 - f(\vec{x}) \to E[1-Y\,|\,\boldsymbol{X} = \vec{x}]$$

$$\frac{f(\vec{x})}{1-f(\vec{x})} = \frac{p(Y=1\,|\,\boldsymbol{X}=\vec{x})}{p(Y=0\,|\,\boldsymbol{X}=\vec{x})}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$$\Rightarrow \text{Optimal solution: } f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y\,|\,\boldsymbol{X} = \vec{x}],\ 1 - f(\vec{x}) \to E[1-Y\,|\,\boldsymbol{X} = \vec{x}]$$

$$\frac{f(\vec{x})}{1-f(\vec{x})} = \frac{p(\boldsymbol{X}\,|\,Y=1)}{p(\boldsymbol{X}\,|\,Y=0)} \frac{P(Y=1)}{P(Y=0)}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$$\Rightarrow \text{Optimal solution: } f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y\,|\,\boldsymbol{X} = \vec{x}],\ 1 - f(\vec{x}) \to E[1-Y\,|\,\boldsymbol{X} = \vec{x}]$$

$$\frac{f(\vec{x})}{1-f(\vec{x})} = \frac{p(\boldsymbol{X}\,|\,Y=1)}{p(\boldsymbol{X}\,|\,Y=0)}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)}\frac{P(A)}{p(A\,|\,\vec{x})P(\vec{x})}$$

$$\hat{r}(\vec{x}) = \frac{\hat{p}(B\,|\,\vec{x})}{\hat{p}(A\,|\,\vec{x})}$$

- Looks like something we may be able to do with a binary classifier!
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

Binary cross entropy loss function (maximum likelihood estimator)

$$\mathscr{L}(g(\vec{x}), Y) = -\,Y\log(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$$

$$f(\vec{x}) = \underset{g}{\text{argmax}}\, E[Y\,|\,\boldsymbol{X} = \vec{x}]\log(g(\vec{x})) + (1 - E[Y\,|\,\boldsymbol{X} = \vec{x}])\log(1-g(\vec{x}))$$

$\Rightarrow$ Optimal solution: $f(\vec{x}) = E[Y\,|\,\boldsymbol{X} = \vec{x}]$

Sigmoid activation function
(correct asymptotic behaviour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y\,|\,\boldsymbol{X} = \vec{x}],\ 1 - f(\vec{x}) \to E[1-Y\,|\,\boldsymbol{X} = \vec{x}]$$

$$\frac{f(\vec{x})}{1-f(\vec{x})} = \frac{p(\boldsymbol{X}\,|\,Y=1)}{p(\boldsymbol{X}\,|\,Y=0)}$$

Optional!

Math interlude

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Use Bayesian statistics to rewrite this

$$r(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})} = \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p(B\,|\,\vec{x})P(\vec{x})}{P(B)} \frac{P(A)}{P(A)}$$

$$\hat{r}(\vec{x}) = \frac{\hat{f}}{1-\hat{f}}$$

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1-f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Looks like somethin
  do with a binary cla
- Yes, with caveats: have to be careful with our choices of loss function and activation function

- $f(\vec{x})$: classifier to differentiate between two datasets
- Optimal form of this classifier:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{\boldsymbol{X},Y}[\mathscr{L}(g(\boldsymbol{X}), Y)]$$

- Without loss of generality, can rewrite as:

$$f(\vec{x}) = \underset{g}{\text{argmin}}\, E_{Y|\boldsymbol{X}=\vec{x}}[\mathscr{L}(g(\vec{x}), Y)\,|\,\boldsymbol{X} = \vec{x}]\ \forall\ \vec{x}$$

ction (maximum likelihood estimator)

$(g(\vec{x})) - (1-Y)\log(1-g(\vec{x}))$

$(\vec{x})) + (1-E[Y|\boldsymbol{X}=\vec{x}])\log(1-g(\vec{x}))$

on: $f(\vec{x}) = E[Y|\boldsymbol{X}=\vec{x}]$

activation function

viour and output between 0 and 1)

$$\sigma(x) = \frac{1}{1-e^{-x}}$$

$$\Rightarrow f(\vec{x}) \to E[Y|\boldsymbol{X}=\vec{x}],\ 1-f(\vec{x}) \to E[1-Y|\boldsymbol{X}=\vec{x}]$$

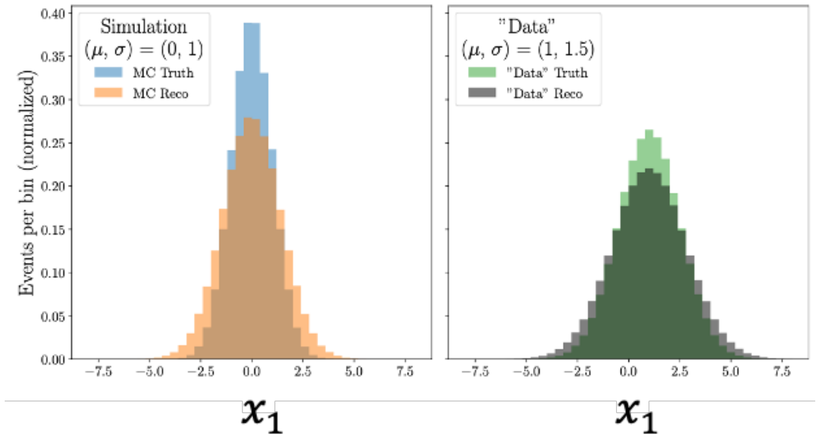
$$\frac{f(\vec{x})}{1-f(\vec{x})} = \frac{p(\boldsymbol{X}\,|\,Y=1)}{p(\boldsymbol{X}\,|\,Y=0)}$$

Sample reweighting

If you're more visual!

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Let's reweight our **reconstructed MC (Dataset A)** to match **data (Dataset B)** using a simple Gaussian example
- Each sample contains a set of events, each with a set of features $\vec{x} = (x_1, \ldots, x_{24})$

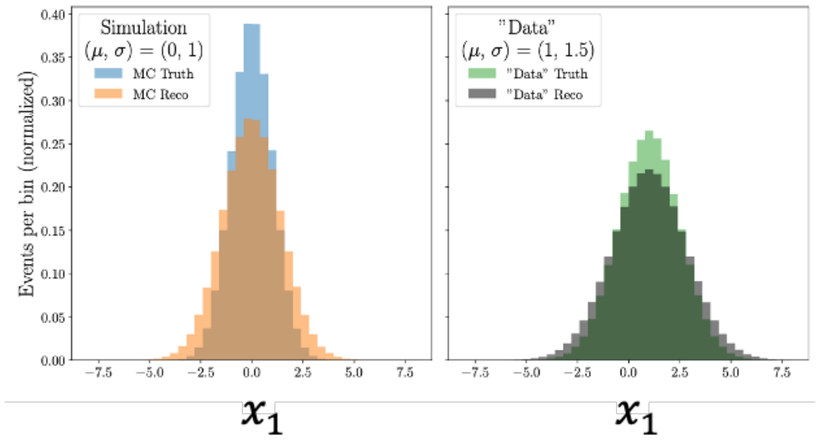


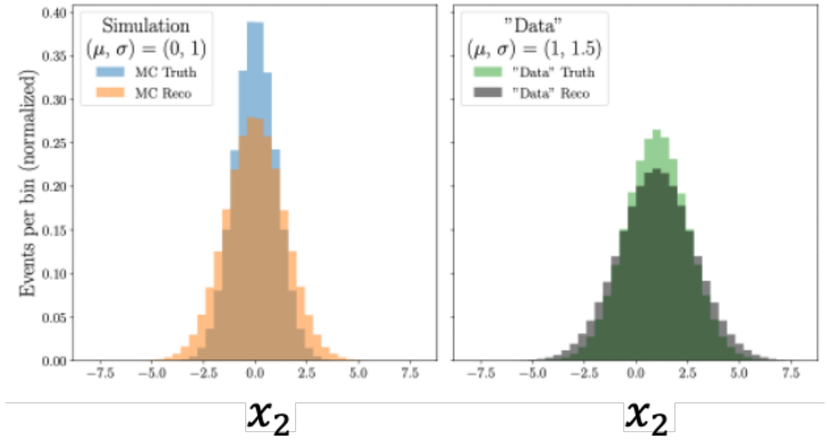
Sample reweighting

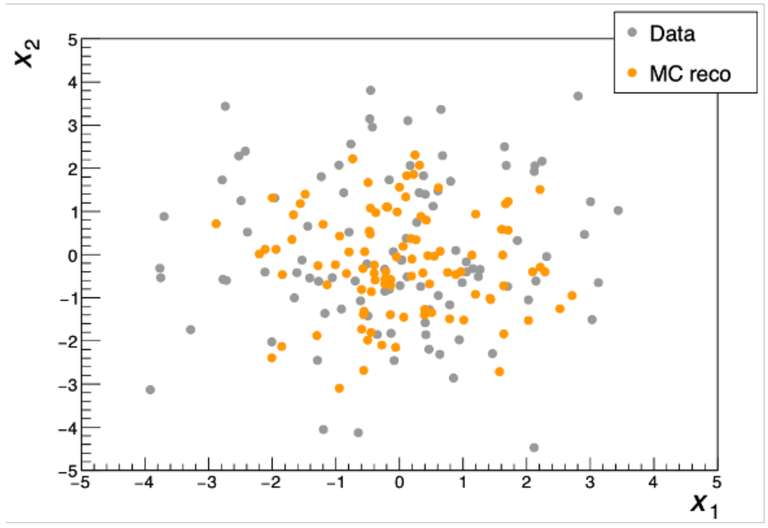
If you're more visual!

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Let's reweight our **reconstructed MC (Dataset A)** to match **data (Dataset B)** using a simple Gaussian example
- Each sample contains a set of events, each with a set of features $\vec{x} = (x_1, \ldots, x_{24})$





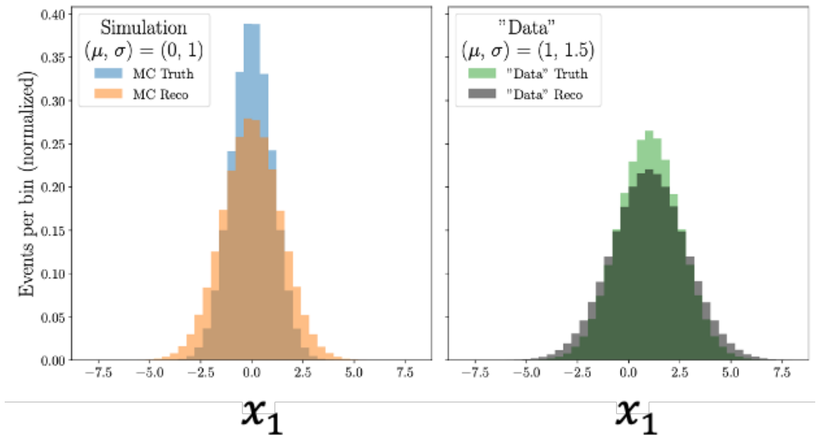


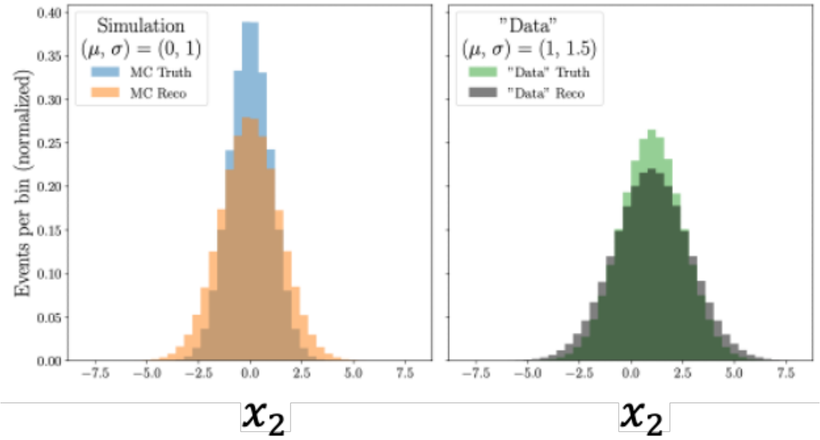
Sample reweighting

If you're more visual!

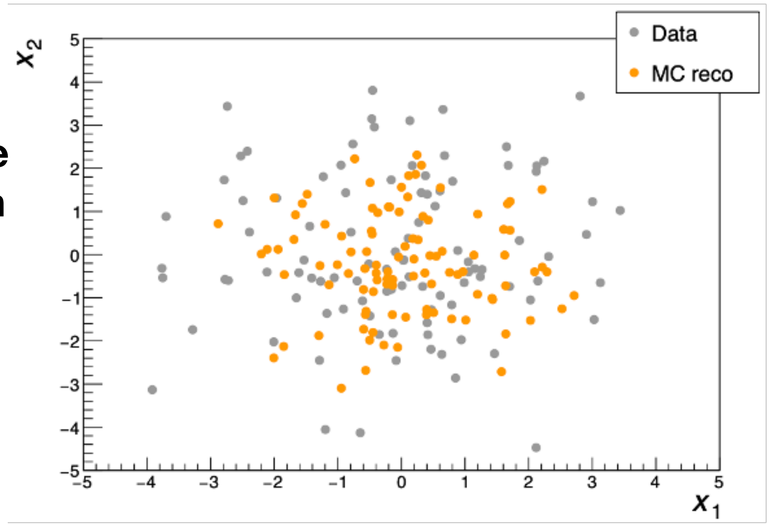
$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

- Let's reweight our **reconstructed MC (Dataset A)** to match **data (Dataset B)** using a simple Gaussian example
- Each sample contains a set of events, each with a set of features $\vec{x} = (x_1, \ldots, x_{24})$





To reweight, can estimate the density ratio between the two samples

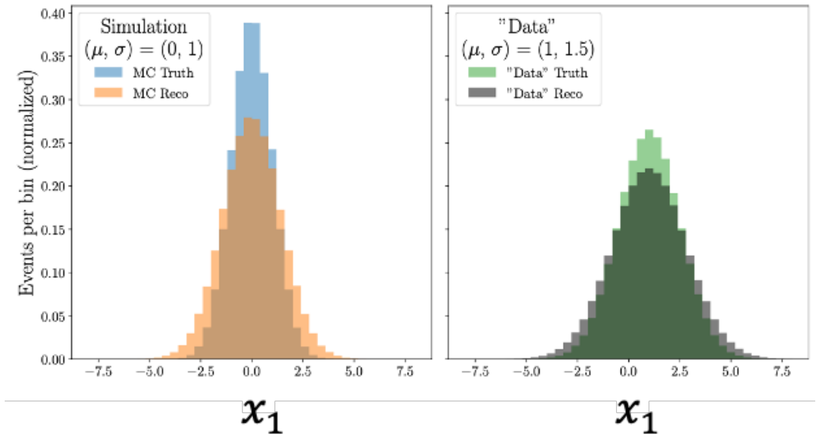


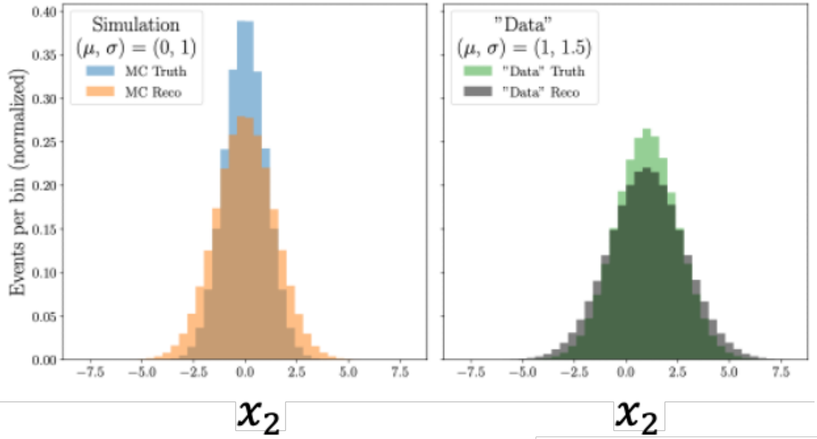
Sample reweighting

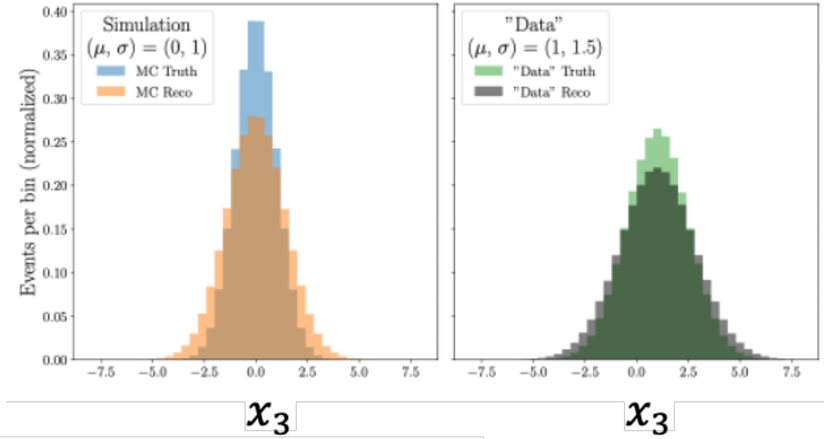
If you're more visual!

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

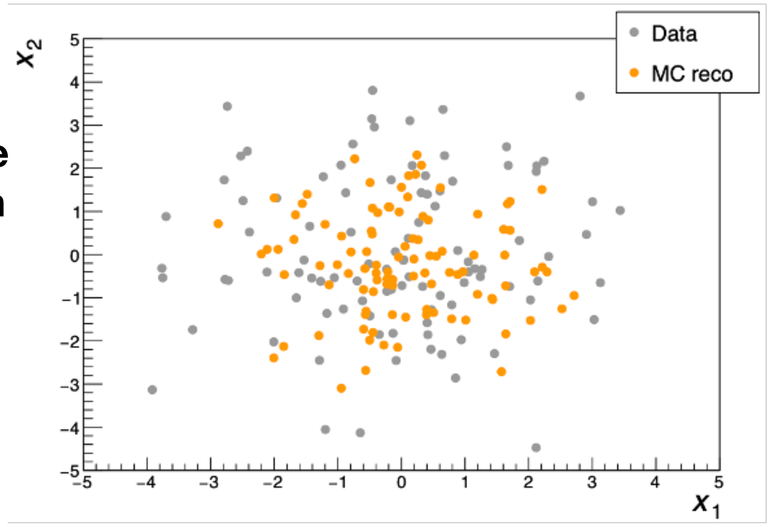
- Let's reweight our **reconstructed MC (Dataset A)** to match **data (Dataset B)** using a simple Gaussian example
- Each sample contains a set of events, each with a set of features $\vec{x} = (x_1, \ldots, x_{24})$

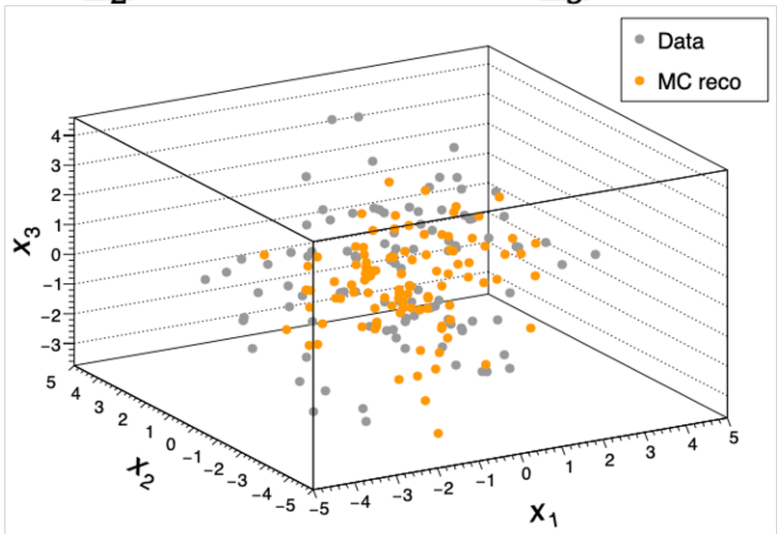






To reweight, can estimate the density ratio between the two samples



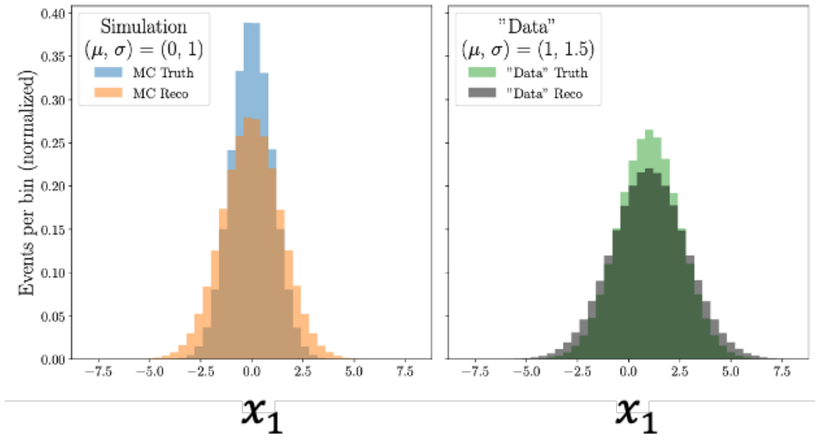


Sample reweighting

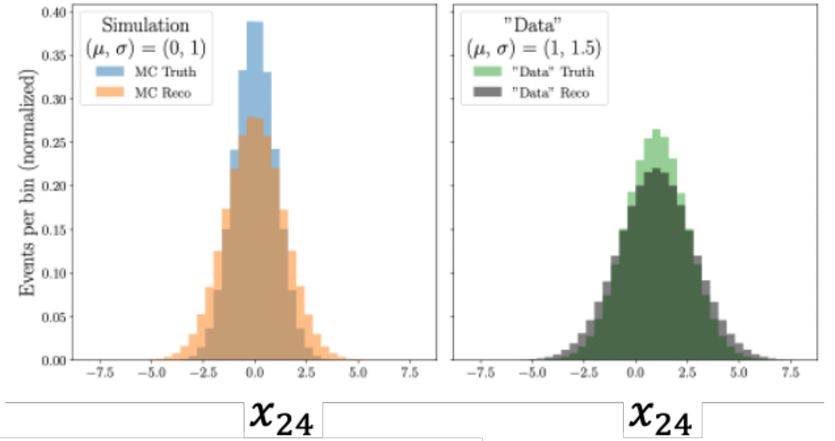
If you're more visual!

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

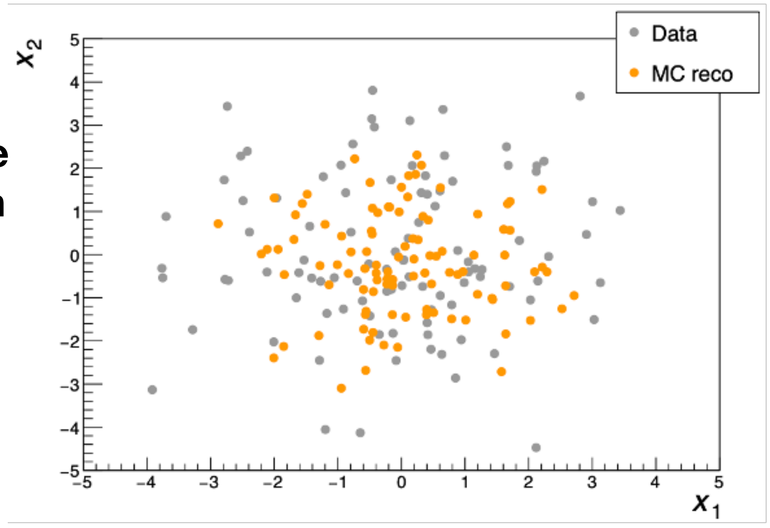
- Let's reweight our **reconstructed MC (Dataset A)** to match **data (Dataset B)** using a simple Gaussian example
- Each sample contains a set of events, each with a set of features $\vec{x} = (x_1, \ldots, x_{24})$

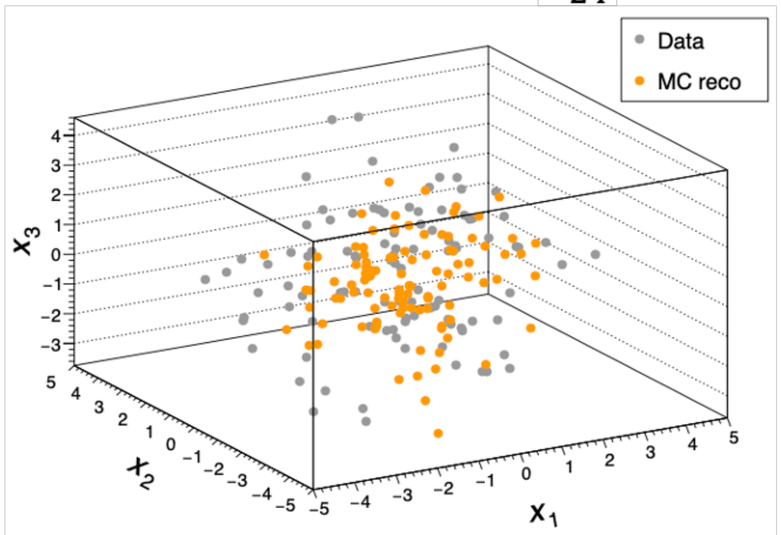


...



To reweight, can estimate the density ratio between the two samples



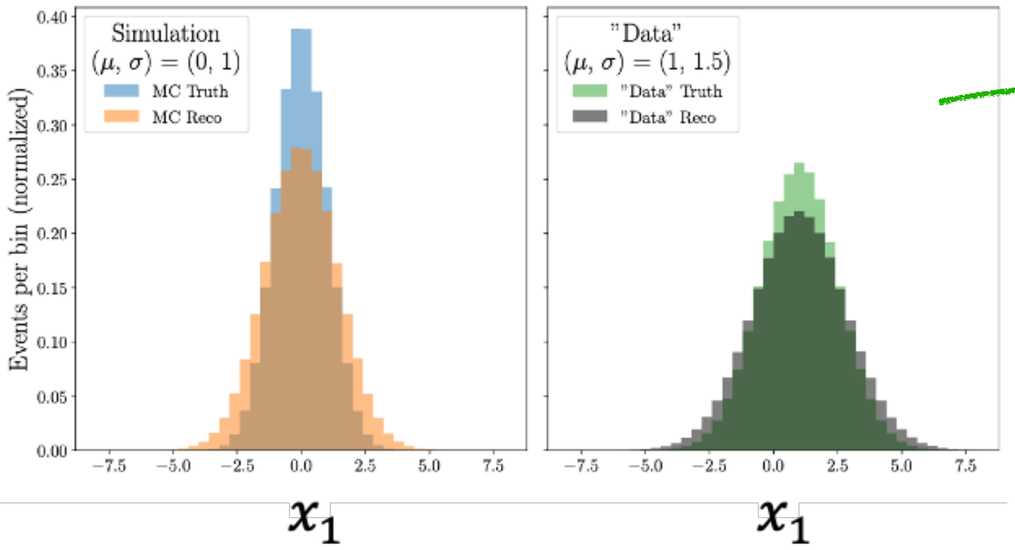


...

Sample reweighting

If you're more visual!

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$



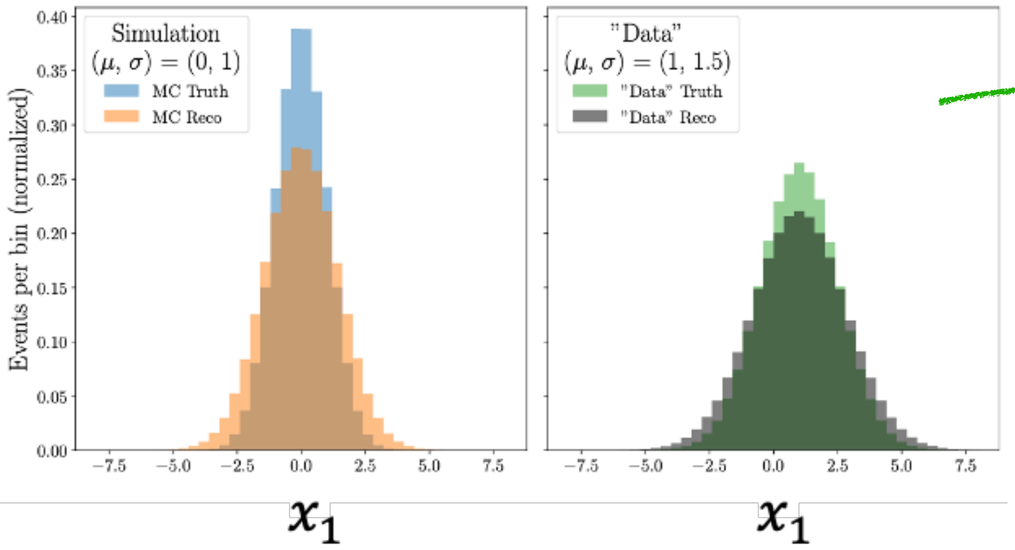
Train a classifier, $f(\vec{x})$, using $\vec{x} = (x_1, \ldots, x_{24})$ as input to differentiate between **reco MC** and **data**

Choose binary cross entropy loss function and sigmoid activation function in output layer

⇒ **output of the classifier, $f(\vec{x}) \in [0,1]$, is the probability that an event is data**

Sample reweighting

If you're more visual!



Train a classifier, $f(\vec{x})$, using $\vec{x} = (x_1, \ldots, x_{24})$ as input to differentiate between **reco MC** and **data**

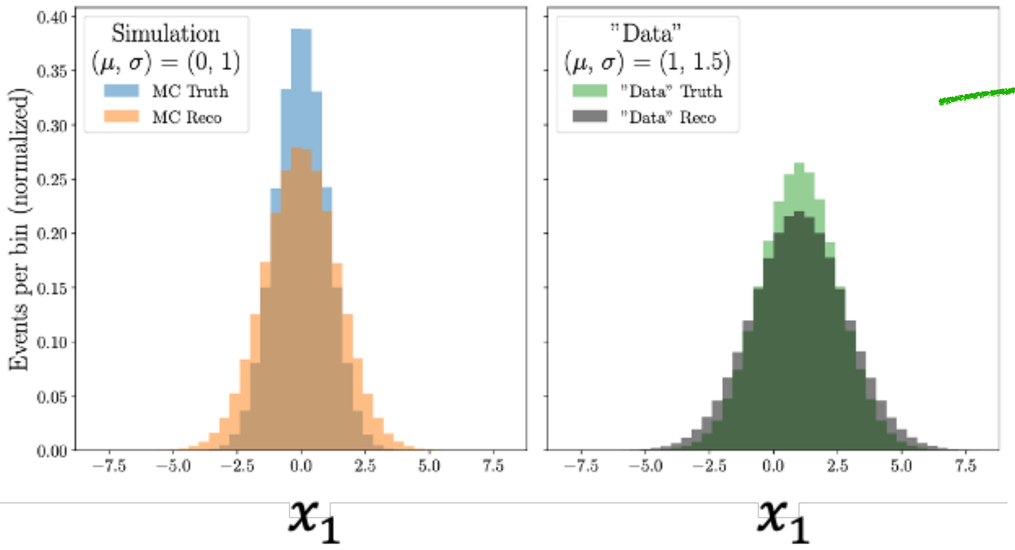
Choose binary cross entropy loss function and sigmoid activation function in output layer

$\Rightarrow$ **output of the classifier, $f(\vec{x}) \in [0,1]$, is the probability that an event is data**

$$\hat{r}(\vec{x}) = \frac{f(\vec{x})}{1 - f(\vec{x})} \approx \frac{p(\vec{x}\,|\,B)}{p(\vec{x}\,|\,A)} = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

Sample reweighting

If you're more visual!



Train a classifier, $f(\vec{x})$, using $\vec{x} = (x_1, \ldots, x_{24})$ as input to differentiate between **reco MC** and **data**

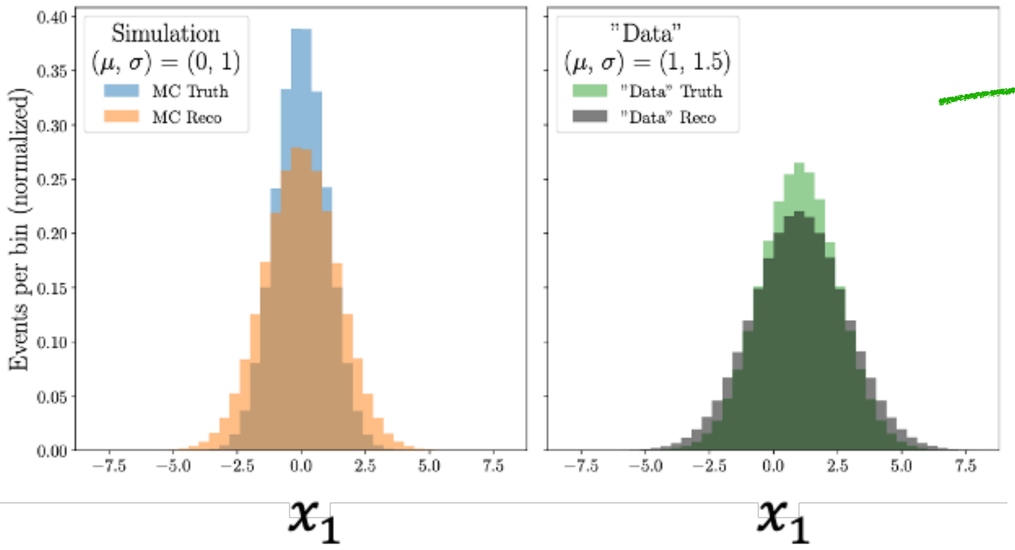
Choose binary cross entropy loss function and sigmoid activation function in output layer

⇒ **output of the classifier, $f(\vec{x}) \in [0,1]$, is the probability that an event is data**

$$\omega = \frac{f(\vec{x})}{1 - f(\vec{x})} \propto \frac{p(\vec{x}\,|\,\text{Data})}{p(\vec{x}\,|\,\text{MC})}$$

Sample reweighting

If you're more visual!



Train a classifier, $f(\vec{x})$, using $\vec{x} = (x_1, \ldots, x_{24})$ as input to differentiate between **reco MC** and **data**

Choose binary cross entropy loss function and sigmoid activation function in output layer

⇒ **output of the classifier, $f(\vec{x}) \in [0,1]$, is the probability that an event is data**

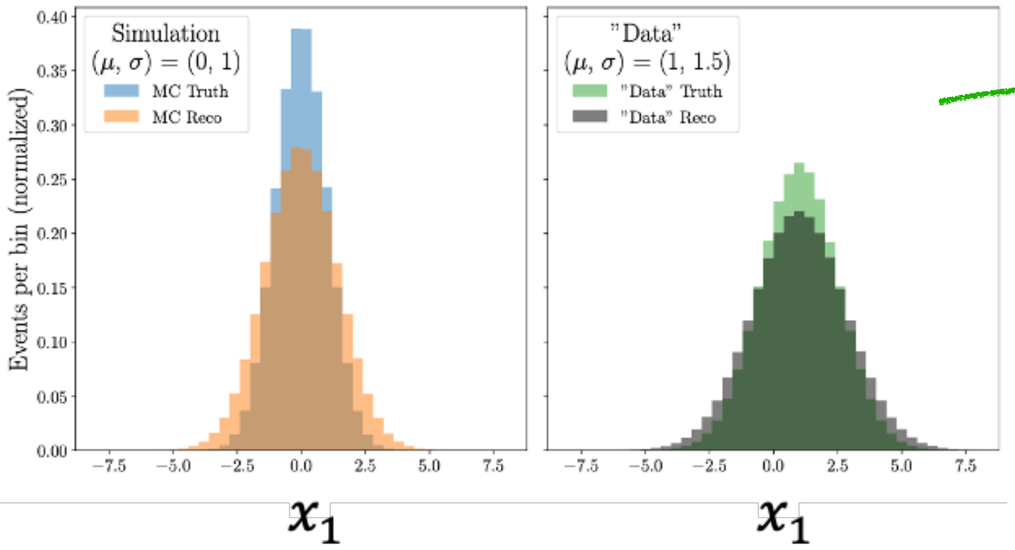
$$\omega = \frac{f(\vec{x})}{1 - f(\vec{x})} \propto \frac{p(\vec{x}\,|\,\text{Data})}{p(\vec{x}\,|\,\text{MC})}$$

Higher weight if phase space represented by $\vec{x}$ is under-represented in MC compared to data

Simultaneously reweights in all 24 dimensions

Sample reweighting

If you're more visual!



Train a classifier, $f(\vec{x})$, using $\vec{x} = (x_1, \ldots, x_{24})$ as input to differentiate between **reco MC** and **data**

Choose binary cross entropy loss function and sigmoid activation function in output layer

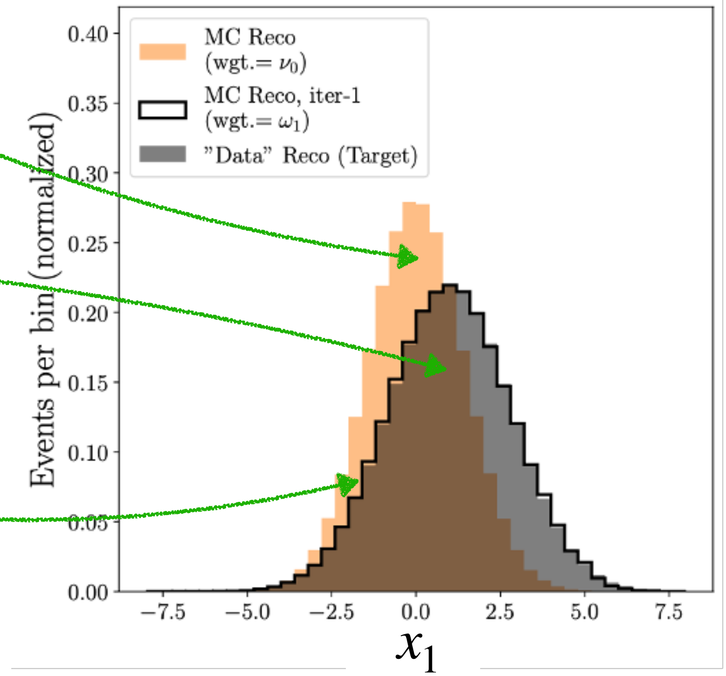
⇒ **output of the classifier, $f(\vec{x}) \in [0,1]$, is the probability that an event is data**

$$\omega = \frac{f(\vec{x})}{1 - f(\vec{x})} \propto \frac{p(\vec{x}\,|\,\text{Data})}{p(\vec{x}\,|\,\text{MC})}$$

Original **reco MC** distribution

Data is the target for our reweighting

Solid black line: **reco MC reweighted with ω**

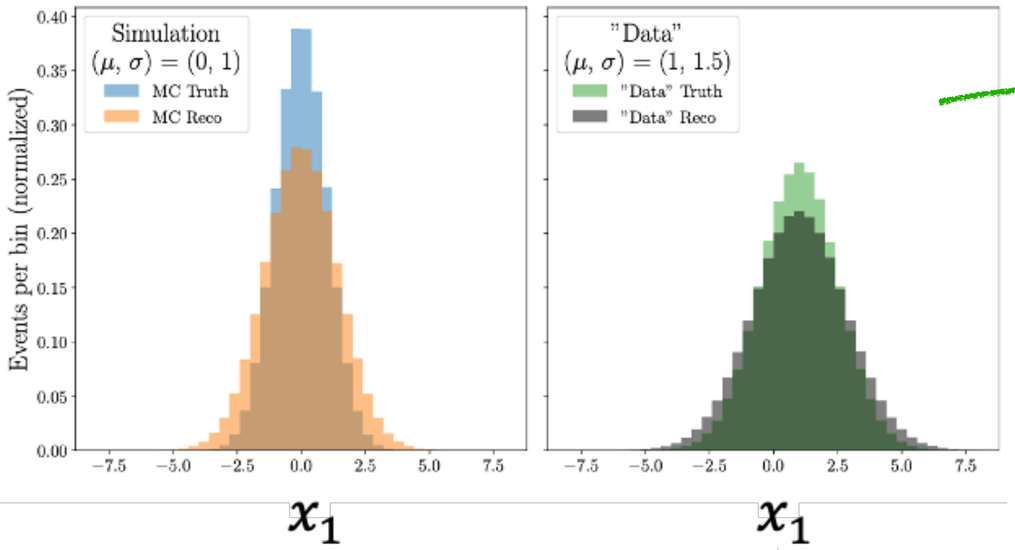


Higher weight if phase space represented by $\vec{x}$ is under-represented in MC compared to data

Simultaneously reweights in all 24 dimensions

Sample reweighting

If you're more visual!

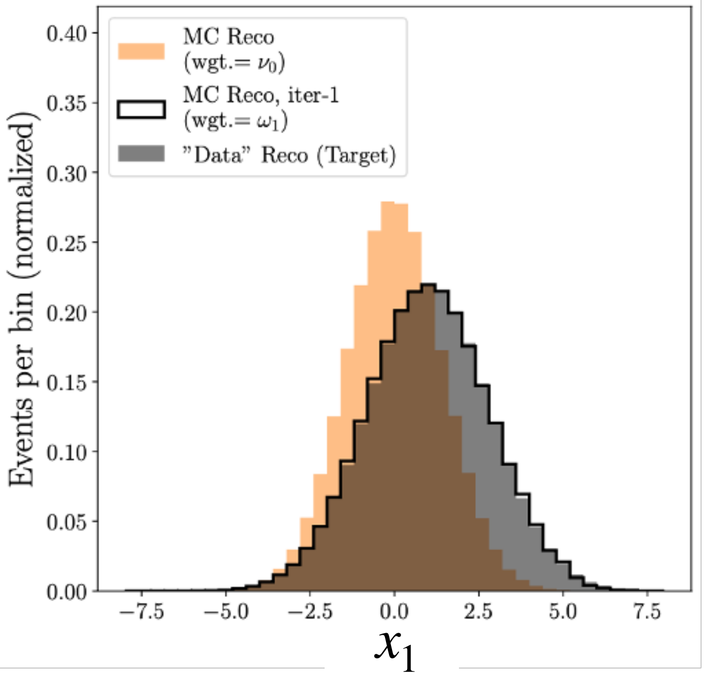


Train a classifier, $f(\vec{x})$, using $\vec{x} = (x_1, \ldots, x_{24})$ as input to differentiate between **reco MC** and **data**

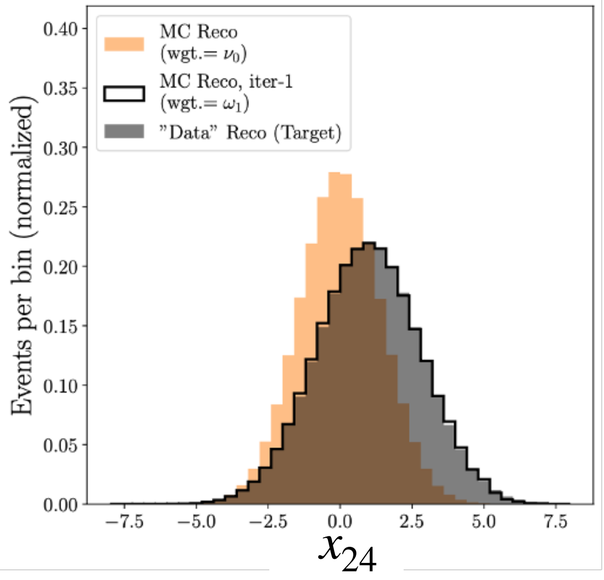
Choose binary cross entropy loss function and sigmoid activation function in output layer

$\Rightarrow$ **output of the classifier, $f(\vec{x}) \in [0,1]$, is the probability that an event is data**

$$\omega = \frac{f(\vec{x})}{1 - f(\vec{x})} \propto \frac{p(\vec{x}\,|\,\text{Data})}{p(\vec{x}\,|\,\text{MC})}$$



...

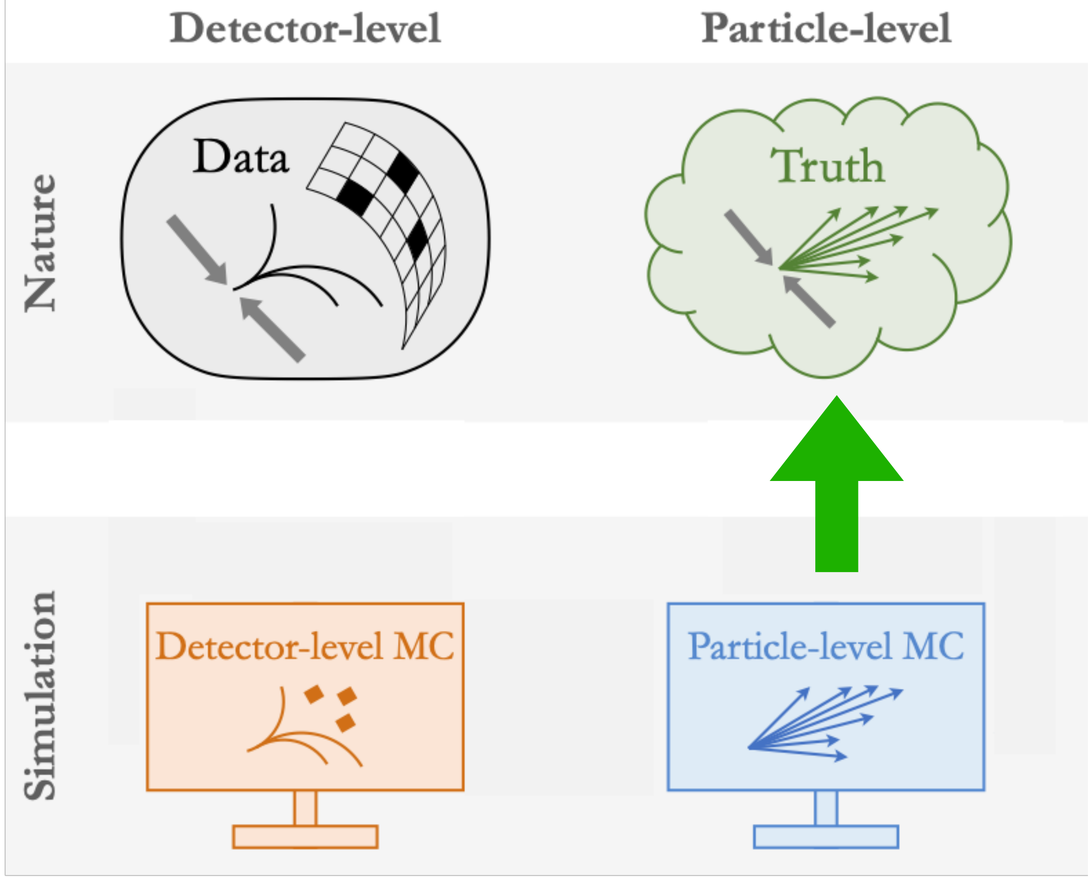


Higher weight if phase space represented by $\vec{x}$ is under-represented in MC compared to data

Simultaneously reweights in all 24 dimensions

The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **“truth data”**

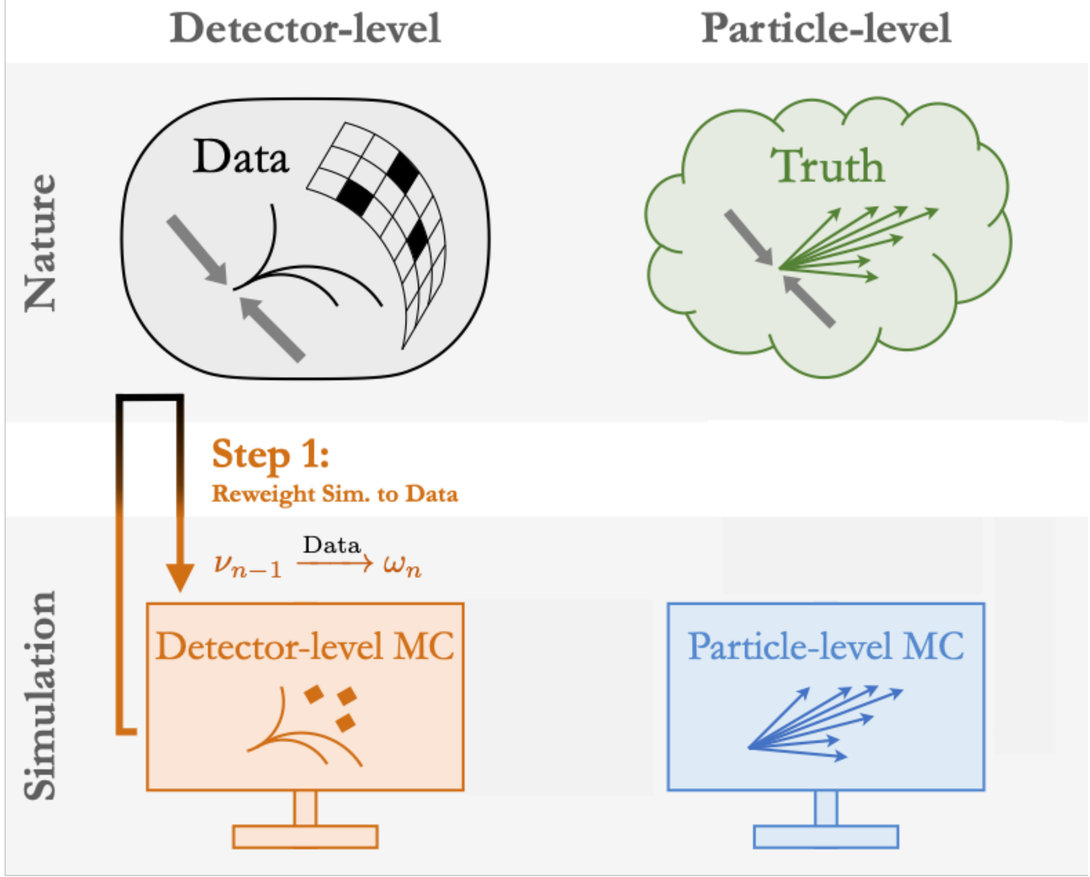


The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

$$\mathscr{MC}^{(0)}_{\text{reco}} = ((w_1^{\text{reco}}, \vec{x}_1^{\text{reco}}), (w_2^{\text{reco}}, \vec{x}_2^{\text{reco}}), \ldots, (w_N^{\text{reco}}, \vec{x}_N^{\text{reco}}))$$

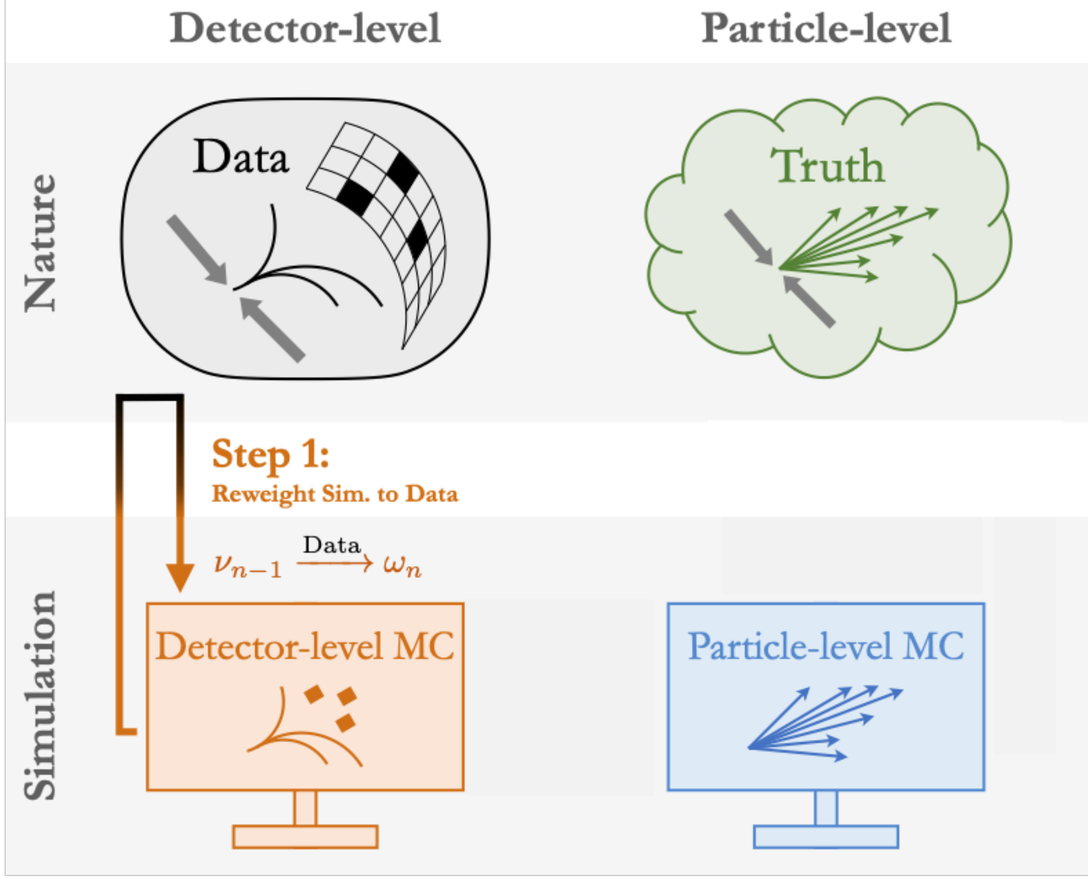


The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

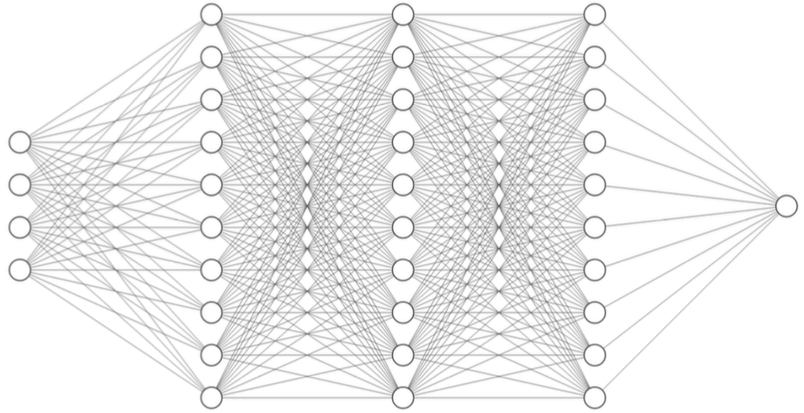
Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

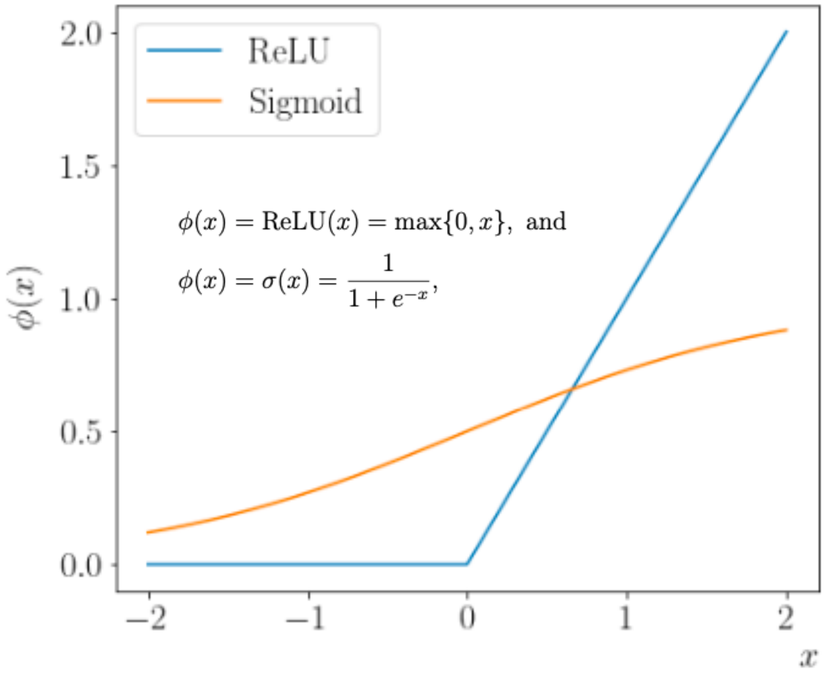
$$\mathscr{MC}^{(1)}_{\text{reco}} = ((w_1^{\text{reco}} \times \omega(\vec{x}_1^{\text{reco}}), \vec{x}_1^{\text{reco}}), (w_2^{\text{reco}} \times \omega(\vec{x}_2^{\text{reco}}), \vec{x}_2^{\text{reco}}), \ldots, (w_N^{\text{reco}} \times \omega(\vec{x}_1^{\text{reco}}), \vec{x}_N^{\text{reco}}))$$



Interlude: neural network setup

- Same NN architecture used for all steps of MultiFold
- NNs implemented in TensorFlow and Keras
- Inputs for each event: 24 standardized features and a standardized weight
- 3 hidden layers with 200 nodes each
- ReLU activation function used for hidden layers
- Necessary for our density ratio estimation to work:
  - Binary cross entropy loss function
  - Sigmoid activation function on the final layer
- Input training data randomly divided into 75%/25% train/validation split

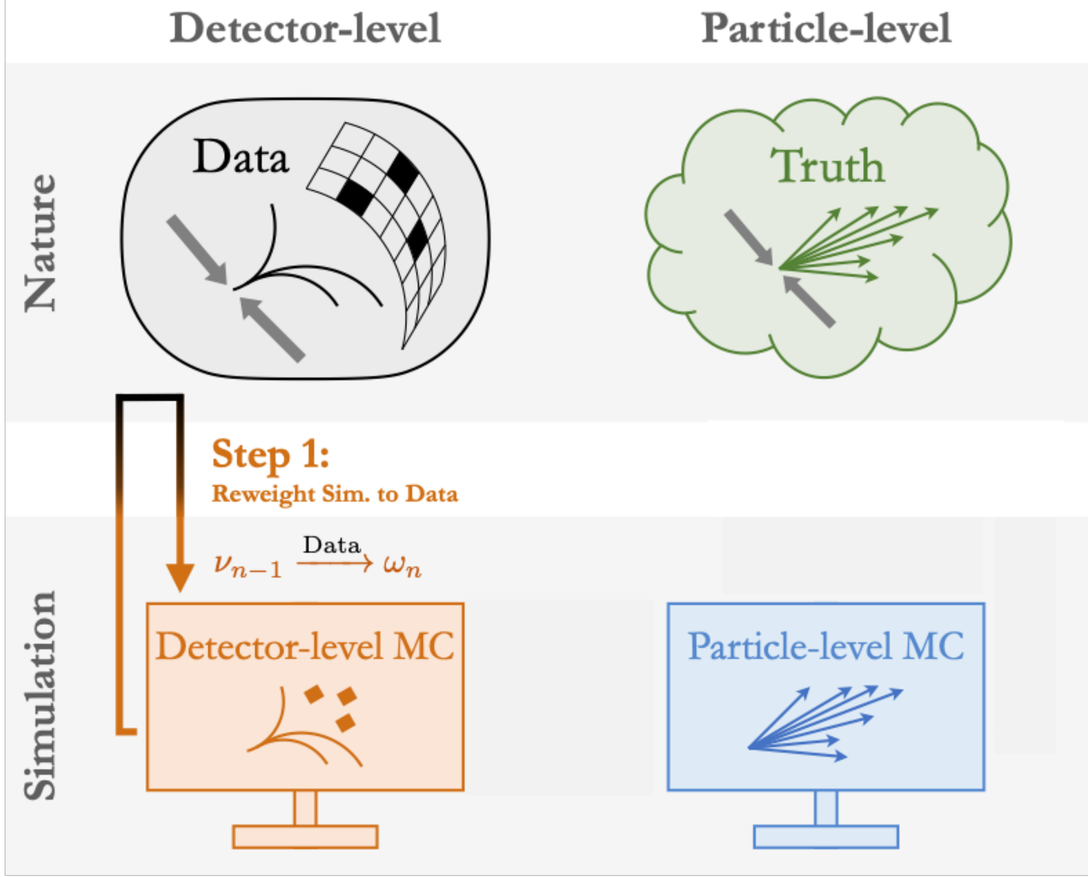




The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **“truth data”**

Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)



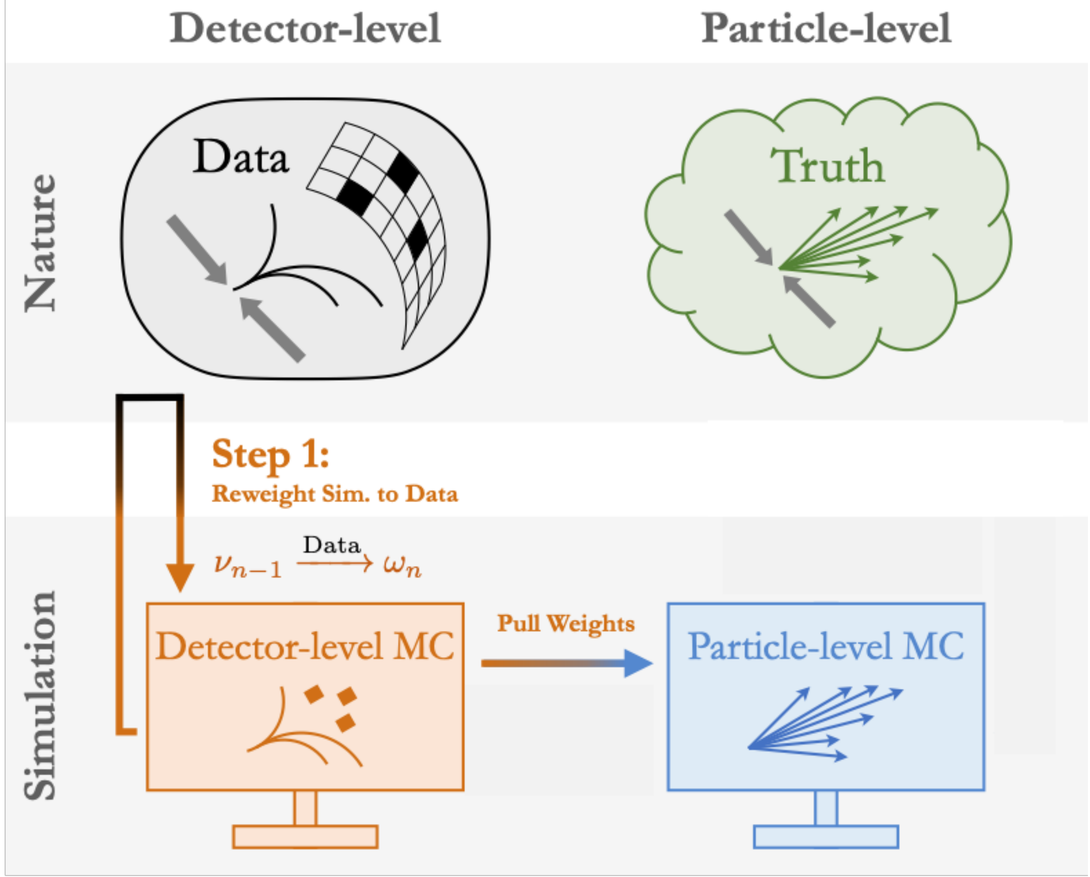
The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

$$\mathscr{MC}^{(0)}_{\text{truth}} = ((w_1^{\text{truth}}, \vec{x}_1^{\text{truth}}), (w_2^{\text{truth}}, \vec{x}_2^{\text{truth}}), \ldots, (w_N^{\text{truth}}, \vec{x}_N^{\text{truth}}))$$



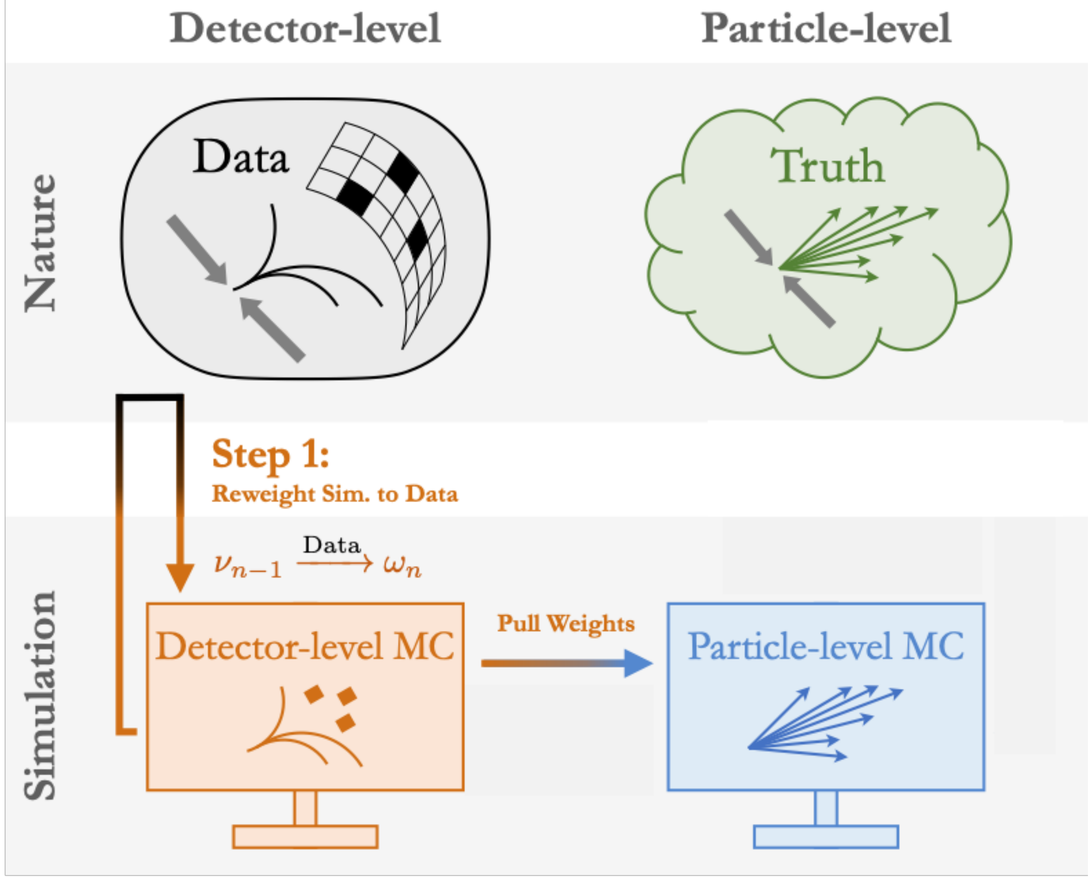
The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

$$\mathscr{MC}^{\text{pull}(1)}_{\text{truth}} = ((w_1^{\text{truth}} \times \omega(\vec{x}_1^{\text{reco}}), \vec{x}_1^{\text{truth}}), (w_2^{\text{truth}} \times \omega(\vec{x}_1^{\text{reco}}), \vec{x}_2^{\text{truth}}), \ldots, (w_N^{\text{truth}} \times \omega(\vec{x}_1^{\text{reco}}), \vec{x}_N^{\text{truth}}))$$



The MultiFold method

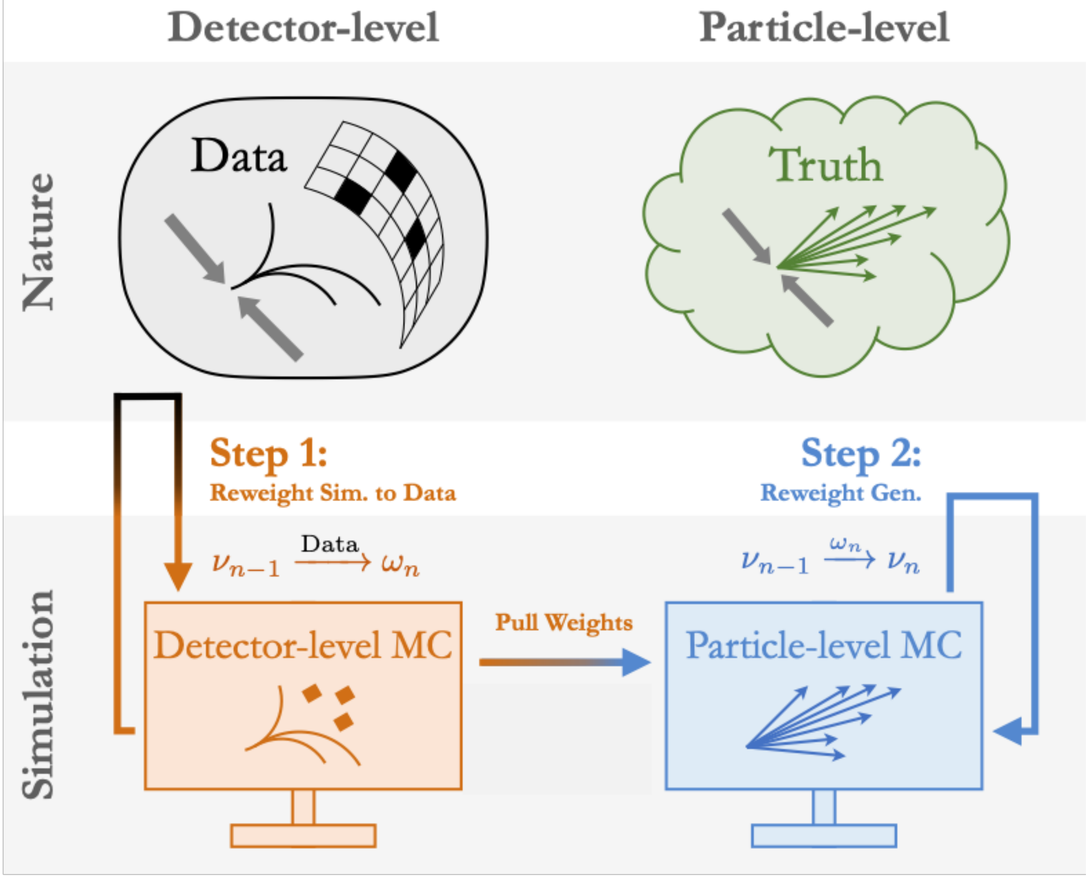
- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

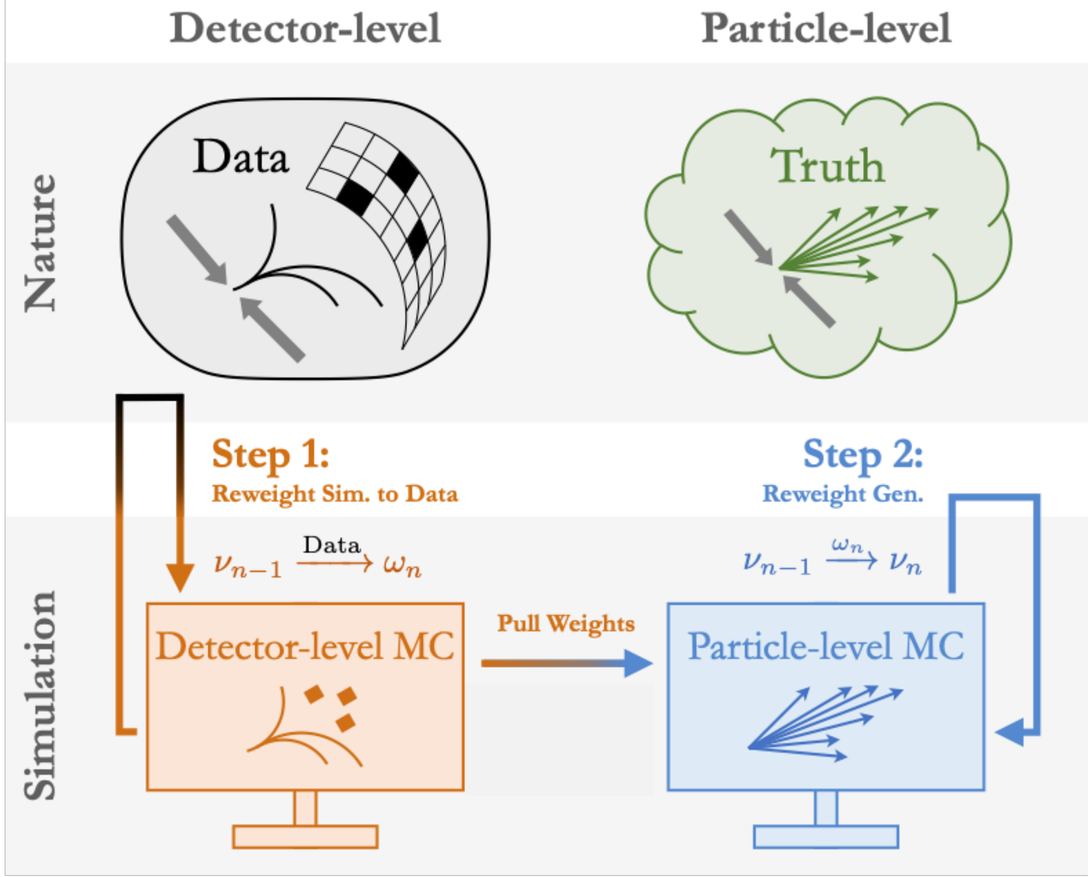
Step 2: Determine a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that replicates the reweighting in Step 1 by reweighting the original **truth MC** sample to the **truth MC sample with the "pull" weights applied**

$$\mathscr{MC}^{(0)}_{\text{truth}} = ((w_1^{\text{truth}}, \vec{x}_1^{\text{truth}}), (w_2^{\text{truth}}, \vec{x}_2^{\text{truth}}), \ldots, (w_N^{\text{truth}}, \vec{x}_N^{\text{truth}}))$$



The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**



Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

Step 2: Determine a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that replicates the reweighting in Step 1 by reweighting the original **truth MC** sample to the **truth MC sample with the "pull" weights applied**

$$\mathscr{MC}^{(1)}_{\text{truth}} = ((w_1^{\text{truth}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_1^{\text{truth}}), (w_2^{\text{truth}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_2^{\text{truth}}), \ldots, (w_N^{\text{truth}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_N^{\text{truth}}))$$

The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

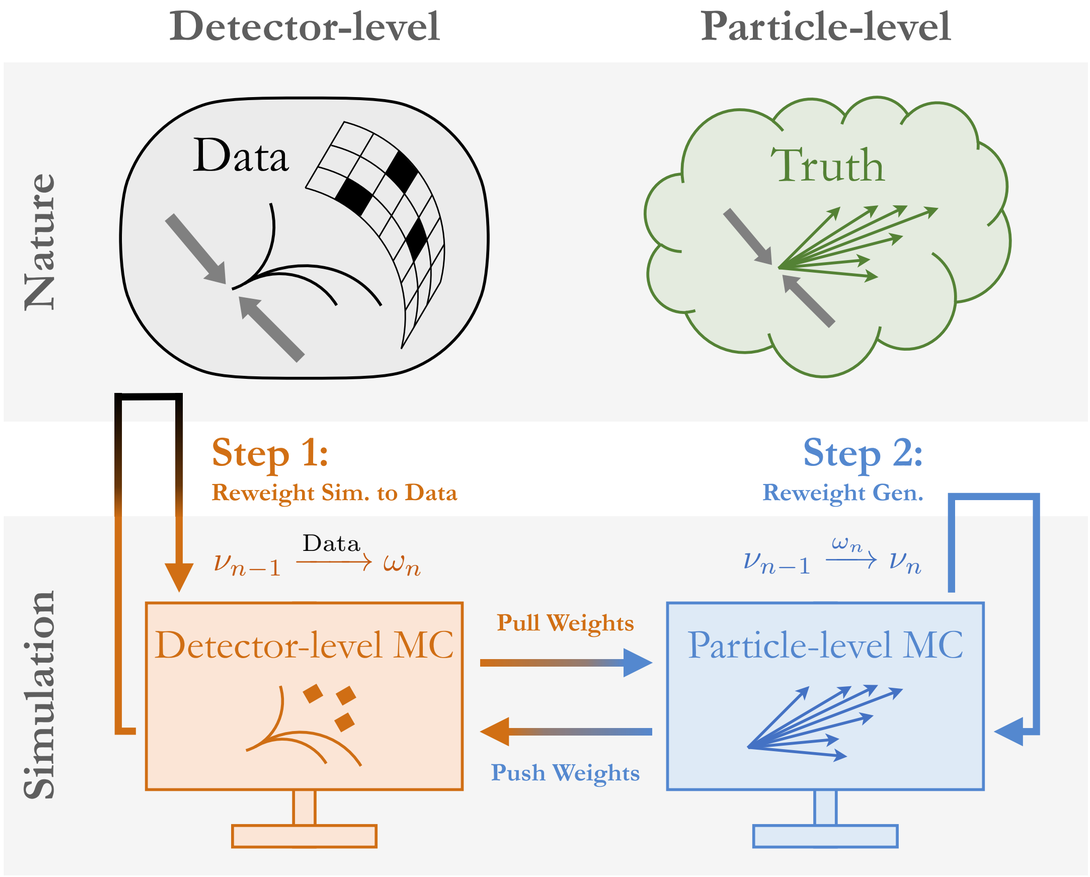
Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

Step 2: Determine a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that replicates the reweighting in Step 1 by reweighting the original **truth MC** sample to the **truth MC sample with the "pull" weights applied**

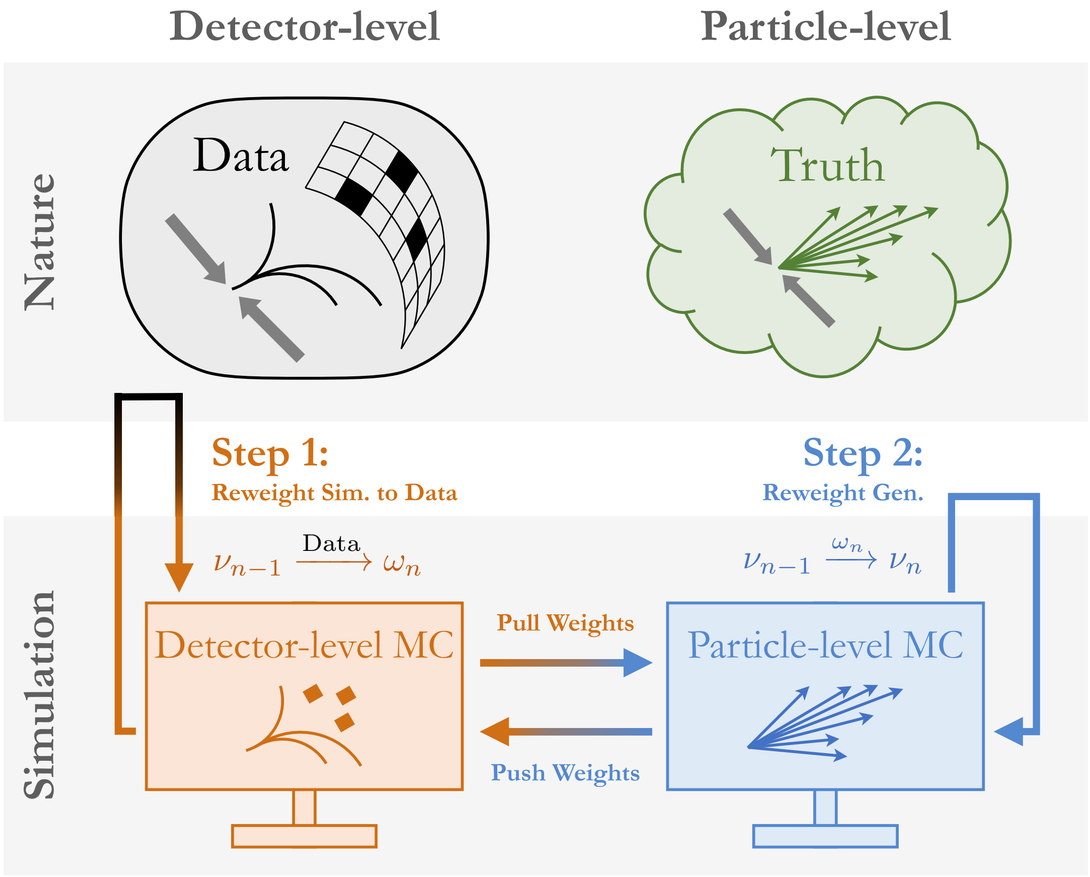
"Push" weights: propagate weights in the form of multiplicative factors to the **reconstructed MC**

$$\mathcal{MC}^{(0)}_{\text{reco}} = ((w_1^{\text{reco}}, \vec{x}_1^{\text{reco}}), (w_2^{\text{reco}}, \vec{x}_2^{\text{reco}}), \ldots, (w_N^{\text{reco}}, \vec{x}_N^{\text{reco}}))$$



The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**



Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

Step 2: Determine a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that replicates the reweighting in Step 1 by reweighting the original **truth MC** sample to the **truth MC sample with the "pull" weights applied**

"Push" weights: propagate weights in the form of multiplicative factors to the **reconstructed MC**

$$\mathscr{MC}^{\text{push(1)}}_{\text{reco}} = ((w_1^{\text{reco}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_1^{\text{reco}}), (w_2^{\text{reco}} \times \nu(\vec{x}_2^{\text{truth}}), \vec{x}_2^{\text{reco}}), \ldots, (w_N^{\text{reco}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_N^{\text{reco}}))$$

The MultiFold method

- **Ultimate goal:** reweight **truth MC** such that it becomes the **"truth data"**

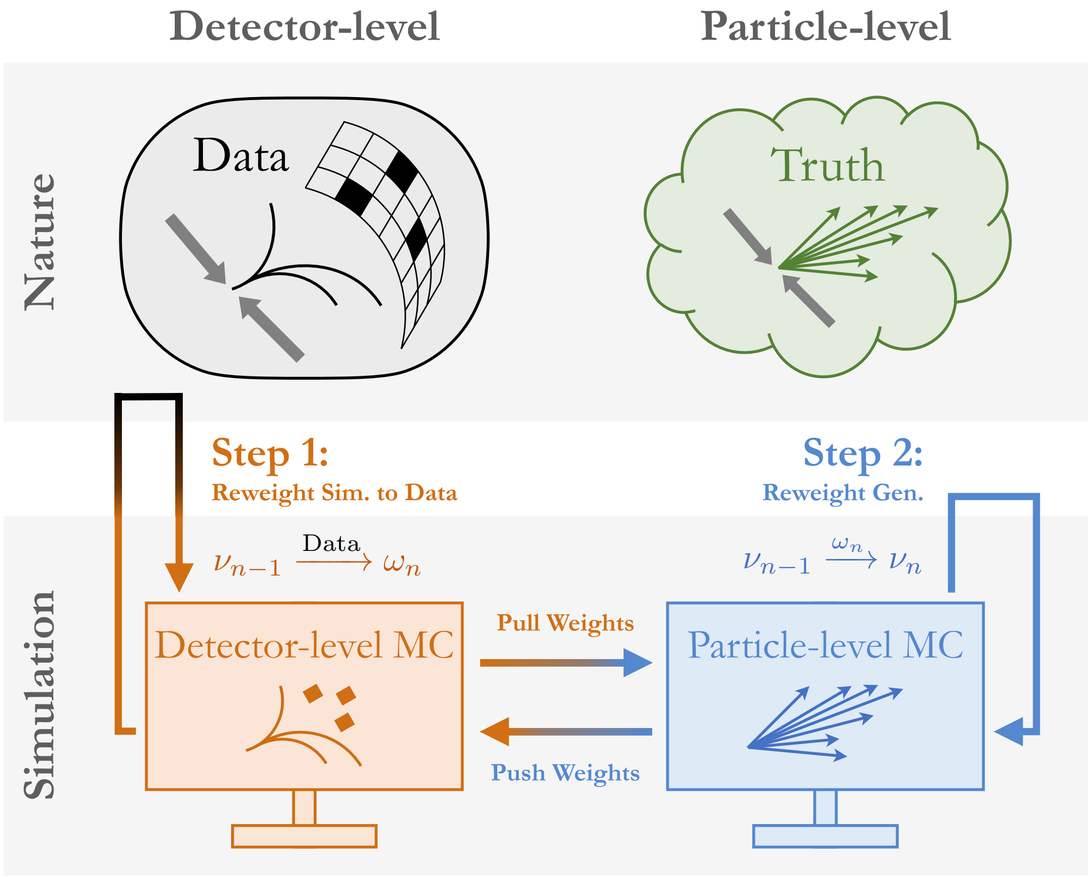
Step 1: Use a neural network to determine a reweighting function, $\omega(\vec{x}^{\text{reco}})$, that will transform the **reconstructed MC** to match **data** for all observables (what we just did!)

"Pull" weights: propagate weights in the form of multiplicative factors to the **truth MC**

Step 2: Determine a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that replicates the reweighting in Step 1 by reweighting the original **truth MC** sample to the **truth MC sample with the "pull" weights applied**

"Push" weights: propagate weights in the form of multiplicative factors to the **reconstructed MC**

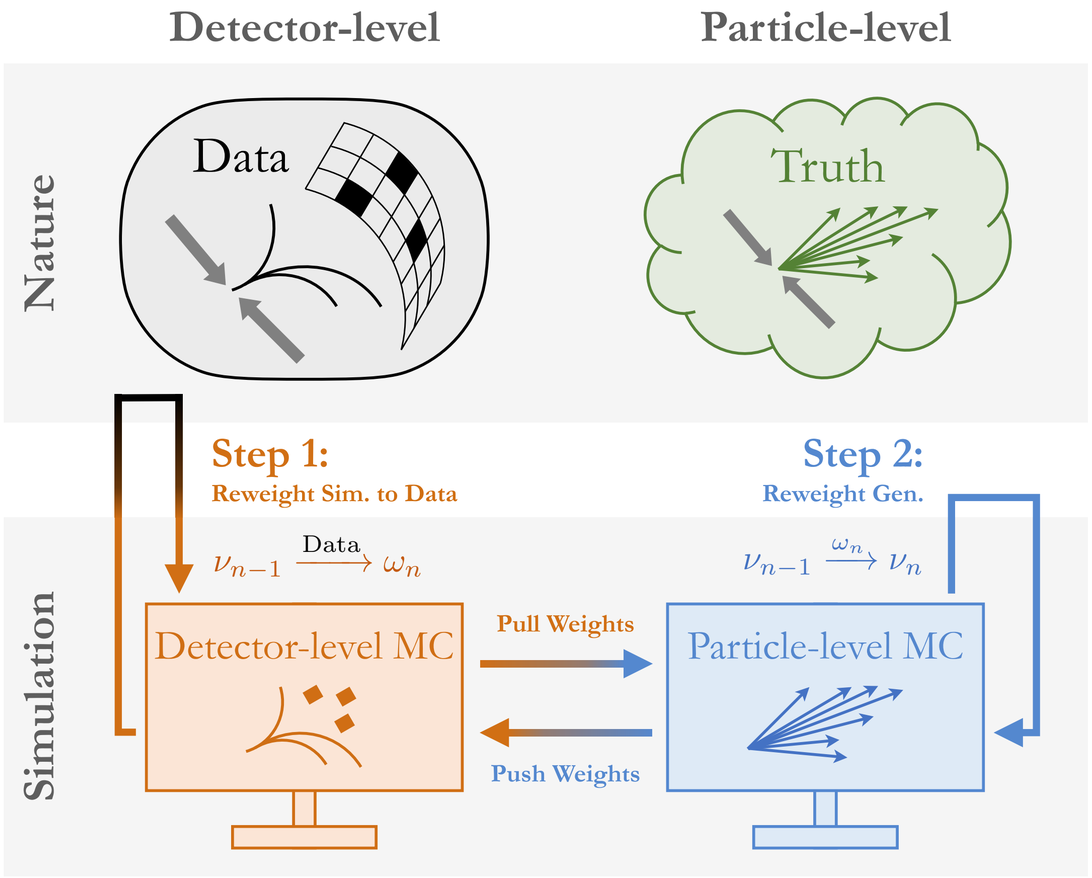
$$\mathscr{MC}^{\text{push}(1)}_{\text{reco}} = ((w_1^{\text{reco}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_1^{\text{reco}}), (w_2^{\text{reco}} \times \nu(\vec{x}_2^{\text{truth}}), \vec{x}_2^{\text{reco}}), \ldots, (w_N^{\text{reco}} \times \nu(\vec{x}_1^{\text{truth}}), \vec{x}_N^{\text{reco}}))$$



And iterate…

The MultiFold method

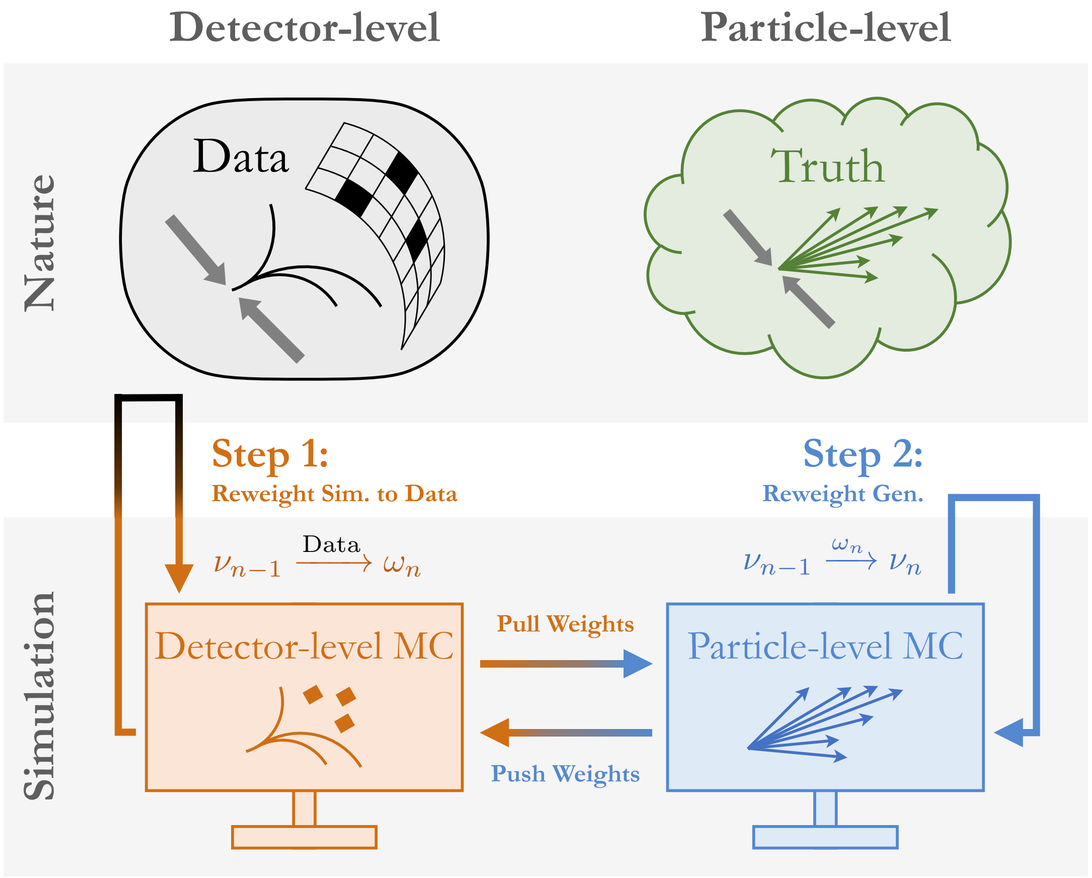
- **Final result:** a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that reweights the **truth MC** to become the **unfolded data**



The MultiFold method

- **Final result:** a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that reweights the **truth MC** to become the **unfolded data**

$$\text{Unfolded Data} = (w_i^{\text{truth}} \times w_i^{\text{MF}}, \vec{x}_i^{\text{truth}})$$

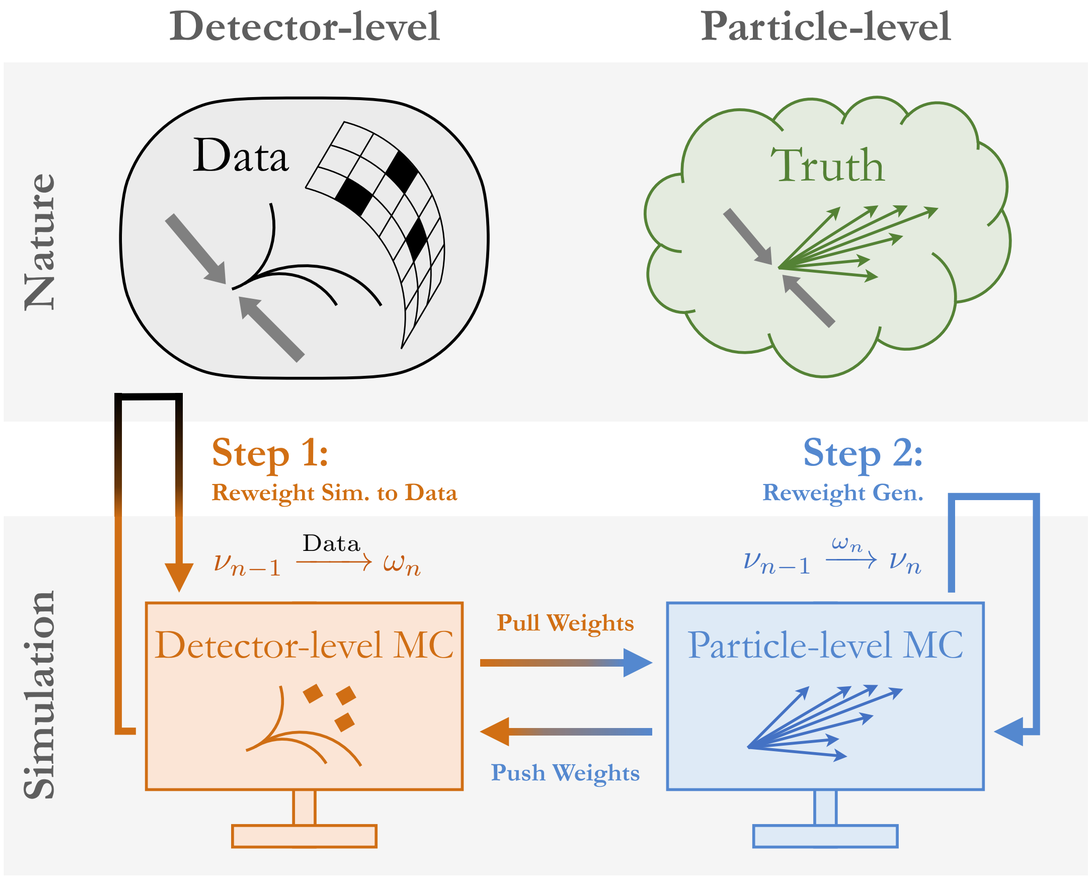


The MultiFold method

- **Final result:** a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that reweights the **truth MC** to become the **unfolded data**

$$\text{Unfolded Data} = (w_i^{\text{truth}} \times w_i^{\text{MF}}, \vec{x}_i^{\text{truth}})$$

$$w_i^{\text{MF}} = \prod_{k=1}^{N_{\text{iter}}} \nu_k(\vec{x}_i^{\text{truth}})$$

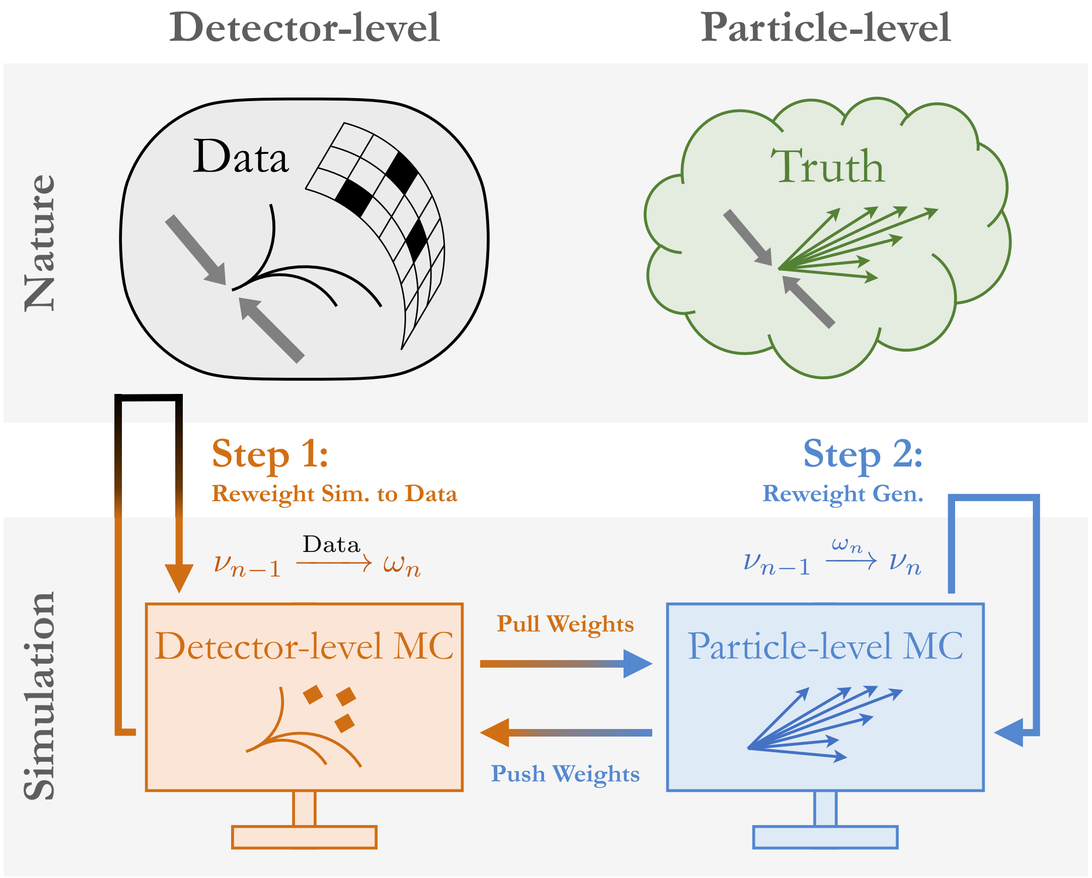


The MultiFold method

- **Final result:** a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that reweights the **truth MC** to become the **unfolded data**

$$\text{Unfolded Data} = (w_i^{\text{truth}} \times w_i^{\text{MF}}, \vec{x}_i^{\text{truth}})$$

- **Result is**
  - **Completely unbinned:** measurement is an event sample instead of a 1D histrogram like in a traditional unfolding algorithm
  - **Multidimensional:** in this analysis, 24 observables as input ⇒ 24 dimensional unfolding

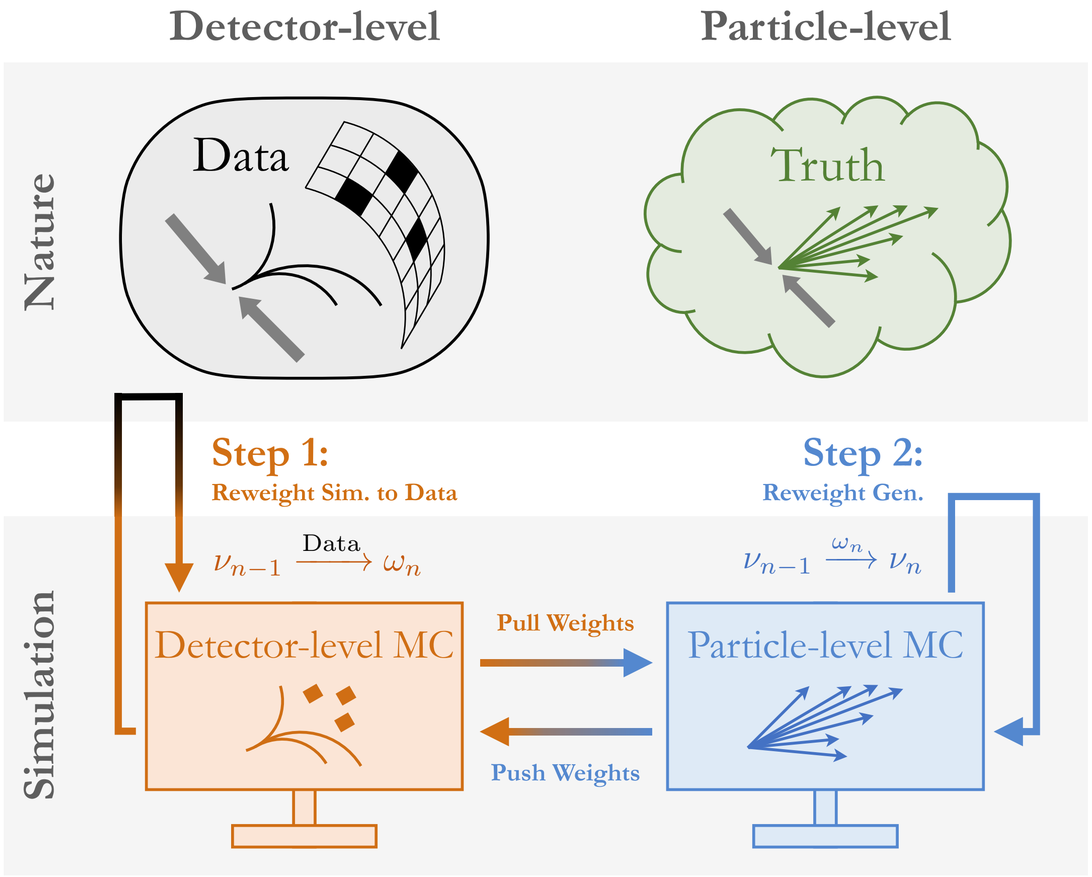


The MultiFold method

- **Final result:** a reweighting function, $\nu(\vec{x}^{\text{truth}})$, that reweights the **truth MC** to become the **unfolded data**

$$\text{Unfolded Data} = (w_i^{\text{truth}} \times w_i^{\text{MF}}, \vec{x}_i^{\text{truth}})$$

- **Result is**
  - **Completely unbinned:** measurement is an event sample instead of a 1D histrogram like in a traditional unfolding algorithm
  - **Multidimensional:** in this analysis, 24 observables as input ⇒ 24 dimensional unfolding



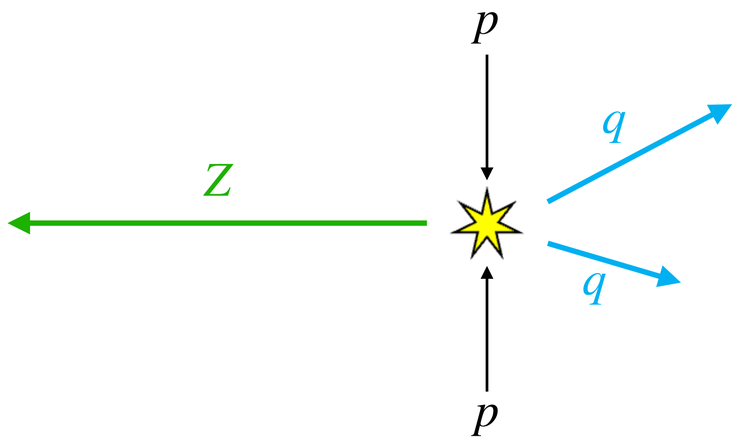
A result in this format is hugely beneficial. For example, can:

- **Freely rebin data after the analysis is complete**
- **Construct new observables after unfolding provided they can be calculated from the input 24**
- **Place cuts on the analysis phase space after unfolding**
- **Easily construct multi-dimensional distributions**

Part 3: Analysis

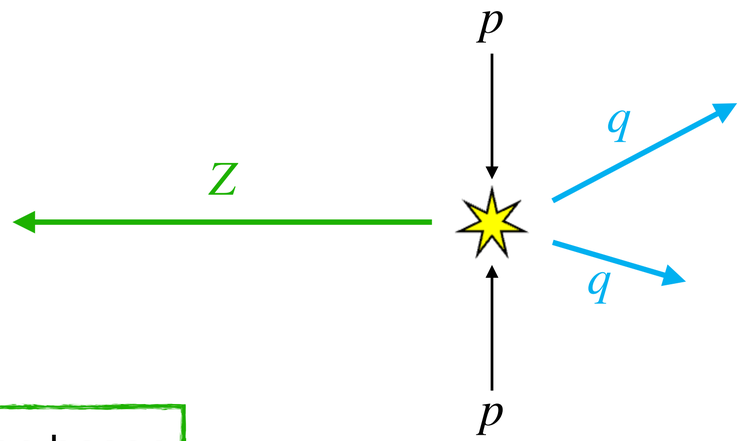
Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model



Z+jets events in the ATLAS detector

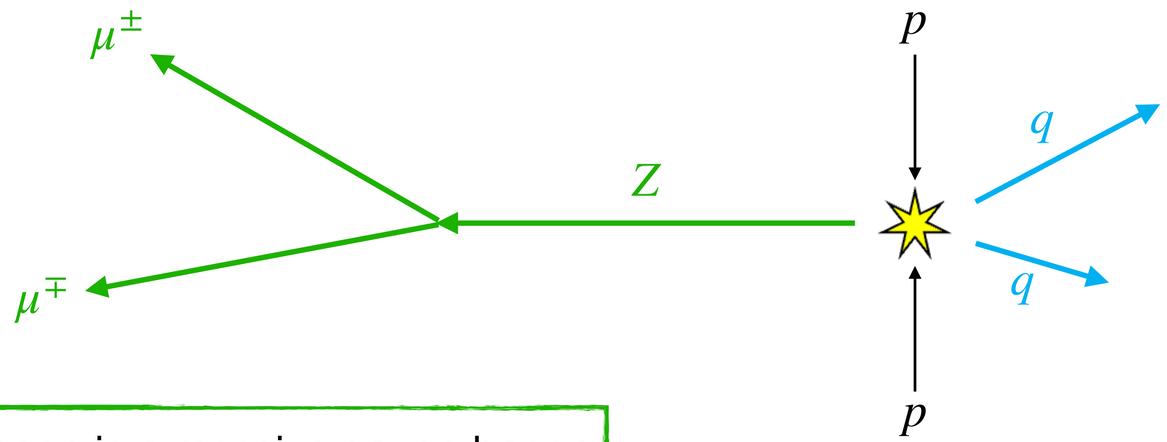
- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model



- The Z boson is a massive gauge boson
- Mediator of electroweak interaction

Z+jets events in the ATLAS detector

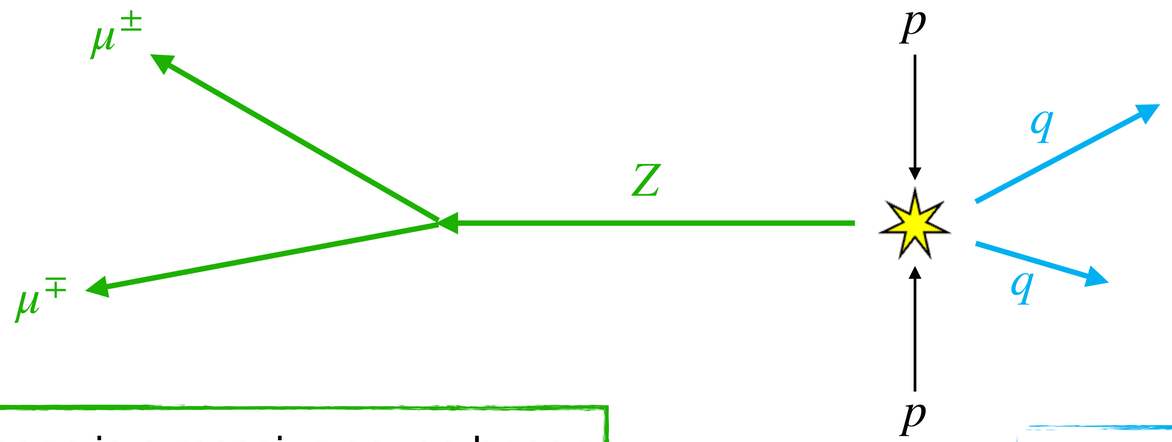
- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model



- The Z boson is a massive gauge boson
- Mediator of electroweak interaction
- In this case, interested in the decay to a muon and an anti-muon
  - ~3.4% of the time
  - “Easy” to reconstruct

Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model

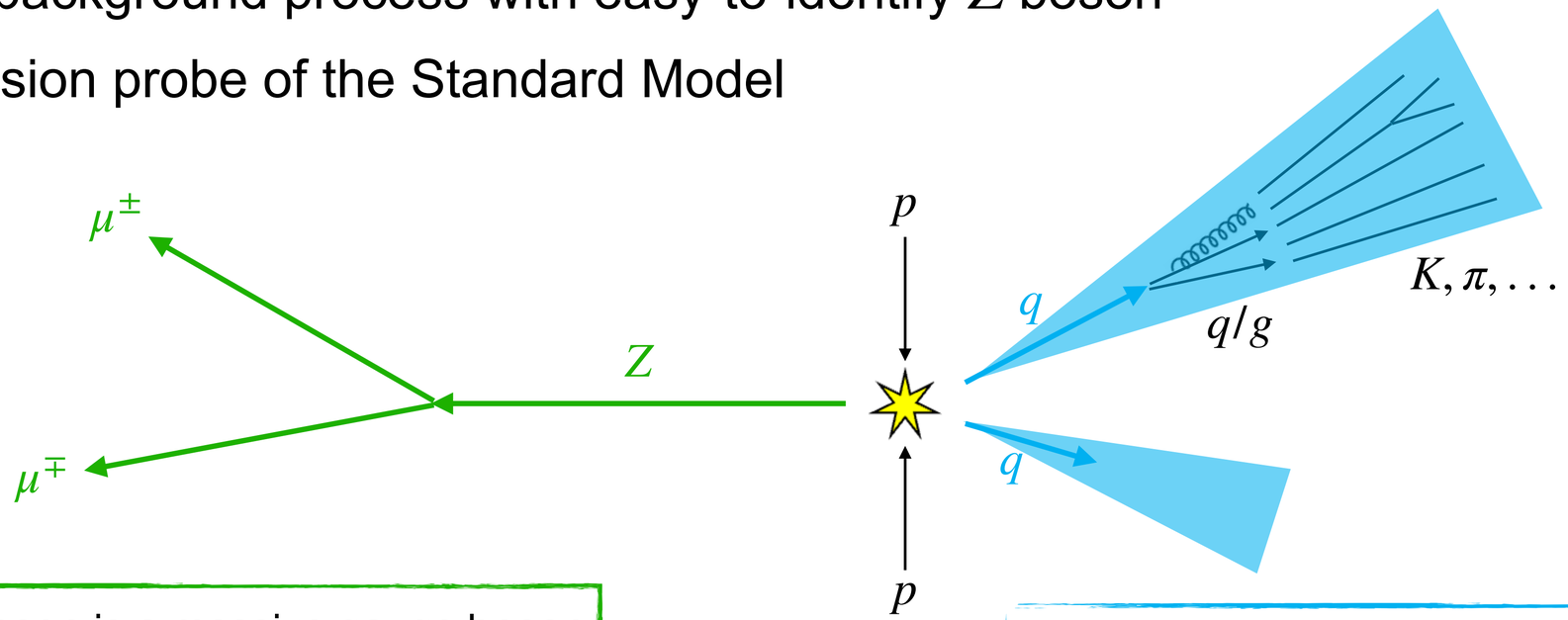


- The Z boson is a massive gauge boson
- Mediator of electroweak interaction
- In this case, interested in the decay to a muon and an anti-muon
  - ~3.4% of the time
  - “Easy” to reconstruct

- Quarks cannot exist as free particles

Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model

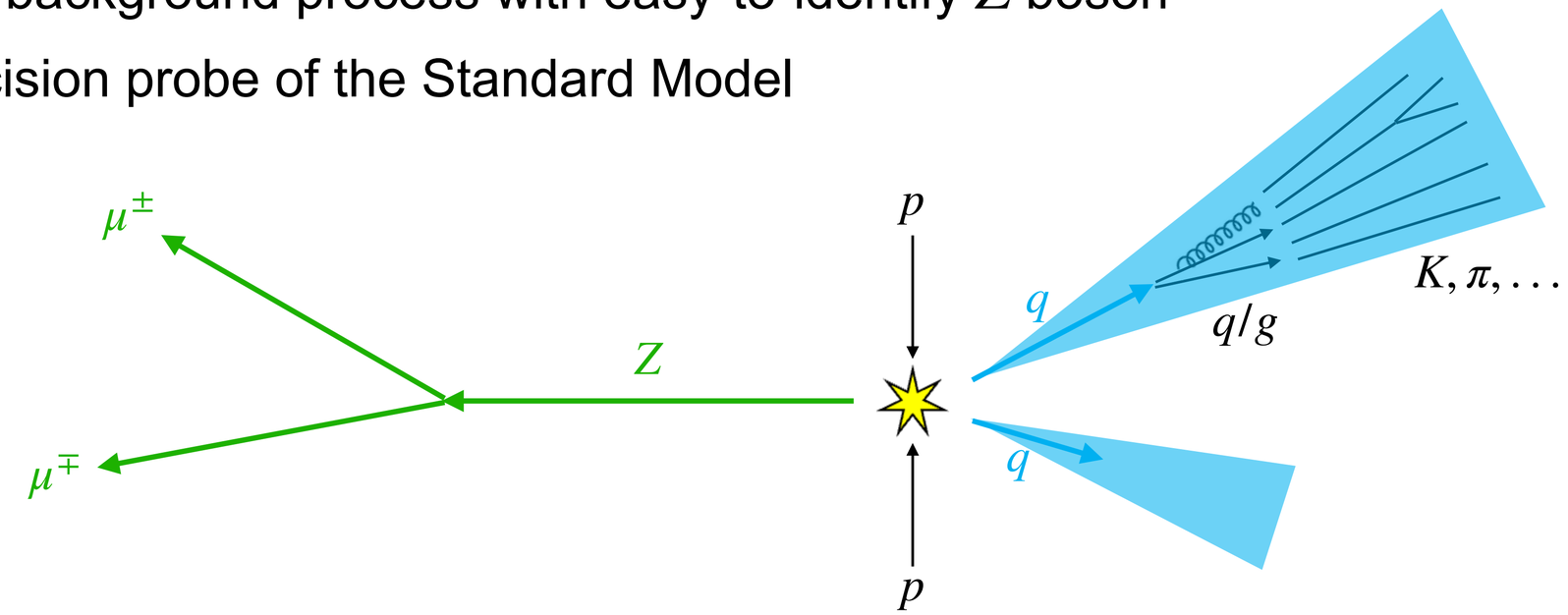


- The Z boson is a massive gauge boson
- Mediator of electroweak interaction
- In this case, interested in the decay to a muon and an anti-muon
  - ~3.4% of the time
  - “Easy” to reconstruct

- Quarks cannot exist as free particles
- Undergo hadronization, producing a collimated shower of particles known as a jet
- Reconstructed by using a clustering algorithm to group together tracks in the inner detector, calorimeter energy deposits, or a combination of both

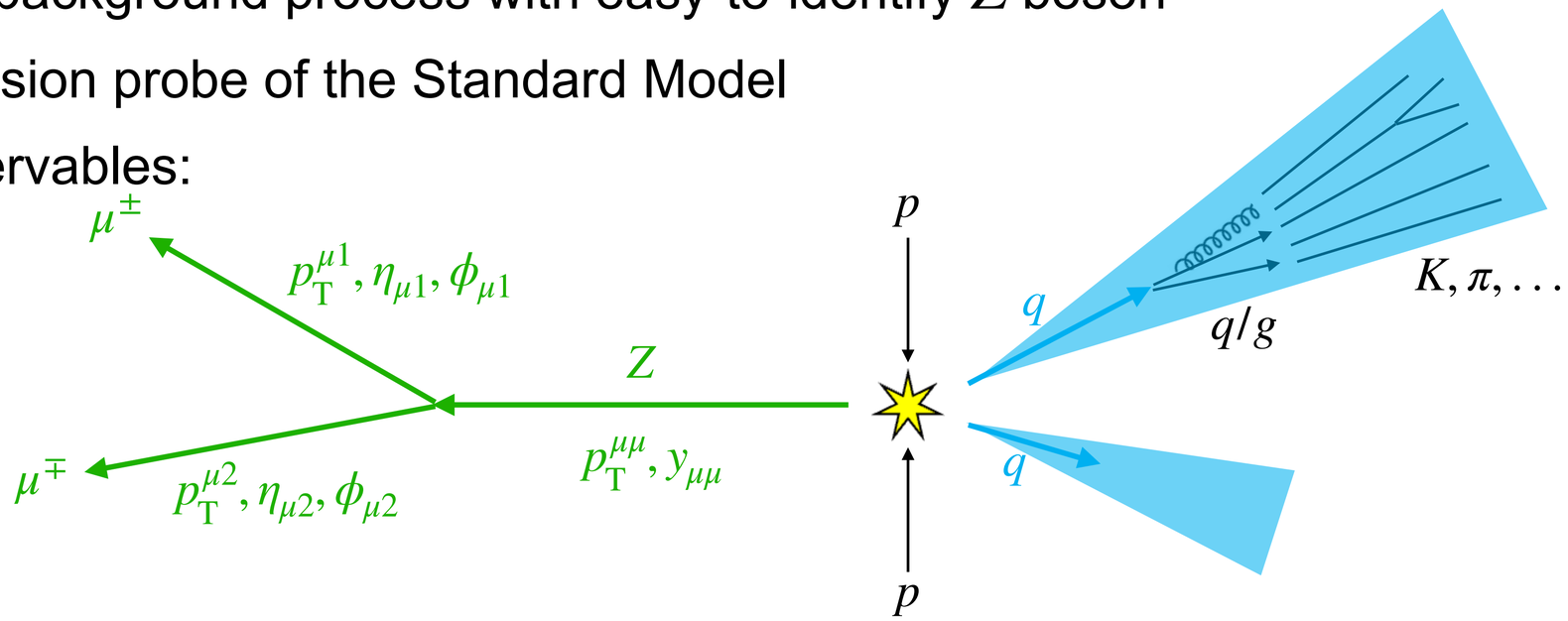
Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model



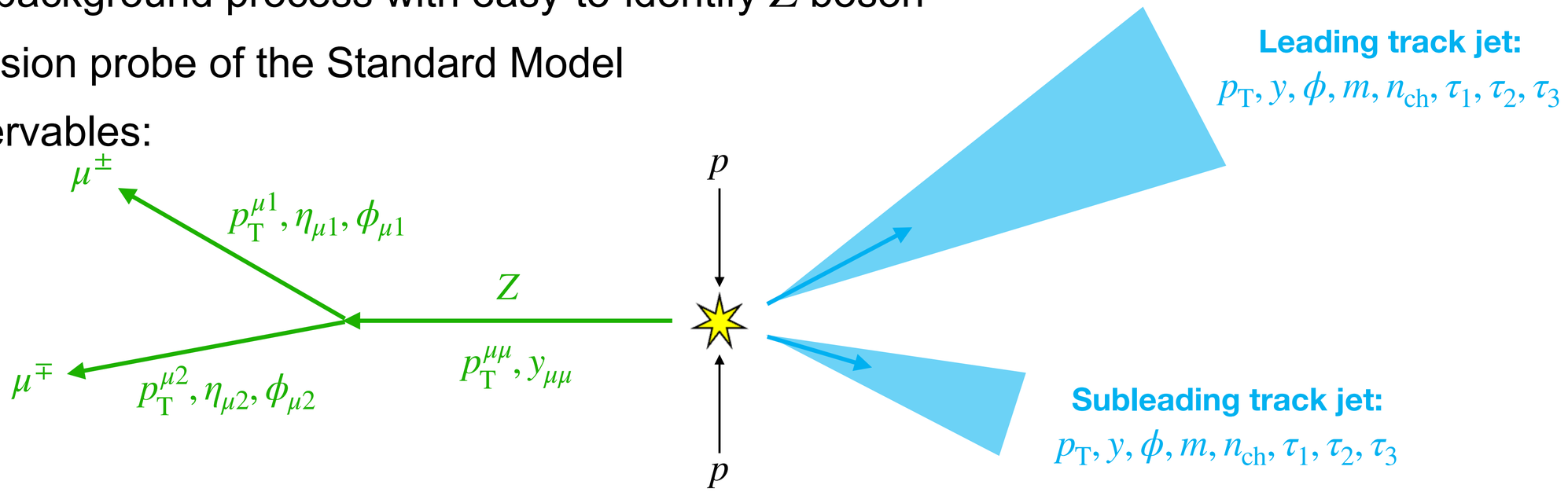
Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model
- Observables:



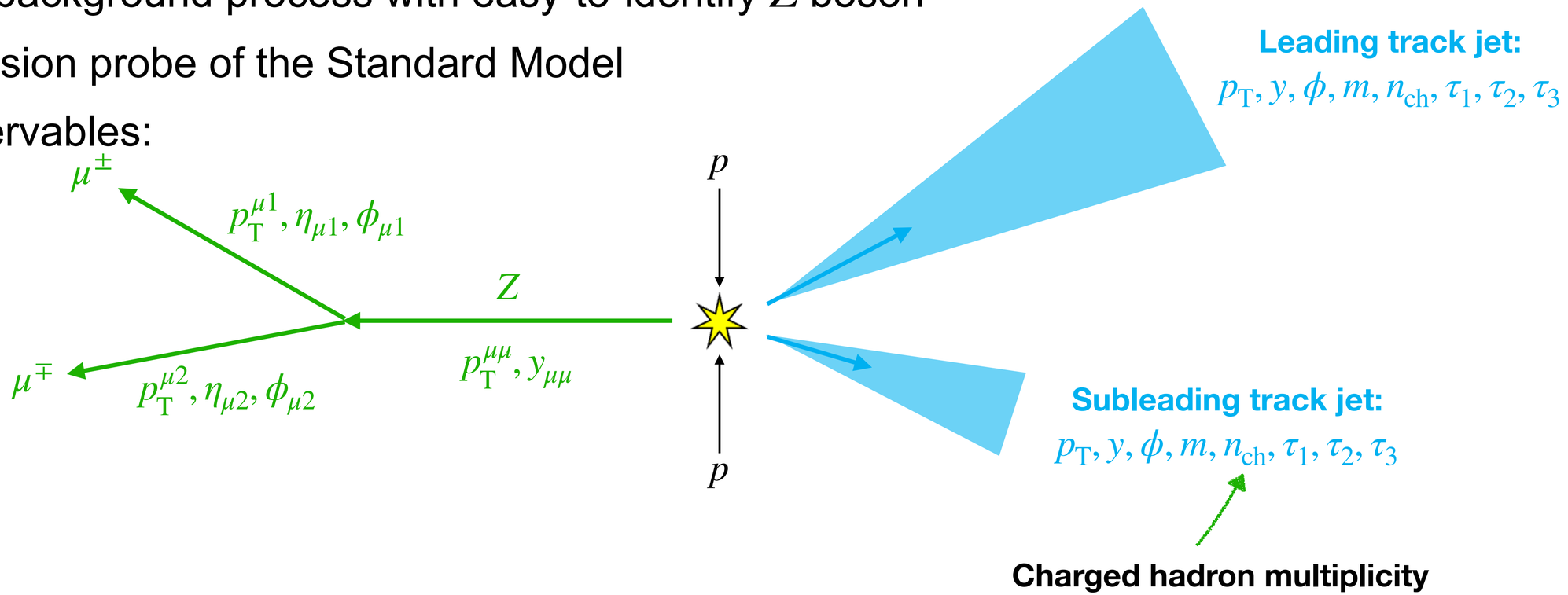
Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model
- Observables:



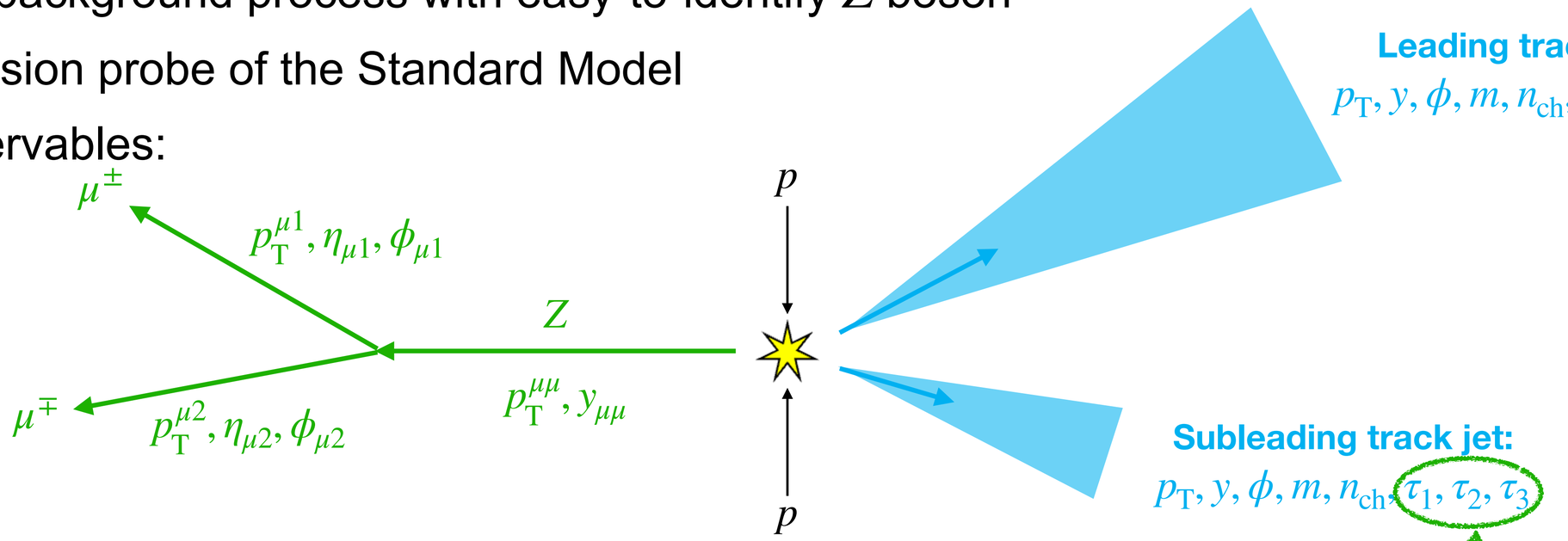
Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model
- Observables:



Z+jets events in the ATLAS detector

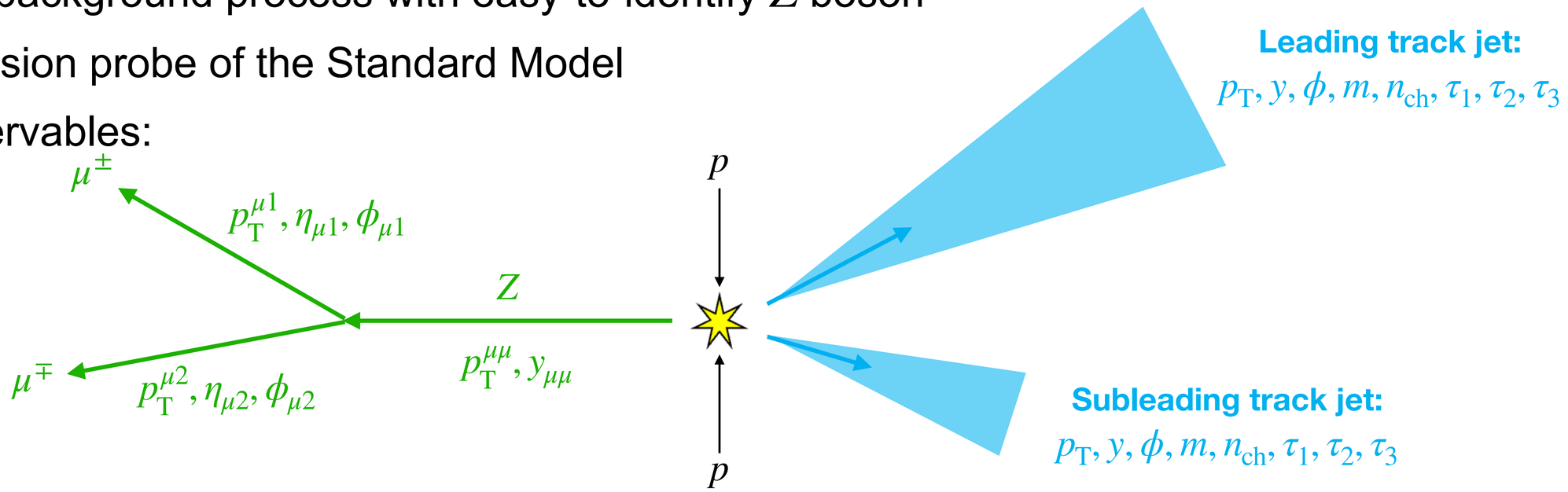
- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model
- Observables:



N-subjettiness: a jet substructure observable that quantifies the degree to which a jet can be classified as N subjects

Z+jets events in the ATLAS detector

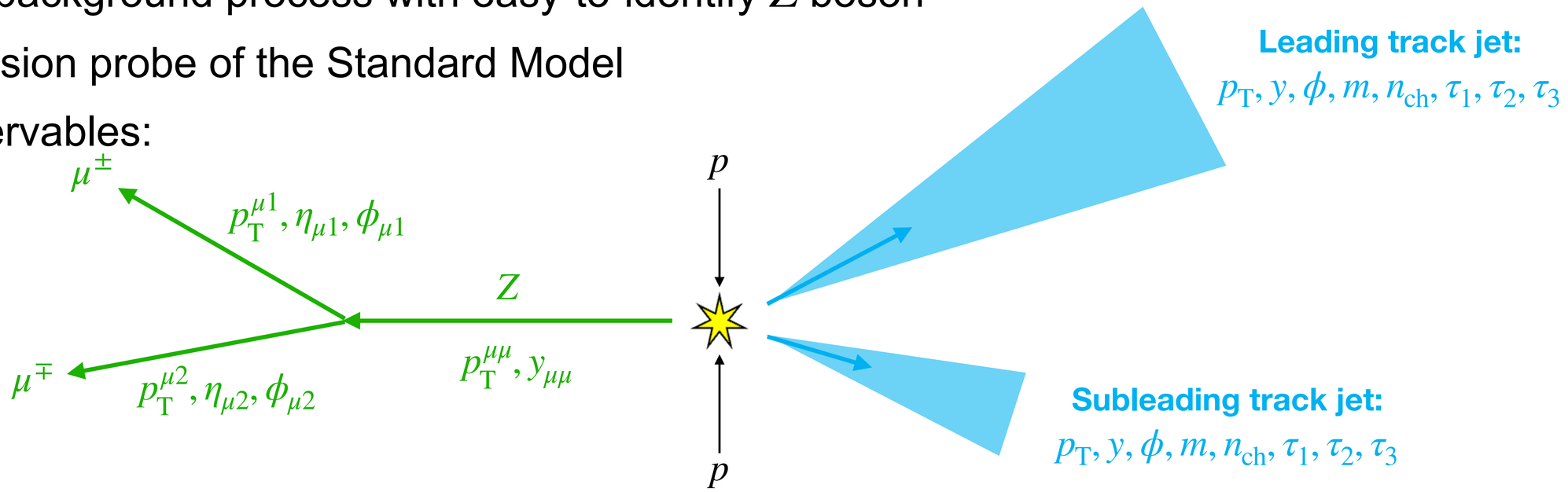
- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model
- Observables:



= 24 total observables

Z+jets events in the ATLAS detector

- Common process in the ATLAS detector and can be measured very precisely
- Low background process with easy-to-identify Z boson
- Precision probe of the Standard Model
- Observables:



= 24 total observables

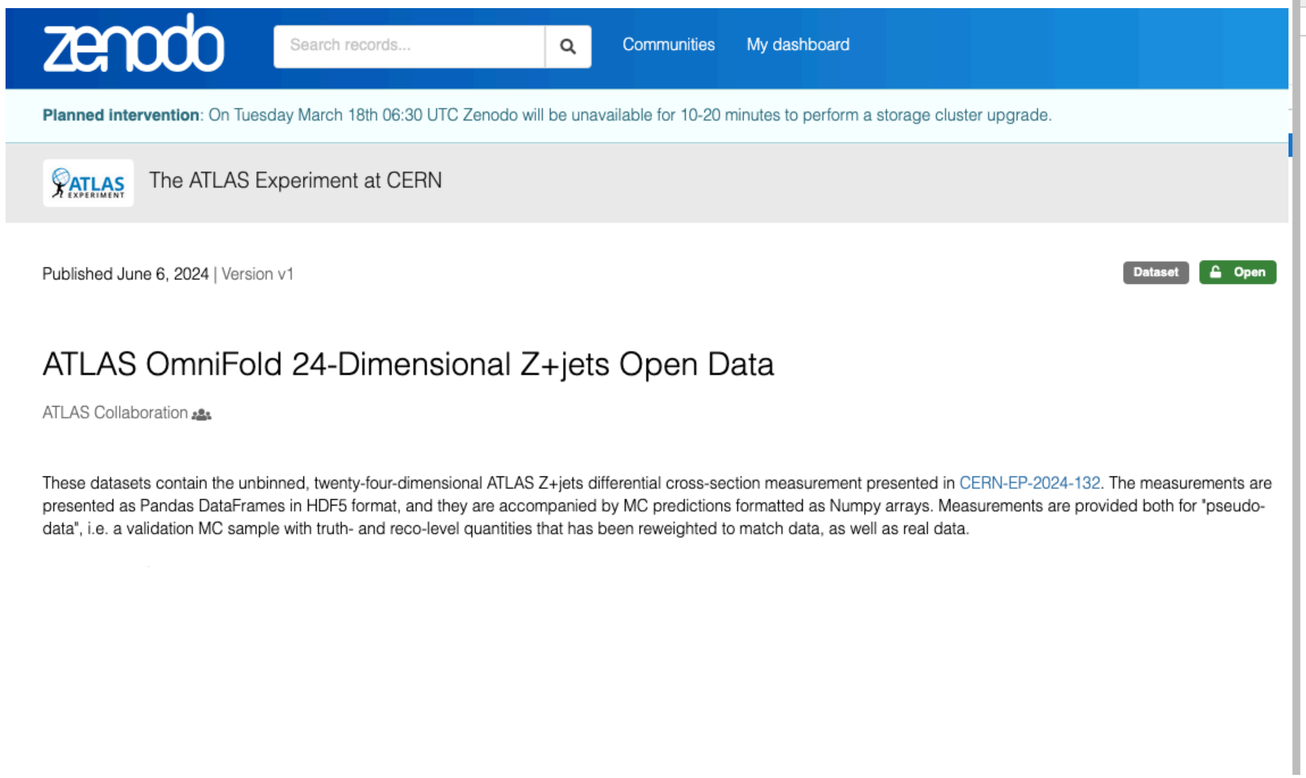
- Analysis utilizes the full ATLAS Run 2 dataset
  - Centre of mass energy of $\sqrt{s} = 13\,\mathrm{TeV}$ and total integrated luminosity of $139.0\,\mathrm{fb}^{-1}$

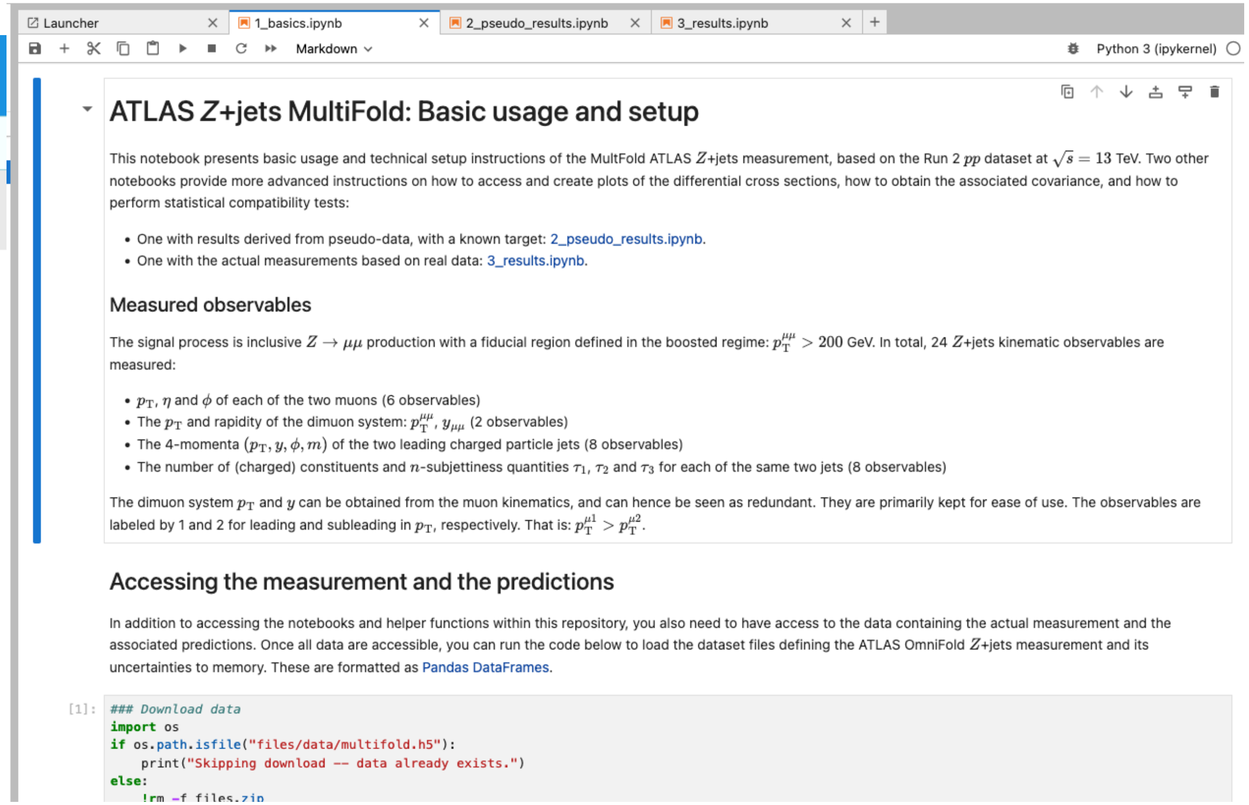
Results

- Two sets of results: one with pseudodata for validation, the other with data
  - Pseudodataset allows access to truth information (target) so can confirm algorithm is functioning as intended

Results

- Two sets of results: one with pseudodata for validation, the other with data
  - Pseudodataset allows access to truth information (target) so can confirm algorithm is functioning as intended
- Datasets are publicly available along with example notebooks for their usage





Results

- Two sets of results: one with pseudodata for validation, the other with data
  - Pseudodataset allows access to truth information (target) so can confirm algorithm is functioning as intended
- Datasets are publicly available along with example notebooks for their usage

A look at the event sample content

Assuming the above worked, we can look at a fraction of the key content of `multifold`. This is the most important data file, as it provides the central value and most uncertainties:

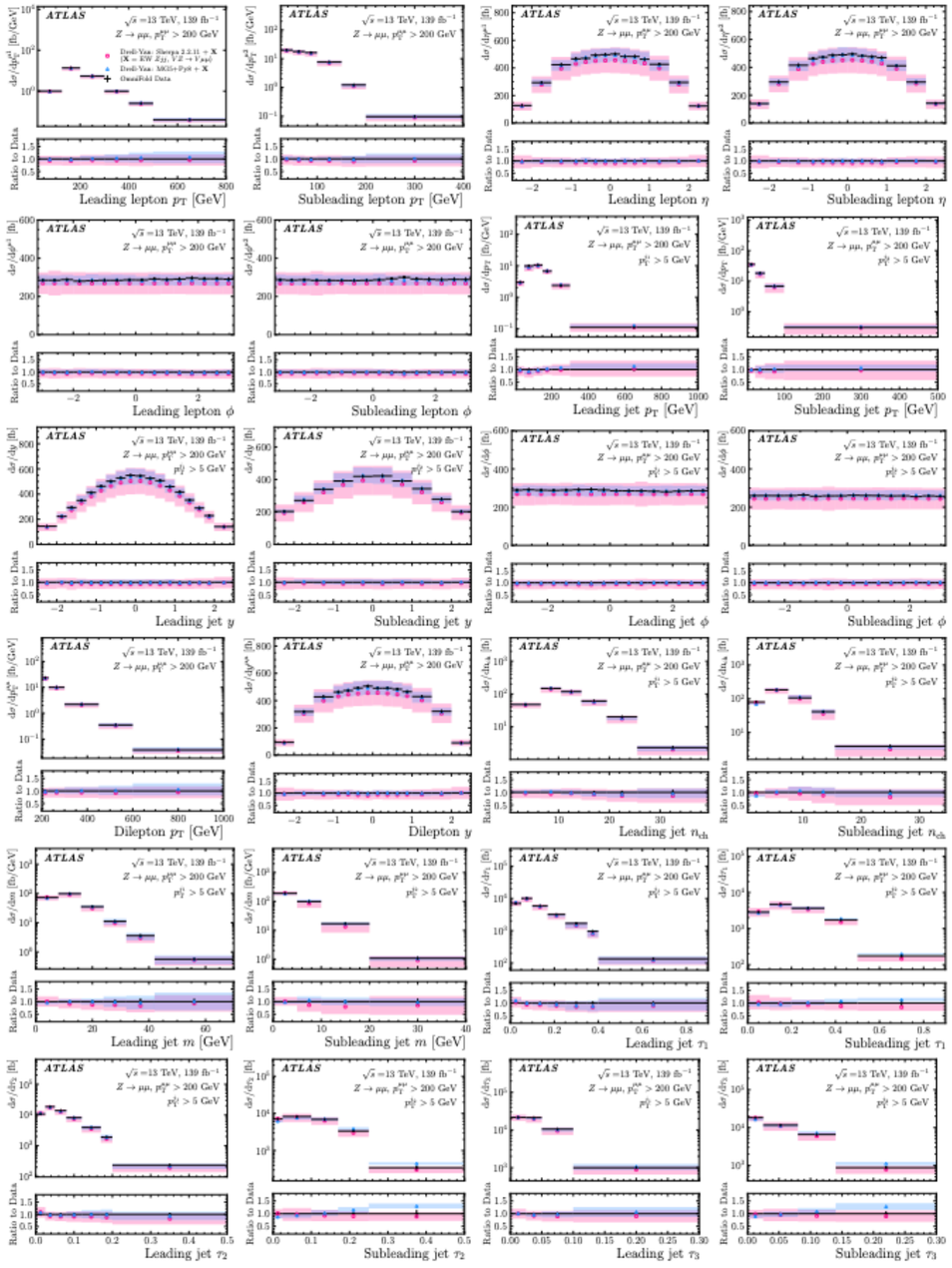
[4]:
```
multifold
```

[4]:

	pT_ll	pT_l1	pT_l2	eta_l1	eta_l2	phi_l1	phi_l2	y_ll	pT_trackj1	y_trackj1	...	weights_trackPtScale	weights_thec
0	479.442780	288.466919	198.183929	-0.117443	-0.334414	0.893844	0.543039	-0.205878	243.584351	-0.924242	...	0.003174	0
1	274.524994	166.120789	125.378044	0.313321	0.368928	2.230892	1.537930	0.337239	96.407021	-1.158080	...	0.008168	0
2	462.713226	335.697479	133.157684	0.766387	0.534180	1.191890	0.832184	0.700256	86.603012	-1.048886	...	0.001638	0
3	215.157608	189.518021	25.711994	1.083798	-0.190828	2.326127	2.406116	0.904586	222.285919	0.166012	...	0.004669	0
4	222.458313	128.850159	108.589226	-0.635713	-0.972127	-2.909612	2.656590	-0.789699	206.385544	0.392547	...	0.002102	0
...	...	...	...	...	...	...	...	...	...	...	...	...	
418009	934.971924	738.464722	196.525192	0.102944	-0.133273	2.388935	2.404151	0.053082	475.055756	0.811634	...	0.000069	0
418010	245.813461	166.847061	93.757919	1.308837	1.533588	1.591105	0.884510	1.389817	713.371887	1.452970	...	0.000193	0
418011	478.670349	378.737518	108.016479	-0.328871	-0.227935	-1.235470	-1.675783	-0.306456	100.177452	0.050614	...	0.001969	0
418012	278.586029	249.255356	43.581135	0.632484	0.687496	-0.966683	-0.071259	0.640673	224.825485	1.490560	...	0.003238	0
418013	244.505249	219.796280	40.357105	1.833223	2.060005	-2.618515	2.682077	1.868578	97.057076	0.686043	...	0.000947	0

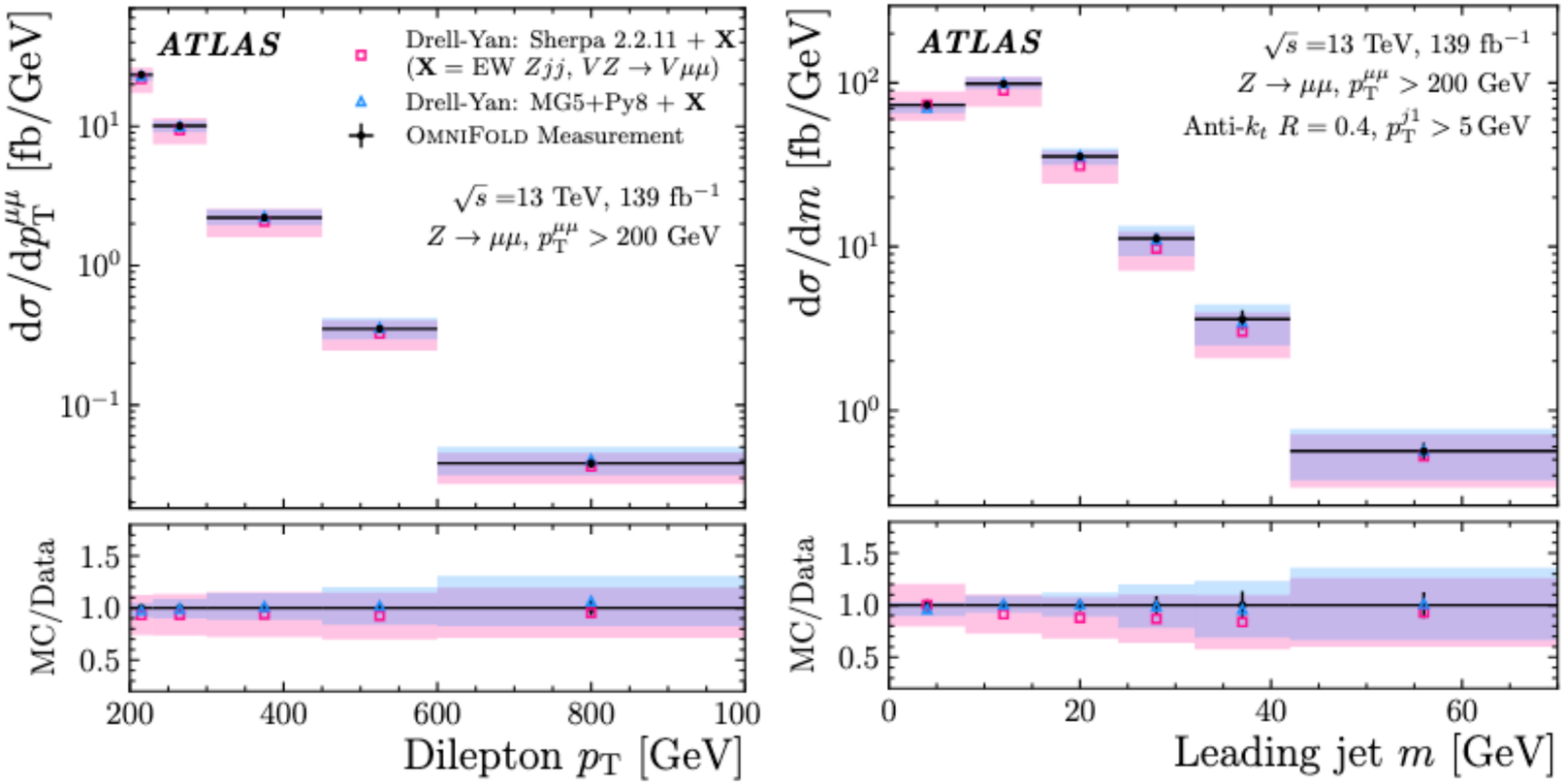
418014 rows × 276 columns

Results: 24 observables



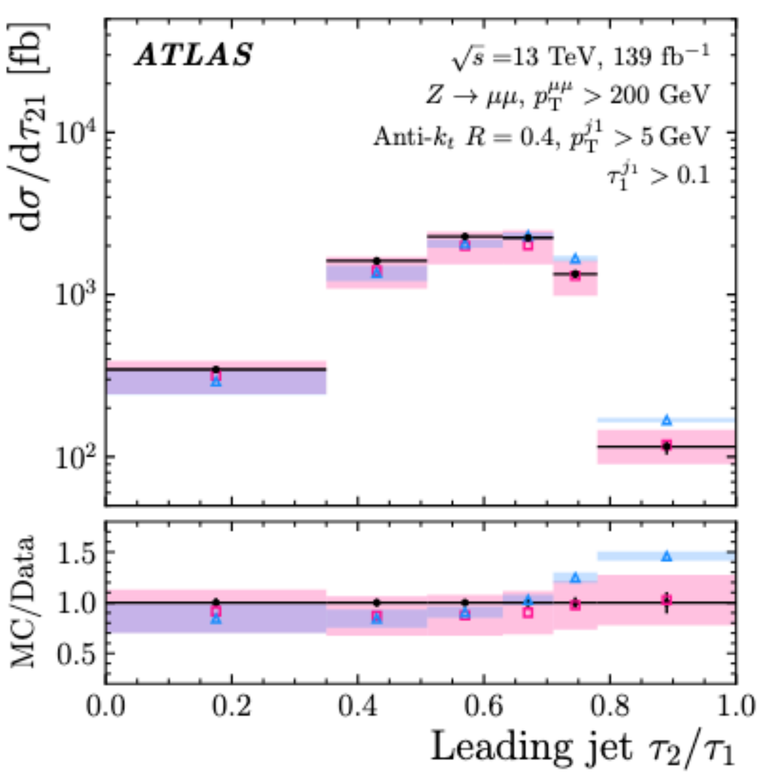
Results: 24 observables

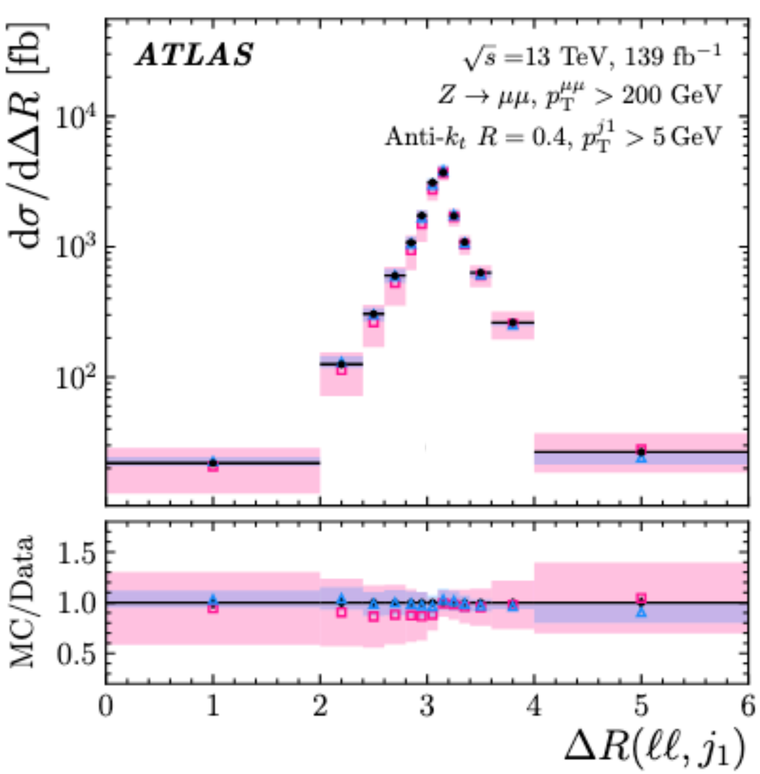
- Histograms crafted from the results dataset

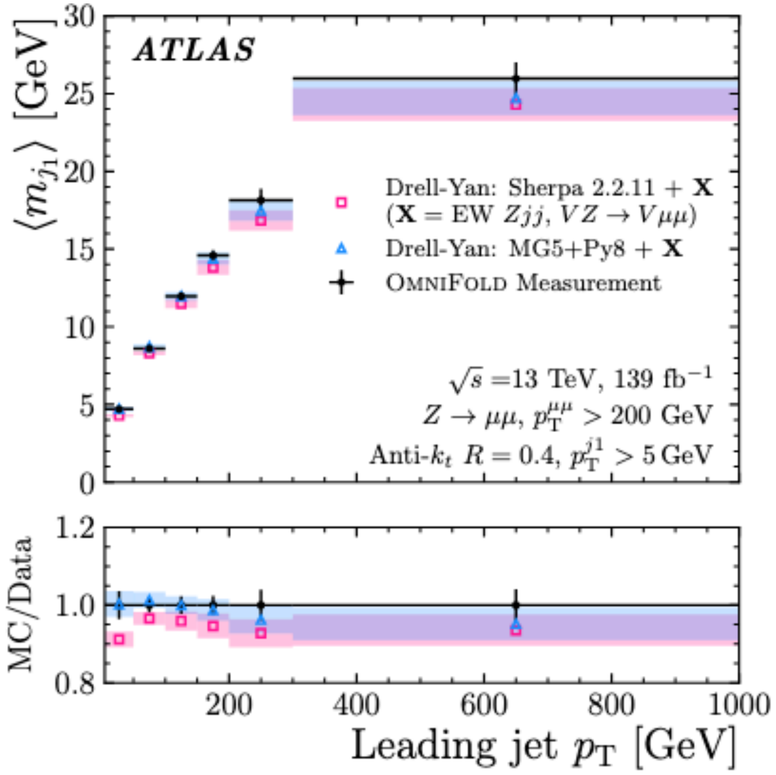


Results: Derived observables

- Observables constructed from the resulting dataset that were not included in the original 24 input observables

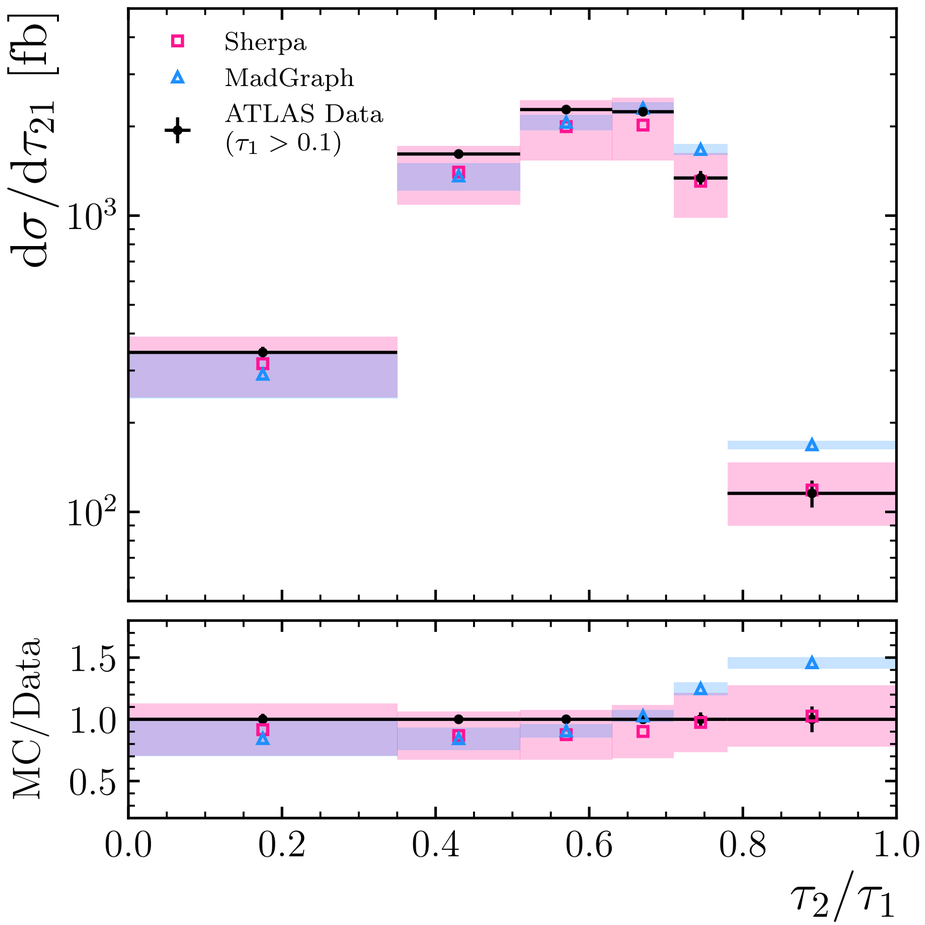






Results: Reduced phase space

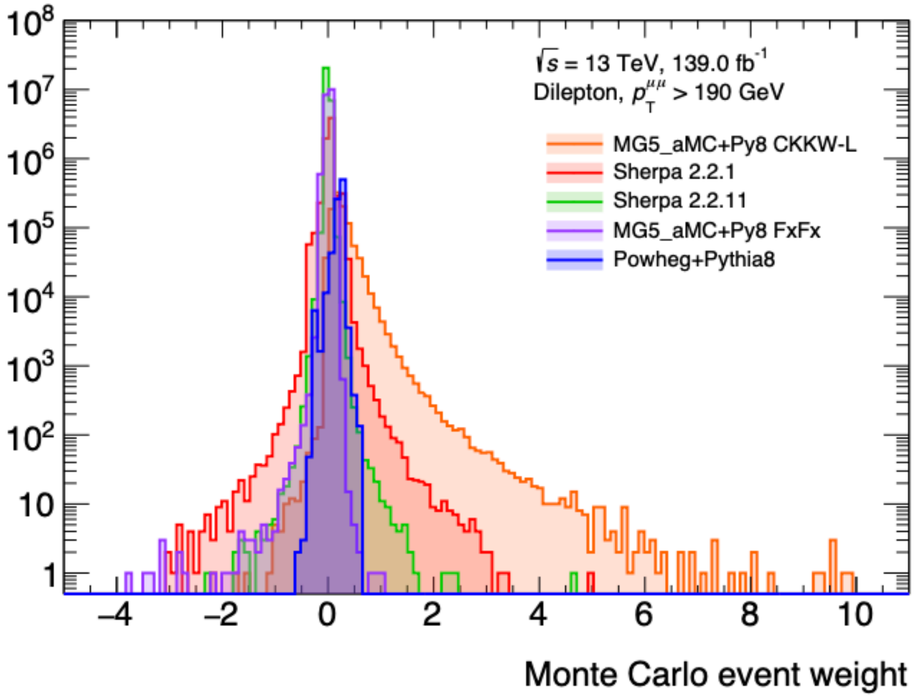
- Examining a subset of the original phase space



Demo!

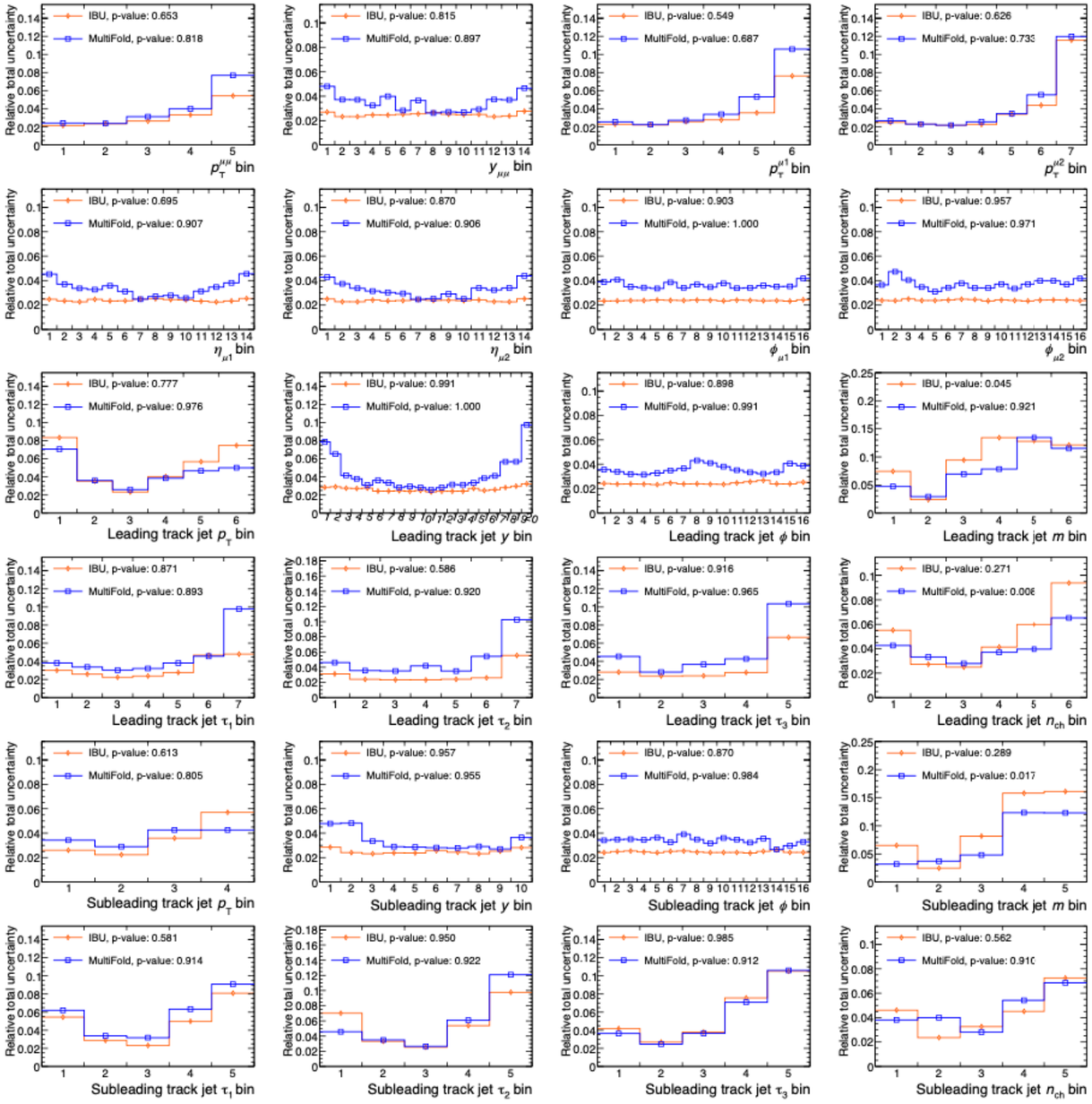
Lessons learned

- Lots of data preprocessing!
  - MC samples at ATLAS have a large fraction of negative weights
  - We had to do an additional reweighting step in the preprocessing to remove negative weights



Lessons learned

- Lots of data preprocessing!
  - MC samples at ATLAS have a large fracti
    of negative weights
  - We had to do an additional reweighting st
    in the preprocessing to remove negative v
- Uncertainties!
  - Neural networks are randomly initialized
  - Slightly different result each time
  - Need to ensemble networks to get a
    nominal weight with an uncertainty
  - Results in an additional uncertainty

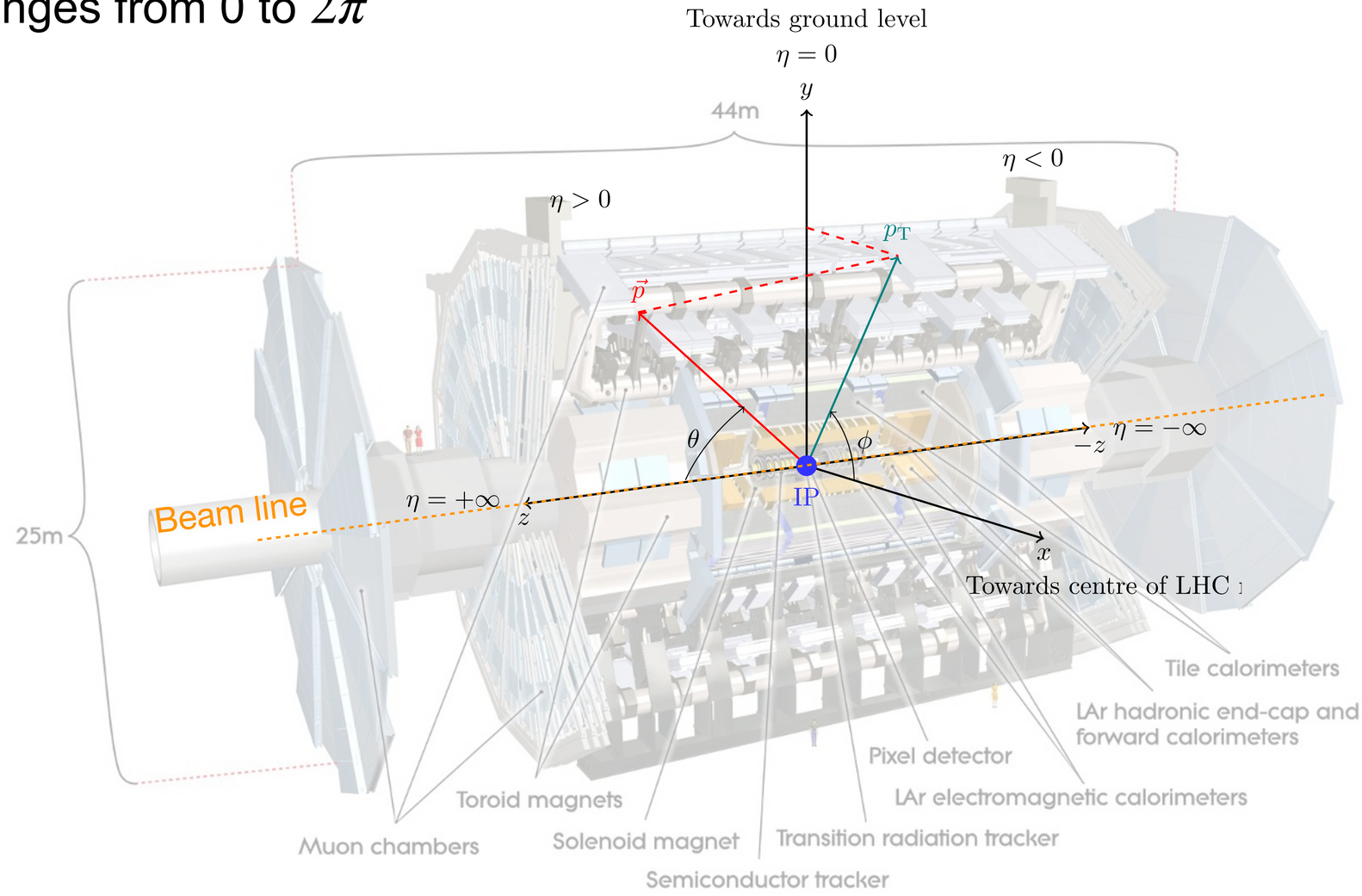


Lessons learned

- Lots of data preprocessing!
  - MC samples at ATLAS have a large fraction of negative weights
  - We had to do an additional reweighting step in the preprocessing to remove negative weights
- Uncertainties!
  - Neural networks are randomly initialized
  - Slightly different result each time
  - Need to ensemble networks to get a nominal weight with an uncertainty
  - Results in an additional uncertainty
- Being careful with our final result to make sure things are independent
  - Split our samples into two, the larger chunk was used to train the NNs
  - The NN was then applied to a completely independent set of events, which is our final result

More lessons learned

- Wraparound effect
  - ϕ coordinate in ATLAS ranges from 0 to 2π



More lessons learned

- Wraparound effect
  - ϕ coordinate in ATLAS ranges from 0 to 2π
  - NNs struggle with this effect
  - Solution: input $\sin(\phi)$ and $\cos(\phi)$ instead
- The ghost of ignored normalization
  - Weights are standardized at each step of the MultiFold process
  - In order to regain the correct cross section for the final measurement had to determine the appropriate scaling factor using properties of the MC and data
  - May be possible to incorporate this into the networks?

Conclusions

- Presented a precision measurement of $Z(\rightarrow \mu\mu)$+jets production at the LHC
- This analysis demonstrates the first use of the MultiFold method in an ATLAS analysis
- The first unbinned and 24 dimensional measurement
- Results are publicly available in the form of a dataset containing:
  - The value of each of the 24 observables
  - A nominal weight
  - Alternative weights associated with each of the uncertainties
- User guides are also published to provide instructions for usage
- MultiFold shows comparable results to standard methods, but with all the added benefits
- All tests performed with pseudodata show good agreement with the target
- Theoretical predictions generally agree well with data
- Results in this format are much more flexible and very useful for the particle physics community

Thanks for listening!