Unfolding Tutorial

Top Workshop

Introduction

- Unfolding and long and complicated a procedure and too much to explain everything in one hour.
- Similarly, setting up and running a realistic unfolding is probably not going to be too useful with the limited time we have.
- Therefore, this tutorial is going to explain unfolding, from a practical point-of-view.
- I will explain the terminology that is used and explain the tests you need to perform in your analysis.
  - ➡ There are no stupid questions! Interrupt me and ask anything.
  - ➡ There will also be time for questions at the end.

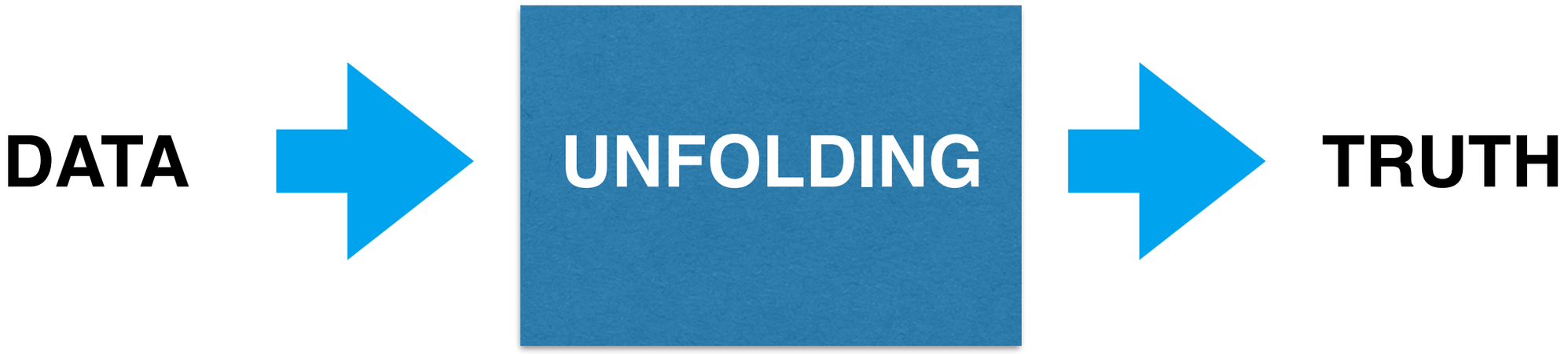
- For a more pedagogical overview, see Baptise's talk: https://indico.cern.ch/event/794004/contributions/3326125/attachments/1802168/2939911/UnfoldingReview.pdf
- And for a walkthrough, see here: http://dpnc.unige.ch/~sfyrla/teaching/Statistics/handsOn4.html

What is Unfolding?

- Unfolding is more generally known as ***"deconvolution"*** where we have some signal **f** convoluted with some noise **G** to give some recorded data **h**.

$$f * G = h$$

- In particle physics **f** and **h** are generally discreet data in the form of binned histograms and **G** is a combination of detector smearing and acceptance effects**.**
- So unfolding is just the process of going from **h** → **f**.
- When **f** and **h** are histograms (i.e. vectors of data), **G** has the form of a matrix and the process of going from **h** → **f** is an example of a matrix inversion problem**.**

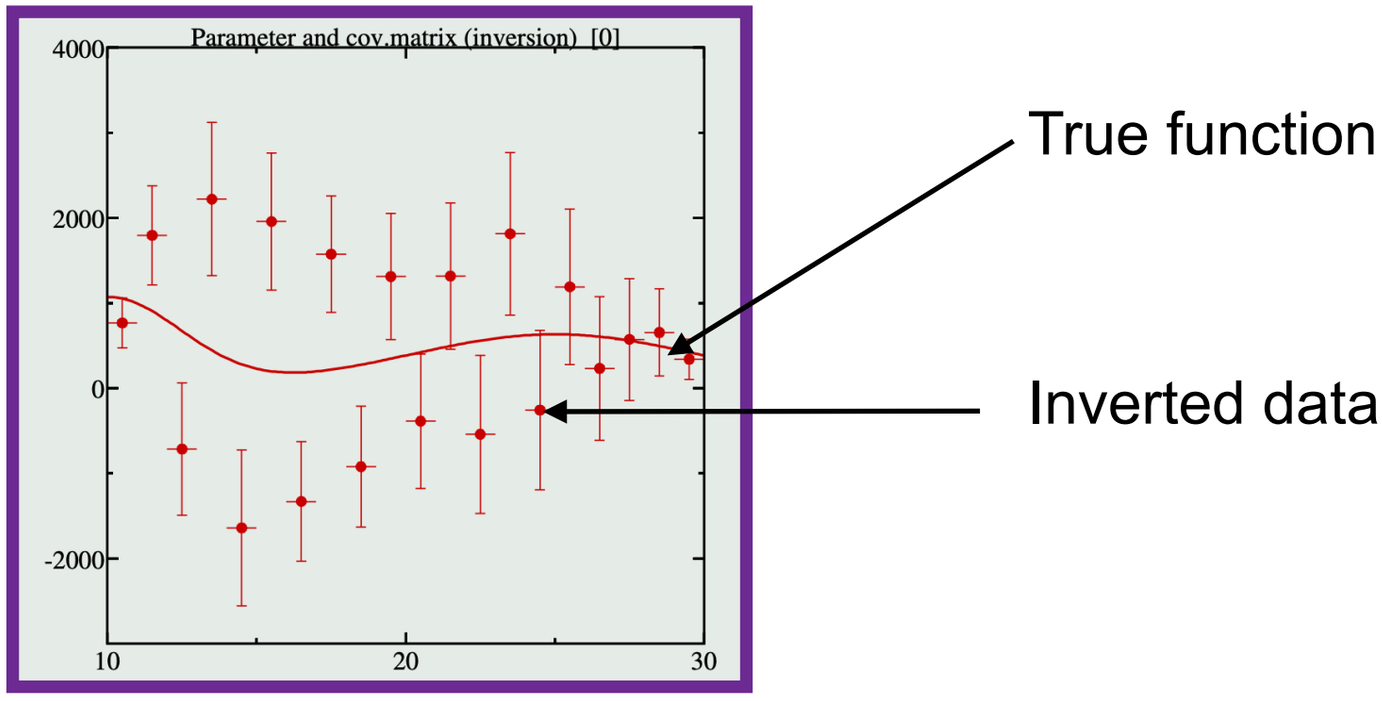


Why Matrix Inversion sucks

- We usually build this matrix (called the response matrix) from Monte Carlo (MC).
- So what happens if we invert the matrix and apply it to our data vector?

$$f = G^{-1}h \qquad V_f = G^{-1}\ V_h\ (G^{-1})^T$$

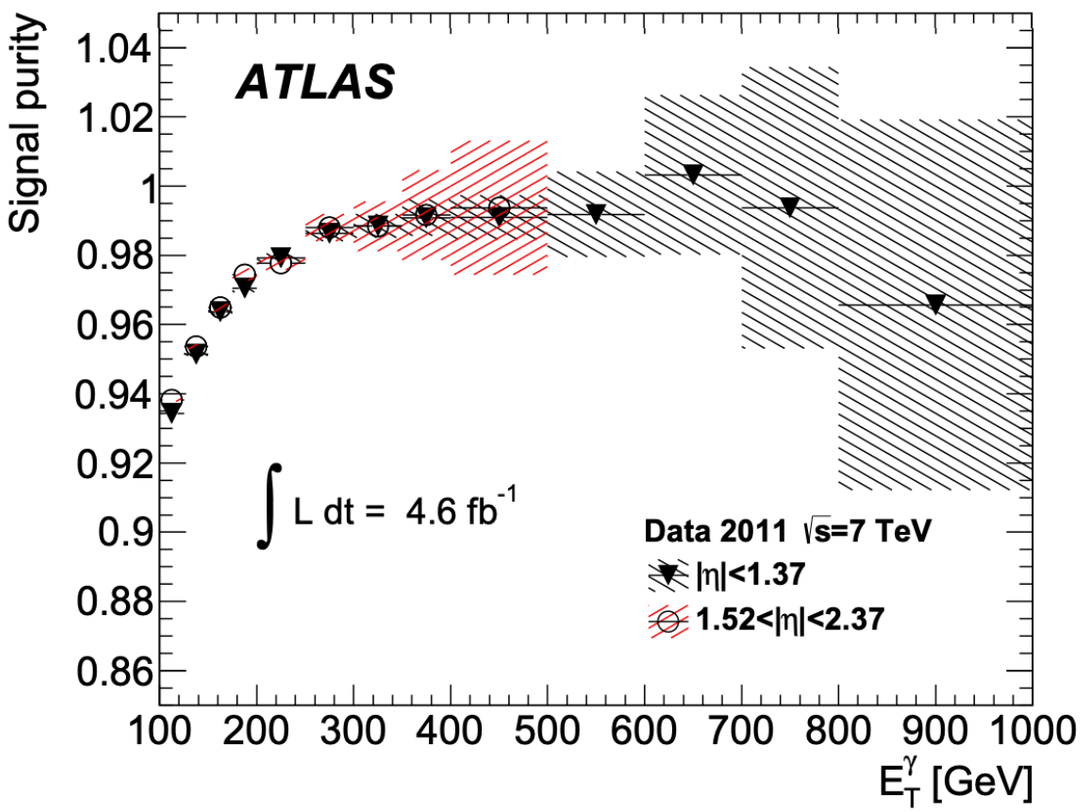
- Basically, this:

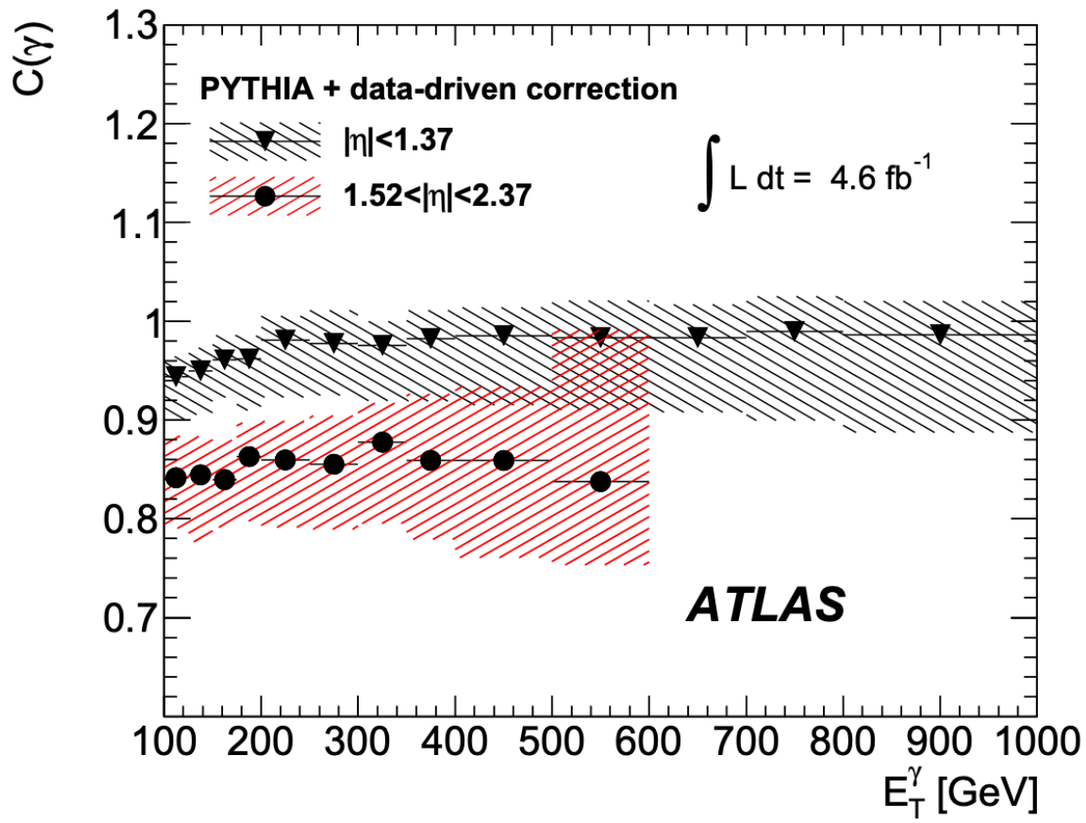


- Large anti-correlations in the covariance matrix allow this result (which actually agrees with the true function). What does that mean?
- It means that statistical fluctuations get amplified and you get stupid results. We need to dampen (regularise) these fluctuations somehow, and this is what almost all unfolding methods do.

Bin-by-bin unfolding

- The first method is the simplest but also the most limited in scope.
- It can only be used when “purity” is very high, i.e. most events at truth level do not migrate to different bins after detector effects.
- Correction factors are then used to correct the data to truth level.
- A simple method that has it’s uses, but not often useful in top.

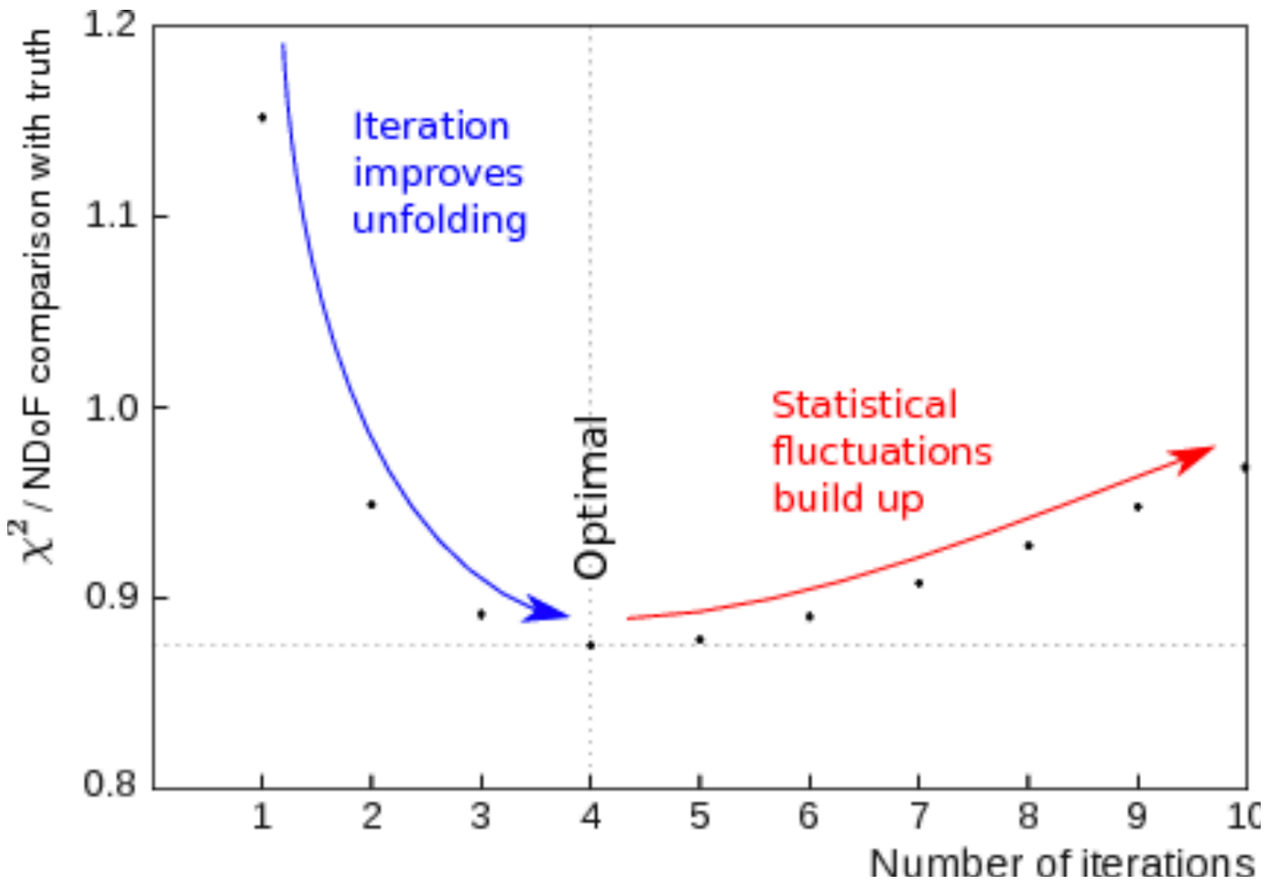




$$C(\gamma) = \frac{\text{truth}}{\text{reco}}$$

Iterative Bayesian Unfolding

- Far more commonly used is Bayesian Iterative Unfolding (IBU).
- In this unfolding method we use Bayes theorem recursively and use our truth distribution as a prior to control the harmonic oscillations.
- Has the slightly unusual feature that the more iterations you do, the weaker the regularisation.



- Finding the optimal stopping point is an important feature of using IBU.

The Recipe

$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\mathrm{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\mathrm{acc}} \cdot (N^j_{\mathrm{obs}} - N^j_{\mathrm{bkg}}),$$

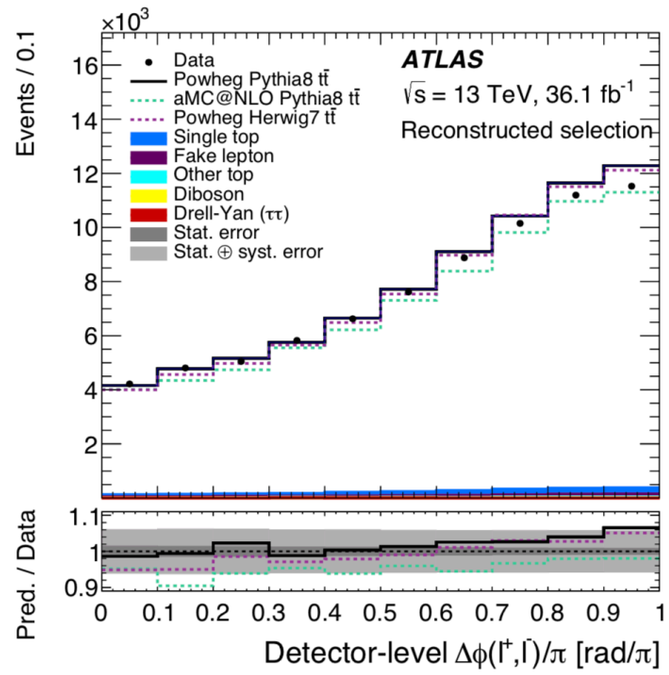
The Recipe

DATA

$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\mathrm{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\mathrm{acc}} \cdot (N^j_{\mathrm{obs}} - N^j_{\mathrm{bkg}}),$$

TRUTH

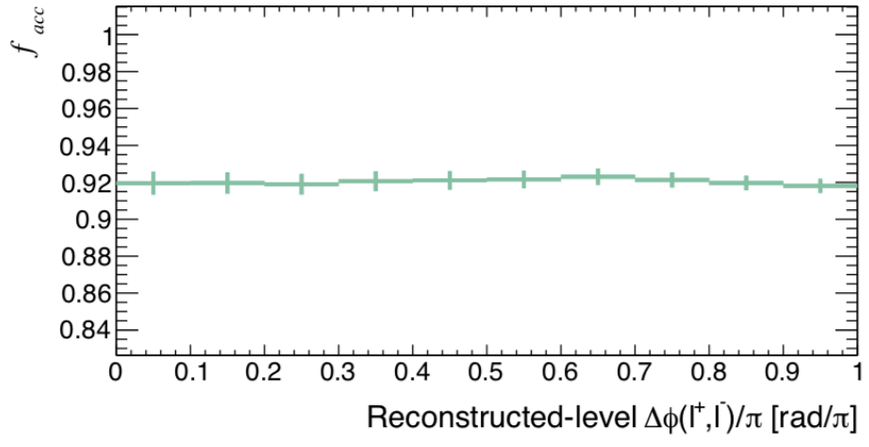
The Recipe



$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\mathrm{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\mathrm{acc}} \cdot (N^j_{\mathrm{obs}} - N^j_{\mathrm{bkg}}),$$

- Step 1: Subtract background events.

The Recipe

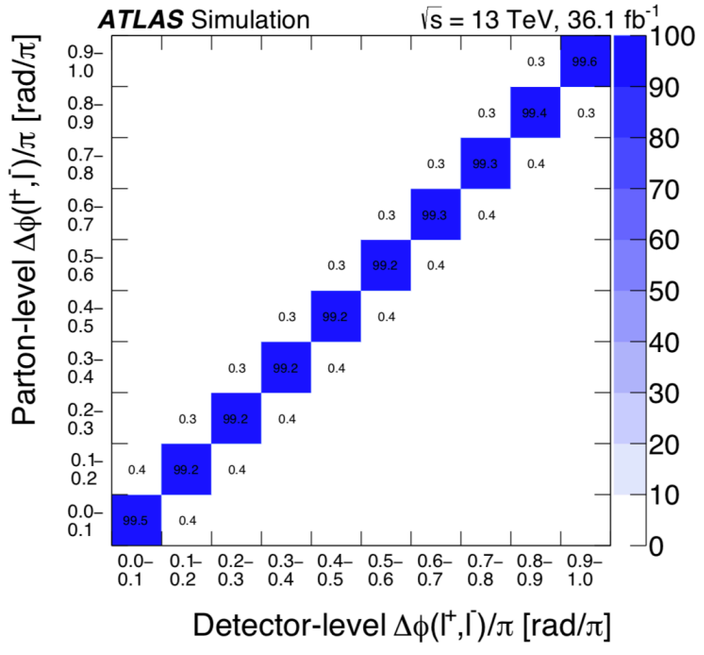


$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L}\cdot\Delta X^i\cdot\epsilon^i_{\mathrm{eff}}}\cdot\sum_j R^{-1}_{ij}\cdot f^j_{\mathrm{acc}}\cdot(N^j_{\mathrm{obs}}-N^j_{\mathrm{bkg}}),$$

- Step 1: Subtract background events.
- Step 2: Correct for ***"non-fiducial"*** events that are in the Reco-level but aren't in truth.

$$f^j_{\mathrm{acc}} = \frac{\mathrm{reco}_j \wedge \mathrm{truth}_j}{\mathrm{reco}_j}$$

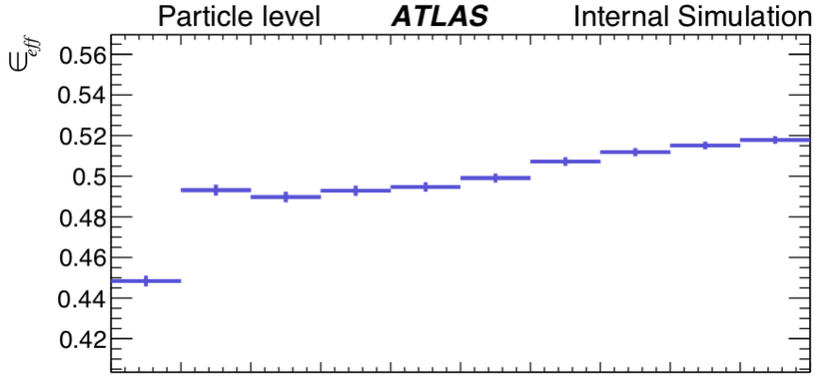
The Recipe



$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\mathrm{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\mathrm{acc}} \cdot (N^j_{\mathrm{obs}} - N^j_{\mathrm{bkg}}),$$

- Step 1: Subtract background events.
- Step 2: Correct for ***“non-fiducial”*** events that are in the Reco-level but aren’t in truth.
- Step 3: Remove the detector smearing.

The Recipe



$$\epsilon^{j}_{\text{eff}} = \frac{\text{truth}_j}{\text{reco}_j \wedge \text{truth}_j}$$

$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\text{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\text{acc}} \cdot (N^j_{\text{obs}} - N^j_{\text{bkg}}),$$

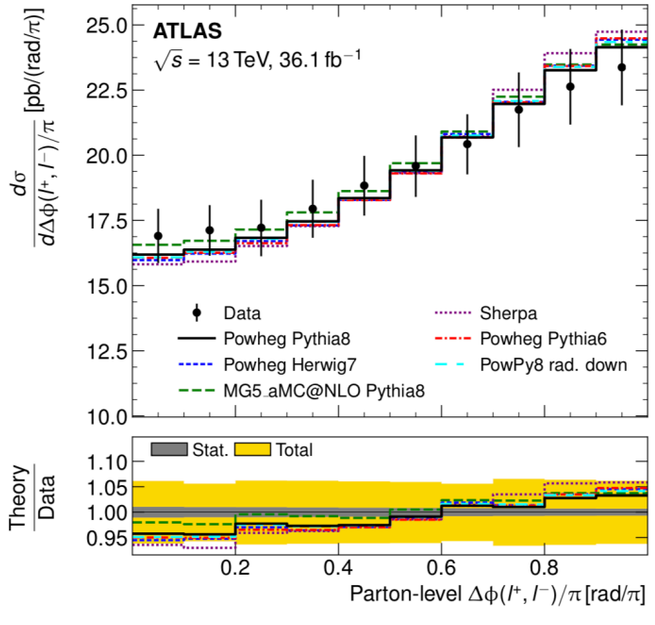
- Step 1: Subtract background events.
- Step 2: Correct for ***"non-fiducial"*** events that are in the Reco-level but aren't in truth.
- Step 3: Remove the detector smearing.
- Step 4.1: Extrapolate to your truth phase-space.

The Recipe

$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\mathrm{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\mathrm{acc}} \cdot (N^j_{\mathrm{obs}} - N^j_{\mathrm{bkg}}),$$

- Step 1: Subtract background events.
- Step 2: Correct for ***"non-fiducial"*** events that are in the Reco-level but aren't in truth.
- Step 3: Remove the detector smearing.
- Step 4.1: Extrapolate to your truth phase-space.
- Step 4.2: Convert event count to cross-section.

The Recipe



$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L}\cdot\Delta X^i\cdot\epsilon^i_{\mathrm{eff}}}\cdot\sum_j R^{-1}_{ij}\cdot f^j_{\mathrm{acc}}\cdot(N^j_{\mathrm{obs}}-N^j_{\mathrm{bkg}}),$$

- Step 1: Subtract background events.
- Step 2: Correct for ***"non-fiducial"*** events that are in the Reco-level but aren't in truth.
- Step 3: Remove the detector smearing.
- Step 4.1: Extrapolate to your truth phase-space.
- Step 4.2: Convert event count to cross-section.
- Step 5: Profit!

The Code

- Most common code that is used is called “RooUnfold”.
- Almost all of the unfolding frameworks that exist in the top group are based on it (with the notable exception of FBU) and usually they are just complicated python wrappers.
- https://svnsrv.desy.de/public/unfolding/RooUnfold/trunk/ <— make sure to get trunk.

- ROOT does have it’s own unfolding class, called TUnfold, which is commonly used by CMS (implements a more frequentist type of unfolding).
- In the end it doesn’t matter too much, most (but not all) of the unfolding methods do the same thing and the recipe + tests are essentially the same.

The Code

```
response = RooUnfoldResponse(hist_reco, hist_truth, migration_matrix)

unfold_bayes = RooUnfoldBayes(response, hist_reco, 4, False)

hist_result = unfold_bayes.Hreco().Clone(random_name)
```

$$\frac{\mathrm{d}\sigma_{t\bar{t}}}{\mathrm{d}X^i} = \frac{1}{\mathcal{L} \cdot \Delta X^i \cdot \epsilon^i_{\mathrm{eff}}} \cdot \sum_j R^{-1}_{ij} \cdot f^j_{\mathrm{acc}} \cdot (N^j_{\mathrm{obs}} - N^j_{\mathrm{bkg}}),$$

- **RooUnfoldResponse** is a class that basically implements R^{-1}_{ij}
- However, the code will also try to do f^j_{acc} and $\epsilon^i_{\mathrm{eff}}$ for you, you shouldn't let it!
- An easy way to prevent this is to simply build the response matrix as follows:

```
response = RooUnfoldResponse(mig_matrix.ProjectionX(), mig_matrix.ProjectionY(), mig_matrix)
```

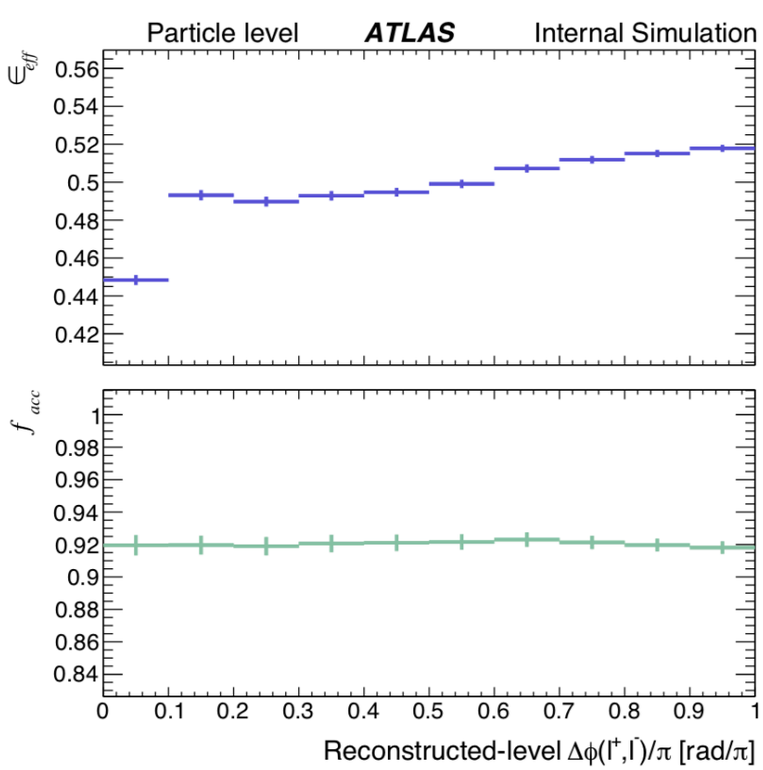
- This way, RooUnfold simply thinks that f^j_{acc} are $\epsilon^i_{\mathrm{eff}}$ equal to 1 for all bins.
- You then obviously have to apply these bin-by-bin corrections yourself.

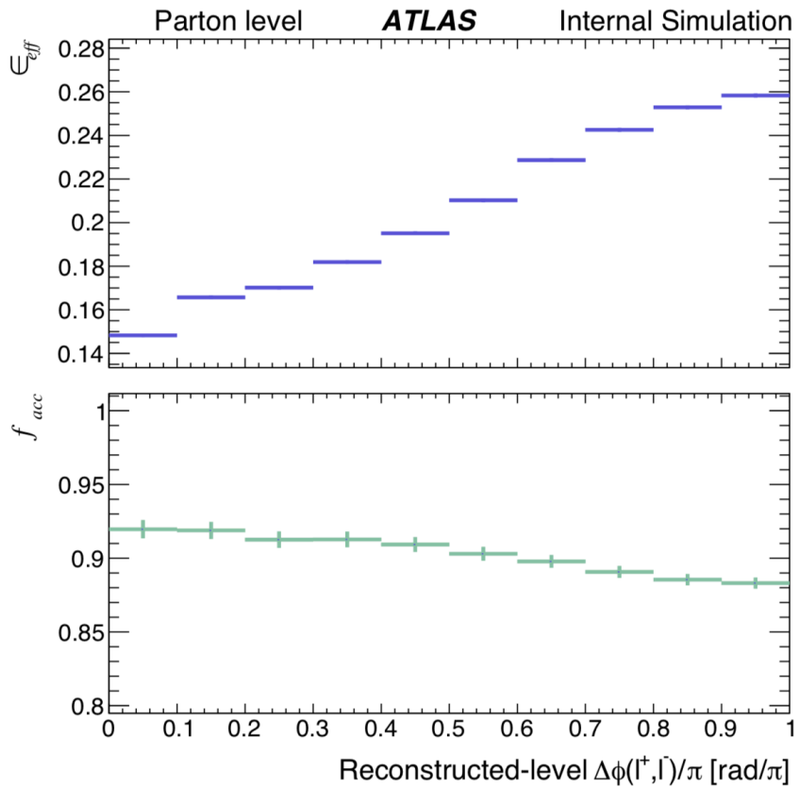
The Code

```
unfold_bayes = RooUnfoldBayes(response, hist_reco, iterations)
unfold_svd   = RooUnfoldBayes(response, hist_reco, kterm)
unfold_bin   = RooUnfoldBayes(response, hist_reco)
unfold_t     = RooUnfoldTUnfold(response, hist_reco)
unfold_inv   = RooUnfoldInvert(response, hist_reco)
```

- RooUnfold allows you to easily switch between several different unfolding options, but all other methods are the same (i.e. getting your histograms, setting error options etc).
- One of the main reasons we like RooUnfold as a tool!

Particle vs. Parton





- From the efficiency and acceptance plots you can really see the difference between parton level in a full phase-space and particle level in a fiducial phase-space.
- In particle level, the acceptance is mostly flat (since our particle and reco level leptons are very very similar) and the extrapolation is also no large (meaning our fiducial phase-space is similar in “size” to our reco one).
- In parton level you can see the extrapolation is much larger, and there are more efficiency effects (since parton leptons are not “dressed”).

Pull Tests

- The pull of an observable tells you two things:
  - ➡ If your results are biased
  - ➡ If your uncertainties are correct
- The pull is constructed by:

$$\text{pull} = \frac{\text{obs.} - \text{exp.}}{\sigma(\text{obs.})}$$

- If σ(obs.) is gaussian, then the pull (if calculated for many statistically independent pseudo-experiments) should take the form of a normal distribution, where the parameters can be interpreted in the following way:
  - ➡ μ = 0: unbiased results, μ != 0, probably some bias
  - ➡ σ = 1: errors correctly estimated, σ < (>) 1 errors are over (under) estimated.
- It is generally a good idea to have pulls with μ = 0 and σ = 1, but it's not the end of the world if they don't, you will just need to apply a correction based on your pull tests.

Pull Tests

- The pull of an observable tells you two things:
  - ➡ If your results are biased
  - ➡ If your uncertainties are correct
- The pull is constructed by:

$$\text{pull} = \frac{\text{obs.} - \text{exp.}}{\sigma(\text{obs.})}$$

- If σ(obs.) is gaussian, then the pull (if calculated for many statistically independent pseudo-experiments) should take the form of a normal distribution, where the parameters can be interpreted in the following way:
  - ➡ μ = 0: unbiased results, μ != 0, probably some bias
  - ➡ σ = 1: errors correctly estimated, σ < (>) 1 errors are over (under) estimated.
- It is generally a good idea to have pulls with μ = 0 and σ = 1, but it's not the end of the world if they don't, you will just need to apply a correction based on your pull tests.

Pull Tests

- The pull of an observable tells
  - ➡ If your results are biased
  - ➡ If your uncertainties are cor
- The pull is constructed by:

- Best calculated from pseudo-experiments yourself.
- RooUnfold does have some options (and the error on the histogram bin it returns isn't bad for a first look).

$$\text{pull} = \frac{\text{obs.} - \text{exp.}}{\sigma(\text{obs.})}$$

- If σ(obs.) is gaussian, then the pull (if calculated for many statistically independent pseudo-experiments) should take the form of a normal distribution, where the parameters can be interpreted in the following way:
  - ➡ μ = 0: unbiased results, μ != 0, probably some bias
  - ➡ σ = 1: errors correctly estimated, σ < (>) 1 errors are over (under) estimated.
- It is generally a good idea to have pulls with μ = 0 and σ = 1, but it's not the end of the world if they don't, you will just need to apply a correction based on your pull tests.

Pull Tests

- The pull of an observable tells
  - ➡ If your results are biased
  - ➡ If your uncertainties are cor
- The pull is constructed by:

- Best calculated from pseudo-experiments yourself.
- RooUnfold does have some options (and the error on the histogram bin it returns isn't bad for a first look).

$$\text{pull} = \frac{\text{obs.} - \text{exp.}}{\sigma(\text{obs.})}$$

- If σ(obs.) is gaussian, then the pull (if calculated for many statistically independent pseudo-experiments) should take the form of a normal distribution, where the

- d of

ll

X	$\frac{d\sigma_{t\bar{t}}}{dX}$ [$\frac{pb}{GeV}$]	$\frac{1}{\sigma_{t\bar{t}}}\frac{d\sigma_{t\bar{t}}}{dX}$ [$\frac{1}{GeV}$]	Stat. (abs.)	Stat. (norm.)	Syst. (abs.)	Syst. (norm.)
$p_T(t)$ [GeV]			[%]	[%]	[%]	[%]
0 – 70	7.1	0.371	± 1.8	± 1.7	+11 -11	+4 -3.2
70 – 150	9.9	0.515	± 1.3	± 1.2	+10 -11	+2.3 -2.7
150 – 250	4.61	0.239	± 1.8	± 1.7	+7 -8	+2.1 -2.0
250 – 400	0.97	0.051	± 3.4	± 3.3	+7 -9	+10 -11
400 – 1000	0.042	0.0022	± 10	± 9	+40 -40	+40 -40

Pull Tests

```
hist_result = unfold_bayes.Hreco(option).Clone(random_name)
```

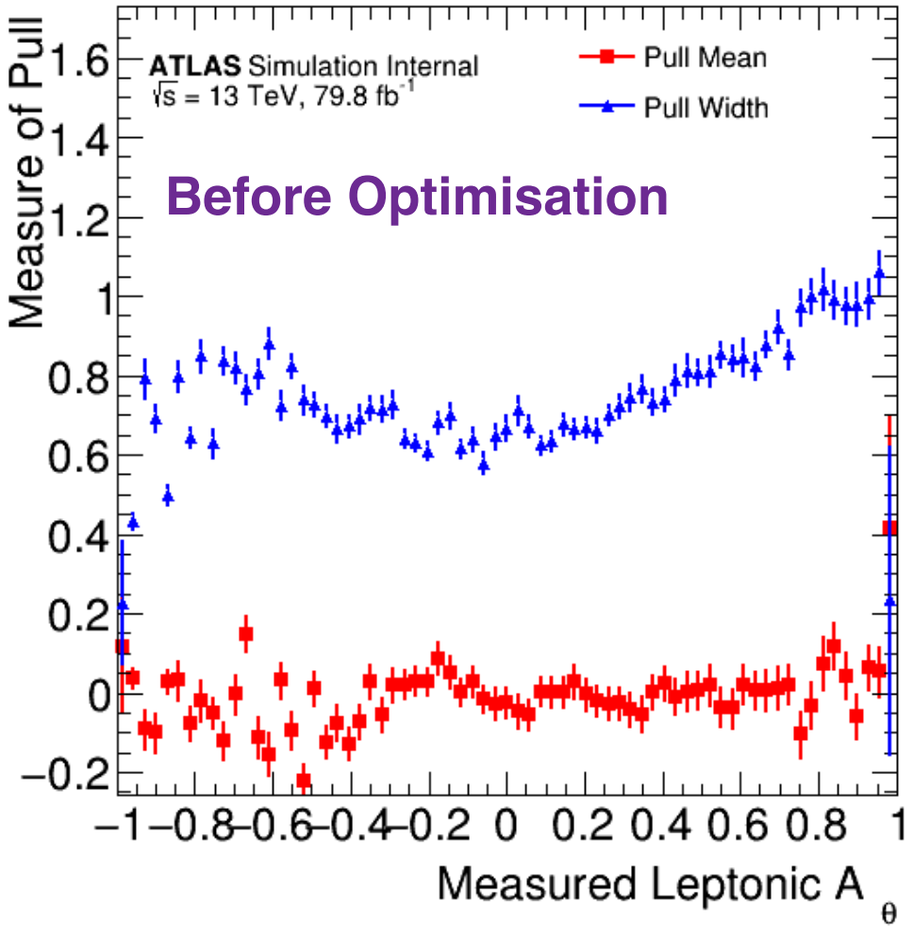
- When retrieving the histogram from RooUnfold, there are several error options:

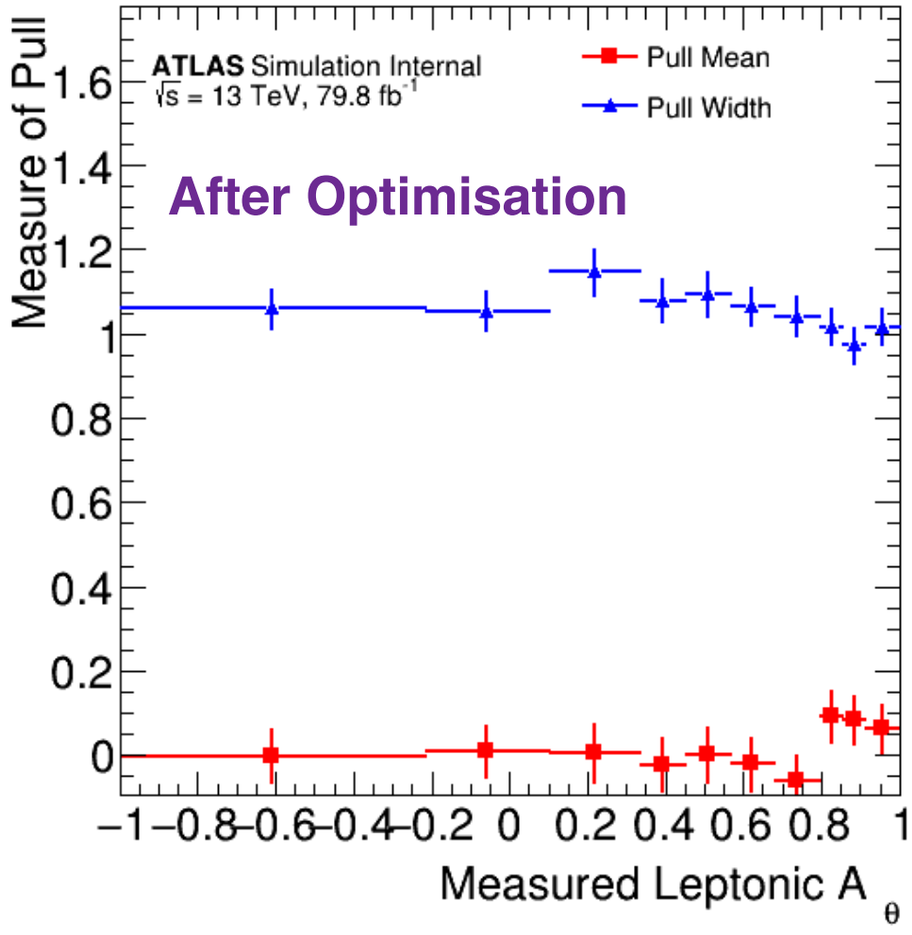
```
Creates reconstructed distribution. Error calculation varies by withError:
    0: No errors
    1: Errors from the square root of the diagonals of the covariance matrix given by the unfolding
    2: Errors from the square root of of the covariance matrix given by the unfolding
    3: Errors from the square root of the covariance matrix from the variation of the results in toy MC tests
```

- Personally, I've never fully trusted these options and usually calculate covariance matrices myself (out of the scope of this tutorial but please ask about it, we have many examples to help you).
- But they should be fine for first tests (at least, option 1 is usually exactly what it says).

Binning Optimisation

- Pull tests can also be used to help you optimise your binning.
- Here is an example from the Deadcone analysis:

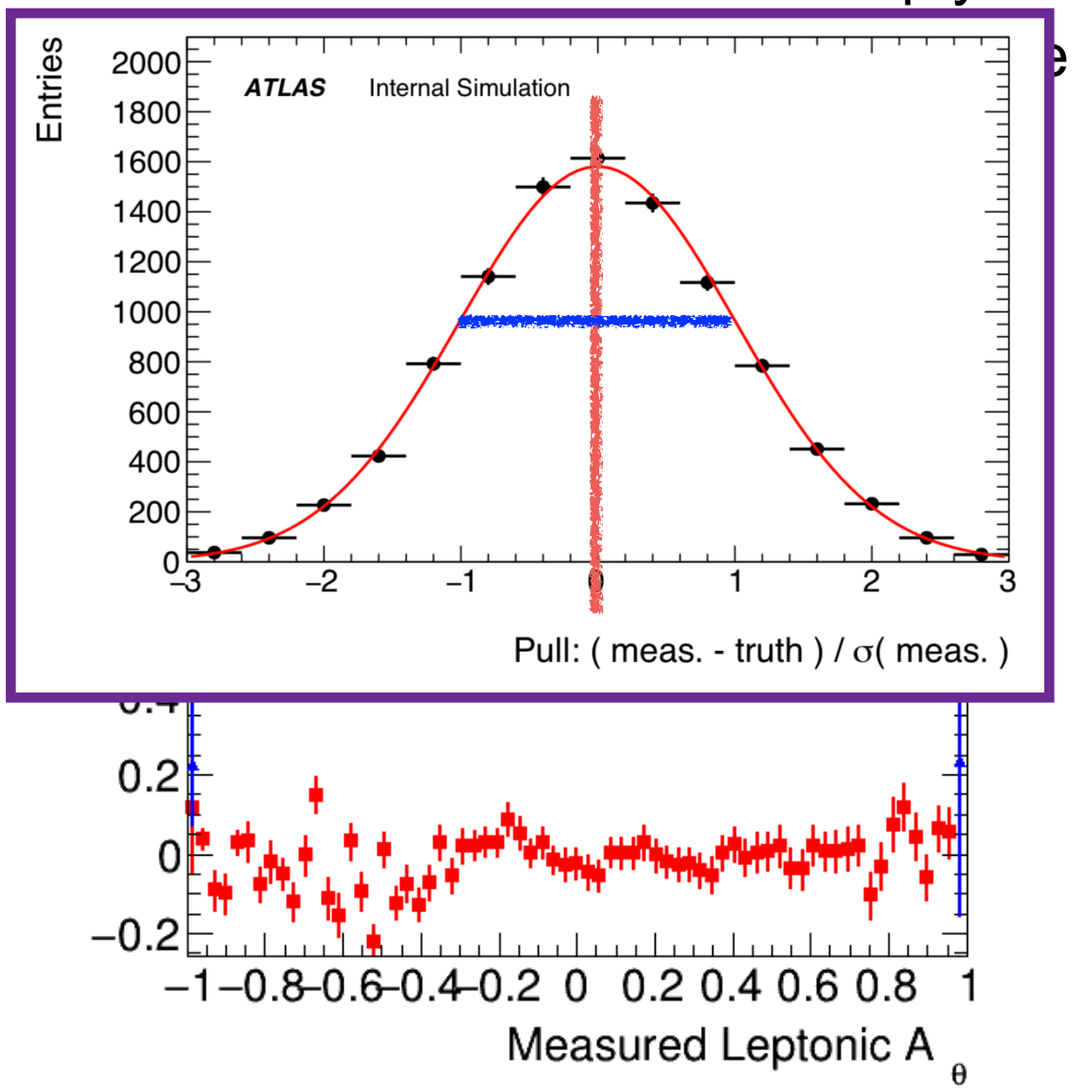


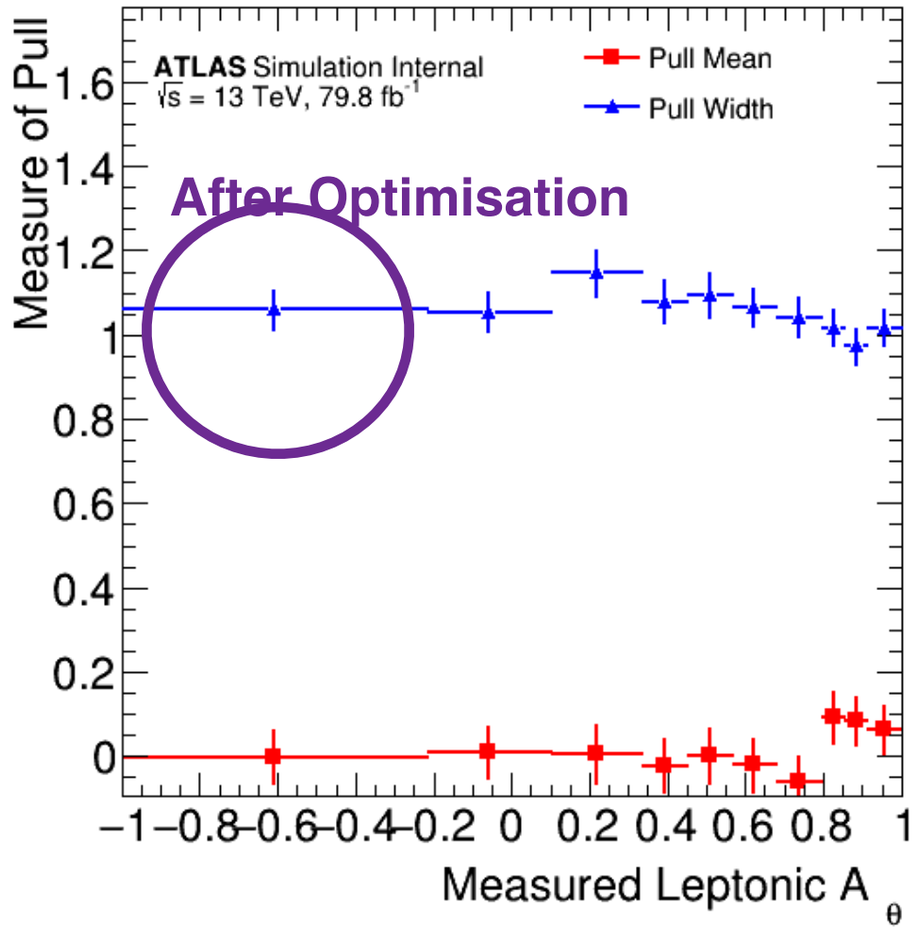


- With a large number of bins, there are obvious biases and underestimated uncertainties.
- After optimisation for the number and width of bins, things look far more reasonable.

Binning Optimisation

- Pull tests can also be used to help you optimise your binning.
- e analysis:

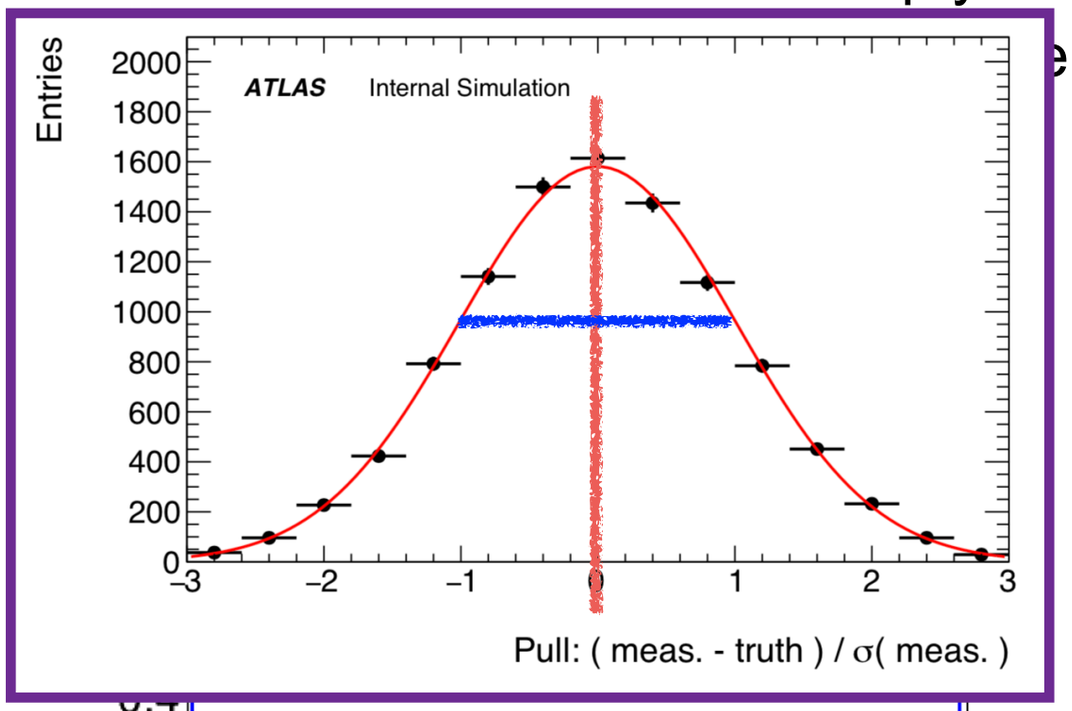


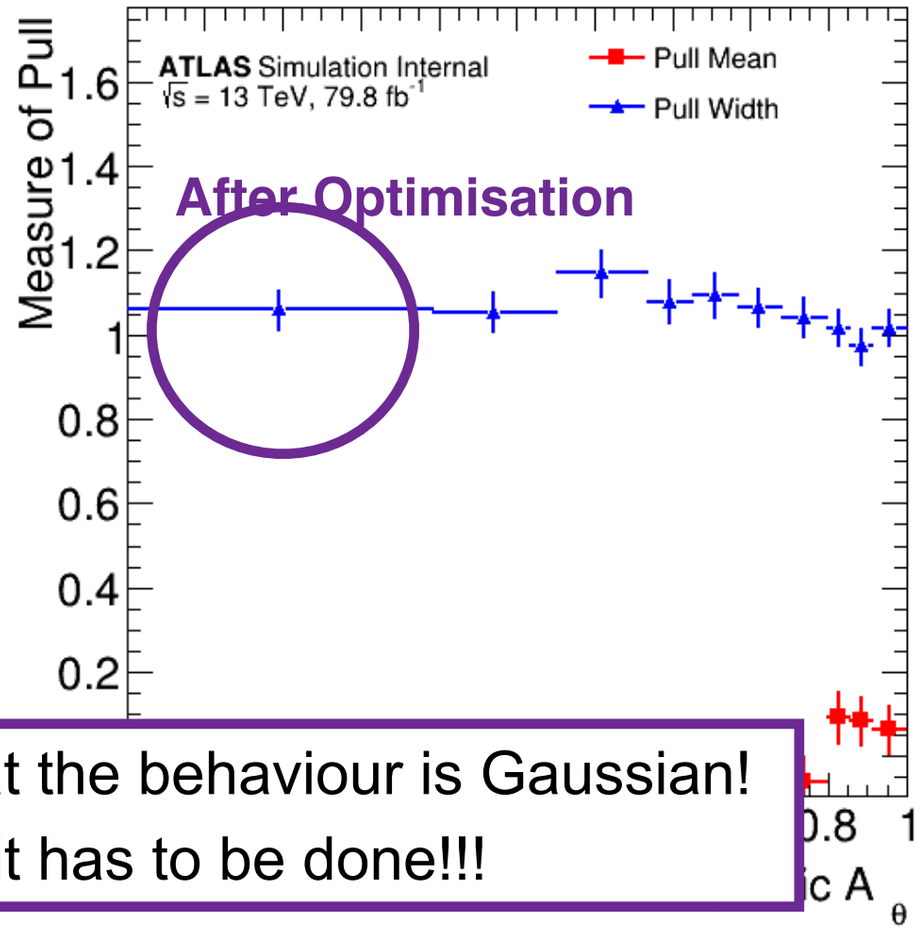


- With a large number of bins, there are obvious biases and underestimated uncertainties.
- After optimisation for the number and width of bins, things look far more reasonable.

Binning Optimisation

- Pull tests can also be used to help you optimise your binning.
- e analysis:



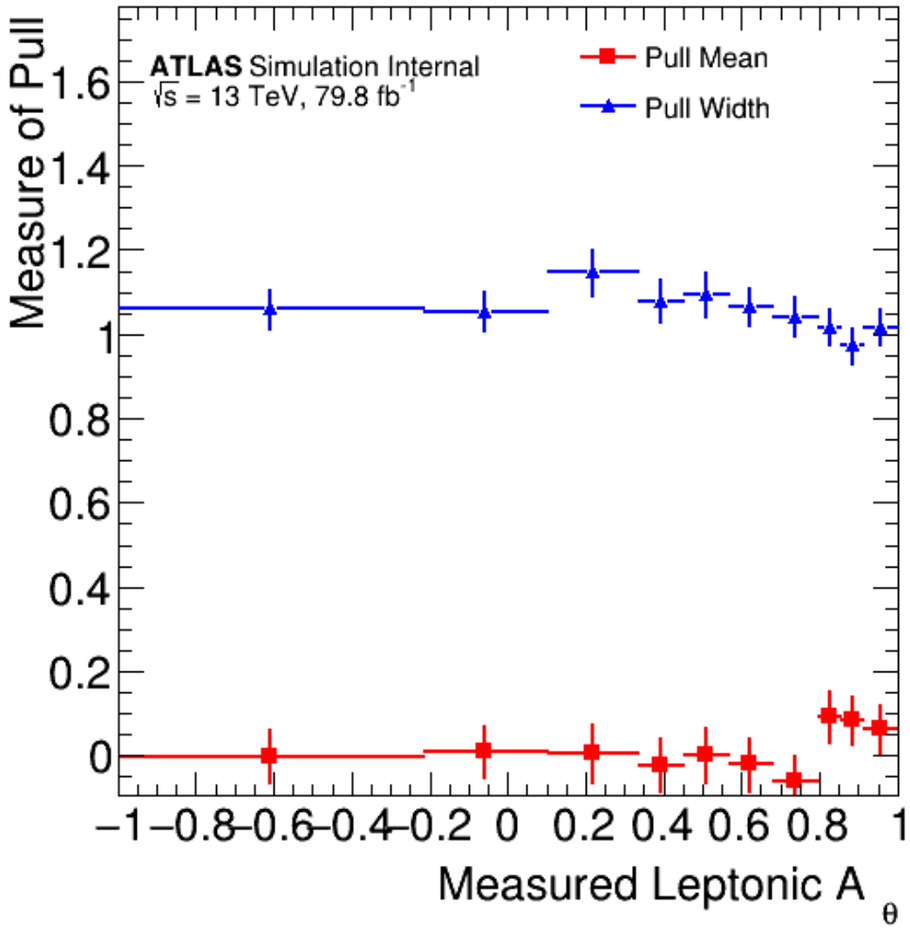


You must check, for each bin, that the behaviour is Gaussian!
This can be a pain but it has to be done!!!

- With a large number of bins, there are obvious biases and underestimated uncertainties.
- After optimisation for the number and width of bins, things look far more reasonable.

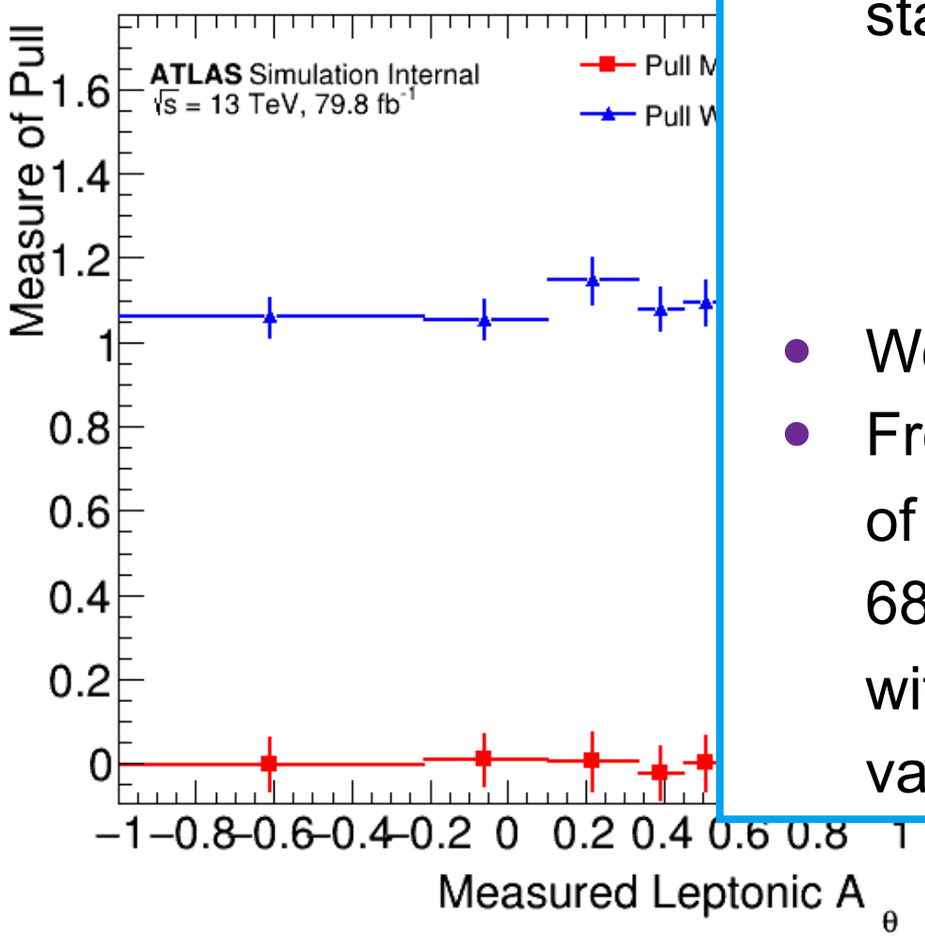
Brief aside on stats…

- We ran 300 statistically independent pseudo-experiments for this binning optimisation test.
- Do we keep optimising until all the blue markers are a at 1 and red markers are at zero?



Brief aside on stats…

- We ran 300 statistically independent pseudo-experiments for this binning optimisation test.
- Do we keep optimising until all the blue markers are a at 1 and red markers are at zero?



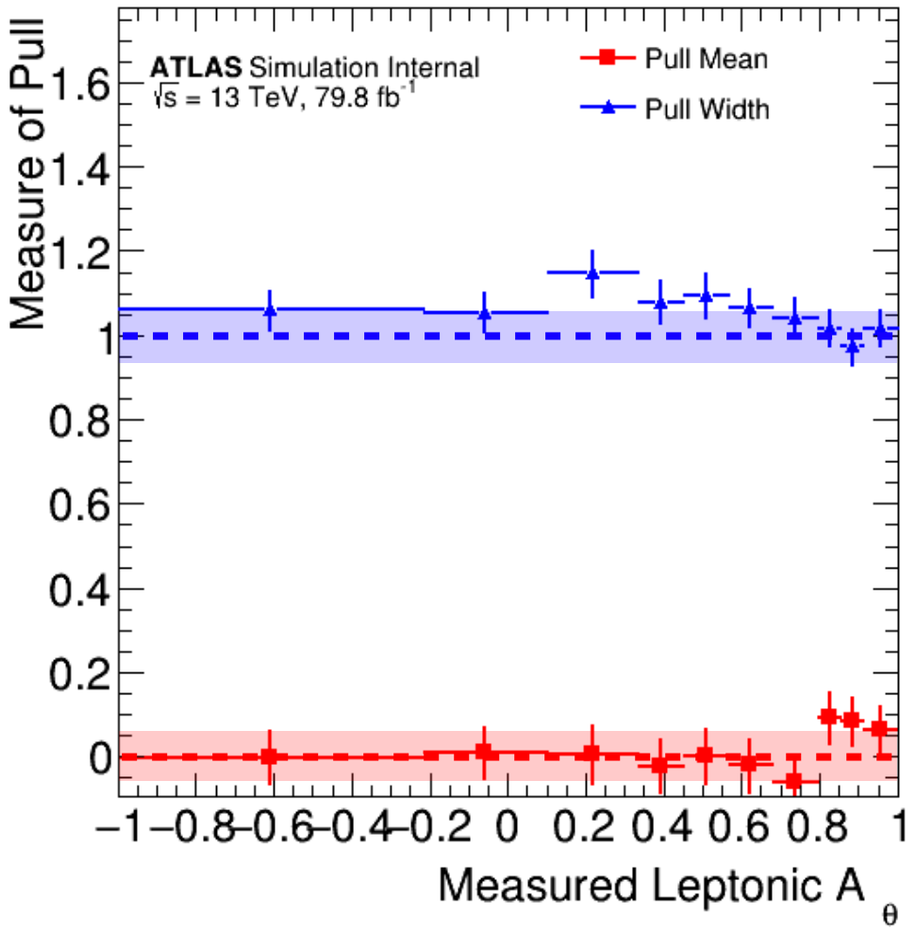
- We have 300 pseudo experiments, so we have a statistical uncertainty of:

$$\frac{1}{\sqrt{300}} * 100 = 5.8\,\%$$

- We also have 10 bins.
- From a pure frequentist point of view, we expect that only 68% (i.e. 7) of the bins lie within 5.8% of their expected values.

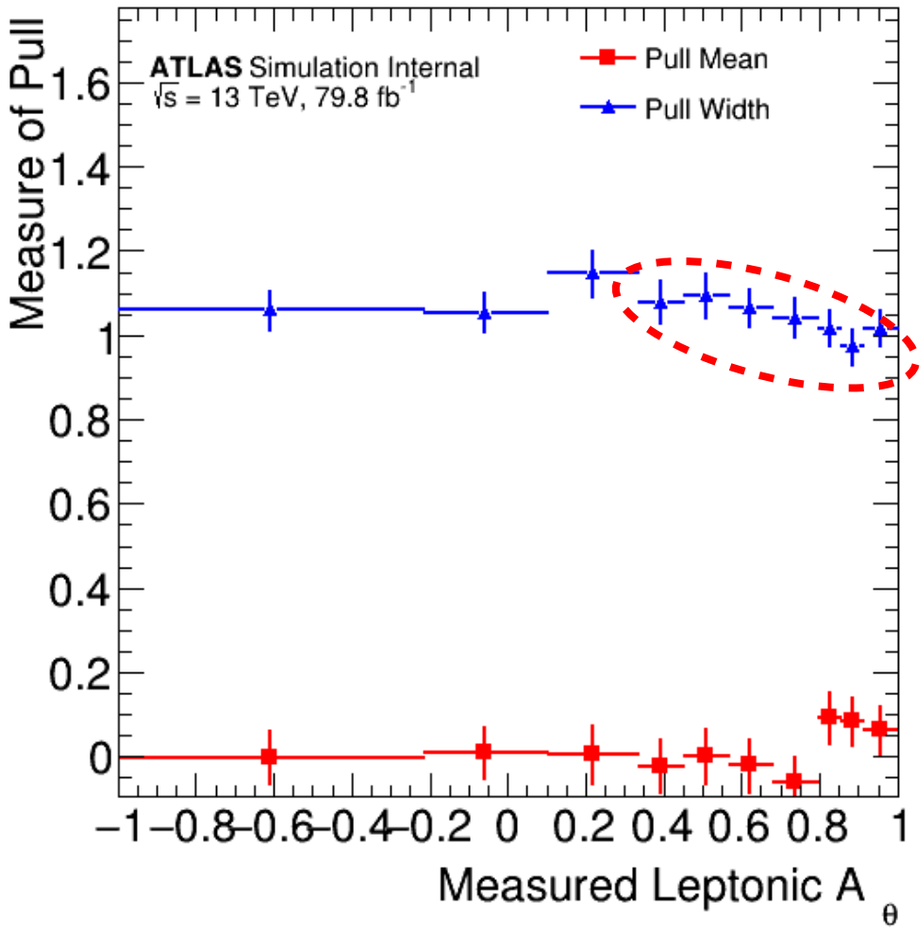
Brief aside on stats…

- So, from a purely statistical counting point of view, the below binning looks like it's (just about) reasonable. There are only 3/4 blue markers outside of the range, and only three red markers, which is what we expect (this is just a basic χ^2 test).



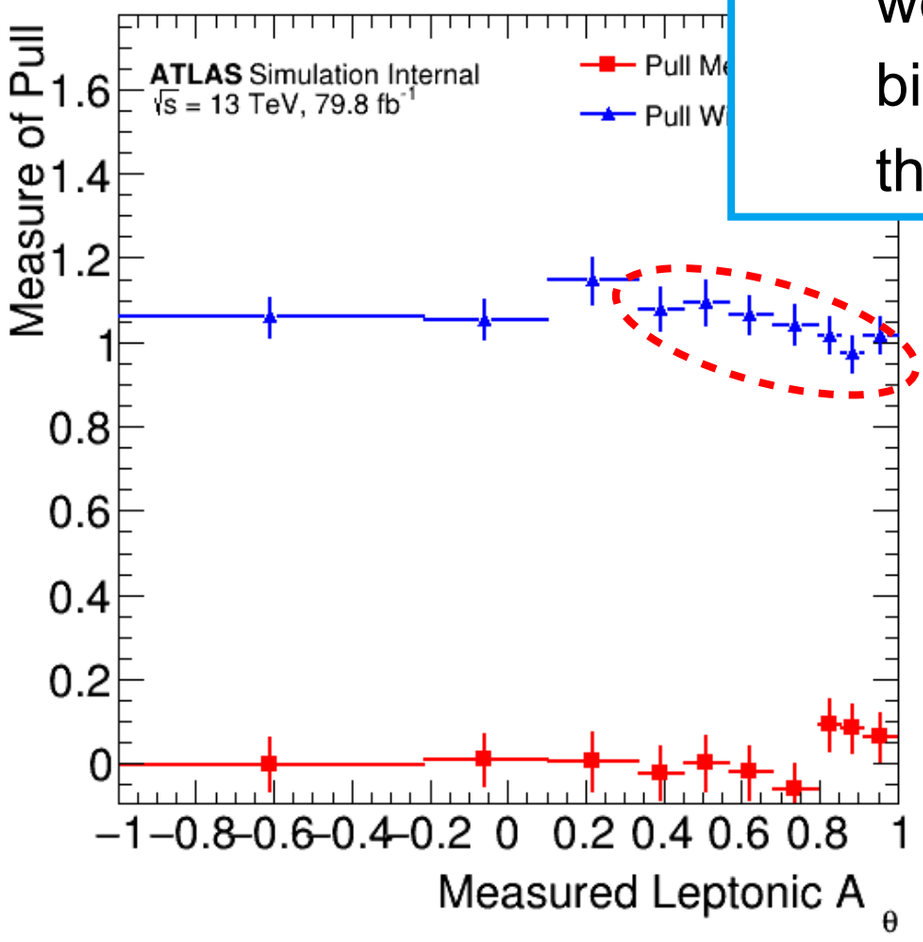
Brief aside on stats…

- So, from a purely statistical counting point of view, the below binning looks like it's (just about) reasonable. There are only 3/4 blue markers outside of the range, and only three red markers, which is what we expect (this is just a basic χ^2 test).
- **But what about this?**



Brief aside on stats…

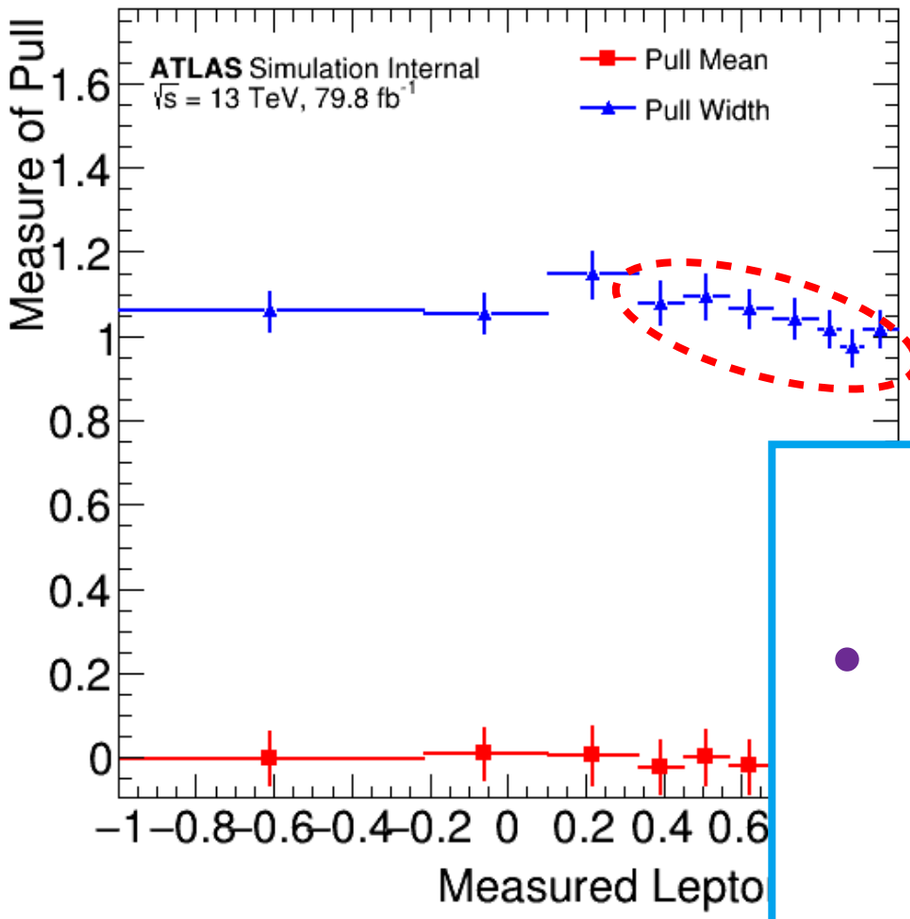
- So, from a purely statistical counting point of view, the below binning looks like it's (just about) reasonable. There are only 3/4 blue markers outside of the range, and only three red markers, which is what we expect (this is just a basic χ^2 test)
- **But what about this?**



- What is the probability that we'd have 5 - 7 sequential bins where each bin is lower than the one preceding it?

Brief aside on stats…

- So, from a purely statistical counting point of view, the below binning looks like it's (just about) reasonable. There are only 3/4 blue markers outside of the range, and only three red markers, which is what we expect (this is just a basic χ^2 test)
- **But what about this?**

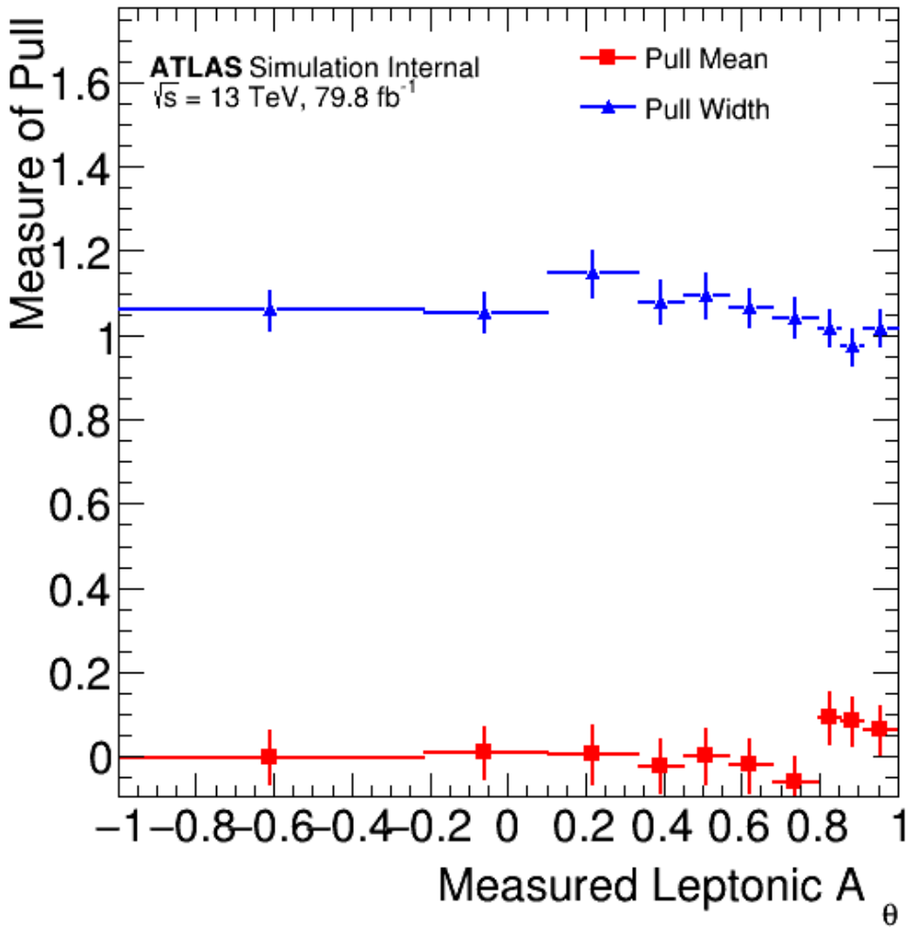


$$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = 0.03$$

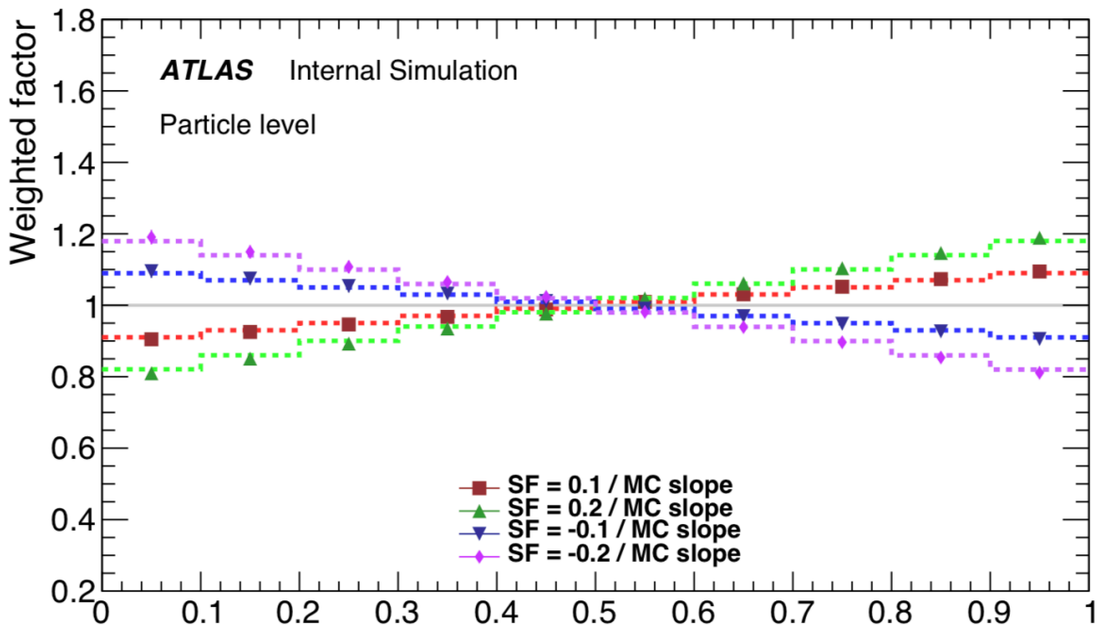
- Since 0.03 < 0.05 (the p-value for 2σ), we'd consider this a significant trend that shouldn't be attributed to random chance.

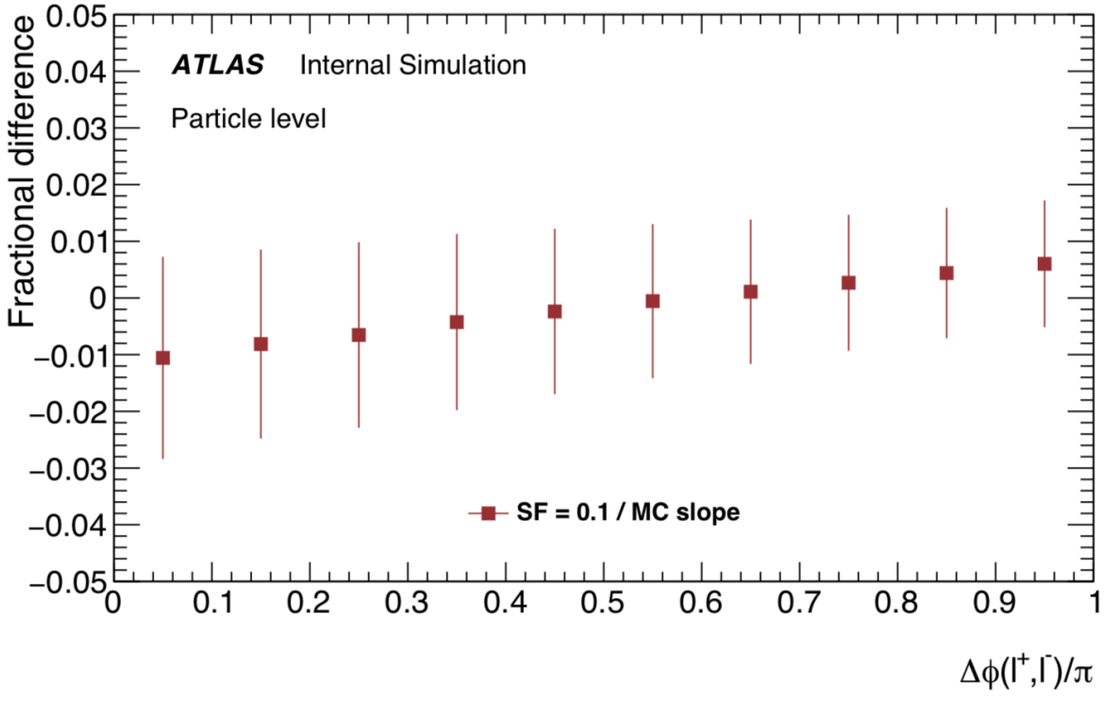
Brief aside on stats…

- Summary on this is, think about each test you do and if it makes statistical sense!
  - ➡ Don't blindly try to make all your bins have exactly a pull μ = 0, σ = 1.
  - ➡ Don't assume just because things are inside an uncertainty band that they're fine, this hides important shape effects!



Stress Tests

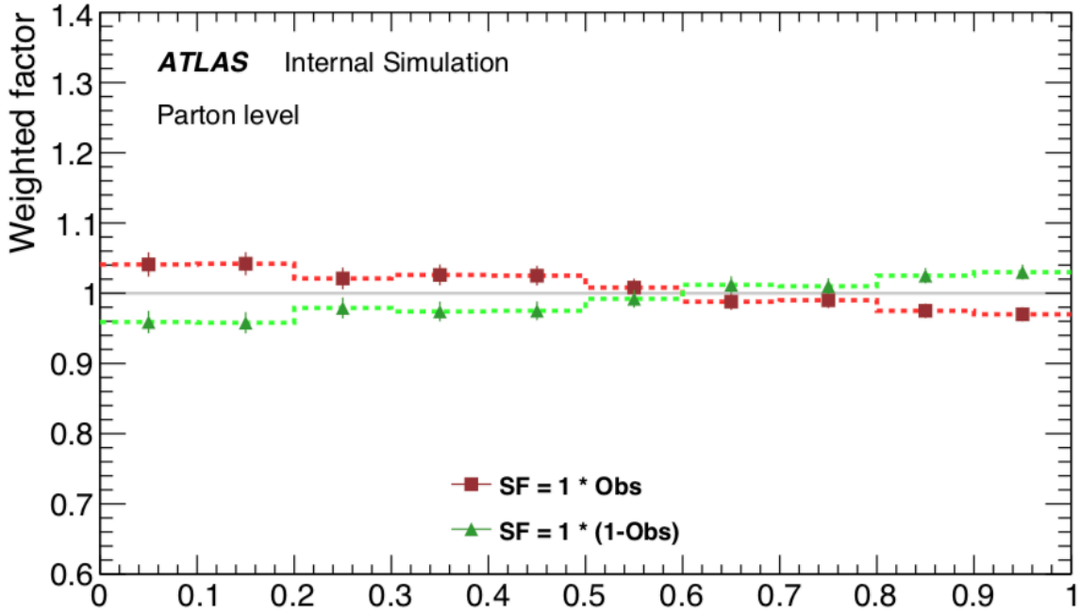


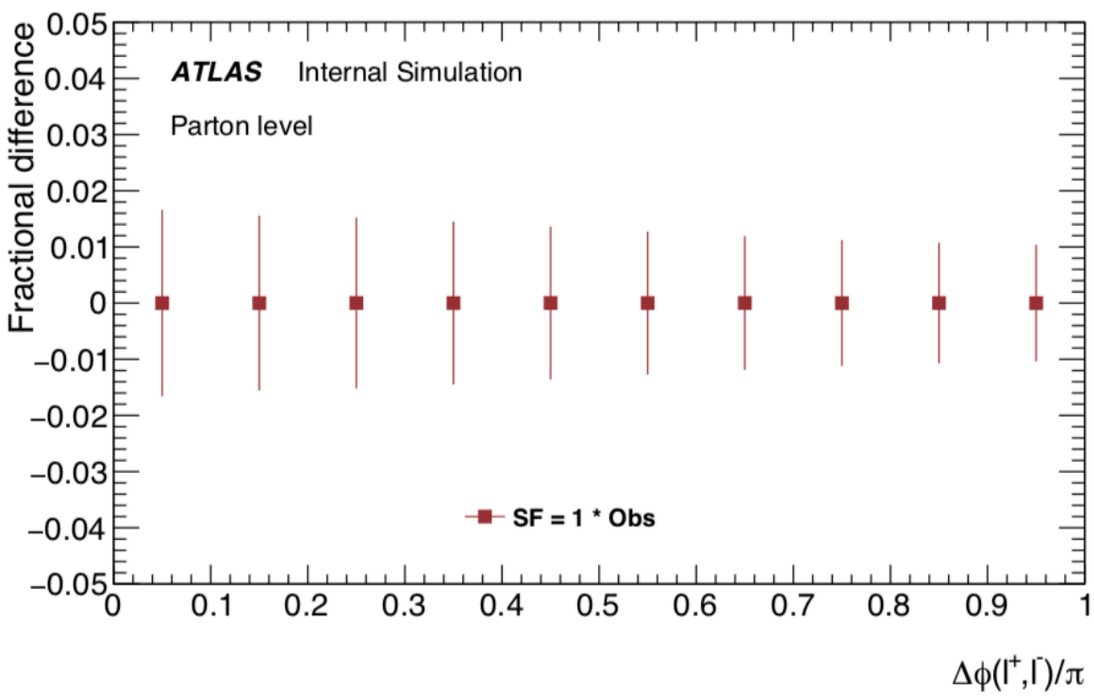


$$S_i = 1 + C \cdot \frac{(\mathrm{Data}_i - \mathrm{MC}_i)}{\mathrm{Data}_i}$$

- Reweight your truth spectra with some function (usually a linear slope).
- Unfold your reweighed reco level distribution and see if you recover that slope.
- The point of this test isn't really to get perfect closure, it's to stress the unfolding and see where it breaks down.
- Other functions may be more appropriate for your analysis (curves, bumps etc.)

Stress Tests

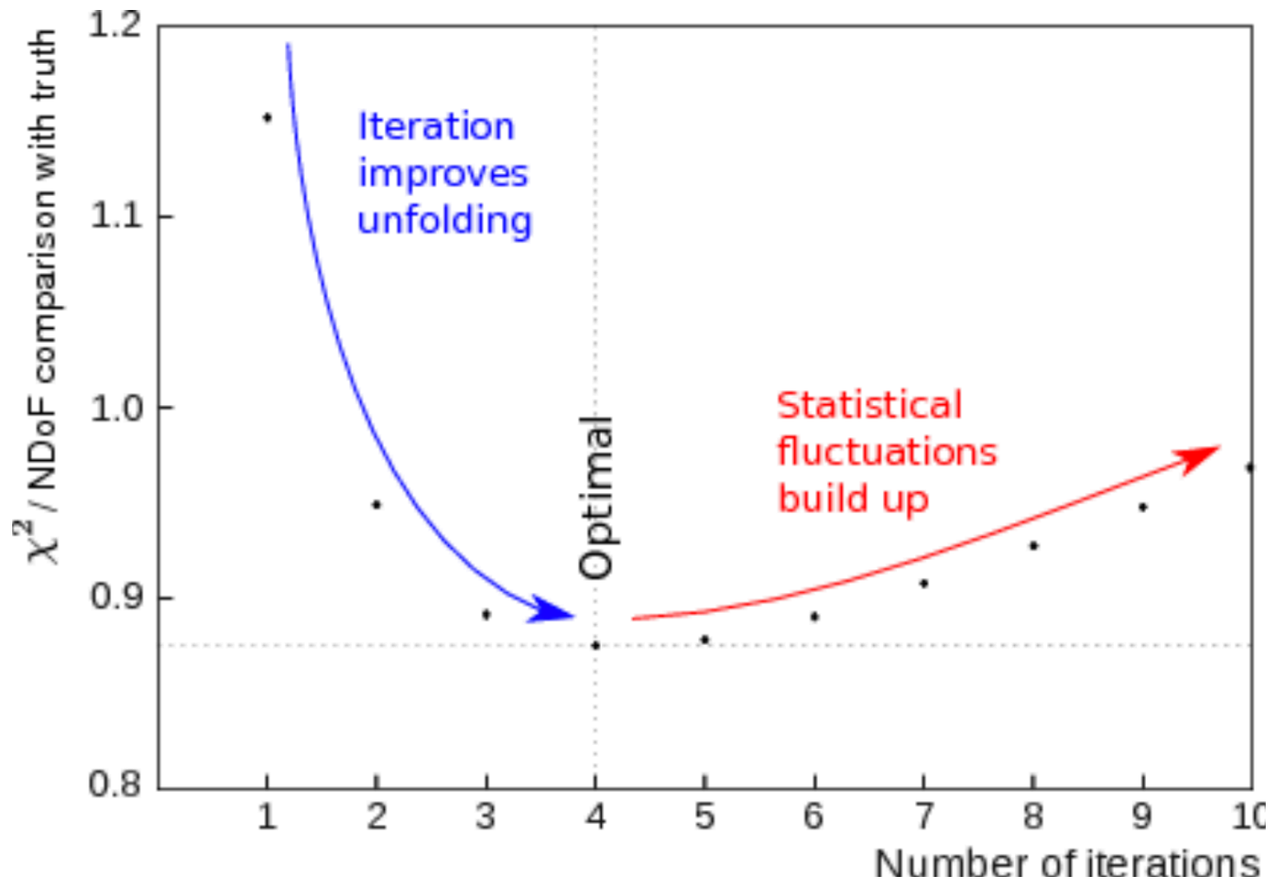




$$S_i = 1 + C \cdot \frac{(\mathrm{Data}_i - \mathrm{MC}_i)}{\mathrm{Data}_i}$$

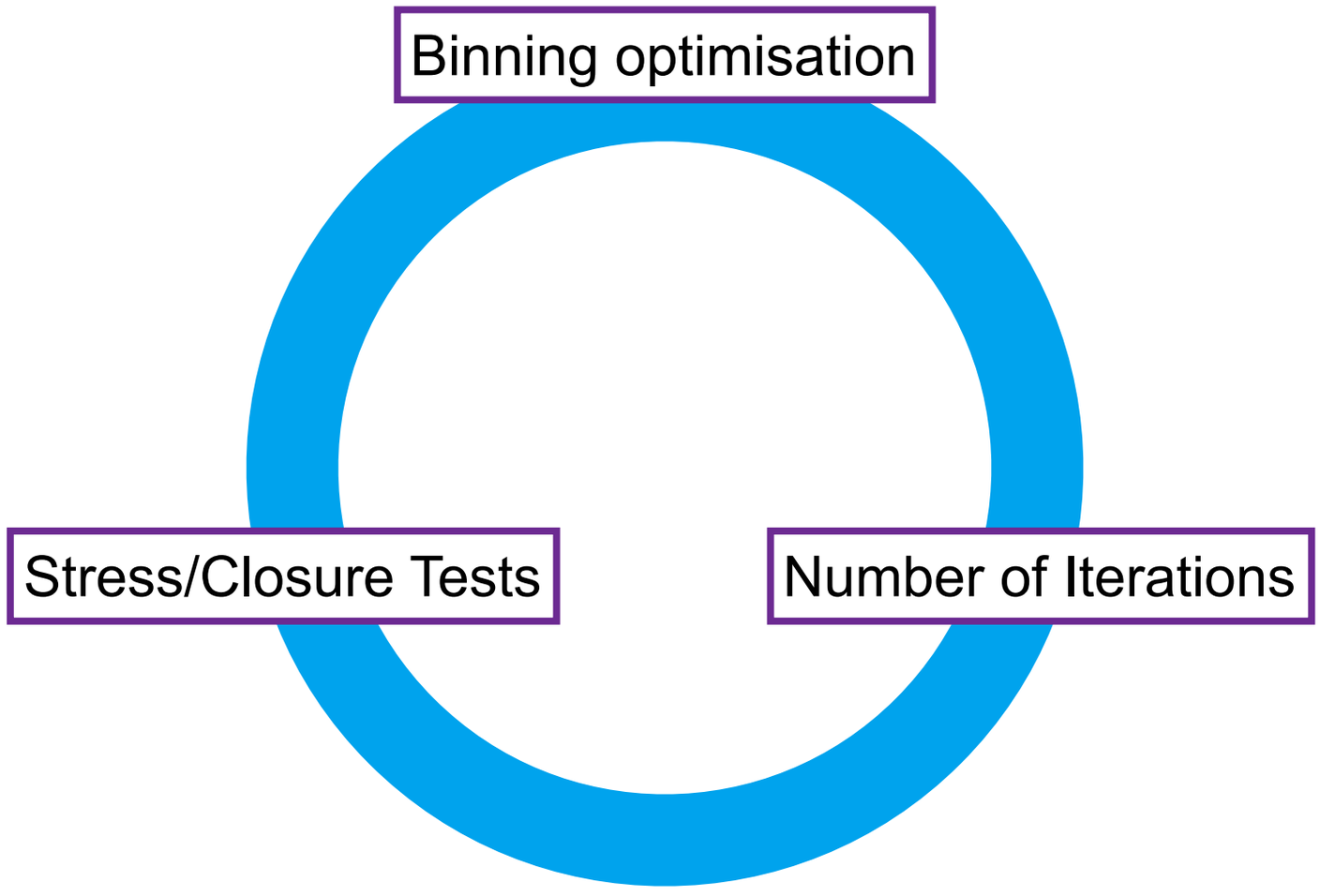
- You can also reweight your truth spectra with the observed Data/MC at reco level.
- This is as close as you can get to saying “can I unfold my data in an unbiased way?”
- If this test doesn’t close, not the end of the world, but you will have to either change the binning/number of iterations or accept a bias correction on your measurement.

Optimising Iterations (IBU)



- Many ways one can determine this, but a Chi2 test is as good as any.
- As a general rule, you want as few iterations as possible in order to minimise the statistical uncertainties.
- You will almost never get a plot as helpful as this one, the changes tend to be quite subtle, but it's usually possible to identify a minima (often between 2 and 4 iterations).

Optimising Iterations (IBU)



- Binning optimisation, number of iterations, stress-tests, closure tests, these are all somewhat inter-dependent, and it's not possible to say "do this one first, then this one" so you will need to use a bit of physics intuition.
- Are small bins the most important to you? If so then optimise that first. If stat uncertainty matters most, optimise the number of iterations first etc.

Unfolding Experts in Top

- If you have any unfolding questions, the following people have frameworks of RooUnfold or FBU implemented and currently running:

➡ **Jay Howarth**	**(IBU spin):**	Manchester
➡ **Federica Fabbri**	**(IBU l+jets xsec):**	Glasgow
➡ **Tom Neep**	**(IBU $t\bar{t}b\bar{b}$):**	Birmingham
➡ **Clement Helsens**	**(FBU CA):**	Somewhere
➡ **Chris Pollard**	**(FBU b-frag):**	Somewhere
➡ **Baptiste Ravina**	**(IBU $t\bar{t}Z$):**	Sheffield
➡ **Maria Costa**	**(IBU t-chan):**	Valencia

- Non-exhaustive list, just people I know that are currently working on unfolding frameworks (biased by who I know).
- Any of them will be happy to help you with your unfolding questions, or you can always contact your sub-conveners.
- We now also have a mailing list for unfolding questions:

atlas-phys-top-unfolding