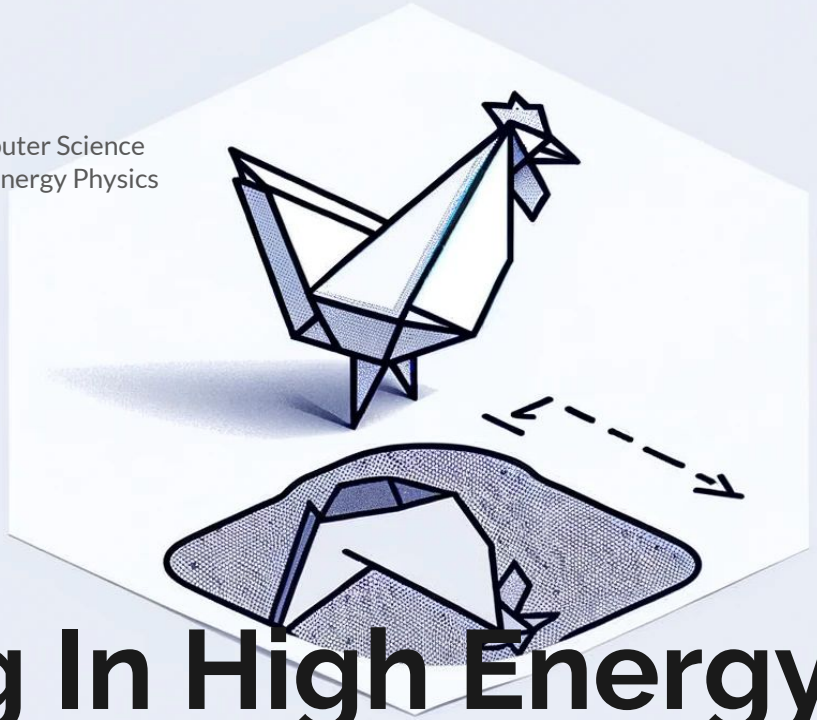
Vincent Croft

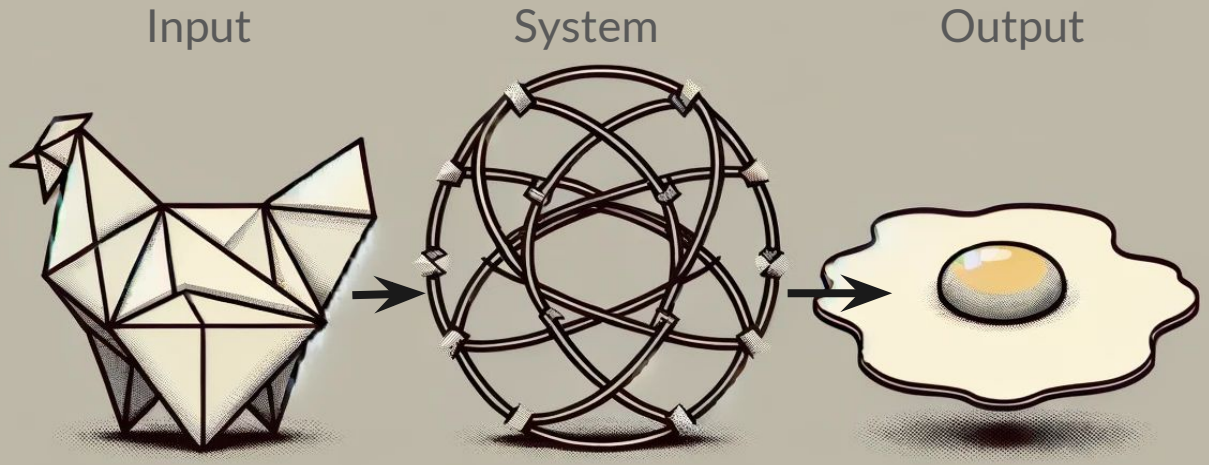
[1] LIACS, Leiden Institute of Advanced Computer Science
[2] Nikhef, Dutch National Institute for High Energy Physics
[3] RooUnfold Development Team



Unfolding In High Energy Physics

An introduction, application to analyses, treatment of uncertainties, and the choice of algorithm and regularisation

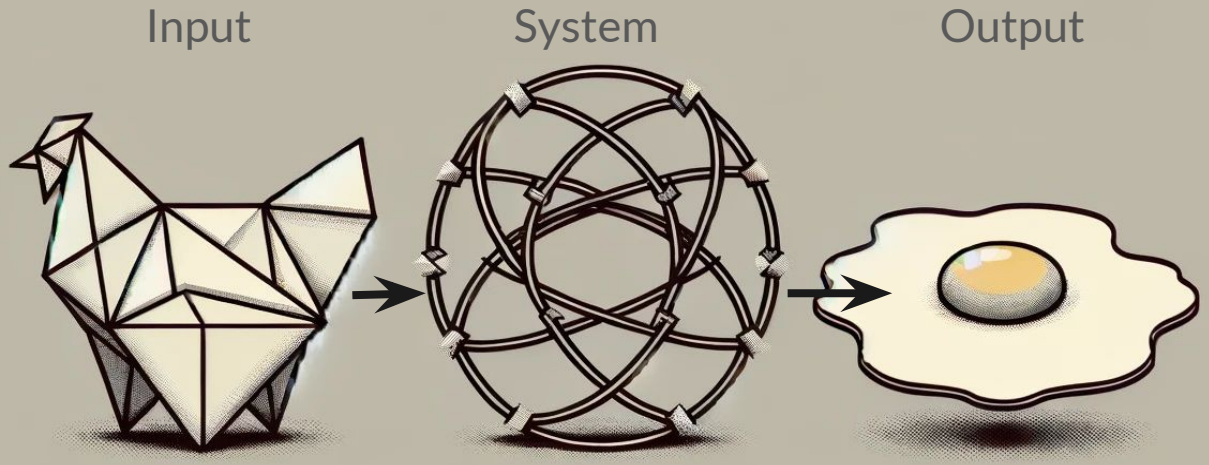
The problem with unfolding



A well defined problem

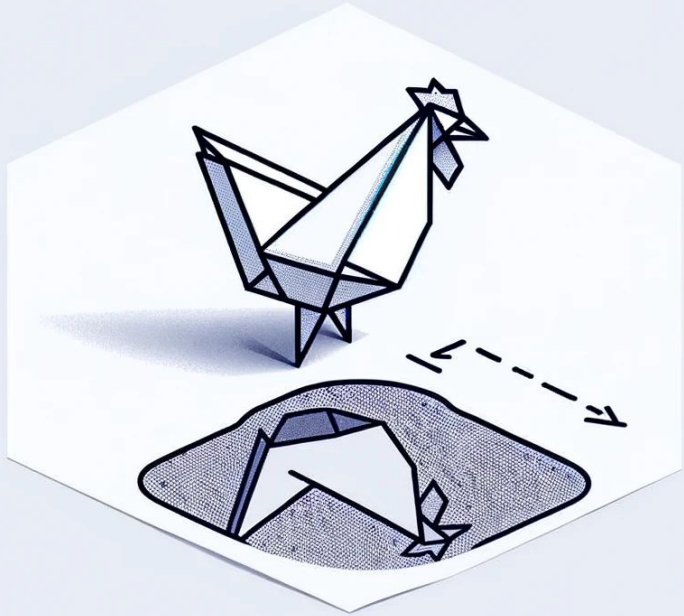
- **Existence**
  - The problem must have a solution
- **Uniqueness**
  - There must be only one solution to the problem
- **Stability**
  - The solution must depend continuously on the data

The problem with unfolding



Simplest ill-posed problem

$$x_1 + x_2 = 1$$



The problem with unfolding

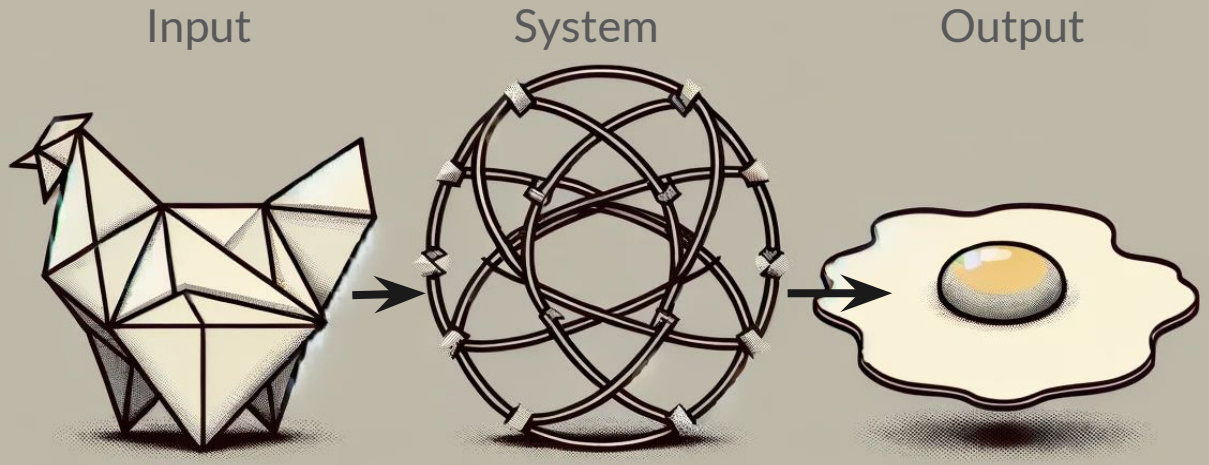


Table of Contents

The Unfolding Problem

- **Medical imaging -** CT scanning, electro-cardiography, etc
- **Geophysical prospecting -** search for oil, land-mines, etc
- **Image deblurring -** astronomy, crime scene investigations, etc
- **Deconvolution of a measurement instruments response.**

Users of RooUnfold

180+ citations

ALICE

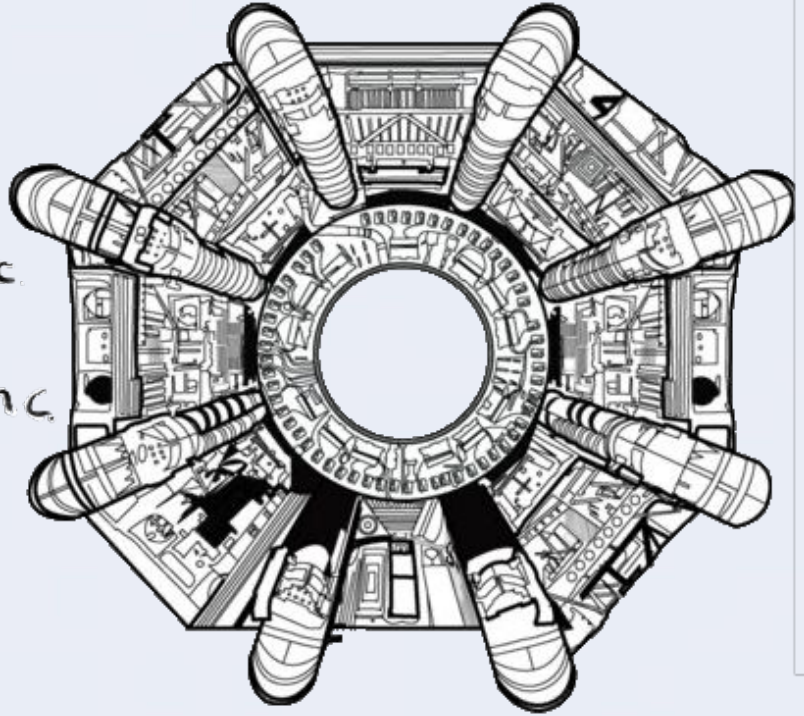
The Unfolding Problem

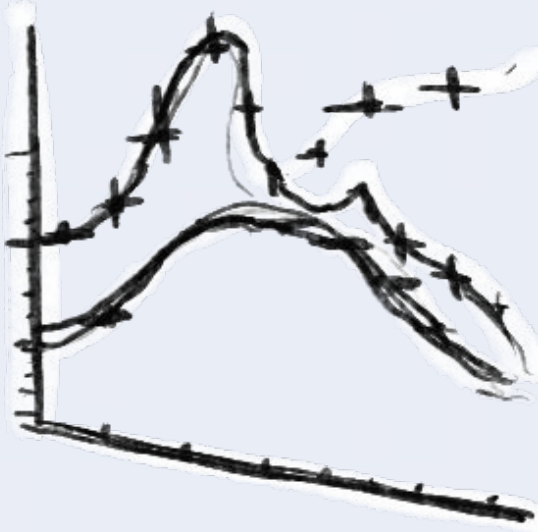
- Estimating the particle-level spectrum
  - Some physical quantity of interest
  - Some basis of observations
  - Smeared by an imperfect measurement device

- Estimating the true distribution when you measure a smeared spectrum

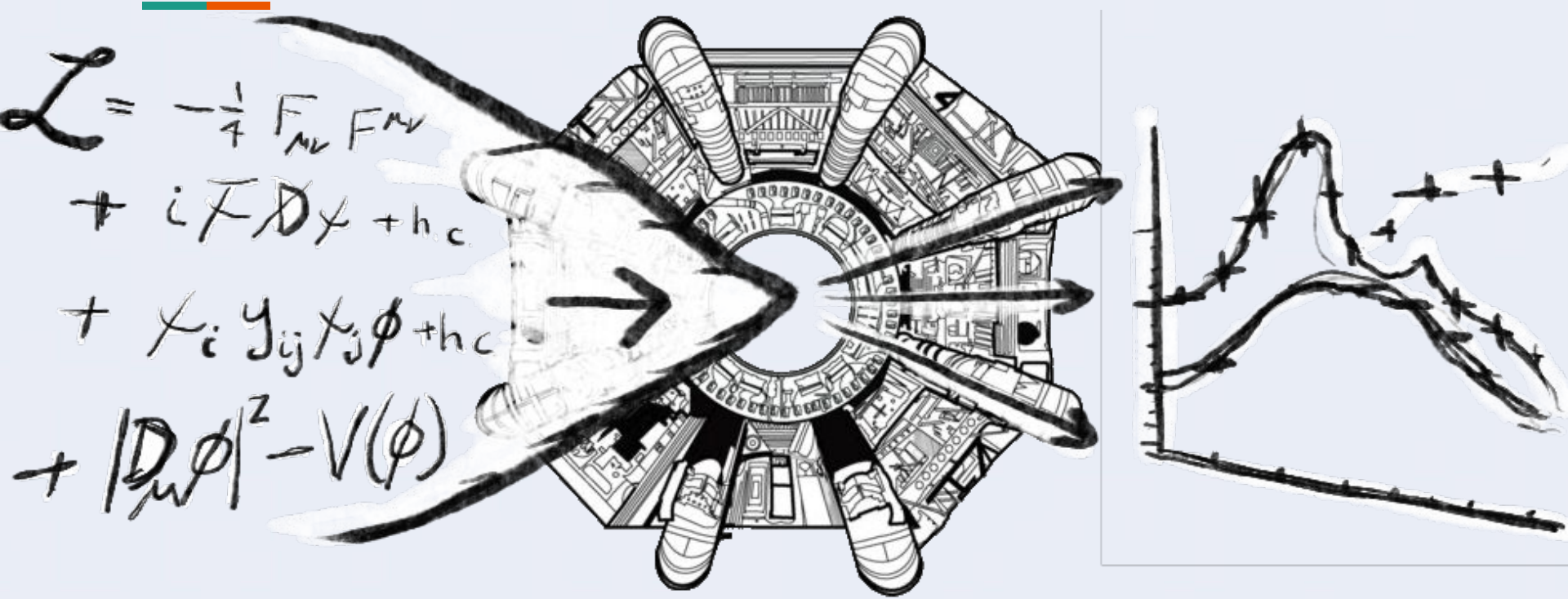
Standard Measurement Roadmap

$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i \bar{\chi} \not{D} \chi + h.c. + \chi_i y_{ij} \chi_j \phi + h.c. + |D_\mu \phi|^2 - V(\phi)$$

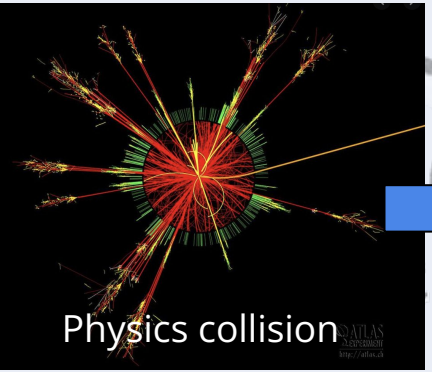




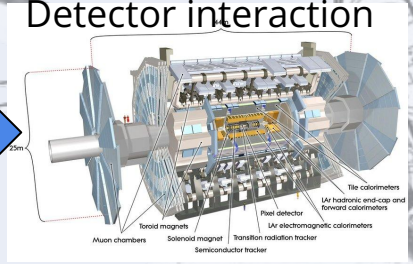
Standard Measurement Roadmap



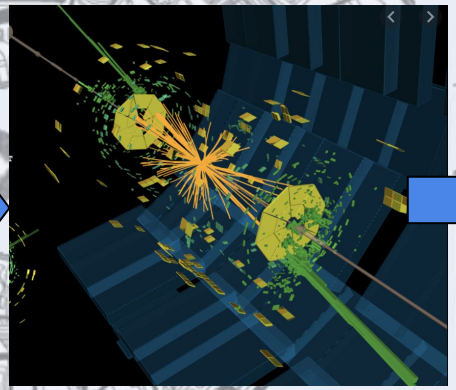
Standard Measurement Roadmap



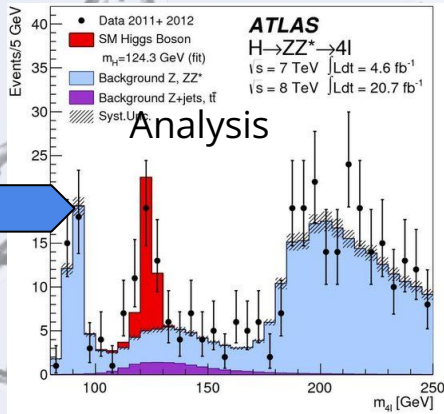
Detector interaction



Detector reconstruction

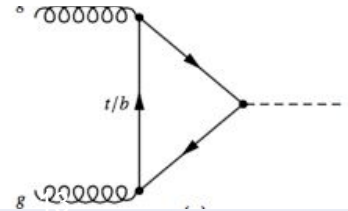


Event selection

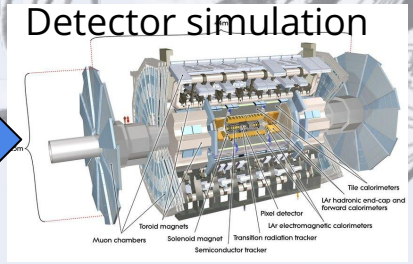


Analysis

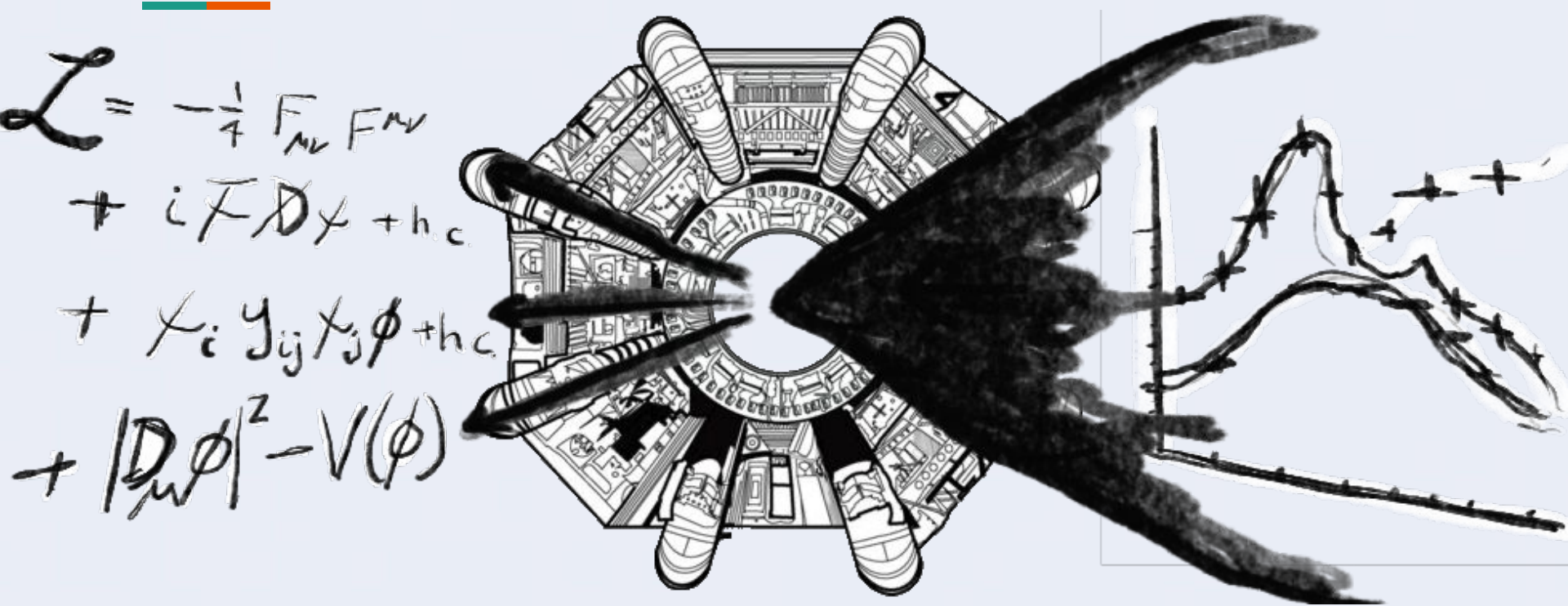
Physics model



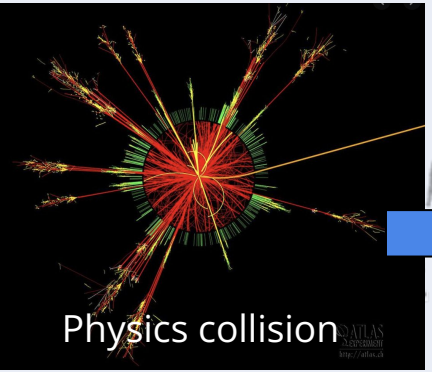
Detector simulation



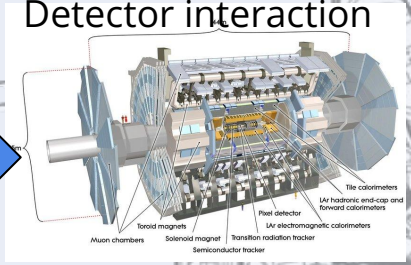
Alternative Measurement Roadmap



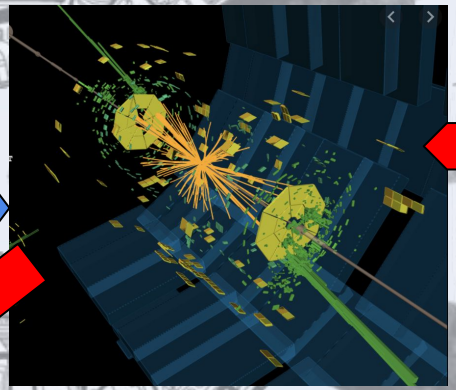
Alternative Measurement Roadmap



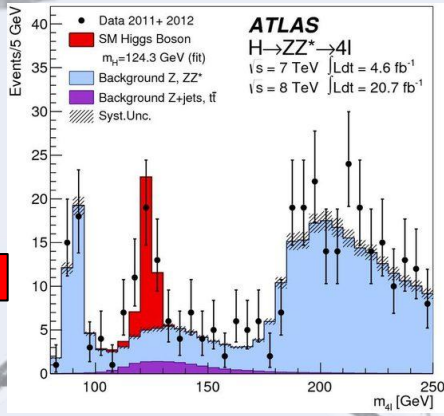
Detector interaction



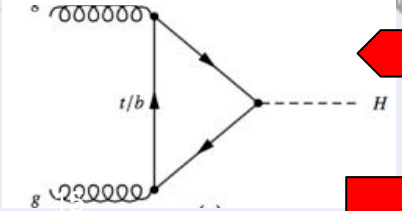
Detector reconstruction



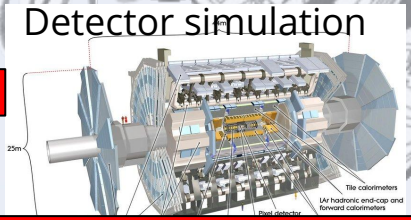
Event selection

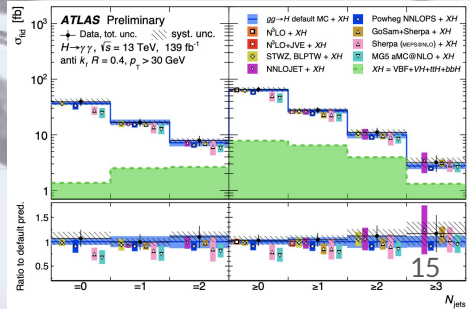


Physics model



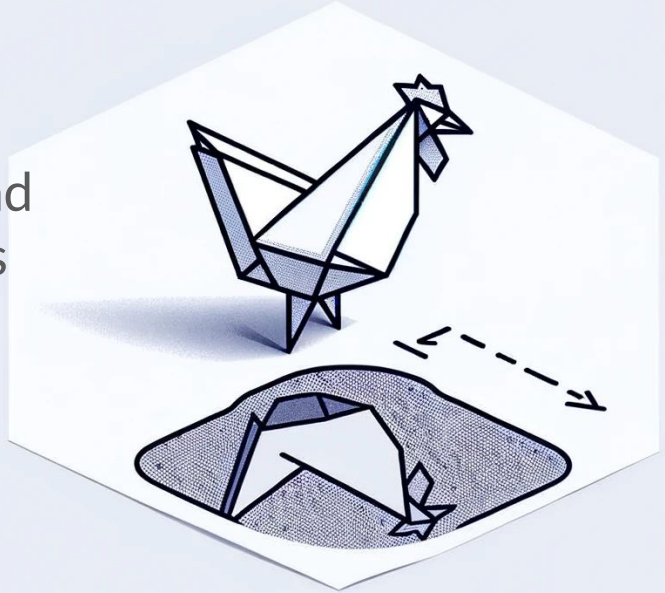
Detector simulation



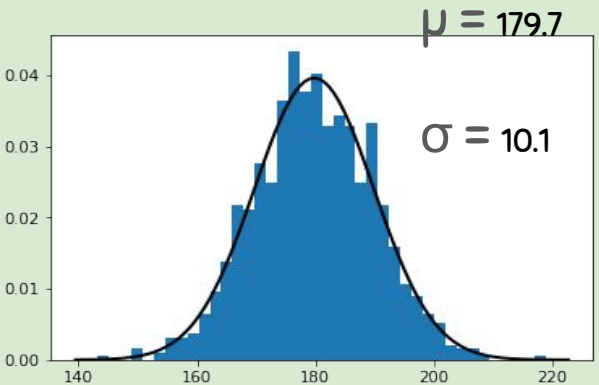


Why use the alternative?

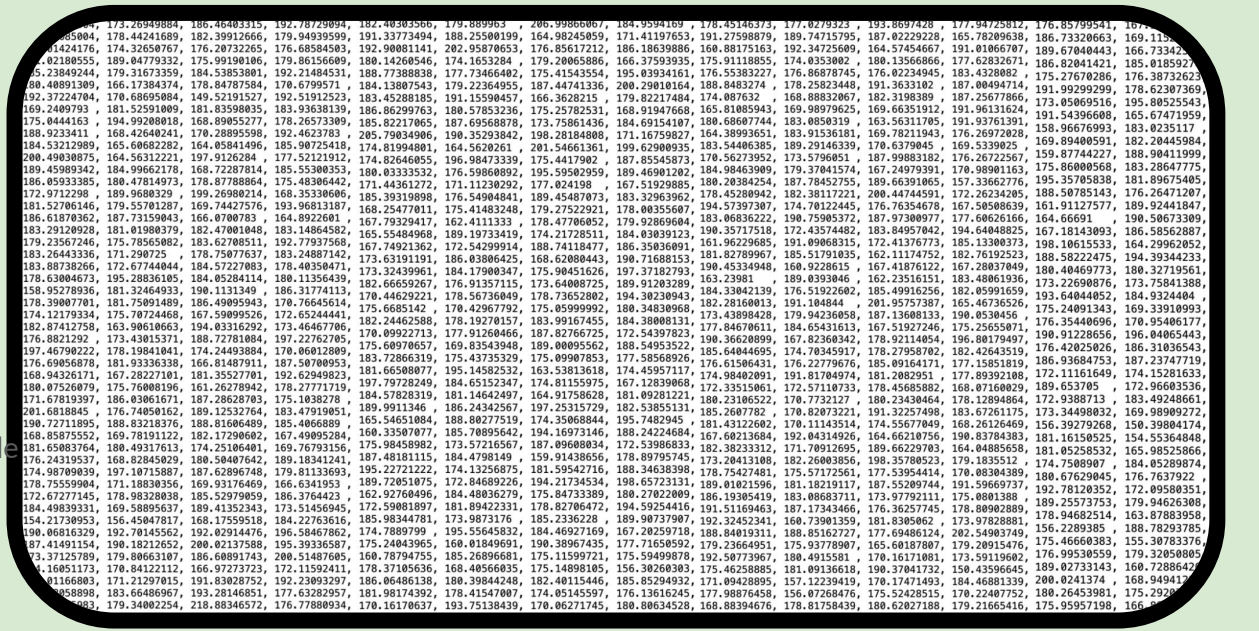
- Multiple predictions differ within the uncertainties
- Ready for future reinterpretations and combinations with other experiments
- Learn about the physics model
  - Not just the likelihood
  - Physics parameters!



Non-parametric Functions - Data Is Hard

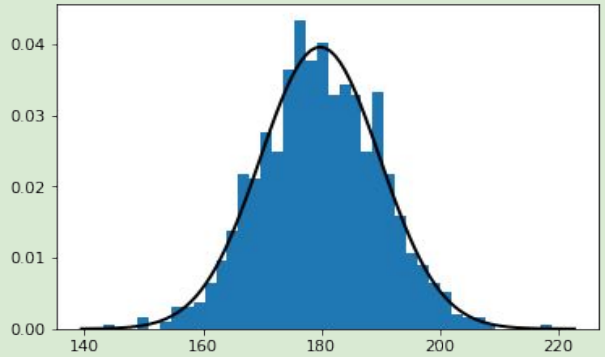


We anticipate that the real (true) nature of these functions is some sort of smooth curve, it is desirable that our estimates be smooth as well – this is why another name for curve estimation is smoothing



Let's revisit histograms! the simplest method of density estimation, and introduce many of the main ideas that will come up consistently in this topic.

Histograms!



Let $[a,b]$ denote an interval which contains the data $\{x_i\}$, and let m be an integer which divides $[a, b]$ into m equal-width bins, $\{B_j\}_{j=1}^{m}$

Let: $h = \frac{b-a}{m}$ denote the binwidth and let y_j denote the number of observations in B_j

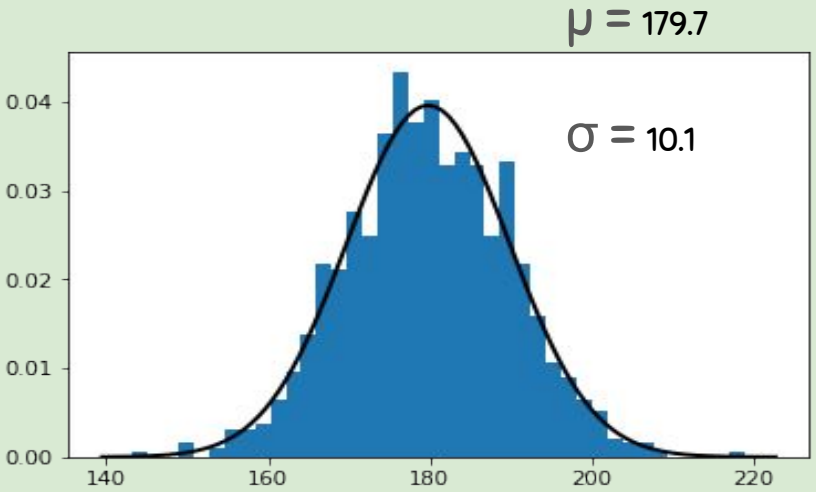
Such that: $\hat{p}_j = \frac{y_j}{n}$ and $p_j = \int_{B_j} f(u)du$ and $\hat{f}(x) = \frac{\hat{p_j}}{h}$

for all $x \in B_j$, where $f(\cdot)$ is the true density of X

Meaning: For a fixed x and m,

$$\mathbb{E}\hat{f}(x) = \frac{p_j}{h} \qquad \mathbb{V}\hat{f}(x) = \frac{p_j(1-p_j)}{nh^2}$$

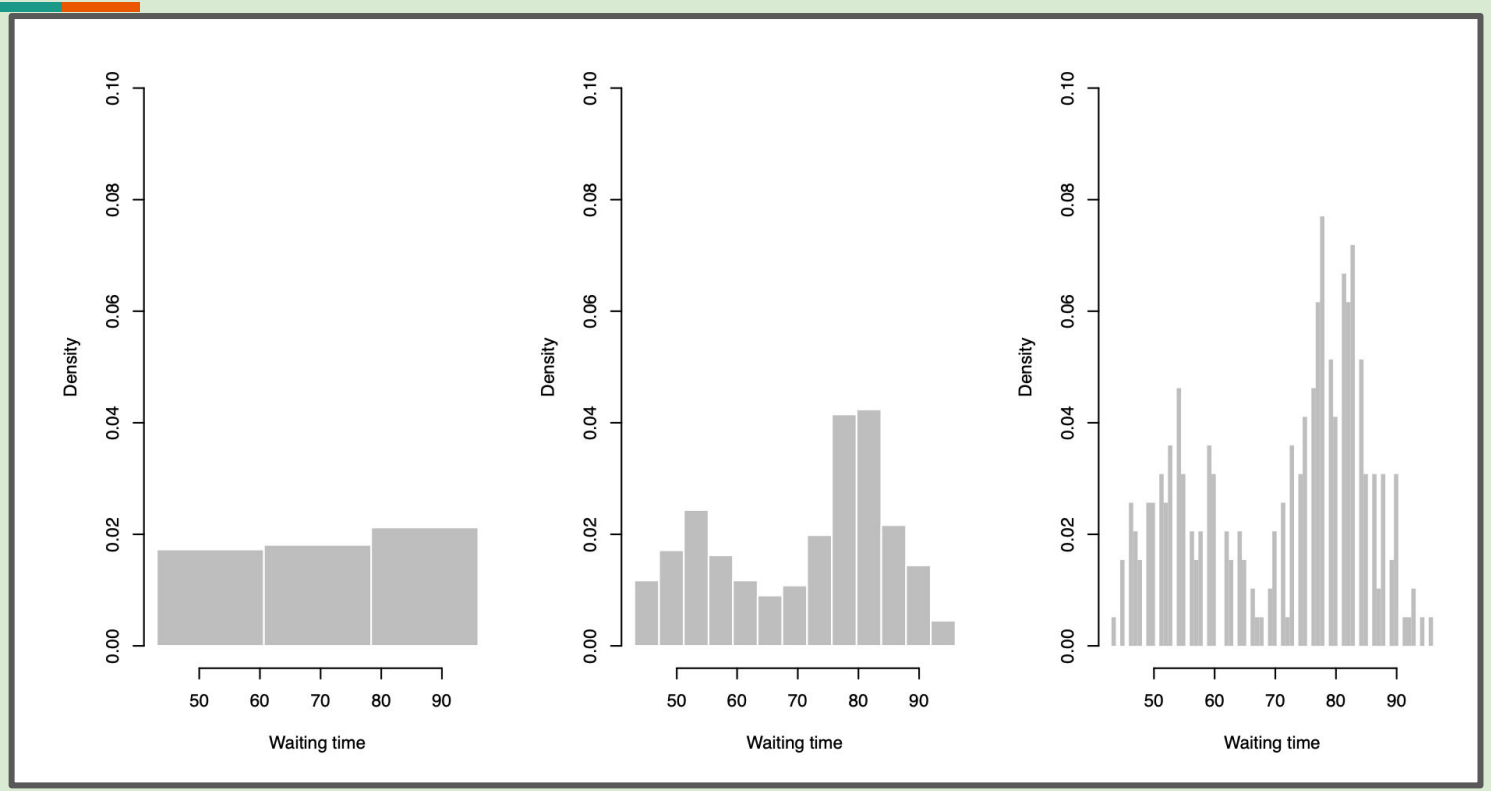
Histogram Bias



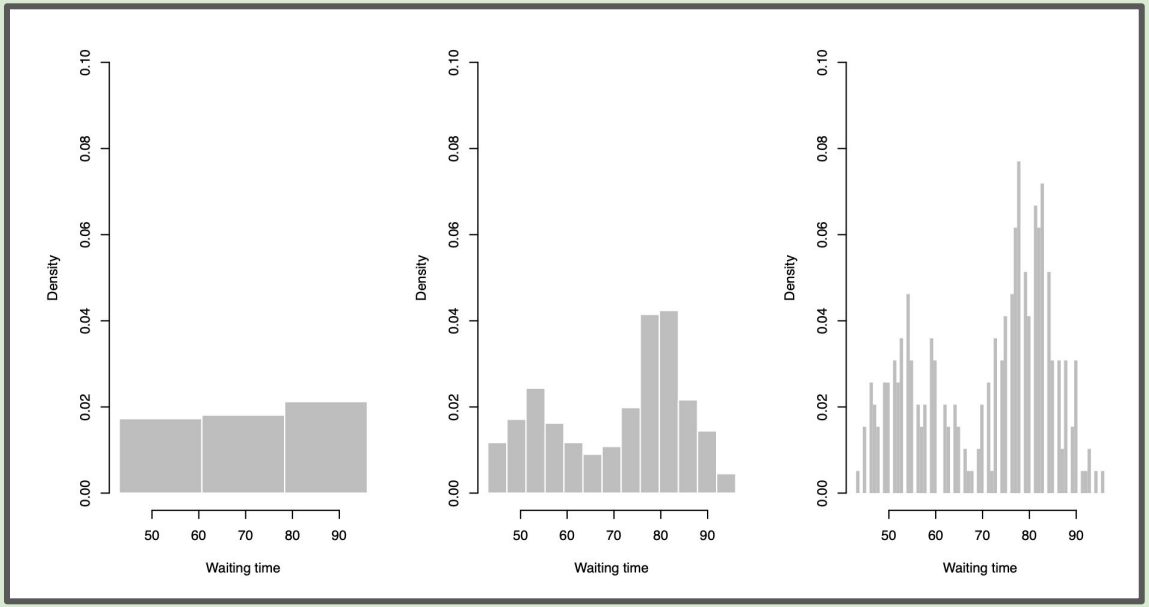
- The histogram is therefore an unbiased estimator of the average density over B_j
- But that isn't the same thing as an unbiased estimator of *f* (unless *f* happened to be constant over B_j)
- If *f* changes over B_j then $\hat{f}$ will be biased

Bias Variance trade-off

The bias can be alleviated by choosing a smaller binwidth



Bias Variance trade-off



However recall that:

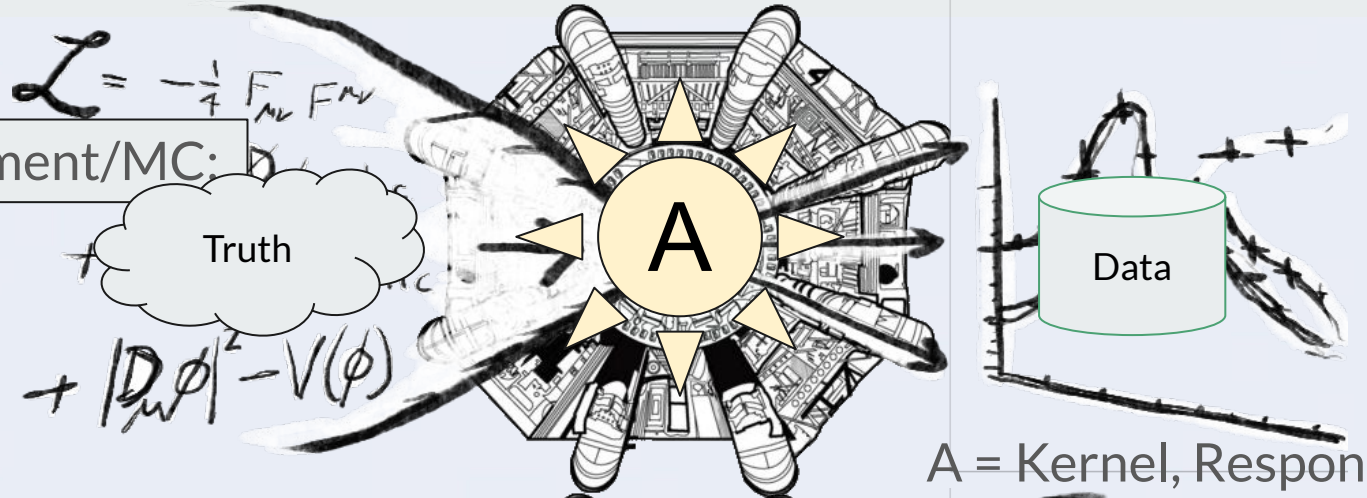
$$\mathbb{V}\hat{f}(x) = \frac{p_j(1-p_j)}{nh^2}$$

- If we make the binwidth twice as small, we quadruple the variance of our estimator
- Therefore, we forced into a difficult tradeoff: if we try to reduce bias, we increase variance, and vice versa

Pause

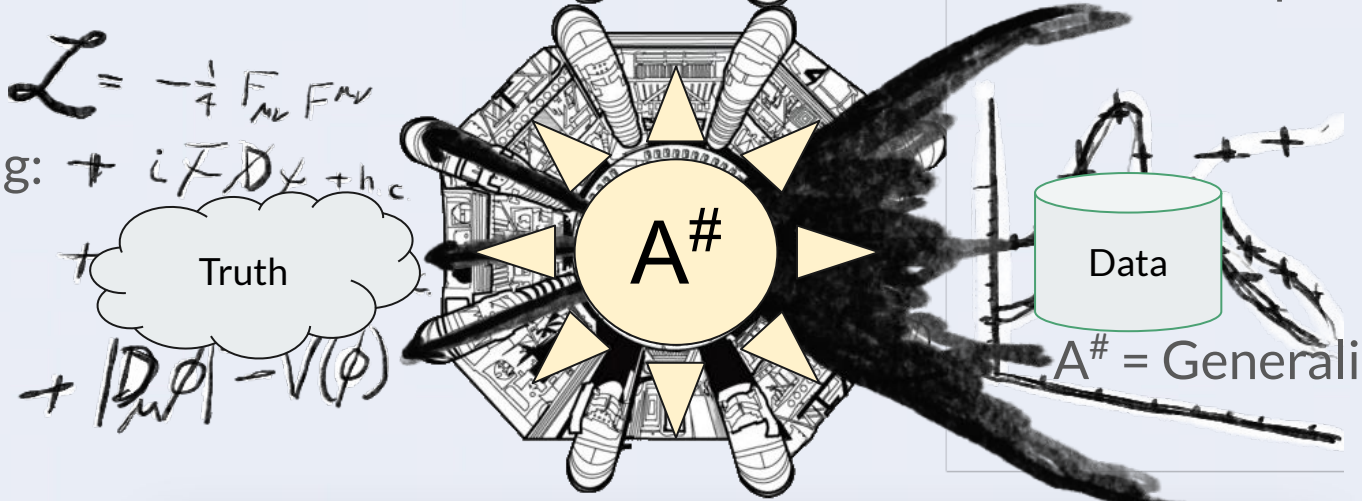
- Estimating functions is risky

Measurement/MC:



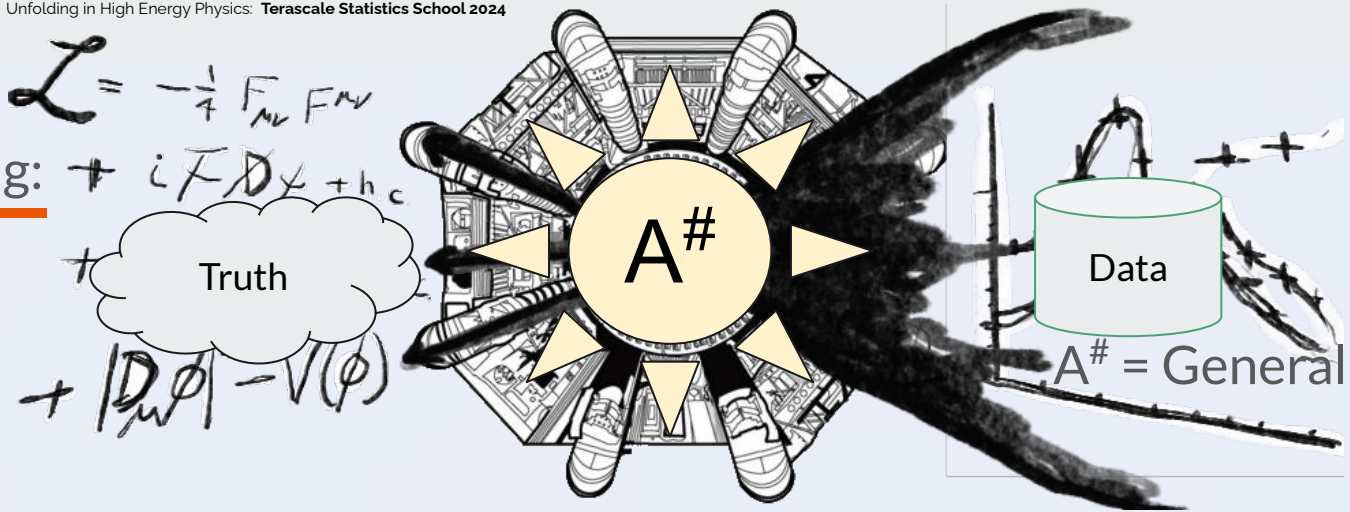
A = Kernel, Response Matrix

Unfolding:



$A^{\#}$ = Generalised Inverse

Unfolding:



A# = Generalised Inverse

Unfolding (deconvolution) with the inverse transition is ill-posed, with an unstable or non-singular solution space. Naïve methods for estimating the truth distribution from measured data can result in chaotic solutions. Methods that constrain to the solution space lead to biased results.

The generalized inverse $A^{\#}$ should depend only on the detector properties, it should not depend on the expected result; it allows to propagate the input errors to the result.

Fredholm integral equation

Consider a random variable y, the goal is to determine $f(y)$

$$\int_0^1 K(x,y)f(y)dy = g(x),$$

Here the kernel K and the right-hand side measured function g are considered known functions.

If f and K are known (as in the forward case) then we can simply compute g by evaluating the integral

Properties of the FIE

$$\int_0^1 K(x,y)f(y)dy = g(x)$$

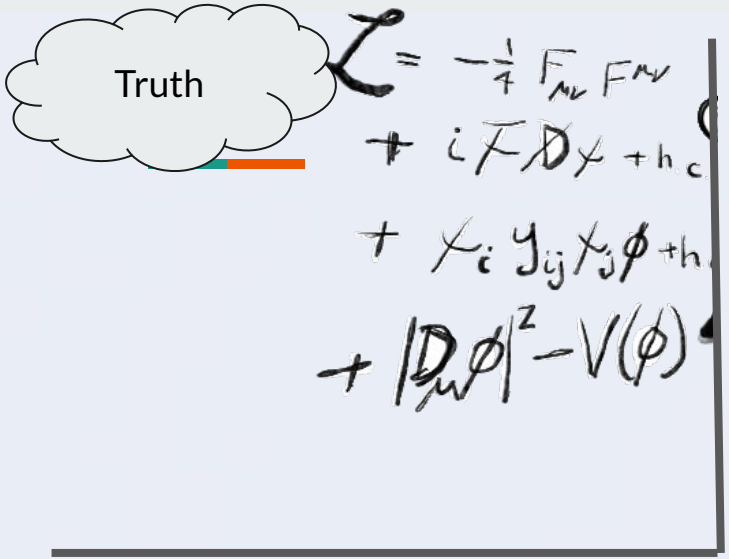
Riemann-Lebesgue Lemma (aka Mercer's Theorem): mapping from f -> g has a dampening/smoothing effect for arbitrary kernels such that the information content of f diminishes with precision of g. As such, estimating f from a measurement of g means that arbitrary levels of noise in g can produce unbounded contributions to the prediction of f.

Smoothing: $K(x,y) = \sum_{i=1}^{\infty} \mu_i u_i(x) v_i(y)$ where μ_i are a decreasing sequence of singular values. The "smoother" the kernel K, the faster the singular values decay to 0. The Picard condition states that for a solution to exist, coefficients singular functions describing the noise have to decay faster than the the singular values of the kernel.

$$\int h(x)dx \approx \sum_k^n w_k h(xk)$$

TL;DR: Lots can go wrong!

Discretisation: Nyström method or quadrature method allows us to approximate an integral equation by replacing the integral with a representative weighted sum. The continuous problem is broken into discrete intervals; quadrature or numerical integration determines the weights and locations of representative points for the integral.



Cross sections are measured by counting events for the specific process:

$$\text{Cross-section} = \frac{\text{Number of events}}{\text{Integrated Luminosity}} = \frac{n}{\int \mathcal{L} dt}$$

Detector acceptance probabilities and [efficiencies], ε_1, ε_2 . . . have to be taken into account:

$$\text{Cross-section} = \frac{n}{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k \int \mathcal{L} dt} = \frac{n}{A \int \mathcal{L} dt}$$

where A is the total acceptance and $\int L\, dt$ is the integrated luminosity.

The candidate event sample may contain background, which has to be subtracted:

$$\text{Cross-section} = \frac{n_{cand} - \tau\, n_{bkg}}{A \int \mathcal{L} dt}$$

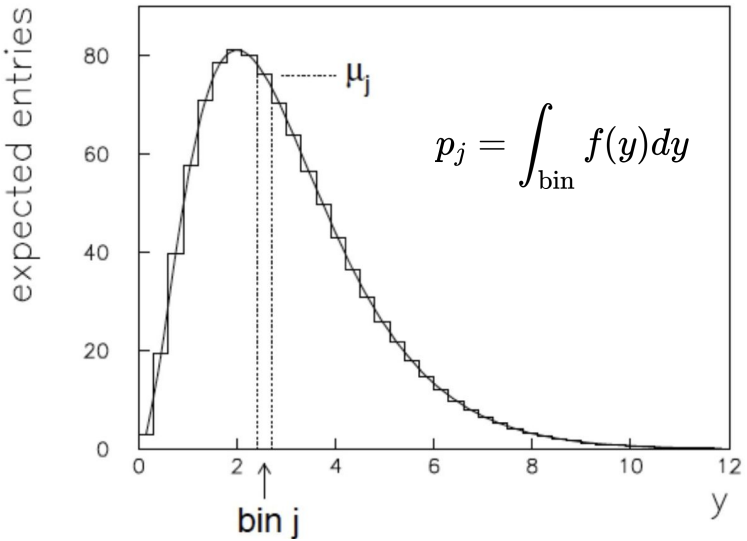
Statistics:

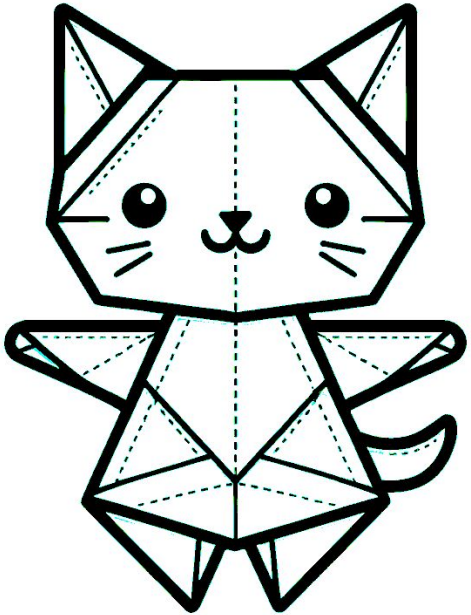
- the number(s) *n* of events follows the Poisson distribution
- the total acceptance factor *A* follows the log-normal distribution (i.e. log *A* is Gaussian)
- the factor τ and the integrated luminosity $\int \mathcal{L} dt$ follow a Gaussian distribution.

Counting Experiments in HEP

If parameterization $f(y;\theta)$ known, find Maximum Likelihood estimators $\hat{\theta}$

If no parameterization available, often construct histogram:
construct estimators for the μ_j (or p_j)
Where μ_j corresponds to the 'true' histogram



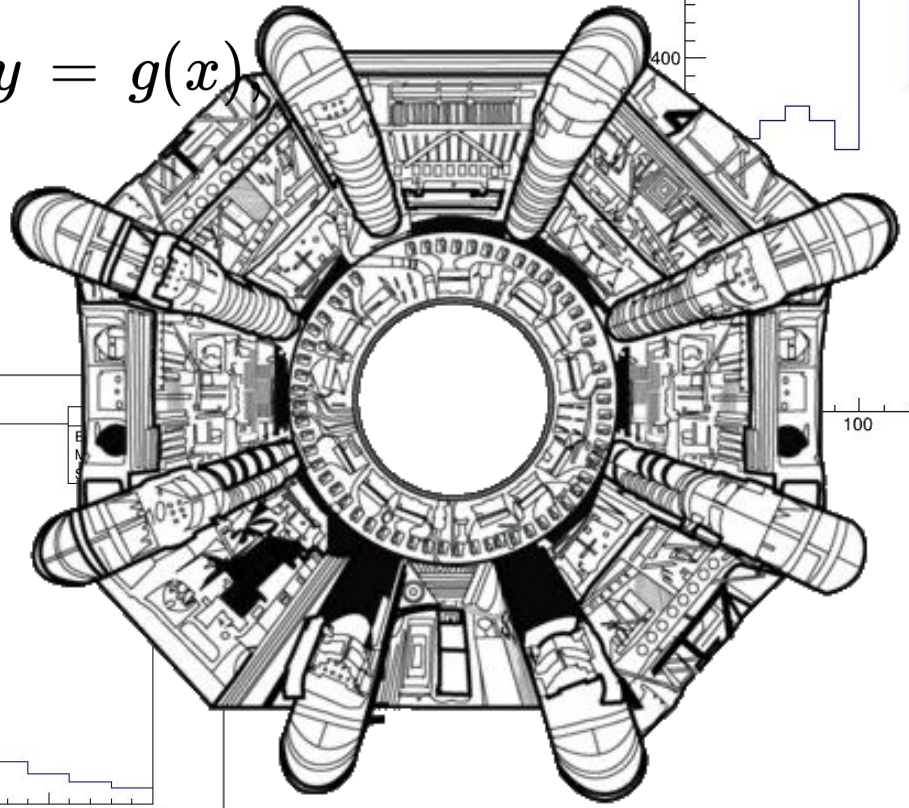


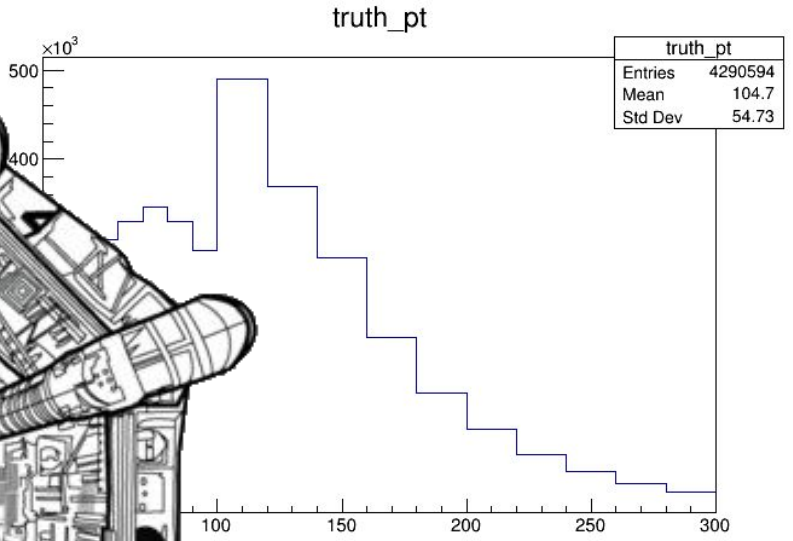
Pause

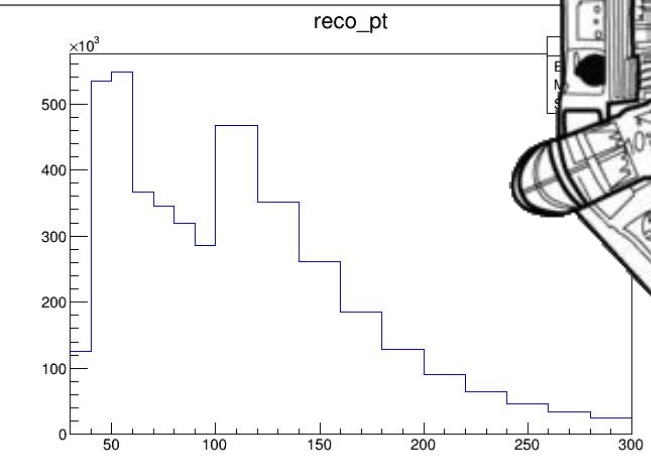
- Fredholm Integral equation describes the unfolding problem
- Need to discretise our functions
- Histograms are practical for HEP and for inverse problems

Counting Experiments in HEP

$$\int_0^1 K(x,y)f(y)dy = g(x),$$



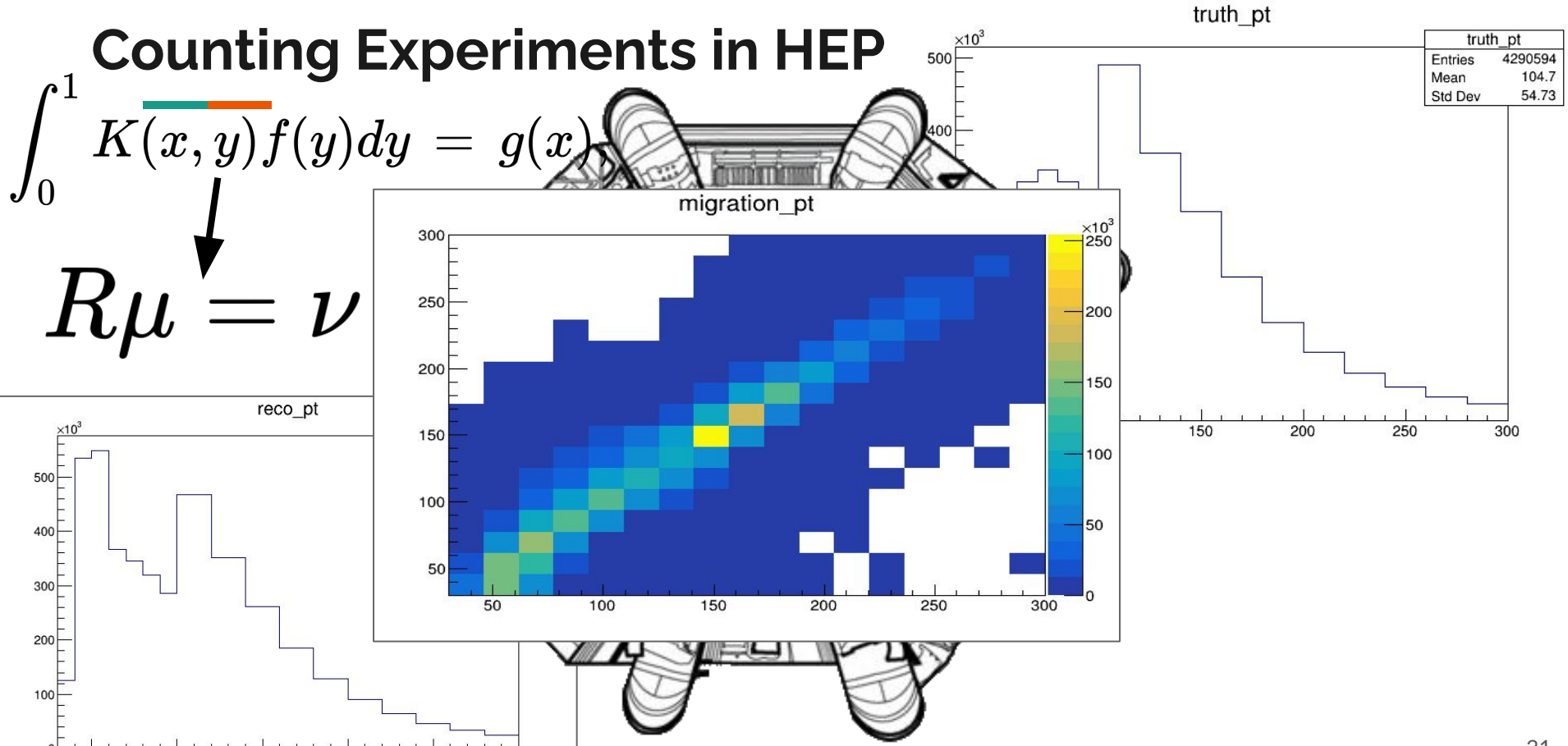




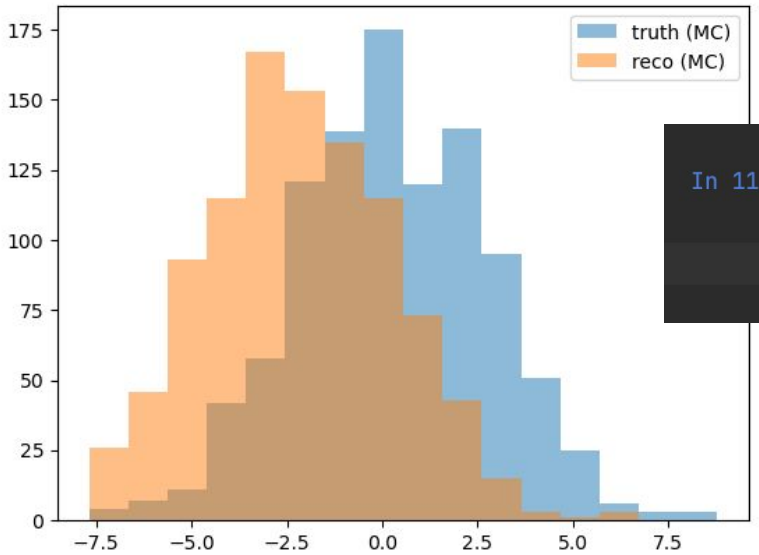
Counting Experiments in HEP

$$\int_0^1 K(x,y)f(y)dy = g(x),$$

$$R\mu = \nu$$

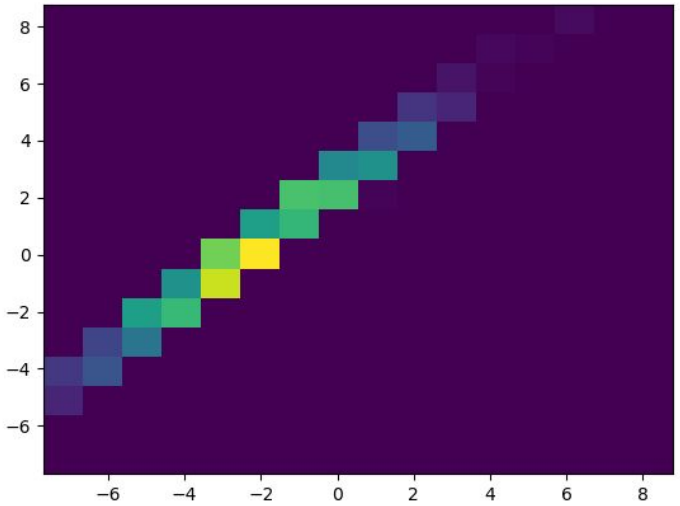


```
In 3
1 #define data and response
2 x = np.random.normal(0.3, 2.5,1000)
3 y = [event+np.random.normal(-2.5,.2) for event in x]
Executed at 2024.04.03 15:01:02 in 4ms
```



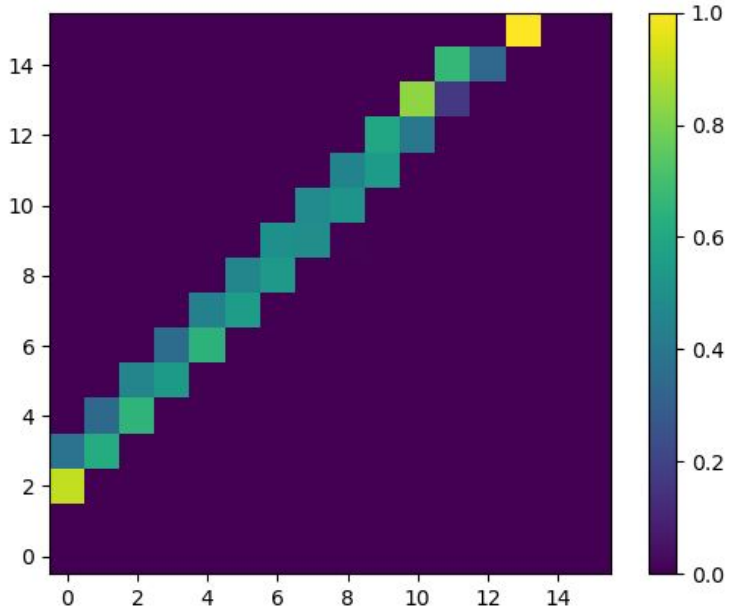

```
In 11
1 truth,bins,_ = plt.hist(x,16,alpha=.5,label='truth (MC)')
2 reco,_,_ = plt.hist(y,bins,alpha=.5,label='reco (MC)')
3 plt.legend();
Executed at 2024.04.03 15:08:05 in 110ms
```

```
In 12  1  response_hist,_,_,_ = plt.hist2d(y,x,bins=[bins,bins])
          Executed at 2024.04.03 15:12:05 in 83ms
```

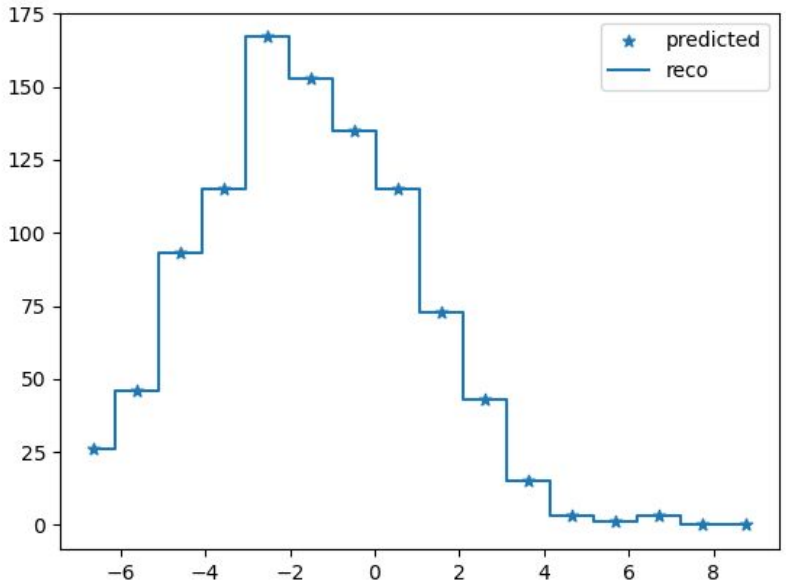


$$R_{ij} = P(\text{observed in bin i} \,|\, \text{produced in bin j})$$

```
In 13  1  probability = np.divide(response_hist,truth)
       2  plt.imshow(probability.T, origin='lower')
       3  plt.colorbar(orientation='vertical');
          Executed at 2024.04.03 15:13:02 in 106ms
```



In 14
```
1 bin_centres = bins[:-1]+np.diff(bins)
2 plt.scatter(bin_centres,np.dot(probability,truth),marker="*", label='predicted')
3 plt.step(bin_centres,reco,where='mid',label='reco')
4 plt.legend();
```
Executed at 2024.04.03 15:15:43 in 90ms



Maximum likelihood (ML) solution

Let us take the most obvious solution;

Assume that the problem $\nu = R\,\mu + \beta$ can be inverted

$$\mu = R^{-1}(\nu - \beta)$$

- But remember; ν are the expectation values of the observed histogram, not the observed histogram itself (which is denoted by n).

Assume also that the data can be considered as independent poisson distributions for each bin

- Now the maximum likelihood estimator is $\hat{\nu} = n$ so then

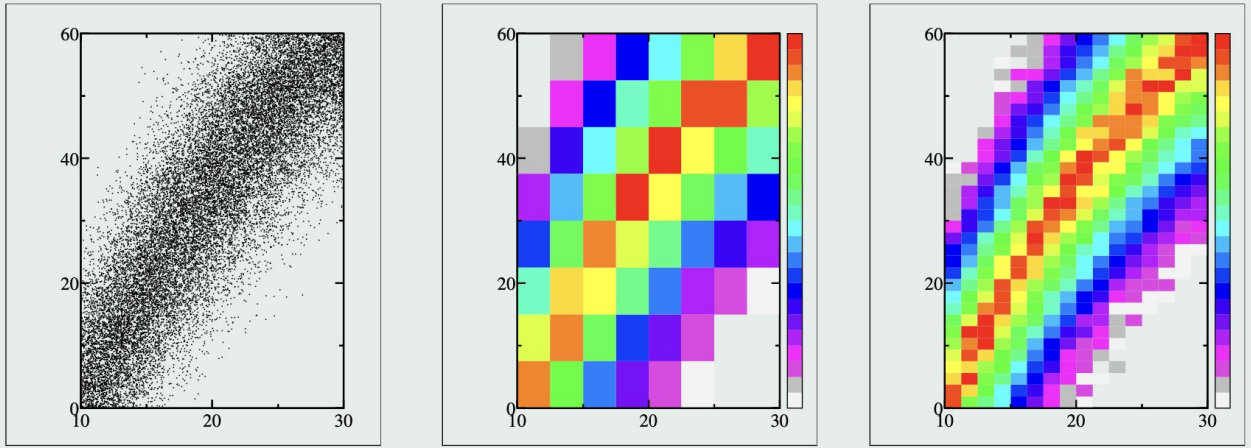
$$\hat{\mu} = R^{-1}(n - \beta)$$

Unfolding Folklore,

https://www.desy.de/~sschmitt/blobel/school_march10.pdf

The response matrix $\boldsymbol{A}$ is generated from Monte Carlo $x - y$-pairs.

The resolution is deteriorated, if too few bins are used (m too small).



scatter plot x raw 8×8 bins x fine 16×32 bins x

Example: plots of x versus y, with small non-linearity in $y = y(x)$ ...

use number of y-bins $m \gtrsim 2n$

Never use $n = m$ with identical bins! – ("inverse crime": ...the numerical methods contain features that effectively render the inverse problem less ill-posed than it actually is, thus yielding unrealistically optimistic results.)

Maximum likelihood (ML) solution

Let us take the most obvious solution;

Assume that the problem $\nu = R\,\mu + \beta$ can be inverted

$$\mu = R^{-1}(\nu - \beta)$$

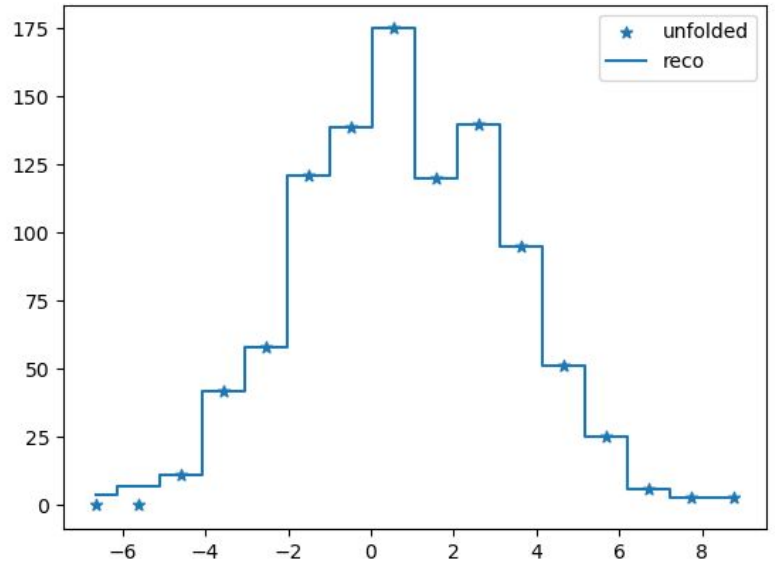
- But remember; ν are the expectation values of the observed histogram, not the observed histogram itself (which is denoted by n).

Assume also that the data can be considered as independent poisson distributions for each bin

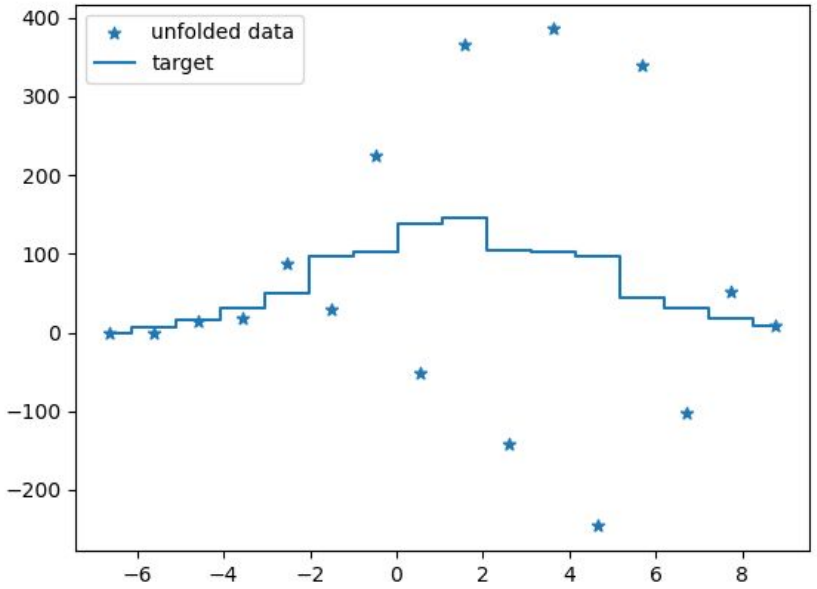
- Now the maximum likelihood estimator is $\hat{\nu} = n$ so then

$$\hat{\mu} = R^{-1}(n - \beta)$$

In 16
```
plt.scatter(bin_centres,np.dot(np.linalg.pinv(probability),reco),marker="*", label='unfolded')
plt.step(bin_centres,truth,where='mid',label='reco')
plt.legend();
```
Executed at 2024.04.03 15:18:50 in 109ms



In 9
```
#same smearing but a different distribution
target = np.random.normal(1.2, 3.,1000)
data = [event+np.random.normal(-2.5,.2) for event in target]
target_hist,_ = np.histogram(target,bins)
data_hist,_ = np.histogram(data,bins)
```
Executed at 2024.04.03 15:01:34 in 4ms



In 10
```
plt.scatter(bin_centres,np.dot(np.linalg.pinv(probability),data_hist),marker="*", label='unfolded data')
plt.step(bin_centres,target_hist,where='mid',label='target')
plt.legend();
```
Executed at 2024.04.03 15:01:36 in 73ms

Remember me?

$$\int_0^1 K(x,y)f(y)dy = g(x)$$

Riemann-Lebesgue Lemma (aka Mercer's Theorem): mapping from f -> g has a dampening/smoothing effect for arbitrary kernels such that the information content of f diminishes with precision of g. As such, estimating f from a measurement of g means that arbitrary levels of noise in g can produce unbounded contributions to the prediction of f.

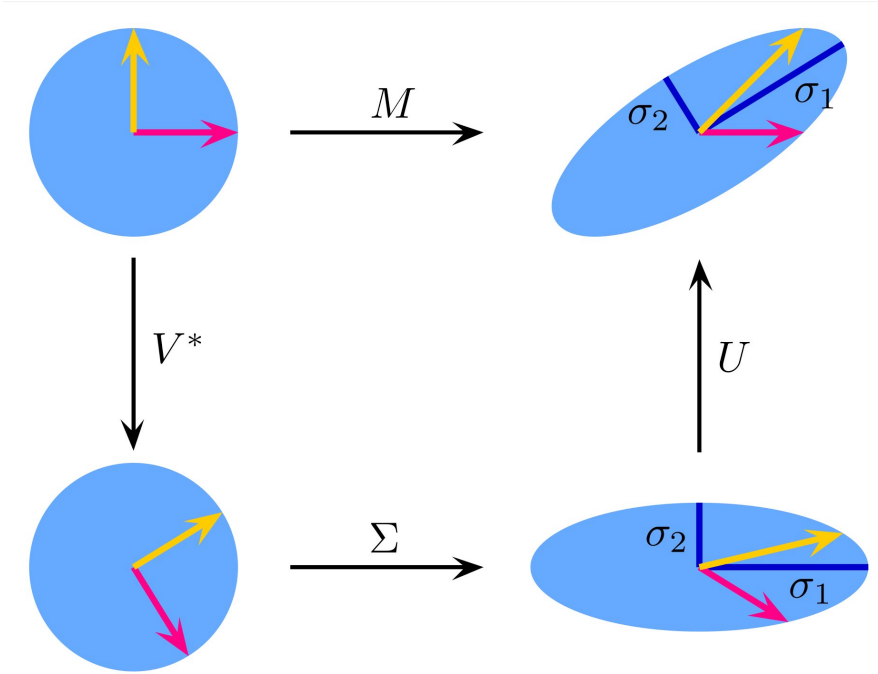
Smoothing: $K(x,y) = \sum_{i=1}^{\infty} \mu_i u_i(x) v_i(y)$ where μ_i are a decreasing sequence of singular values. The "smoother" the kernel K, the faster the singular values decay to 0. The Picard condition states that for a solution to exist, coefficients singular functions describing the noise have to decay faster than the the singular values of the kernel.

Singular Values

Eigenvalues $\lambda_j \geq 0$ (diagonals of Σ)are in decreasing order, with $\lambda_1 \geq \lambda_2 \geq ...\lambda_n \geq 0$.

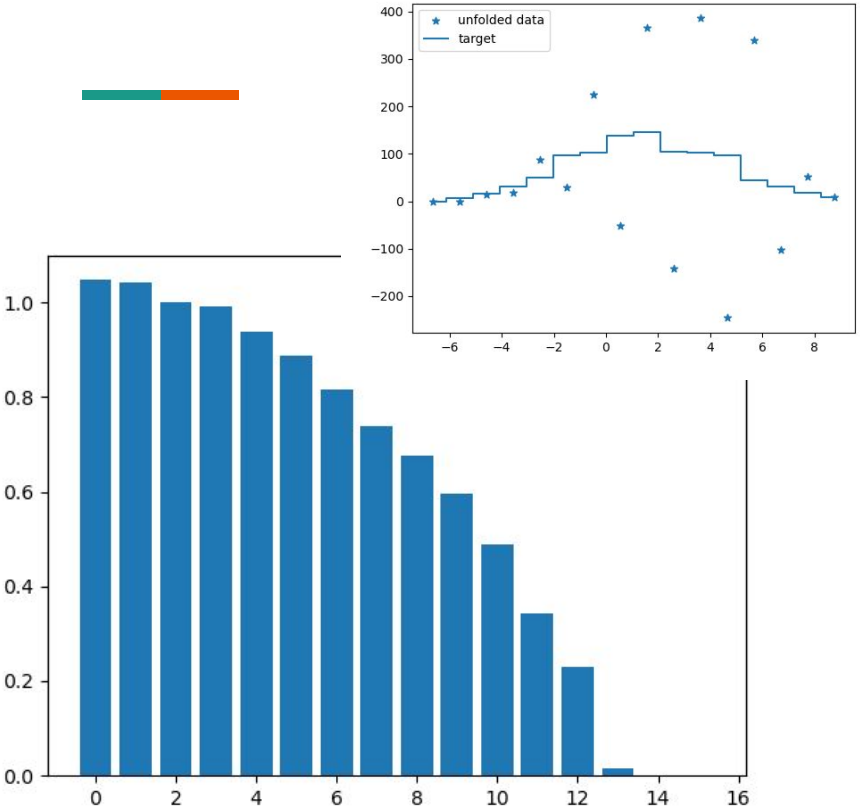
The orthogonal matrix $U = [u_1, u_1, \ldots, u_n]$ is an array of eigenvectors u_j of the matrix M

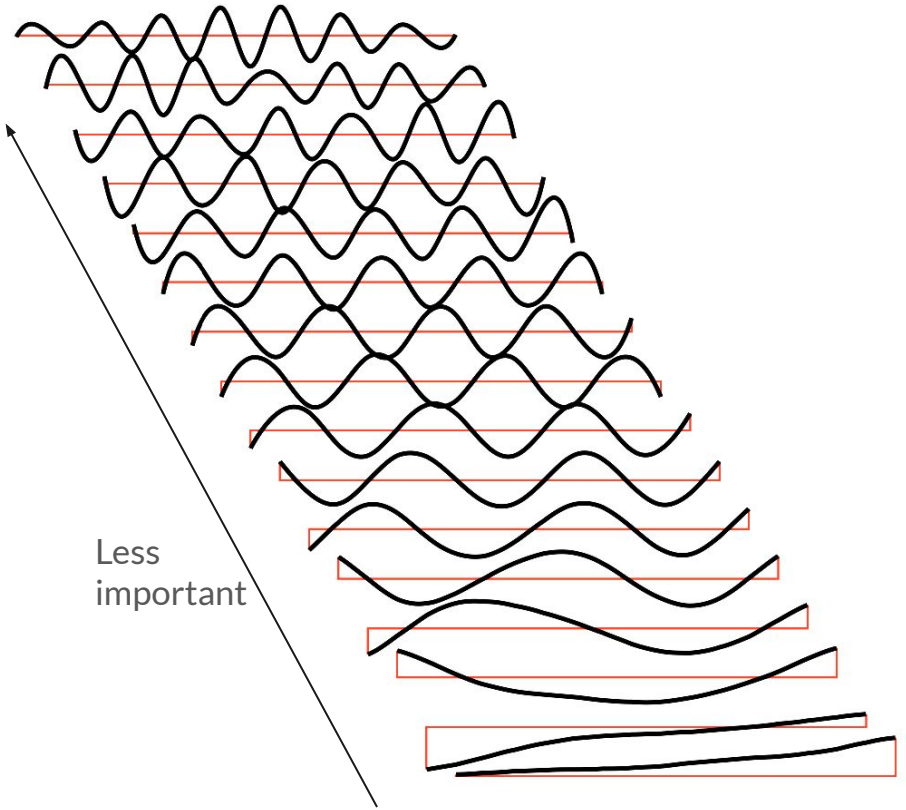
If the inverse M^{-1} exists (i.e. all $\lambda_j > 0$) the solution is expressed by a vector c of Fourier coefficients.



$$M = U \cdot \Sigma \cdot V^*$$

Specifically, the singular value decomposition of an $m \times n$ complex matrix $\mathbf{M}$ is a factorization of the form $\mathbf{M} = \mathbf{U\Sigma V^*}$, where $\mathbf{U}$ is an $m \times m$ complex unitary matrix, $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, $\mathbf{V}$ is an $n \times n$ complex unitary matrix, and $\mathbf{V}^*$ is the conjugate transpose of $\mathbf{V}$. Such decomposition always exists for any complex matrix. If $\mathbf{M}$ is real, then $\mathbf{U}$ and $\mathbf{V}$ can be guaranteed to be real orthogonal matrices; in such contexts, the SVD is often denoted $\mathbf{U\Sigma V}^{\mathrm{T}}$.


unfolded data
target
400
300
200
100
0
−100
−200
−6
−4
−2
0
2
4
6
8
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
16


Less
important

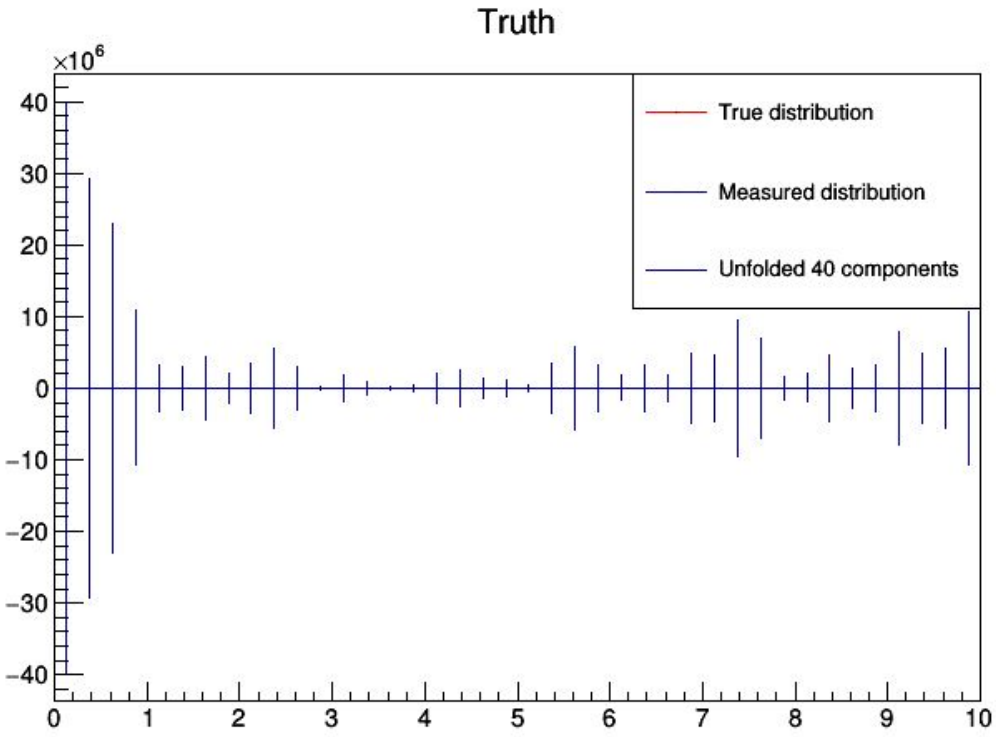
Pause

- Discretisation with quadrature methods allows linear algebra solutions.
- Noise is highly problematic
- Problems come from small singular values.

Example in practice

The problem: We don't have ***v***, only ***n***.

- R^{-1} interprets fluctuations in ***n*** as the residual of original fine-structure and puts this back.
- Causes breakdown
  - In this example 40 components corresponds to the matrix inversion solution



Likelihood formalism and regularisation function.

The data are described with a given probability model that determines the likelihood functio $L(\vec{\mu}) = P(\vec{n}|\vec{\mu})$.
Often the n_i are modelled as independent and Poisson distributed, so that the likelihood is

$$L(\vec{\mu}) = \prod_{i=1}^{N} \frac{\nu_i^{n_i}}{n_i!} e^{-\nu_i}$$

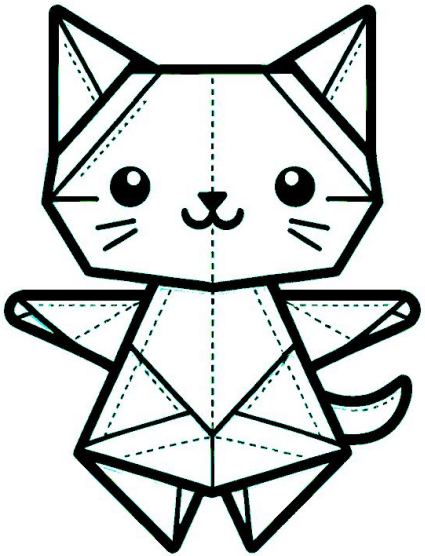
Where you can remember $\nu_i = \sum_{j=i}^{M} R_{ij}\mu_j + \beta_i$

To suppress the large variance of the maximum-likelihood estimator one chooses a μ that does not correspond to the maximum of the log-likelihood in L_{max}, but rather one considers a region of μ-space where ln L(μ) is within some threshold achieved by maximising not ln L(μ) but rather a linear combination of it and a regularisation function S(μ), which represents the smoothness of the histogram μ

$$\varphi(\vec{\mu}) = \ln L(\vec{\mu}) + \tau S(\vec{\mu})$$

E.g Tikhonov regularisation

$$S(\vec{\mu}) = -\sum_{i=1}^{M-2} (-\mu_i + 2\mu_{i+1} - \mu_{i+2})^2$$



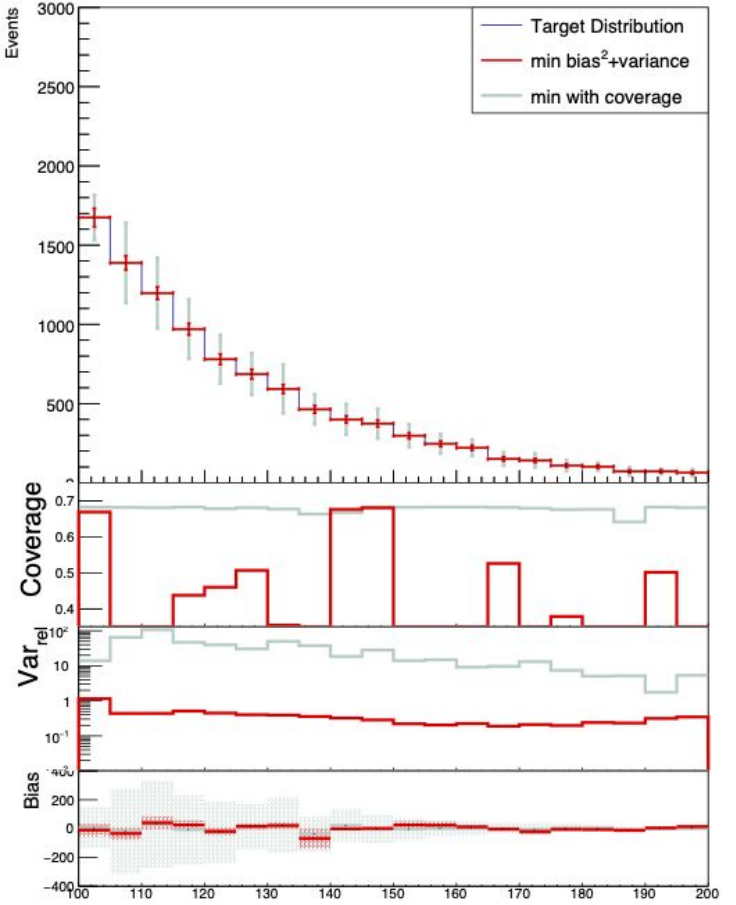
Demonstration

- https://statisticalmethods.web.cern.ch/StatisticalMethods/tutorial/
- https://github.com/vincecr0ft/RooUnfoldTutorials/

Choosing regularisation strength

So how do you take all these things into account in a smart way?

- Choose a regularisation strength that unconditionally minimises the bin-averaged MSE (the mean squared error, aka sum of bias squared and variance, averaged over the bins)
- Require the that the bin-averaged estimate of the coverage of the unfolded data reaches the target coverage of 68.3% within 1%.



Bias, variance and coverage

Conclusion: In unfolding one must accept some bias in exchange for a (hopefully large) reduction in variance. So what do we need to consider?

- Bias: the difference between the estimator's expectation and the parameter's true value.
- Variance: the diagonal elements of the covariance matrix of the estimators for μ
- Coverage: the probability that the true value o μ_i falls between plus and minus one standard deviation about the estimator of μ_i

Note: the bias, variance and coverage are always defined with respect to the chosen true and expected data histograms.

Variance, Bias, and Coverage

Use a *toy* dataset with K events, the variance for bin *i* is then calculated as

$$\sigma^2_{\hat{\mu}_i} = \frac{1}{K-1}\sum_{\mathrm{k}}^{K}\left((\hat{\mu}_i)_k - \frac{1}{K}\sum_{\mathrm{k}}(\hat{\mu}_i)_k\right)$$

Use a *toy* dataset with K events, the bias for bin *i* is then calculated as

$$b_i = \frac{1}{K}\sum_{\mathrm{k}}^{K}(\hat{\mu}_i)_k - \mu_i$$

Under the assumption that t $\hat{\vec{\mu}}$ are Gaussian distributed coverage probability can be calculated in closed form as

$$P_{\mathrm{cov}} = \Phi\left(\frac{b_i}{\sigma_{\hat{\mu}_i}} + 1\right) - \Phi\left(\frac{b_i}{\sigma_{\hat{\mu}_i}} - 1\right)$$

Where Φ is the Standard Gaussian cumulative distribution function.

Study on smeared exponential distribution

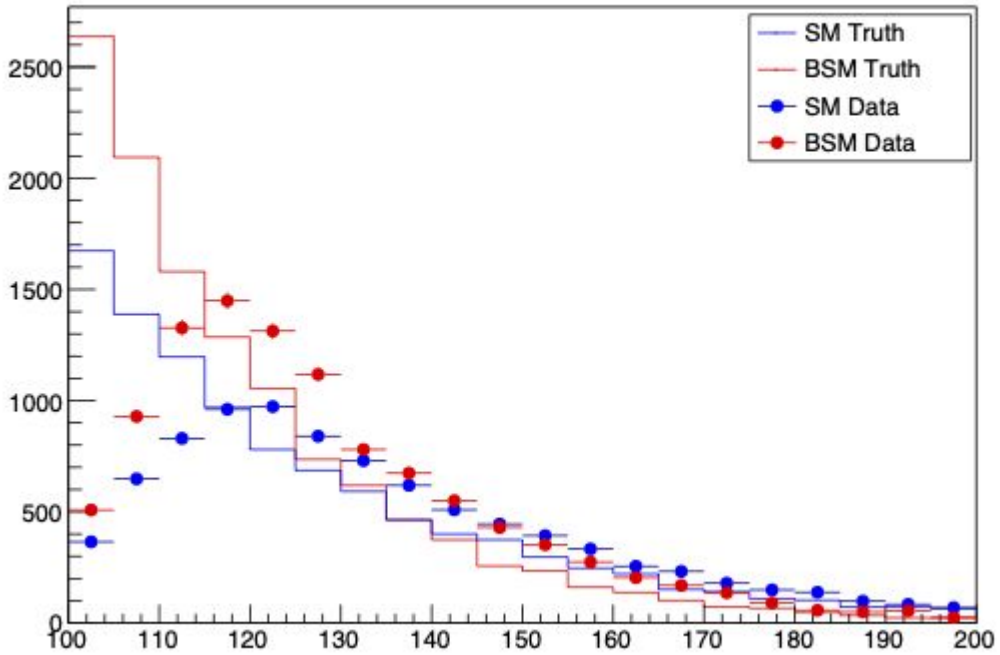
We define as benchmark model an exponential decay distribution, smeared with a resolution function that is loosely inspired on a calorimeter response:

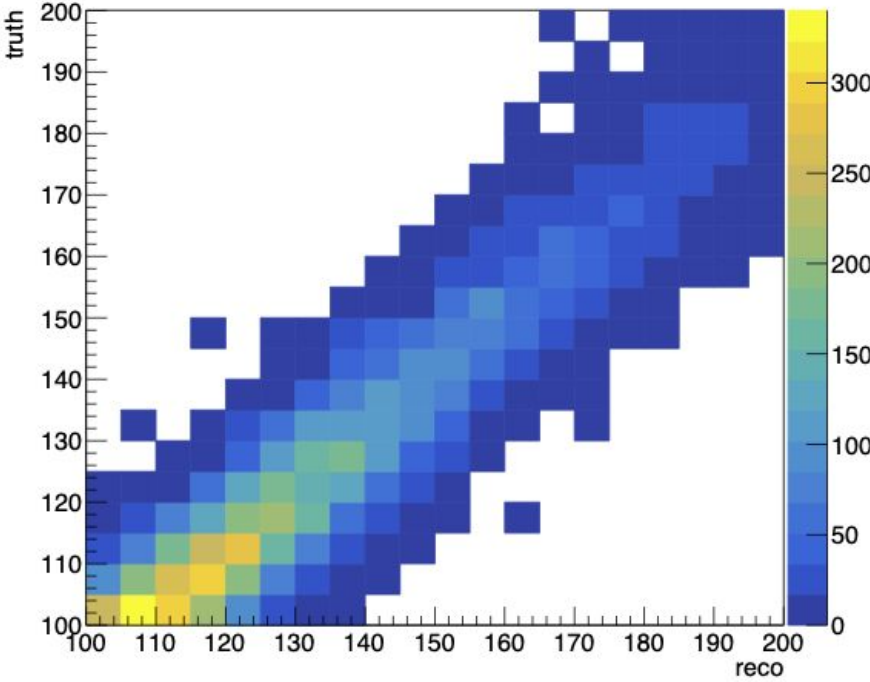
$$\begin{aligned} f(x|\alpha) &= f_{\text{physics}}(x_{\text{true}}|\alpha) * f_{\text{detector}}(x_{\text{true}}, x) \\ &= (\alpha \cdot \exp(-\alpha \cdot x_{\text{true}})) * \text{Gauss}\,(x - x_{\text{true}}, 7.5, 0.5 \cdot \sqrt{x_{\text{true}}} + 2.5) \end{aligned}$$

where the $*$ symbol represents the convolution operator.

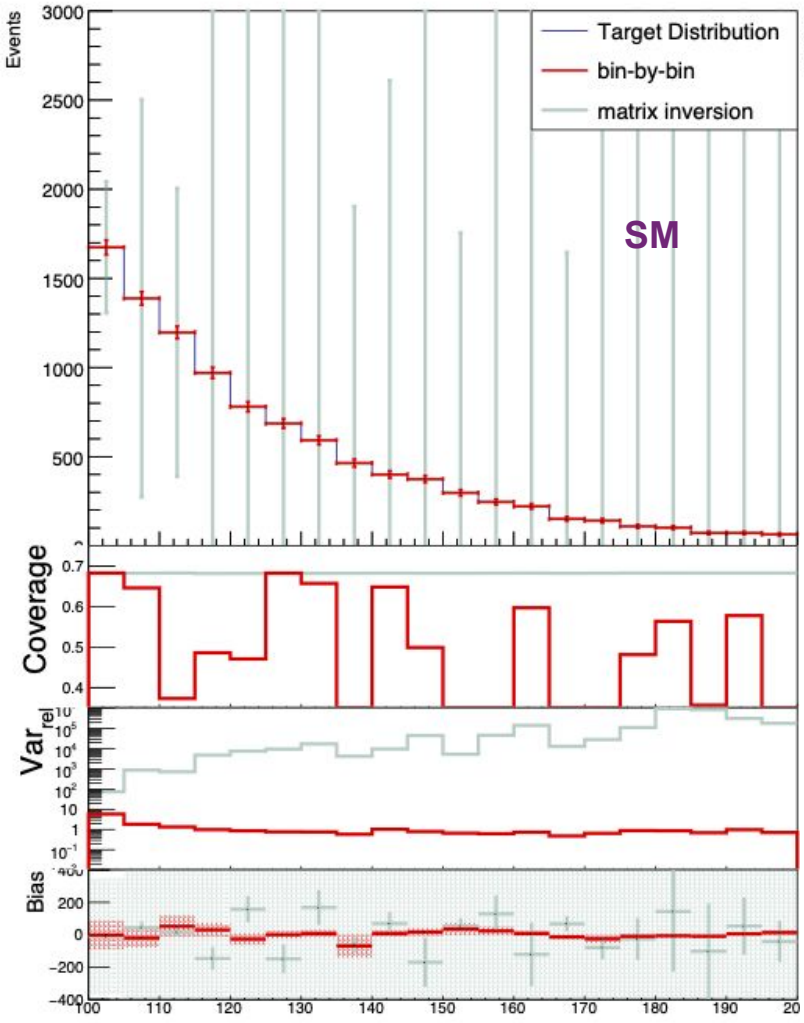
We define two models variants, labeled SM (‘Standard Model’) and BSM (‘Beyond the Standard Model’), that correspond to an exponential distribution with a slope α of 0.035 and 0.05 respectively

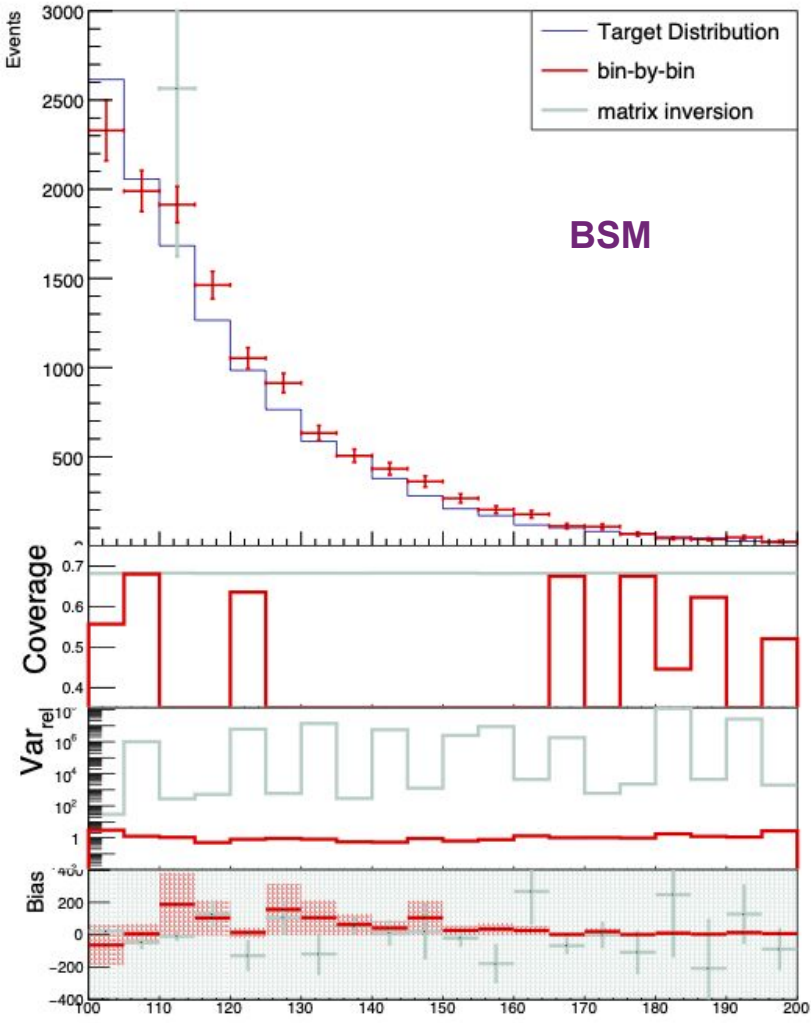
Study distributions





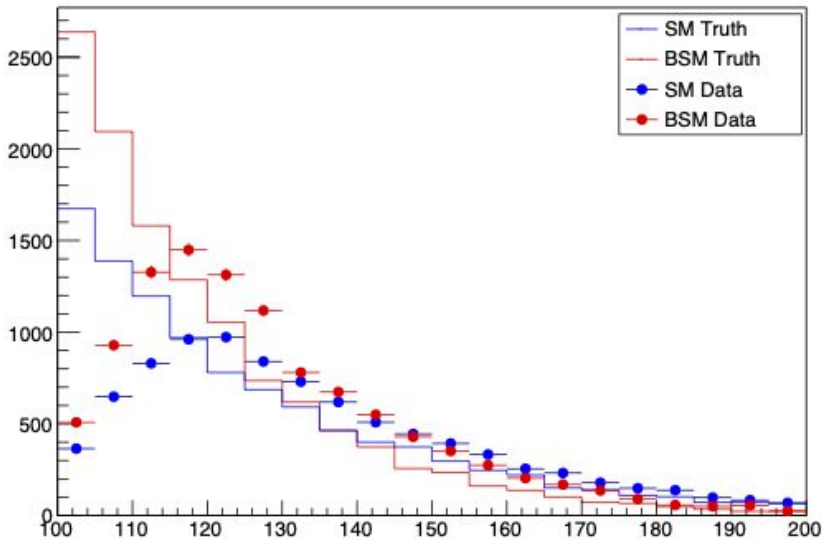
Methods that are not tunable

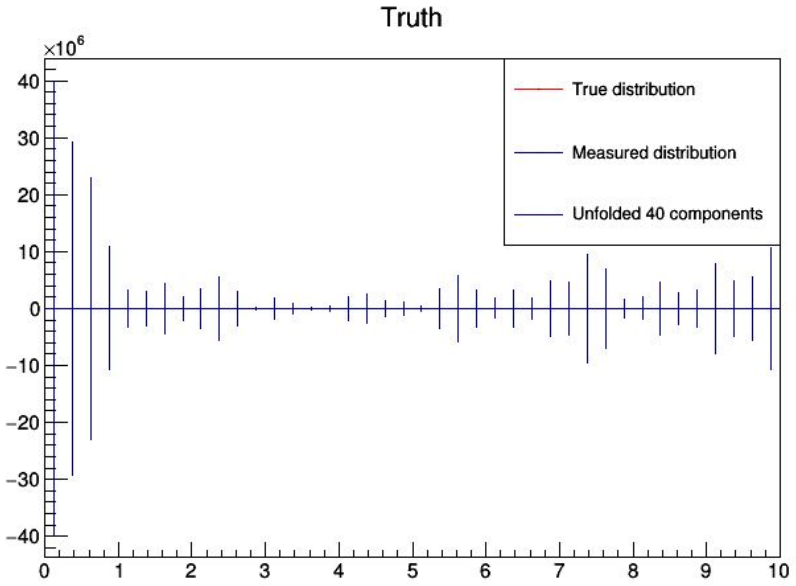




Single Value Decomposition (SVD)

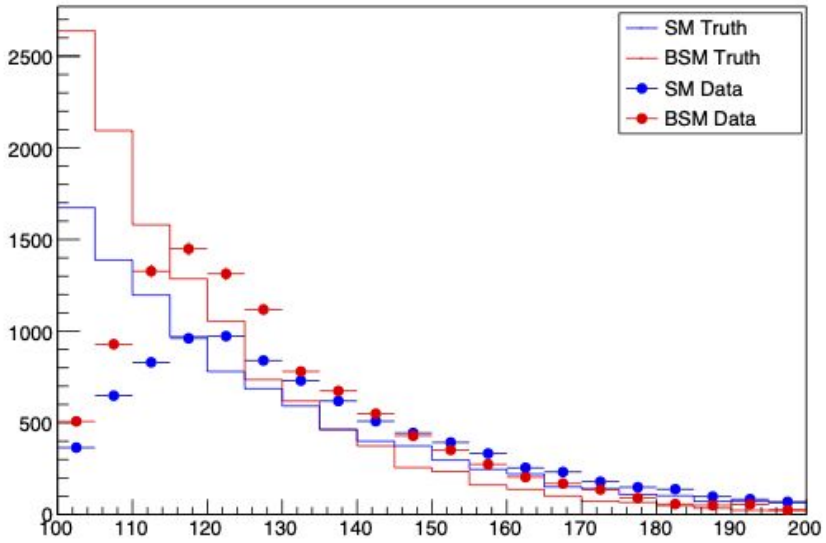
Remember

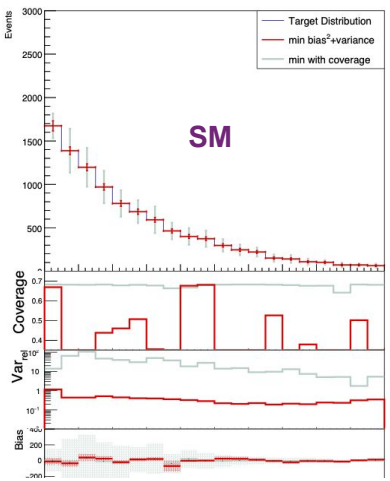


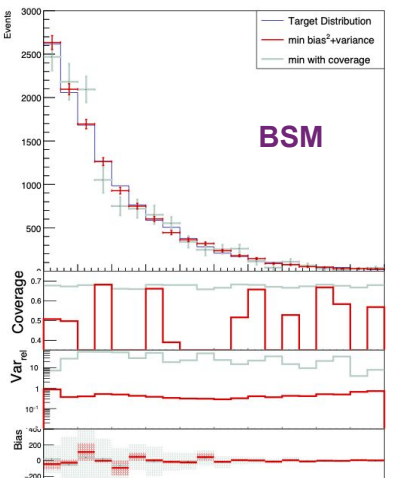


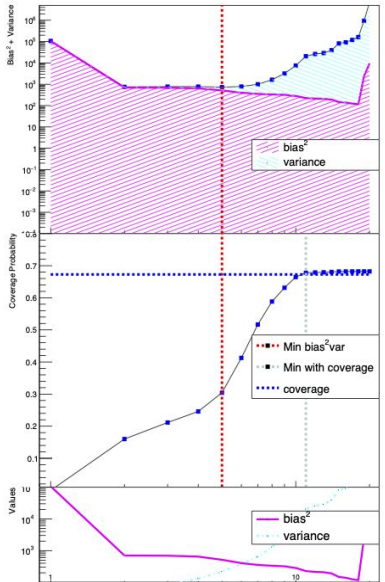
Single Value Decomposition (SVD)

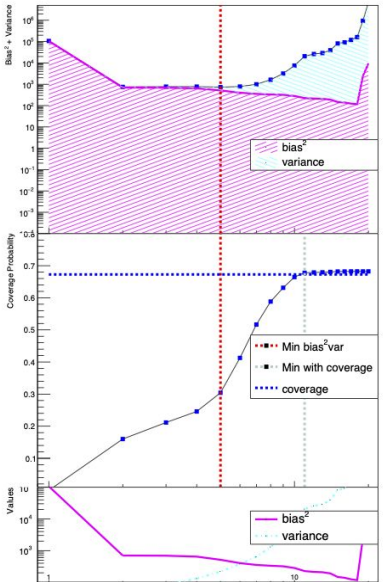
Looking at Coverage, Variance and Bias



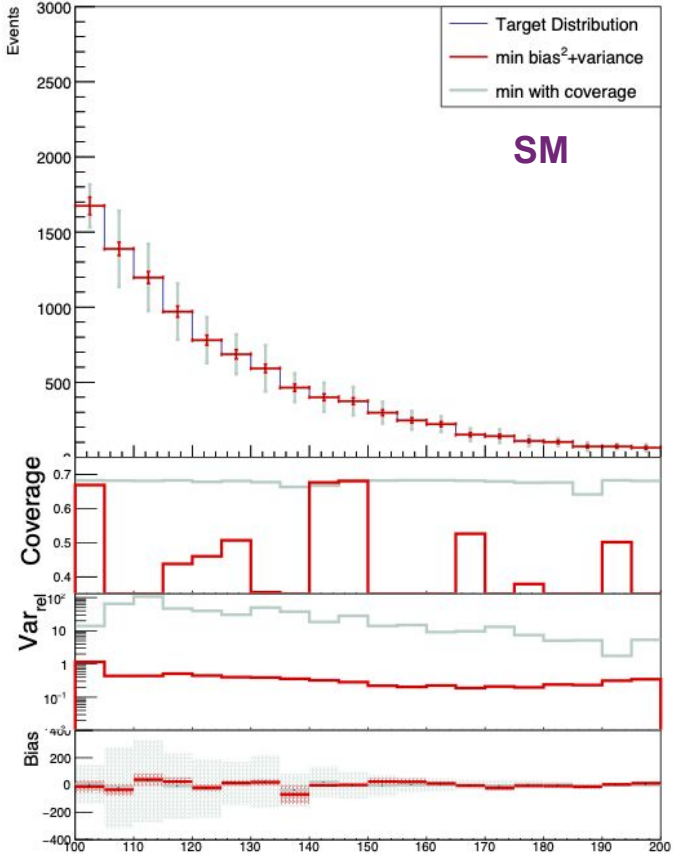


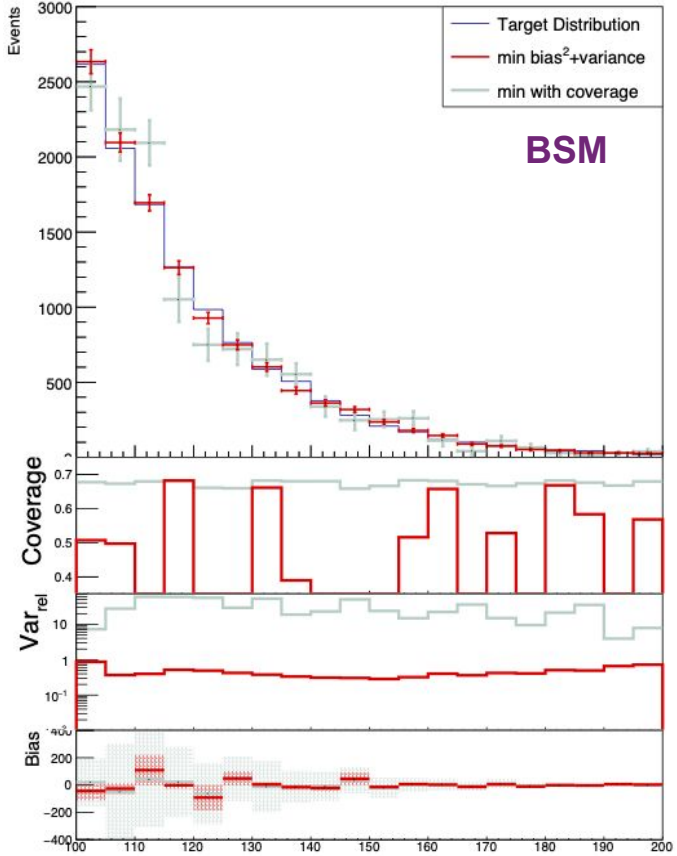






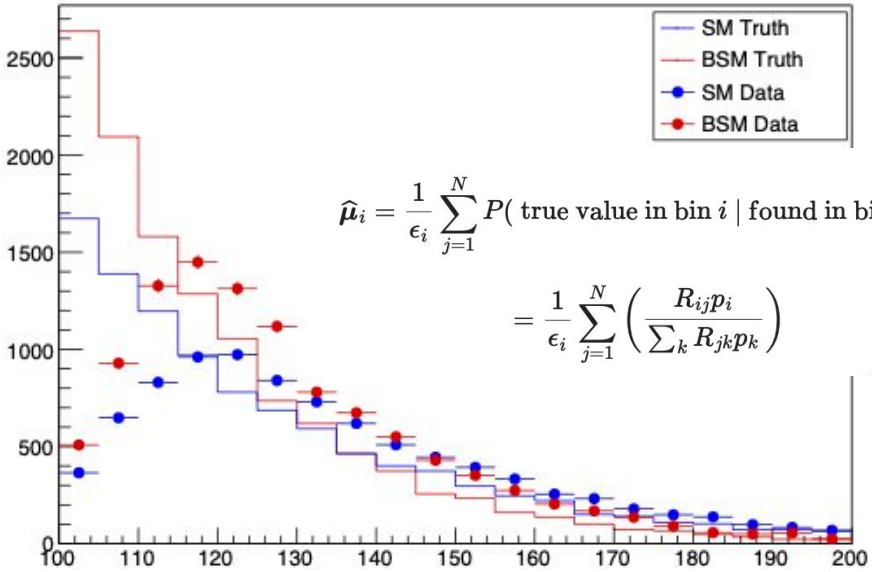
Single Value Decomposition (SVD)





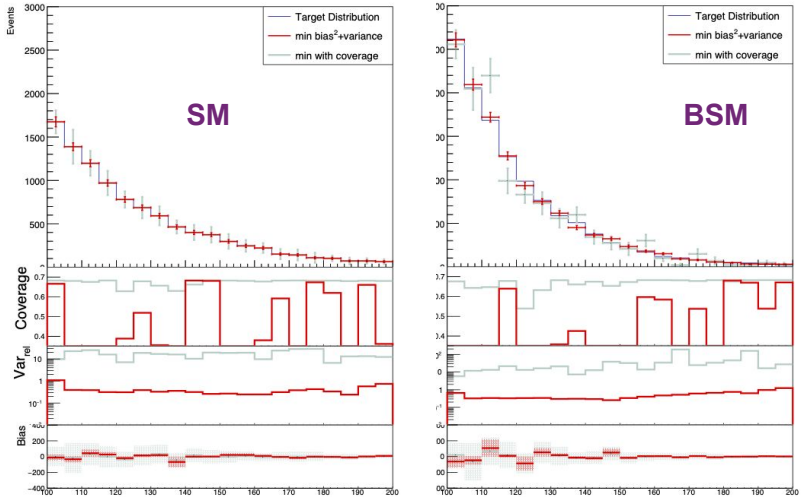
Richardson-Lucy (D'Agostini, IBU)

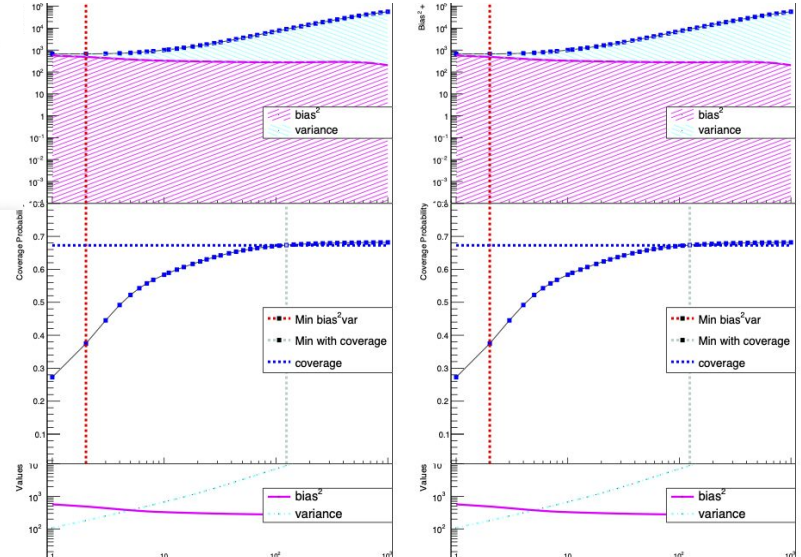
Looking at Coverage, Variance and Bias



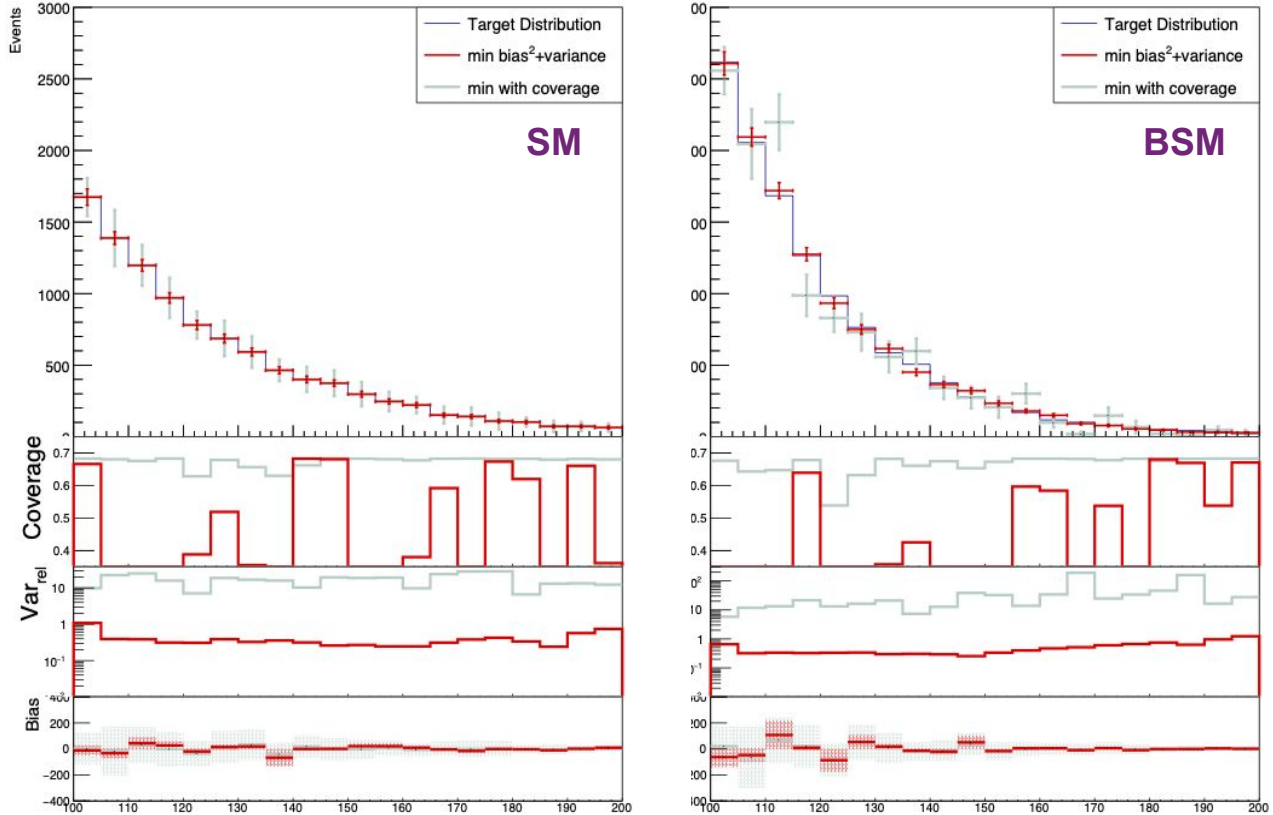
$$\widehat{\boldsymbol{\mu}}_i = \frac{1}{\epsilon_i} \sum_{j=1}^{N} P(\text{ true value in bin } i \mid \text{found in bin } j\ )\, n_j$$

$$= \frac{1}{\epsilon_i} \sum_{j=1}^{N} \left(\frac{R_{ij} p_i}{\sum_k R_{jk} p_k} \right)$$





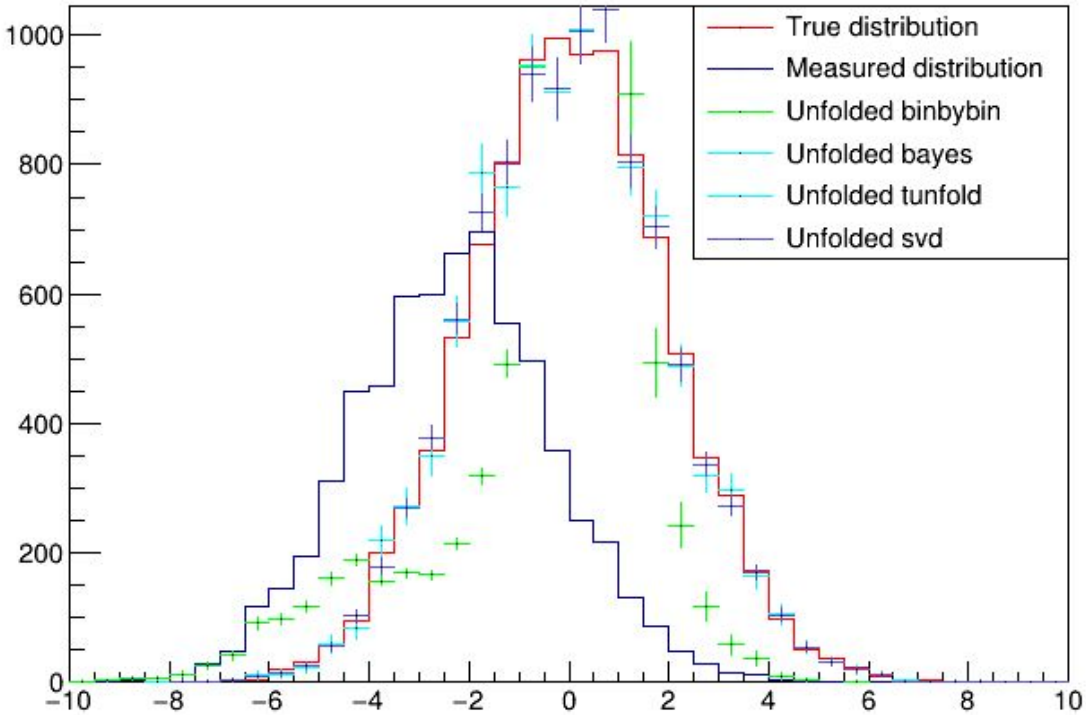
Richardson-Lucy (D'Agostini, IBU)



A Summary of Unfolding Methods in RooUnfold

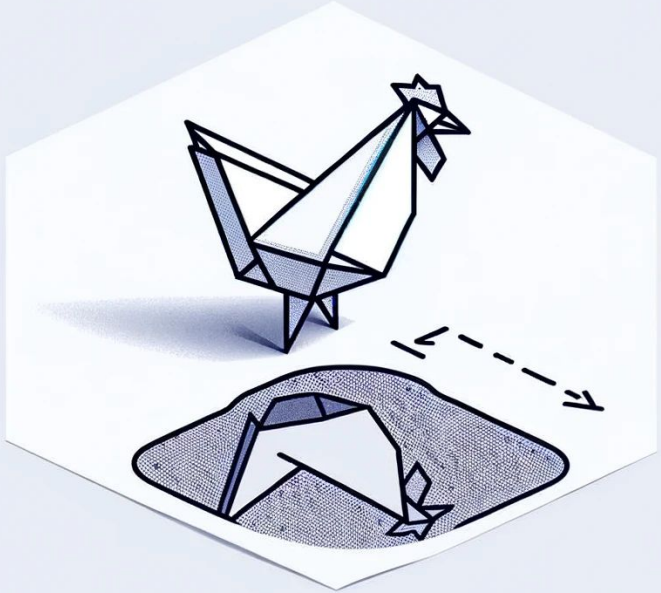
- Common interface to multiple methods
- Each with different error propagation
- Each with different responses to distributions
- Each with different regularisation parameters.

Open in SWAN



Detector uncertainties in the Likelihood

$$\mathcal{L}(\mu, \sigma \mid n)$$

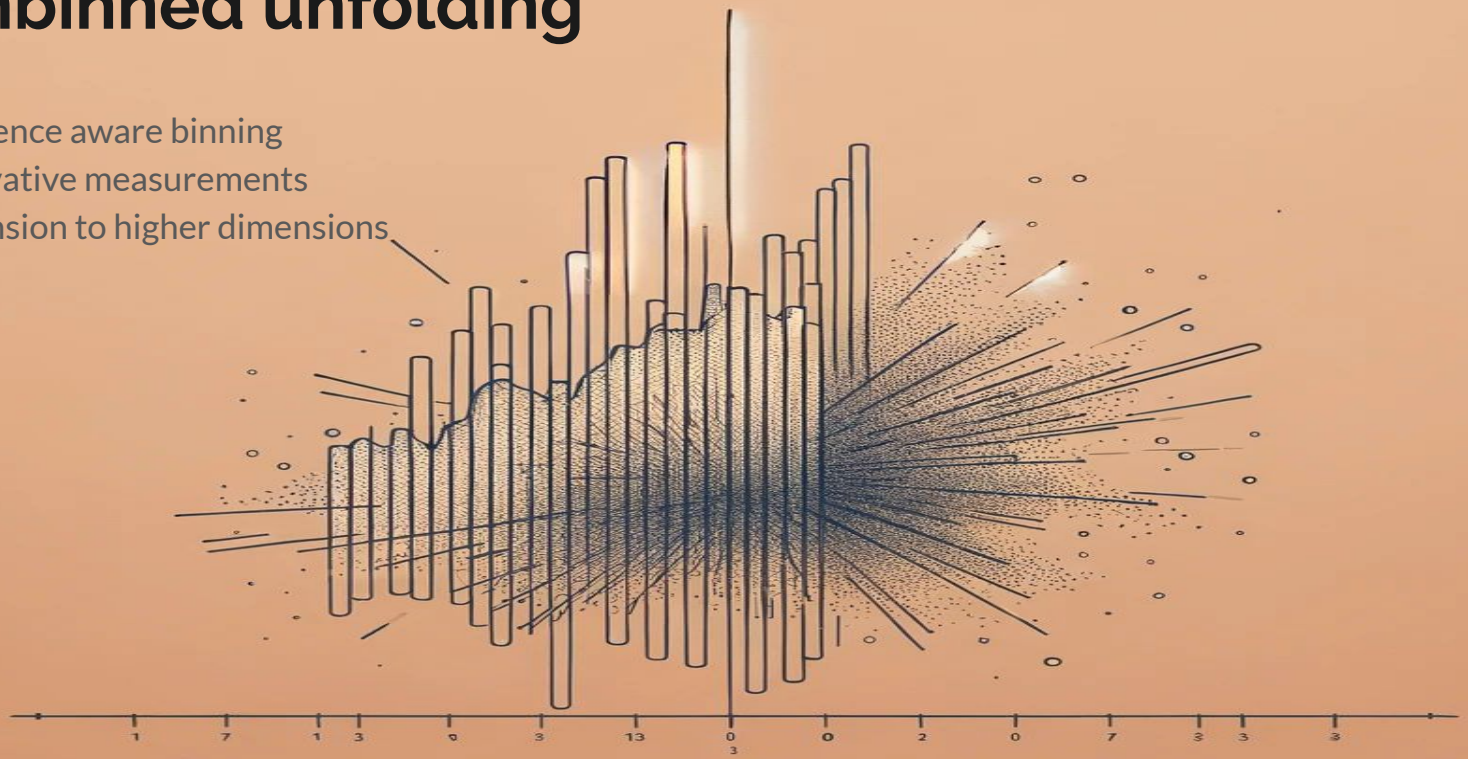


$$\mathcal{L}(\mu \mid \sigma, n)$$

$$\mathcal{L}(\mu \mid \hat{\sigma}, \bar{\nu})$$

Unbinned unfolding

- Inference aware binning
- Derivative measurements
- Extension to higher dimensions

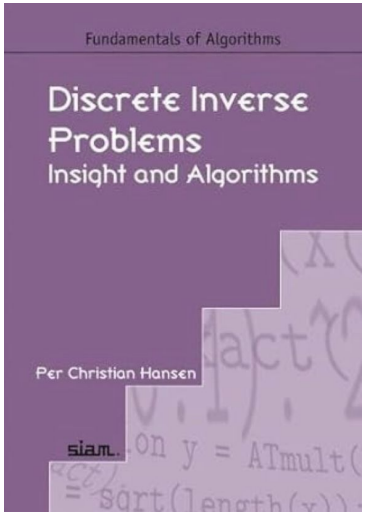


Summary

- Unfolding isn't so bad.
- Need a very precise understanding of the detector response.
- Unfolded distribution should not depend on the input data.
- Gives statistical estimators with expectation values and variances.

Where do I find more information and code?

- RooUnfold; https://gitlab.cern.ch/RooUnfold (RooUnfold and RooUnfoldML)
- Paper; Comparison of unfolding methods using RooFitUnfold. International Journal of Modern Physics A, Vol. 35, No. 24, 2050145 (2020) https://arxiv.org/abs/1910.14654
- Paper; Publishing unbinned differential cross section results. JINST 17 (2022) 01, P01024 https://arxiv.org/abs/2109.13243
- Paper; An algorithm for automatic unfolding of one-dimensional data distributions, Nuclear Instruments and Methods in Physics Research A 729 (2013) 410-416. PHYSTAT 2011
- Per Christian Hansen: "Discrete Inverse Problems: Insight and Algorithms"
- Lectures by Volker Blobel https://www.desy.de/~sschmitt/blobel/unfoldpaper.html



Issues or questions?

Email roounfold-support@cern.ch, vincent.croft@cern.ch

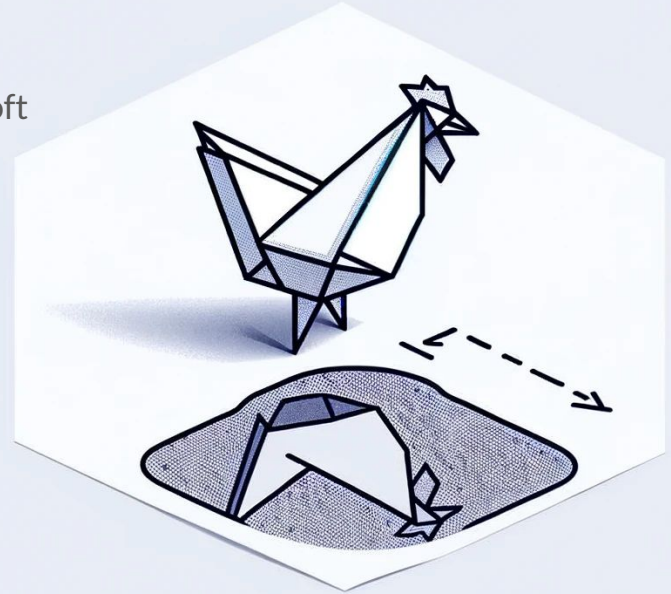
Thanks!

The whole RooUnfold team

Lydia Brenner, Tim Adye, Carsten Burgard, Vincent Croft

Study collaborators

Pim Verschuuren, Glen Cowan, Wouter Verkerke



Backup

Counting Experiments in HEP (Notation)

The 'true' histogram $\mu = (\mu_1, \ldots, \mu_M)$

Expectation values for observed histogram $\nu = (\nu_1, \ldots, \nu_N)$

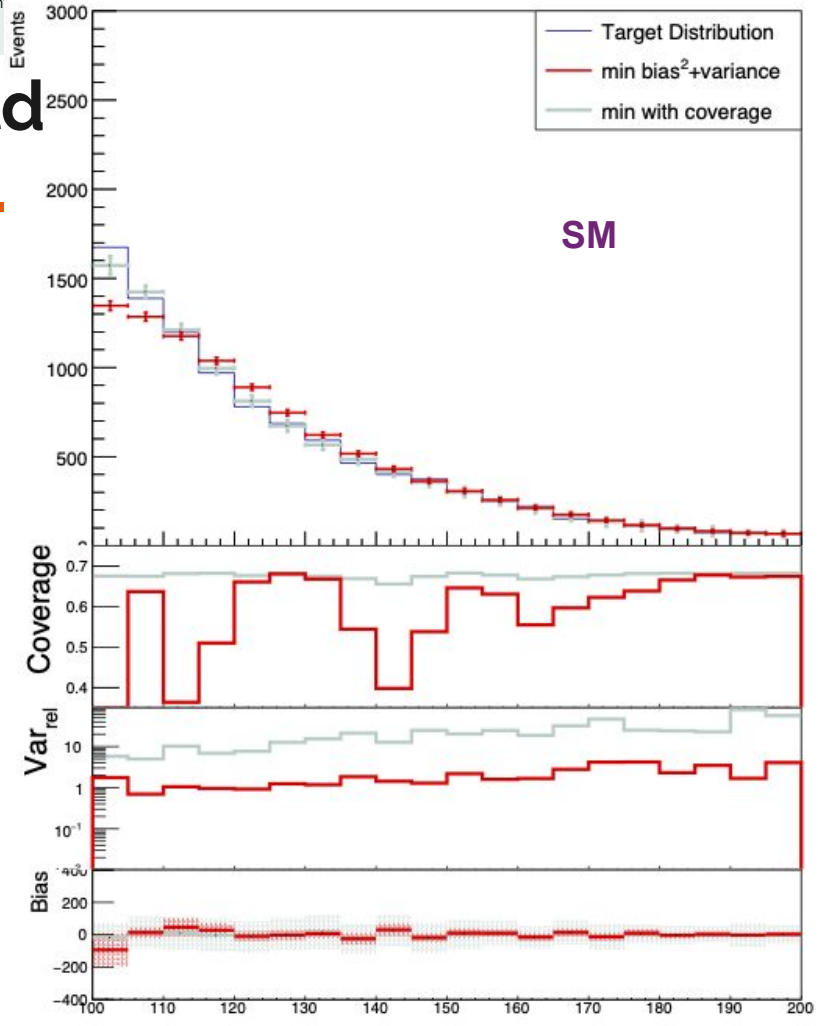
Observed histogram $n = (n_1, \ldots, n_N)$

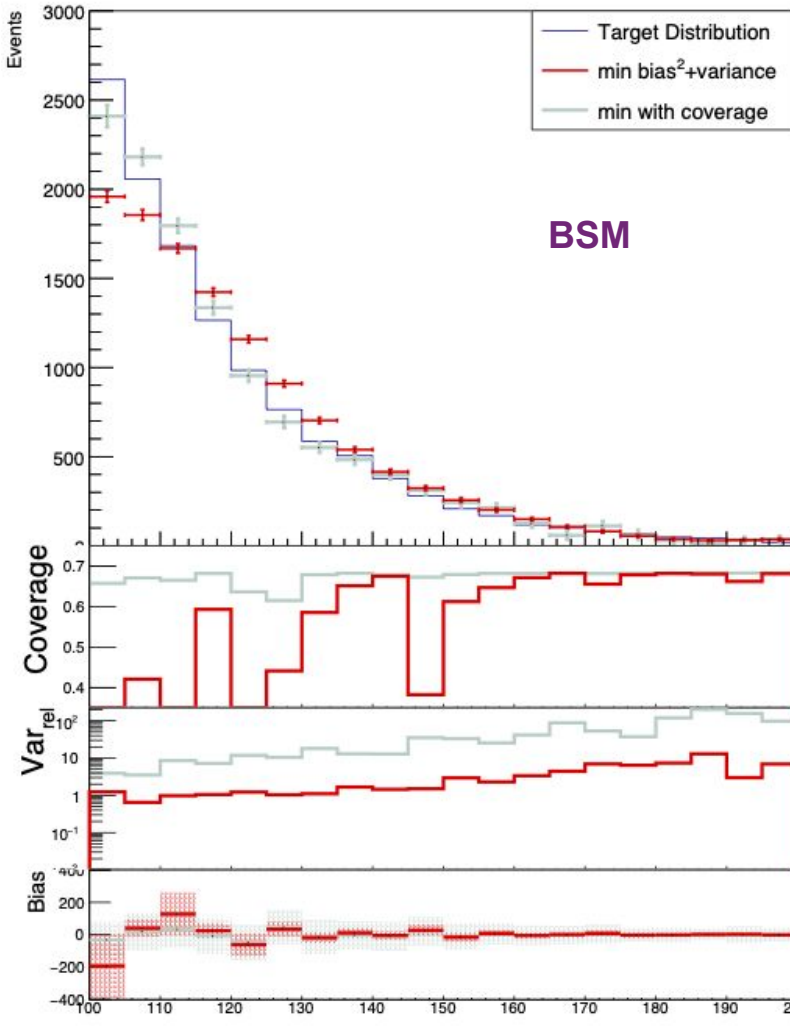
Expected backgrounc $\beta = (\beta_i, \ldots, \beta_N)$

Response matrix $R_{ij} = P(\text{observed in bin i} \,|\, \text{produced in bin j})$

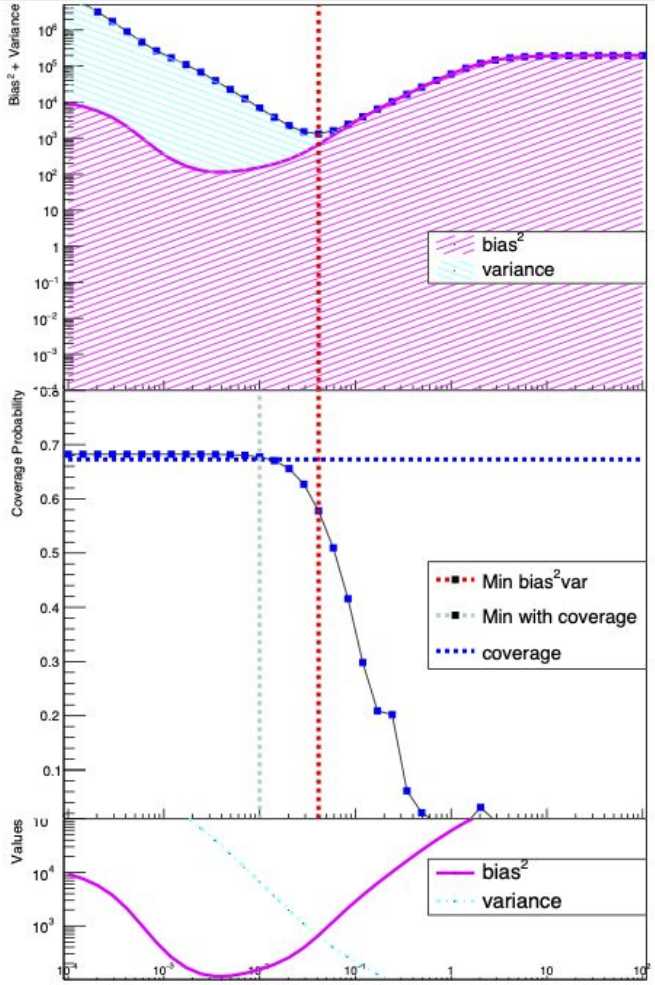
Related by $E[n] = \nu = R\,\mu + \beta$

TUnfold

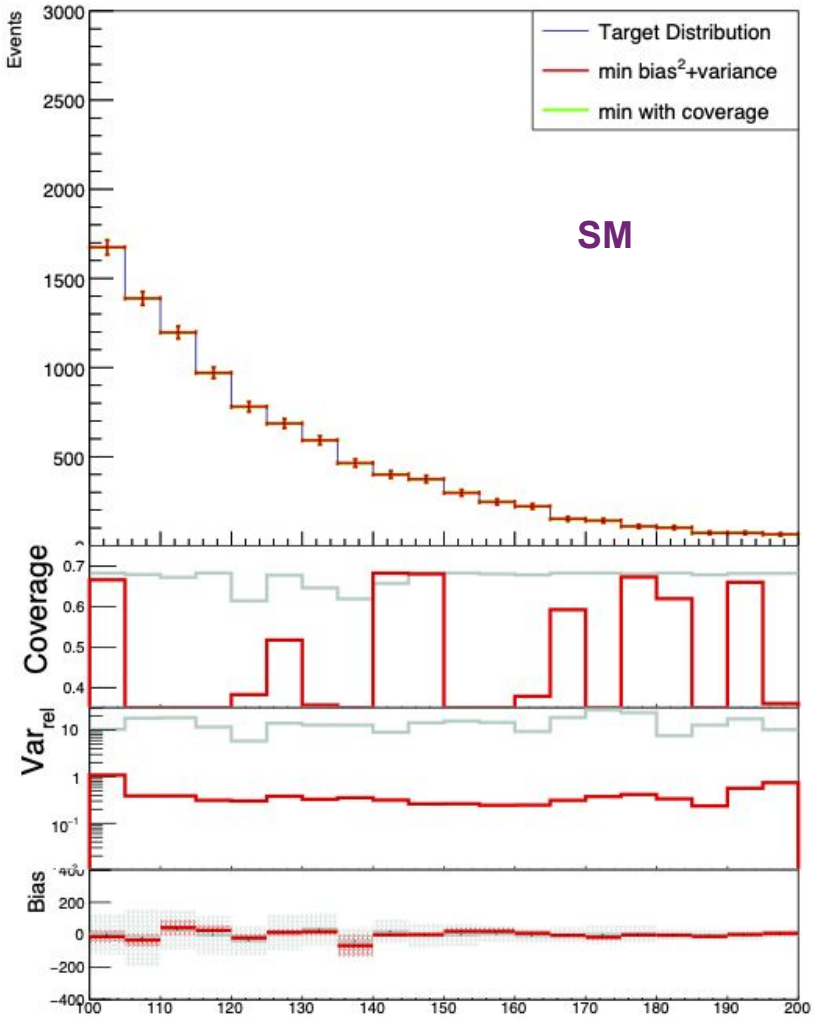

Events
Target Distribution
min bias²+variance
min with coverage
SM
Coverage
Var_rel
Bias

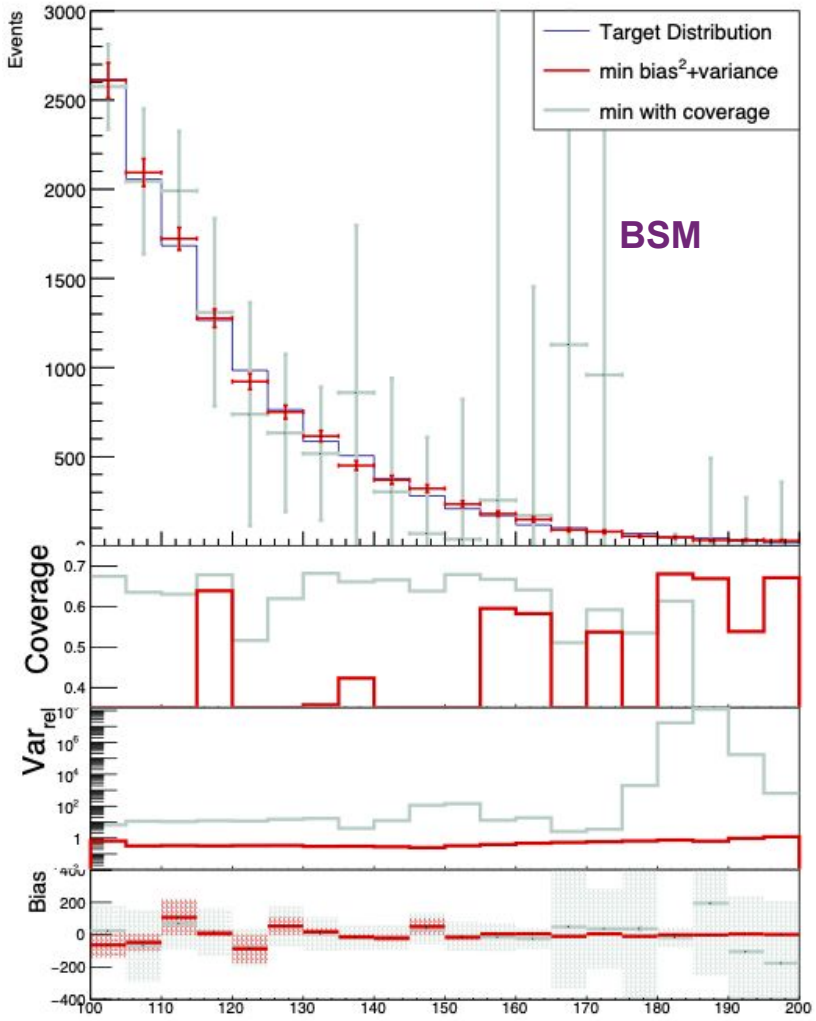

Events
Target Distribution
min bias²+variance
min with coverage
BSM
Coverage
Var_rel
Bias

TUnfold

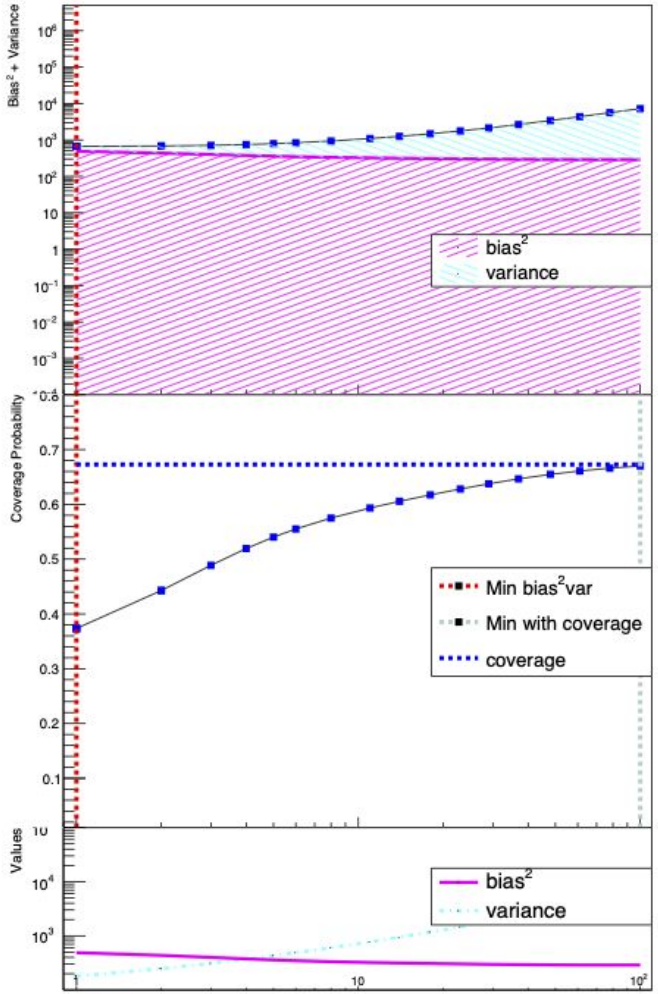

Bias² + Variance
bias²
variance
Coverage Probability
Min bias²var
Min with coverage
coverage
Values
bias²
variance

IBS


Events
Target Distribution
min bias²+variance
min with coverage
SM
Coverage
Var_rel
Bias


Events
Target Distribution
min bias²+variance
min with coverage
BSM
Coverage
Var_rel
Bias

IBS


Bias² + Variance
bias²
variance
Coverage Probability
Min bias²var
Min with coverage
coverage
Values
bias²
variance

A different study: Bimodal distribution and biases

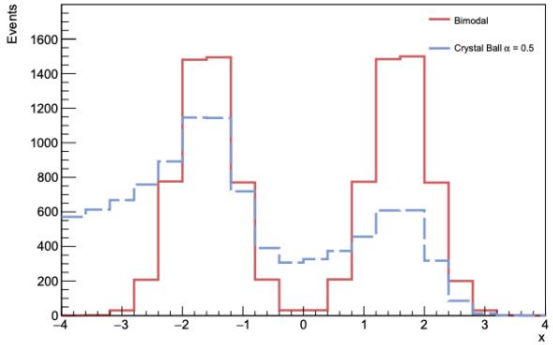
Look deeper into situations where the response matrix and the data are not sampled from the same model.

The bimodal model for this study is the sum of two Crystal Ball functions smeared by a a Gaussian resolution model

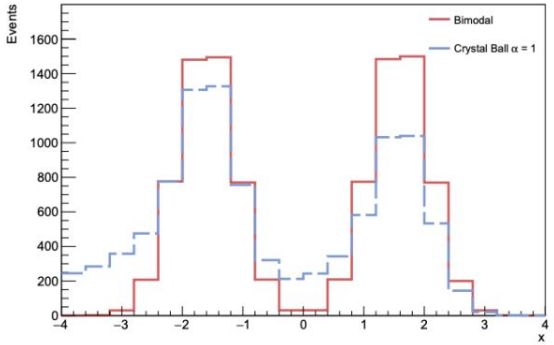
$$
\begin{aligned}
f(x|\alpha) &= f_{\text{physics}}(x_{\text{true}}|\alpha) * f_{\text{detector}}(x_{\text{true}}, x) \\
&= (0.5 \cdot f_{CB}(x_{\text{true}}|\mu = 2.4, \sigma = 0.48, \alpha, n = 1) + \\
&\quad 0.5 \cdot f_{CB}(x_{\text{true}}|\mu = 5.6, \sigma = 0.48, \alpha, n = 1)) \\
&\quad * Gauss(x - x_{\text{true}}, 0, 0.4)
\end{aligned}
$$

A different study: Bimodal distribution and biases

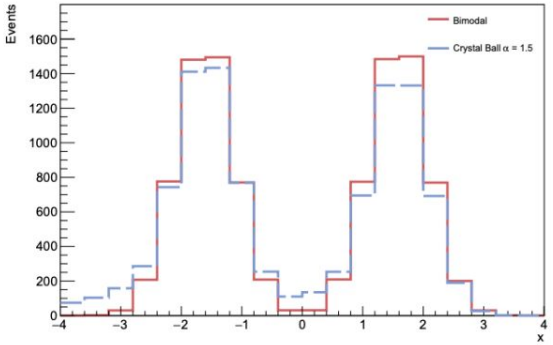
$\alpha = 0.5$



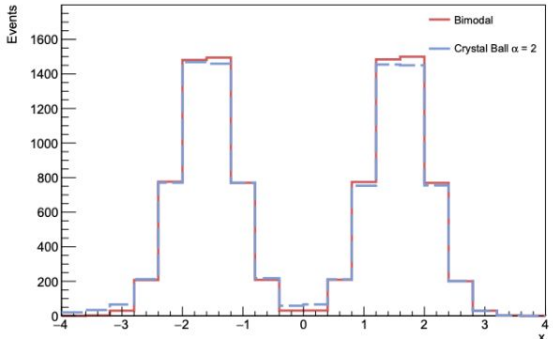
$\alpha = 1$



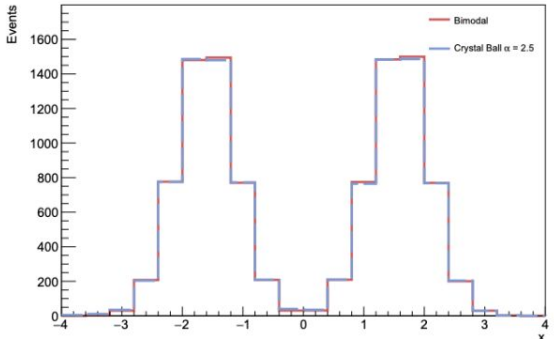
$\alpha = 1.5$



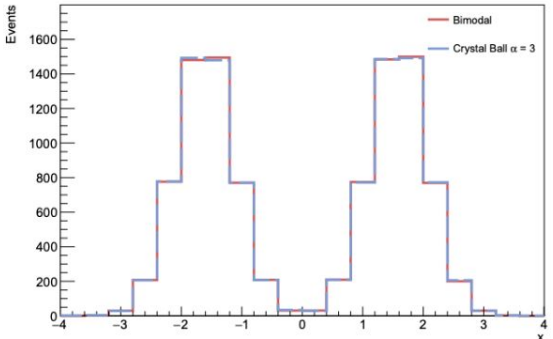
$\alpha = 2.0$



$\alpha = 2.5$

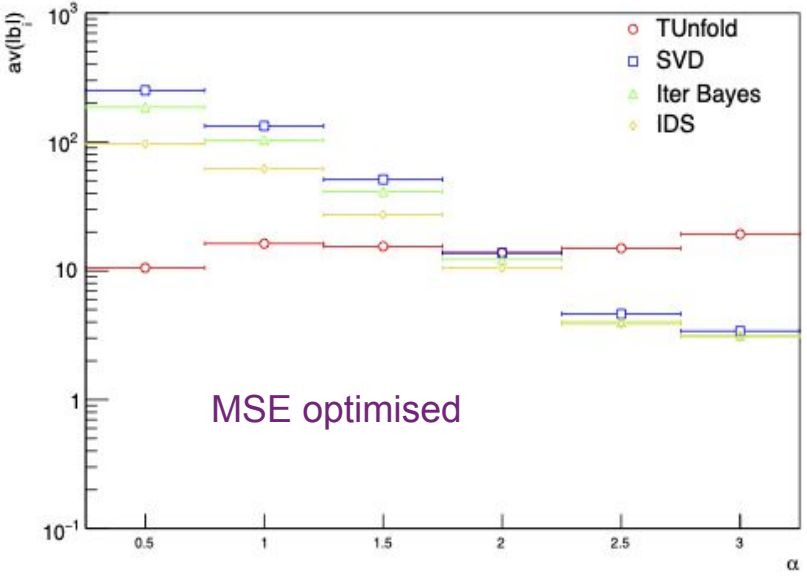


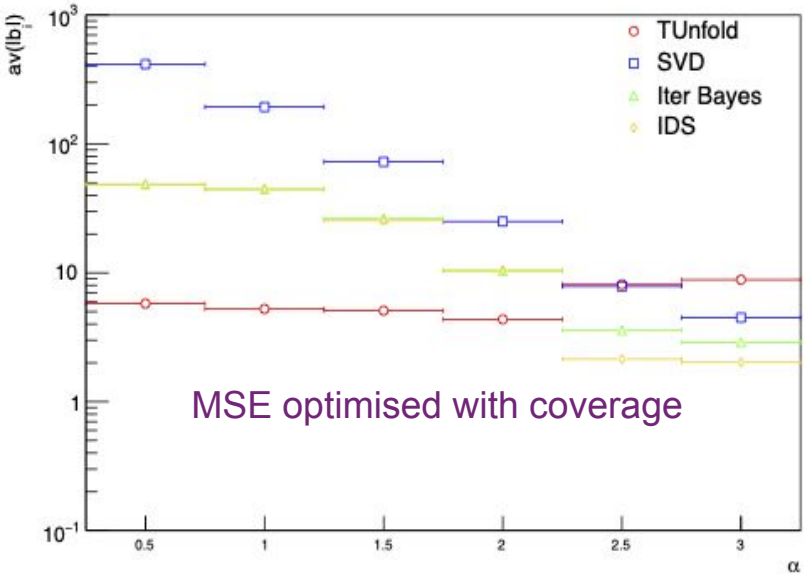
$\alpha = 3.0$



A different study: Bimodal distribution and biases

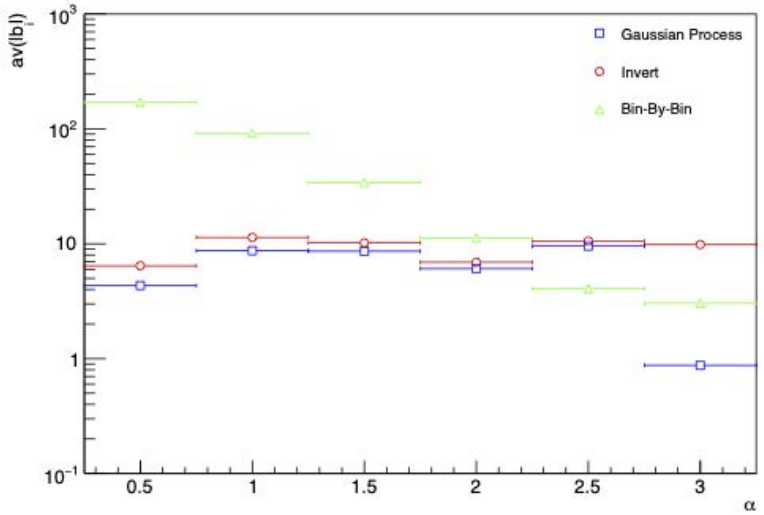
Bin-averaged unfolding bias for data from the distorted distribution unfolded with a response matrix for an undistorted distribution.





A different study: Bimodal distribution and biases

Bin-averaged unfolding bias for data from the distorted distribution unfolded with a response matrix for an undistorted distribution.



RooUnfold update

- Many frameworks / implementations for unfolding exist
  - Most use RooUnfold as a backend internally
  - Main focus of many of these frameworks: uncertainty handling
- Updates to RooUnfold itself
  - Integration with RooFit
    - Easier uncertainty handling
  - Workspace handling
    - Easy for combination
- Make RooUnfold “future proof”
  - Ready for possible unbinned unfolding methods in the future
  - Improved user friendliness
- End product
  - Saved in a way to allow changing of method at later time

Unfolding and fitting

- Unfold taking systematic variations into account
  - Also done by other advanced frameworks
- Can unfold after fitting signal and background contributions
  - Need to bring fit results into a format suitable as unfolding input
  - Non-trivial to propagate uncertainties
- Why not do both at the same time?
  - Ideal solution: RooFit implementation of unfolding!

Introduction to RooFitUnfold

- Idea: Updated implementation of RooUnfold directly in RooFit
- Includes: Improved handling of uncertainties
  - Uses error propagation from any NPs to the unfolded distribution
  - Allows for inclusion of uncertainties coming from migration matrix
- Handels different input formats
  - Histograms (as already RooUnfold did)
  - pdfs -> Means unbinned distributions can now be unfolded
    - Binned methods allow setting of internal binning
    - unbinned methods can technically be included in the future
- Lives in workspaces

Methods included

All RooUnfold methods included

- Iterative Bayes
- IDS
- SVD
- TUnfold
- Gaussian Processes unfolding (NEW)
- Poisson unfolding, a simple likelihood unfolding (NEW)
- Unregularised
  - Bin-by-bin
  - Matrix inversion
- Can easily include more methods

Documentation:

https://gitlab.cern.ch/roofitunfold-tutorial-2019/RooUnfold/blob/master/README.md

https://arxiv.org/pdf/1105.1160.pdf

Implementation

- RooUnfold uses TH1 objects as basis
  - Very user-friendly, but internally not ideal with RooFit
- Templated to use RooAbsReal as a base object
  - Can easily be plugged on top of an existing workspace
- Created RooUnfoldFunc
  - a RooAbsReal wrapper around RooUnfold

Implementation: Inputs

- Truth distristributions
  - Histograms (TH1) or pdf (RooFit/Workspace)
- Reco distributions
  - Histograms (TH1) or pdf (RooFit/Workspace)
- Response matrix
  - 2D Histogram (TH2) or pdf (RooFit/Workspace)
- Data: background subtracted if needed
  - binned (TH1 or RooDataHist) or unbinned (TTree or RooDataSet)

Implementation: Features

- RooUnfoldFunc can be imported into workspace
  - Can use any existing workspace as an input
  - Can update reco level workspace after unfolding
  - Can unfold and fit (on reco-level) simultaneously
  - Easy persistence
  - Extremely useful for combinations
- RooUnfoldSpec can be used to construct RooUnfoldFunc
  - Helper class similar to HistFactory
- Unfolding result is only cached
  - Can switch to a different unfolding method a-posteriori

Workspace write out

- Directly written out into a workspace
  - At any level of the analysis
  - Saves all information to be able to do a change of unfolding method on the fly
  - Includes error propagation
  - Writes out for ALL unfolding methods
    - So also for regularised methods

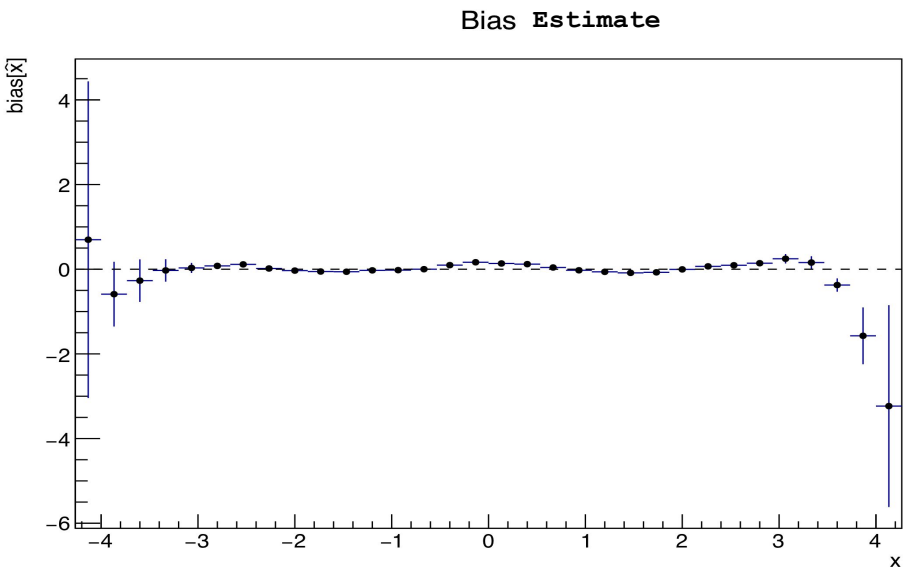
Error propagation

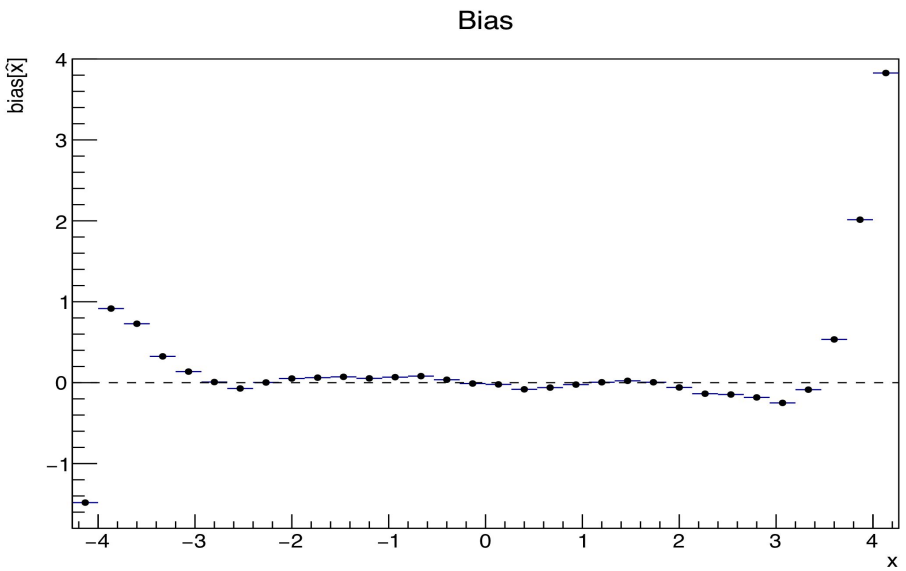
- Default RooUnfold can propagate simple uncertainties
  - Statistical uncertainties on Data
  - Bin-by-bin correlations
  - No handling of systematic uncertainties!
- RooFit functions (pdfs) can depend on arbitrarily many parameters
  - automatic error propagation from input parameters to all outputs by RooFit
  - ony requirement: the output needs to be a RooFit object
  - Nuisance parameter treatment comes “for free” with RooUnfold integration in RooFit
- No explicit handling of systematic uncertainties needed in RooUnfold
  - RooUnfold+RooFit handles uncertainties neatly :)
  - Some toy sampling methods required for bias calculation, but error bands on plots come directly from RooFit

Bias

Two bias calculations included

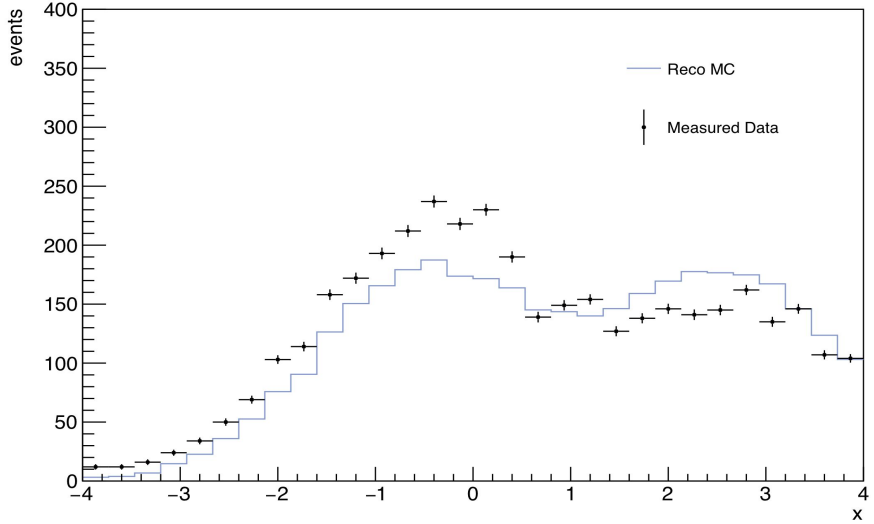
Bias estimate without toys and a full bias calculation

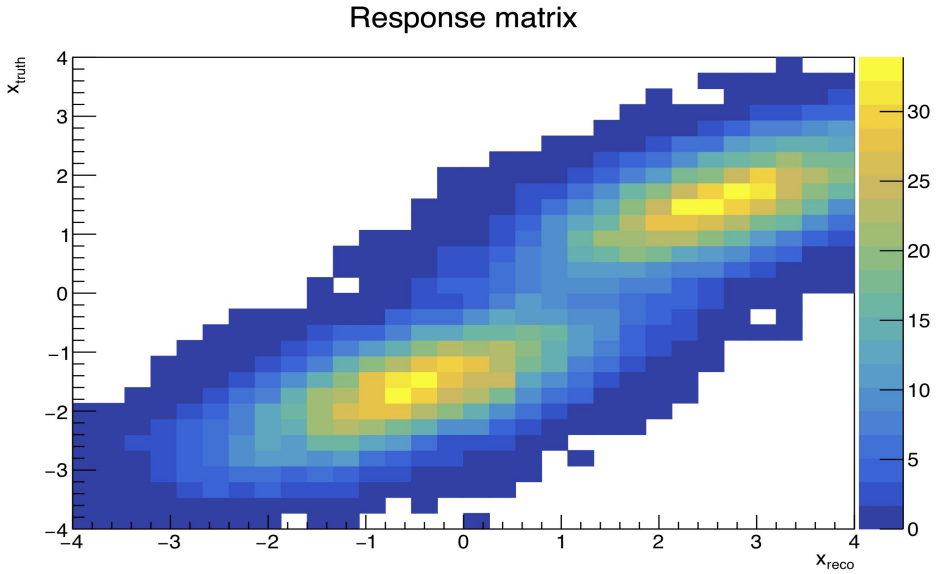




Example: Bimodal distribution

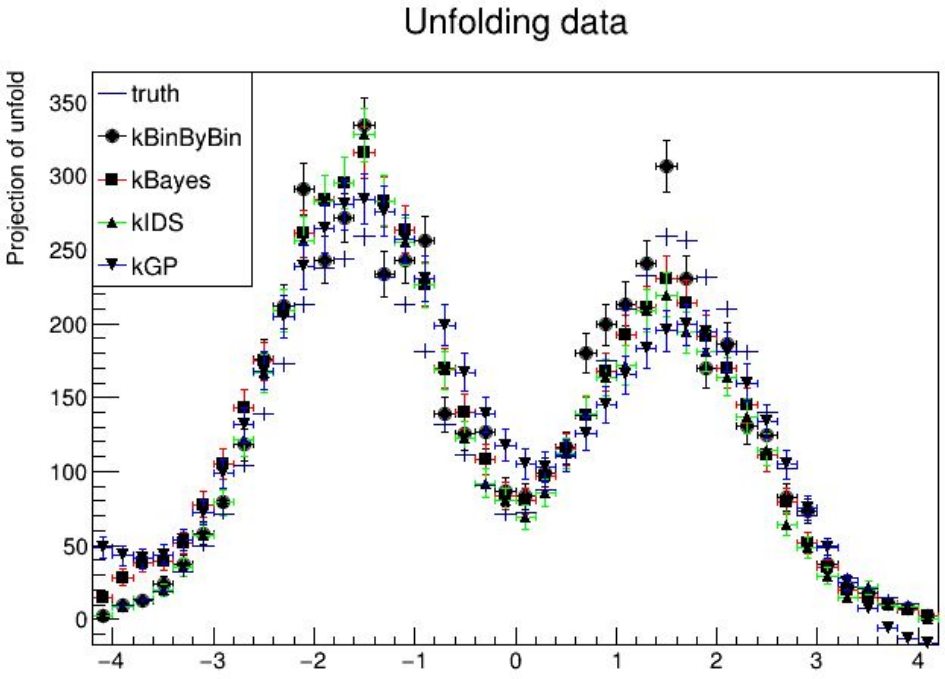
Reconstruction level plots





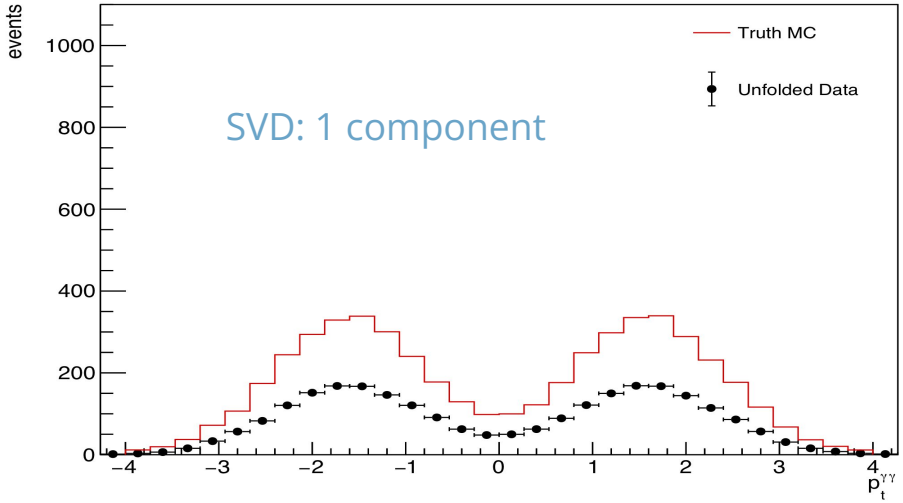
Example: Bimodal distribution

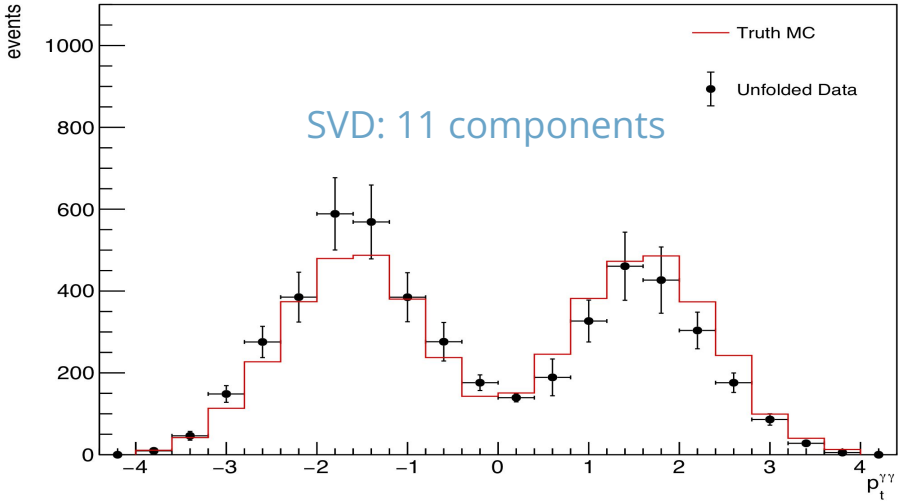
Compare different unfolding methods



Example: Bimodal distribution

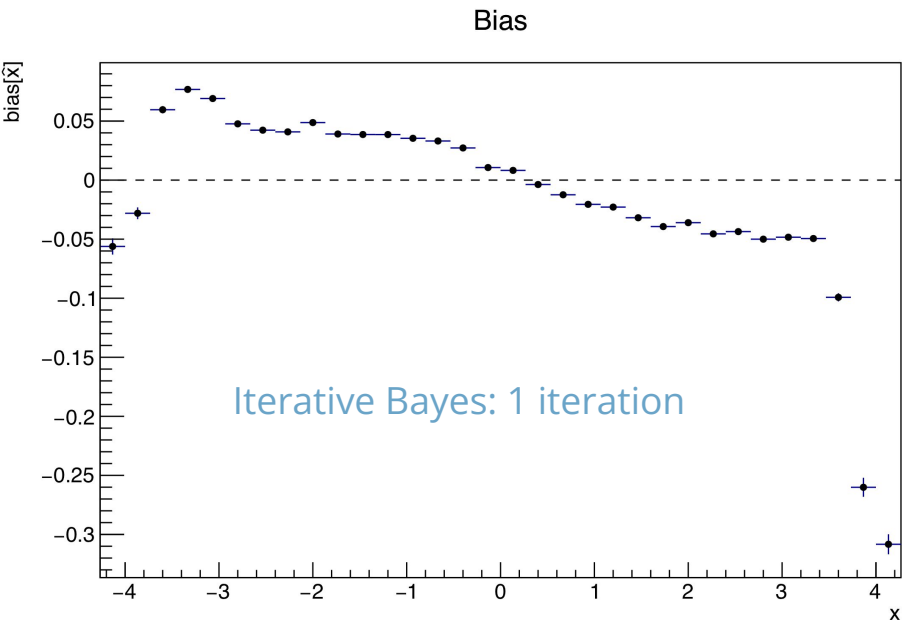
Compare different regularisation strengths

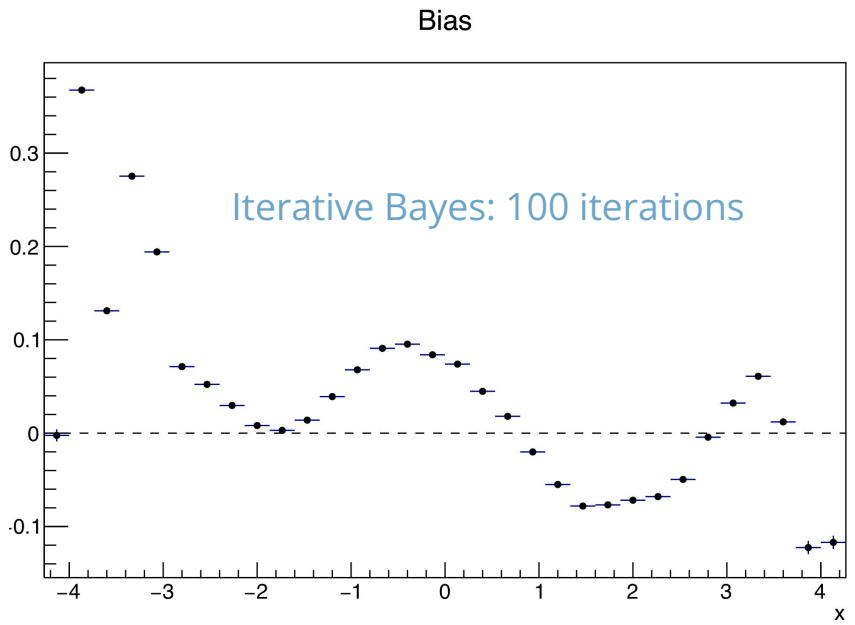




Example: Bimodal distribution

Compare different regularisation strengths: Don't forget the bias!





Bias calculation

(Asimov) data-driven

First the uncertainties are taken truth level from the unfolded Asimov dataset. Toys are thrown in each bin around the Asimov truth values based on the full uncertainty. These toys are called level 1 toys. For each level 1 toy further toys are thrown, called level 2 toys. Each of the level 2 toys is folded and then unfolded with the chosen unfolding method. The bias for each level 2 toy is calculated as

$$\text{bias}_{l2} = (\sigma_{\text{refold}} - \sigma_{\text{truth}})/\sigma_{\text{truth}},$$

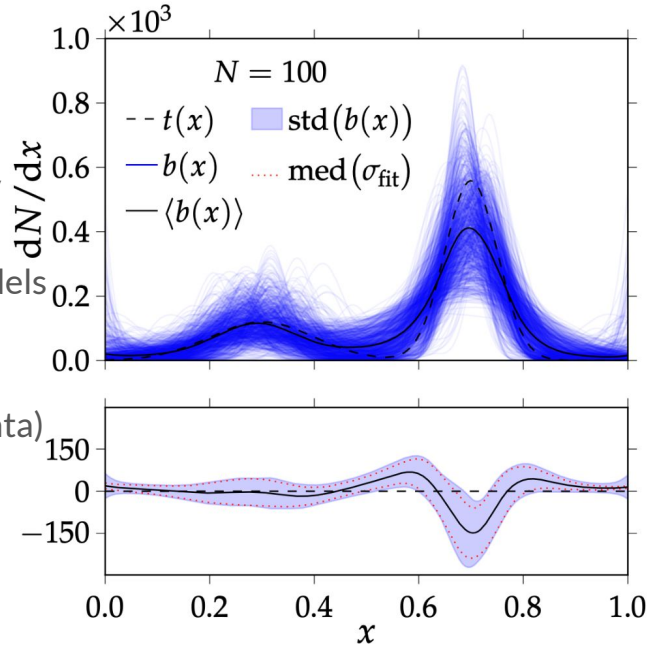
where the truth refers to the value of the level 1 toy at truth level from which the level 2 toy is thrown and refold refers to the value of the level 2 toy after folding and unfolding. The bias of each bin in the distribution that is being unfolding is the average over all bias_{l2}.

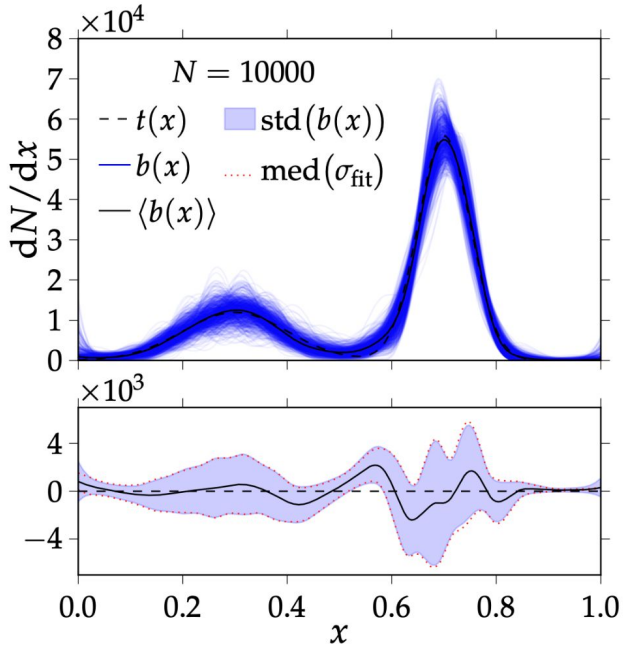
Function based unfolding

Goal: Train model to produce a function to approximate P(truth), evaluate conditional on data: P(truth|data)

Three approaches:

- Fit a non-parametric density estimator.
- GAN/VAE learn implicit models for $\hat{P}$
- NF/DDIMs learn representation of P(truth|data)

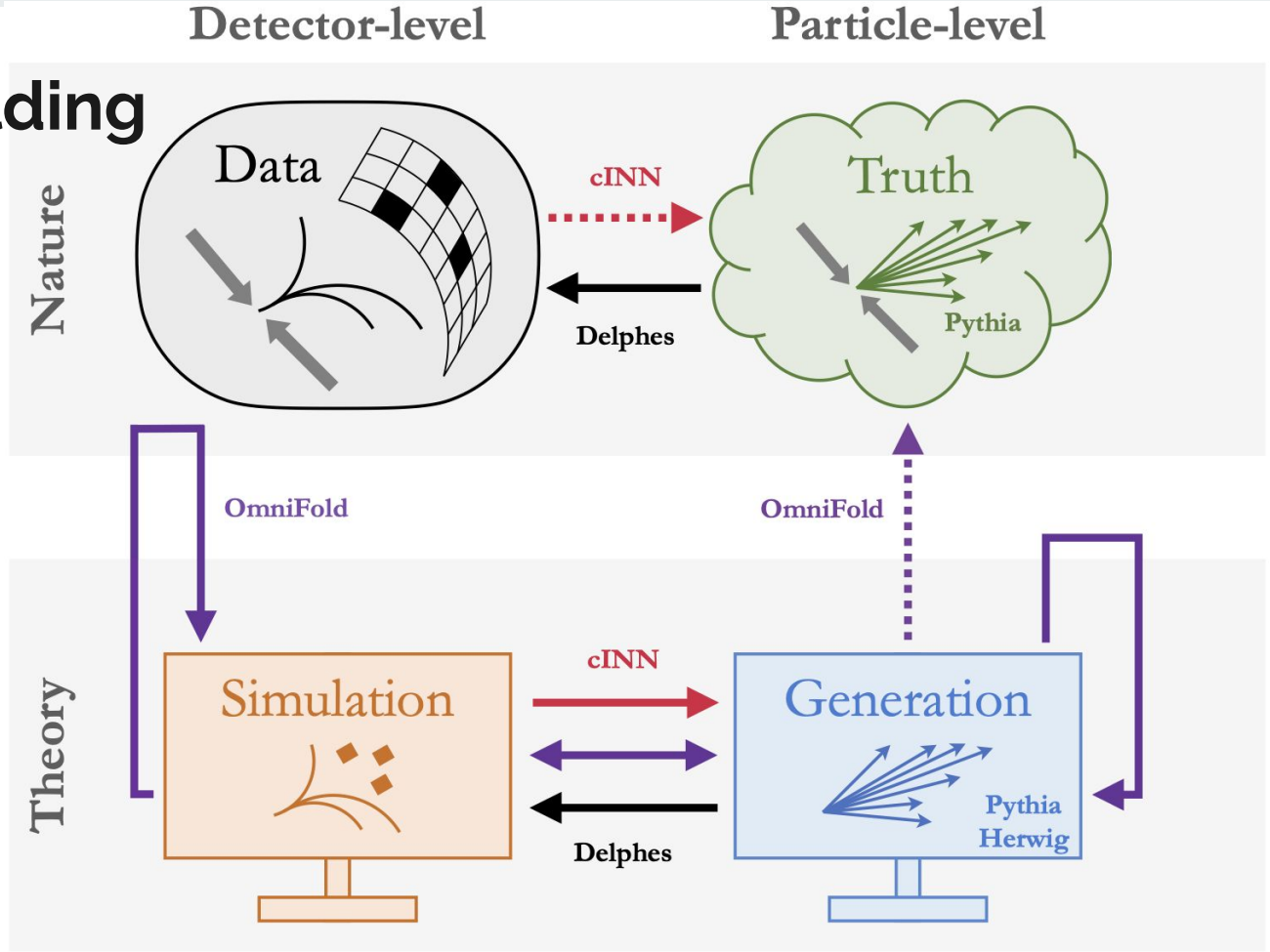




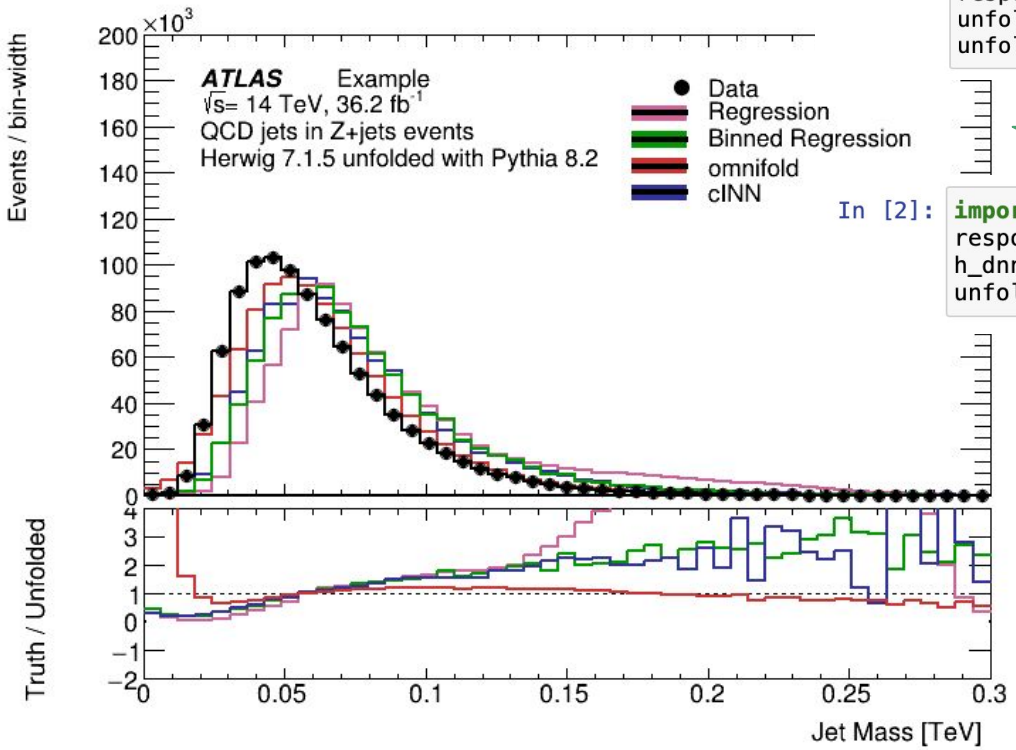
Classifier unfolding

Goal: learn a function $\hat{\omega}$ that approximates the likelihood ratio

$$\omega(x) = \frac{p(x)}{p_{\mathrm{MC}}(x)}$$



RooUnfoldML - https://gitlab.cern.ch/RooUnfold/RooUnfoldML/




```
In [1]: import ROOT
response = ROOT.RooUnfoldResponse(reco,truth,migration)
unfold = ROOT.RooUnfoldBayes(response, data)
unfolded_hist = unfold.Hunfold()
```

```
In [2]: import RooUnfoldML
response = RooUnfoldML.RooUnfoldMLResponse(reco,truth)
h_dnn = RooUnfoldML.RooUnfoldBinnedRegression(response,data)
unfolded_hist = h_dnn.Hunfold()
```

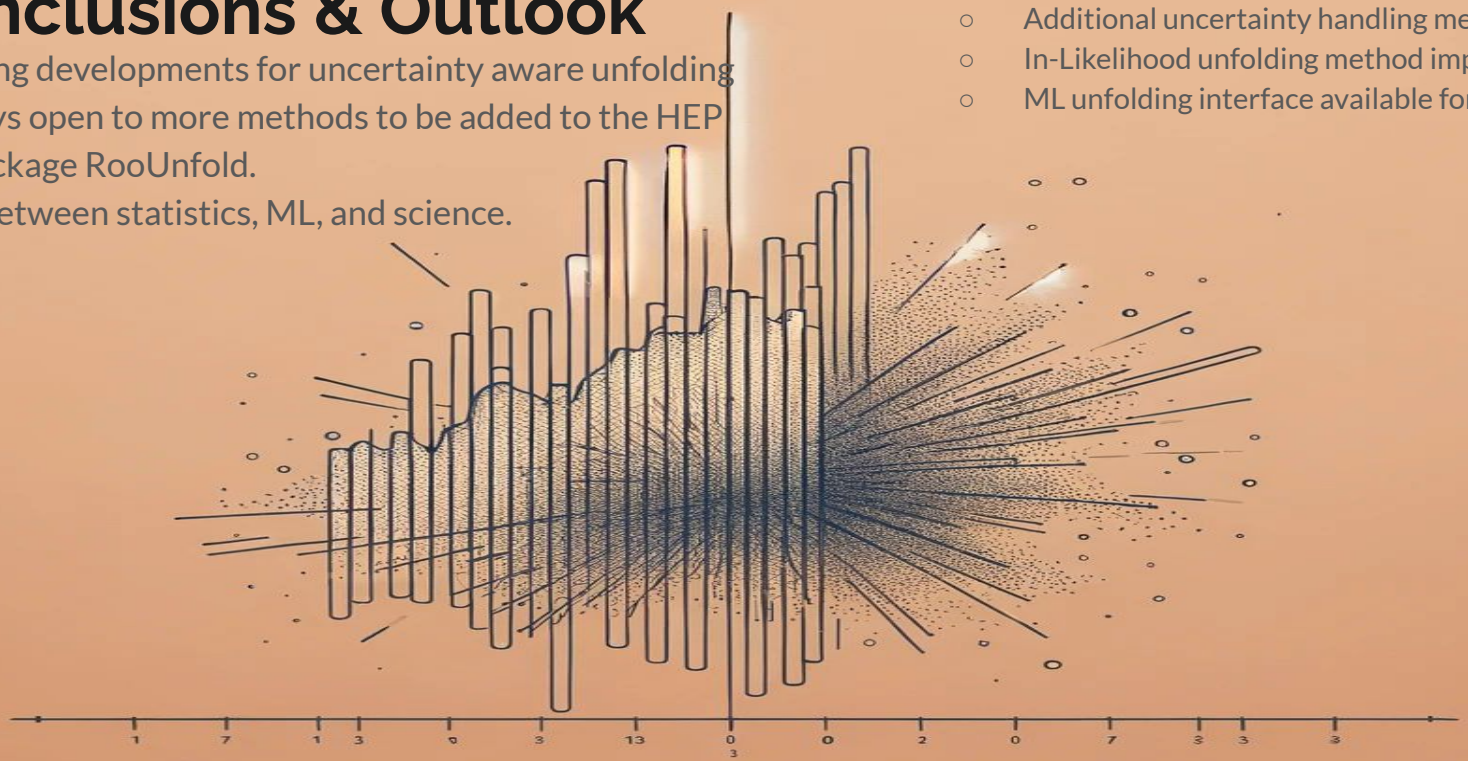
Conclusions & Outlook

Lots of exciting developments for uncertainty aware unfolding

We are always open to more methods to be added to the HEP unfolding package RooUnfold.

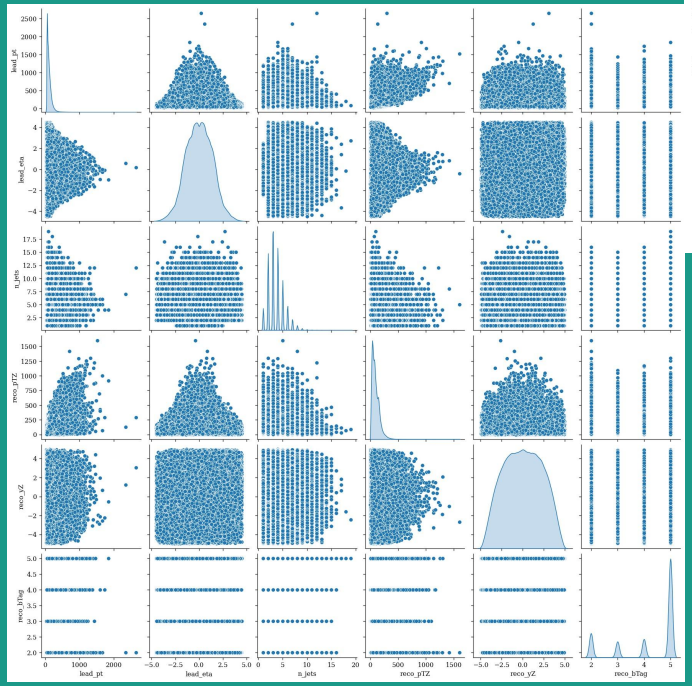
A deep link between statistics, ML, and science.

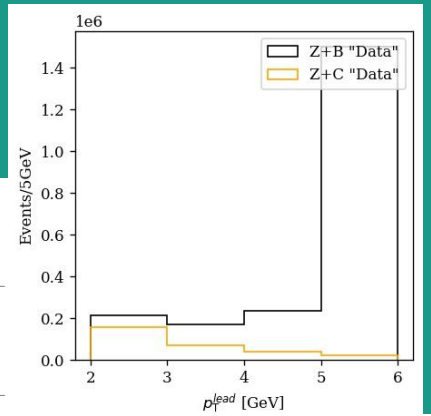
- If you have suggestions, please let us know
- Wide-bin and RUN unfolding methods incoming
- Additional uncertainty handling methods
- In-Likelihood unfolding method implemented
- ML unfolding interface available for development.

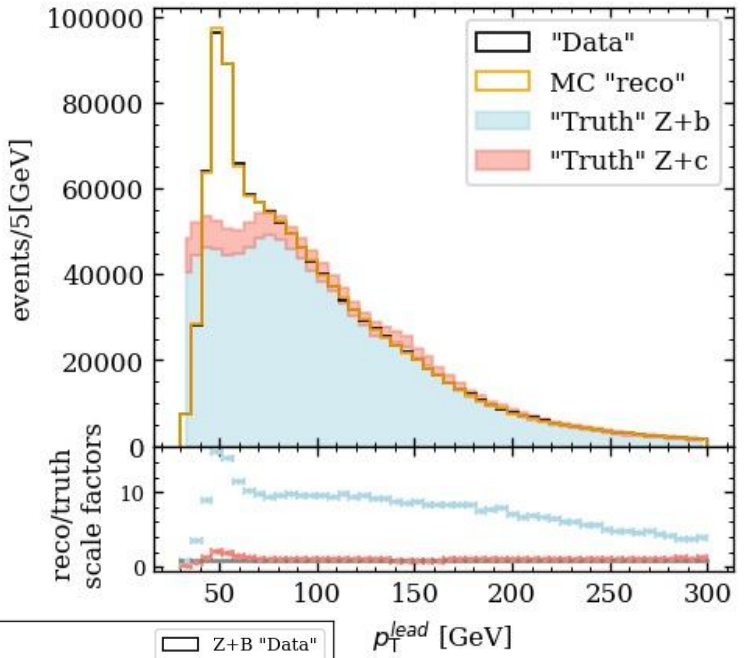


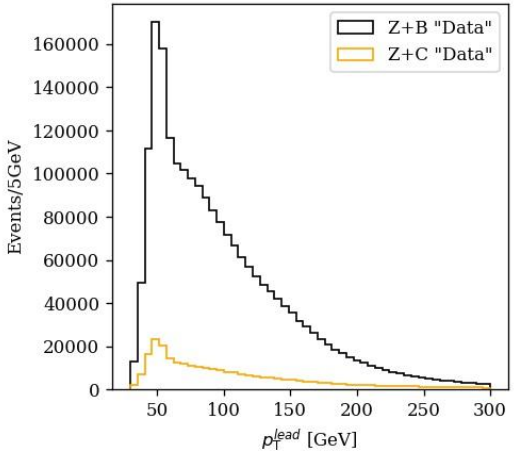
Problem has lots of correlated

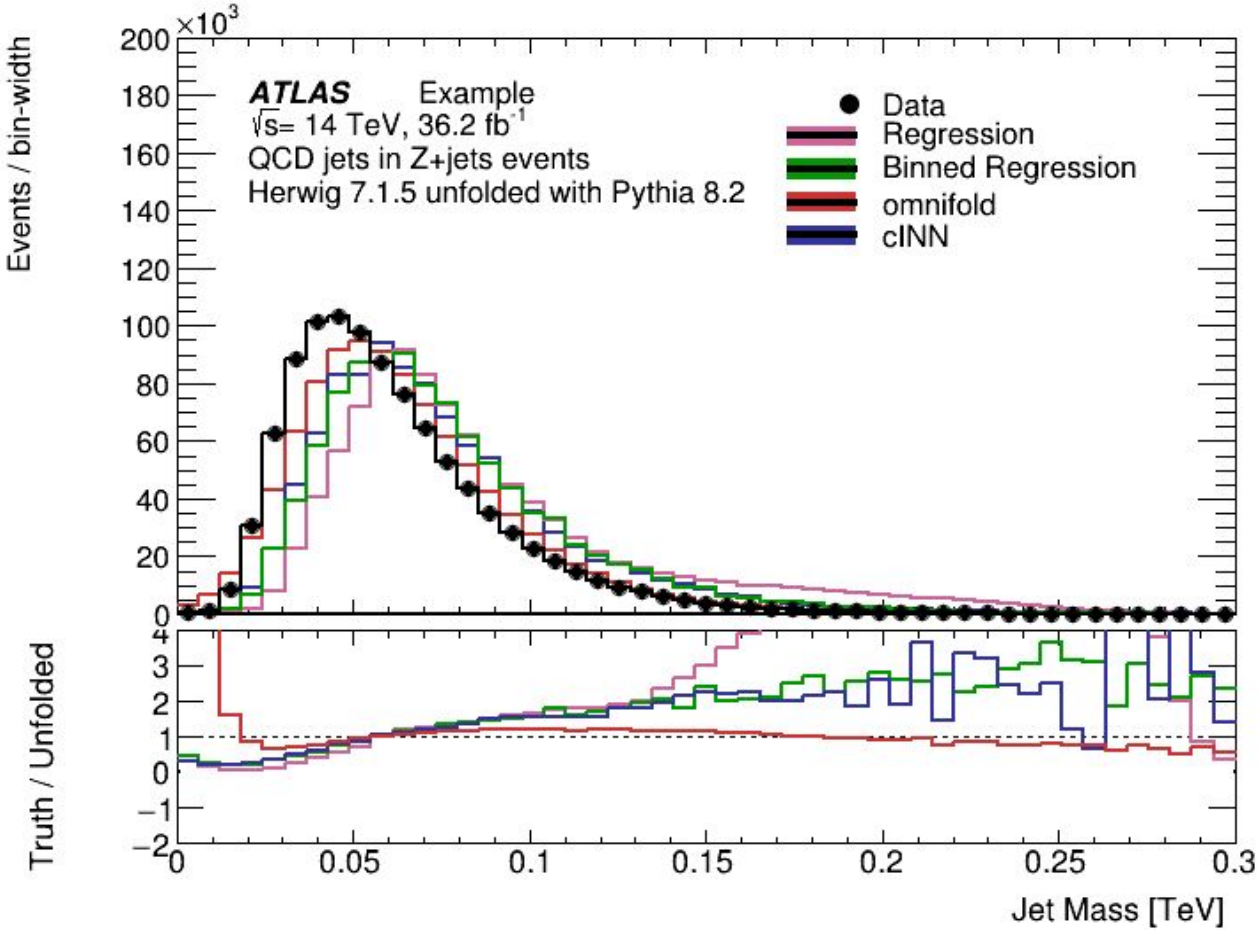
Information - Why not include it?











Lots of possibilities!

Common Interface!

```
In [1]: import ROOT
        response = ROOT.RooUnfoldResponse(reco,truth,migration)
        unfold = ROOT.RooUnfoldBayes(response, data)
        unfolded_hist = unfold.Hunfold()
```

```
In [2]: import RooUnfoldML
        response = RooUnfoldML.RooUnfoldMLResponse(reco,truth)
        h_dnn = RooUnfoldML.RooUnfoldBinnedRegression(response,data)
        unfolded_hist = h_dnn.Hunfold()
```

Easy to use

- Will Ship as a lightweight extension to RooUnfold
- Optional extra flag on compilation
- Minimal Dependencies
- Meaningful default settings

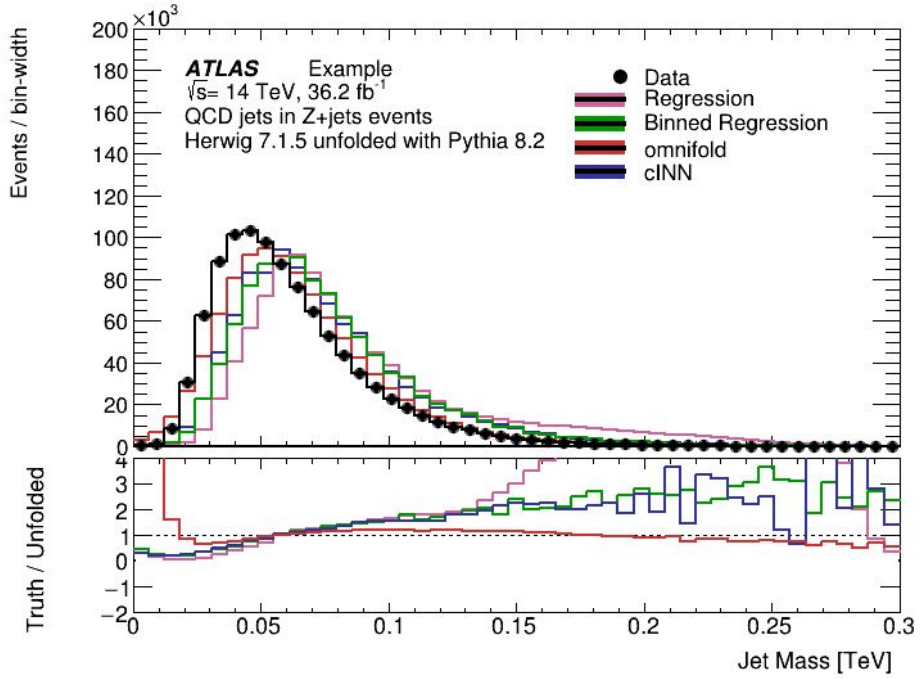
Common Interface!

```
r = roounfoldml.RooUnfoldMLResponse(simobs, genobs)
cINN = roounfoldml.RooUnfoldCondition(r, dataobs)
cINN_data = cINN.Vunfold()
h_cINN = cINN.Hunfold("Mass",bin_edges)
```

```
omnifold = roounfoldml.RooUnfoldReweight(r, dataobs)
omnifold_data,omnifold_weights = omnifold.Vunfold()
h_omni = omnifold.Hunfold("Mass",bin_edges)
```

```
reg = roounfoldml.RooUnfoldRegression(r, dataobs)
reg_data = reg.Vunfold()
h_reg = reg.Hunfold("Mass",bin_edges)
```

- Will Ship as a lightweight extension
- Optional extra flag on compilation
- Minimal Dependencies
- Meaningful default settings



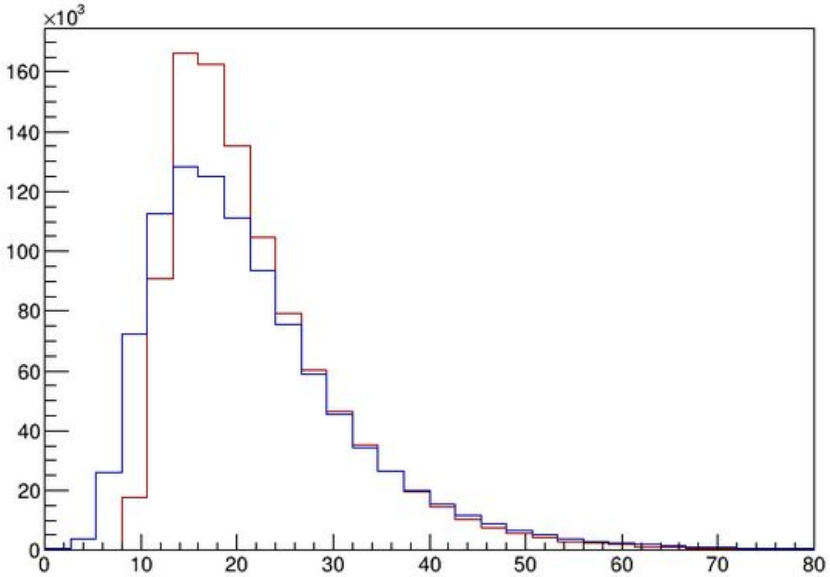
Root Interface

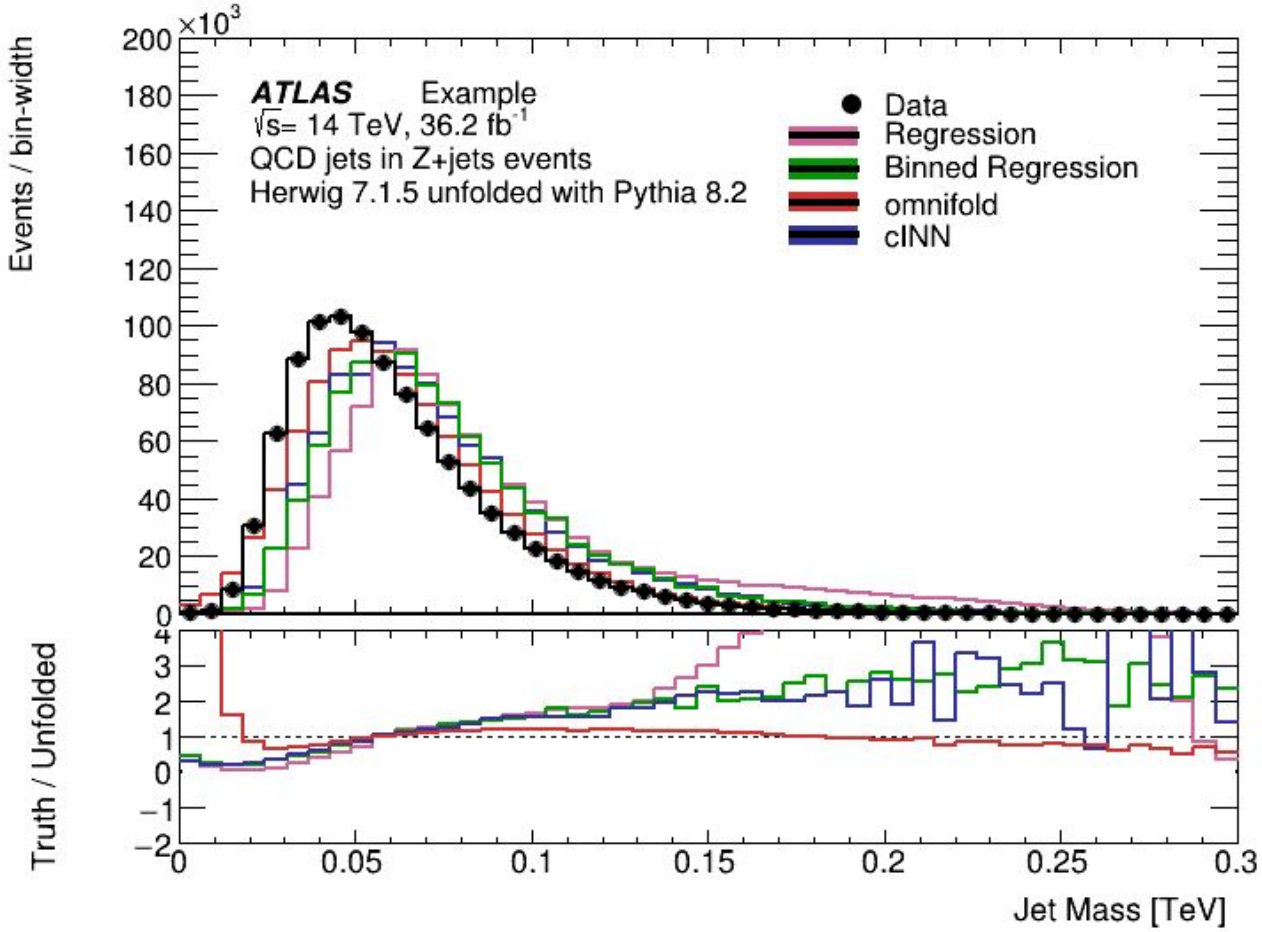
- Standard input is pandas dataframes
- Automatic conversion for:
  - TTree
  - RDataFrame
- Some internal conversions to tf.data.Datasets
- Might be inefficient.
- Easy integration with existing frameworks

```
dataobs = ROOT.RDF.MakeCsvDataFrame("/home/vcroft/dataobs.csv")
simobs = ROOT.RDF.MakeCsvDataFrame("/home/vcroft/simobs.csv")
genobs = ROOT.RDF.MakeCsvDataFrame("/home/vcroft/genobs.csv")
truthobs = ROOT.RDF.MakeCsvDataFrame("/home/vcroft/truthobs.csv")
```

```
r = roounfoldml.RooUnfoldMLResponse(simobs, genobs)
u = roounfoldml.RooUnfoldRegression(r, dataobs)
```

```
data_hist = truthobs.Histo1D(("","",30,0,80), 'Mass')
c = ROOT.TCanvas()
h = u.Hunfold("Mass",30,0,80)
h.SetStats(0)
h.SetLineColor(ROOT.kRed+2)
h.Draw()
data_hist.Draw("same")
c.Draw()
```





Lots of possibilities!