

SymbolFit: Automatic parametric modeling with symbolic regression

Ho Fung Tsoi

Apr 30, 2025

[arxiv:2411.09851](https://arxiv.org/abs/2411.09851)

[github:hftsoi/symbolfit](https://github.com/hftsoi/symbolfit)

symbolfit.readthedocs.io

SymbolFit: Automatic Parametric Modeling with Symbolic Regression

Ho Fung Tsoi¹, Dylan Rankin¹, Cecile Caillol², Miles Cranmer³, Sridhara Dasu⁴,
Javier Duarte⁵, Philip Harris^{6,7}, Elliot Lipeles¹, Vladimir Loncar^{6,8}

¹University of Pennsylvania, USA

²European Organization for Nuclear Research (CERN), Switzerland

³University of Cambridge, UK

⁴University of Wisconsin-Madison, USA

⁵University of California San Diego, USA

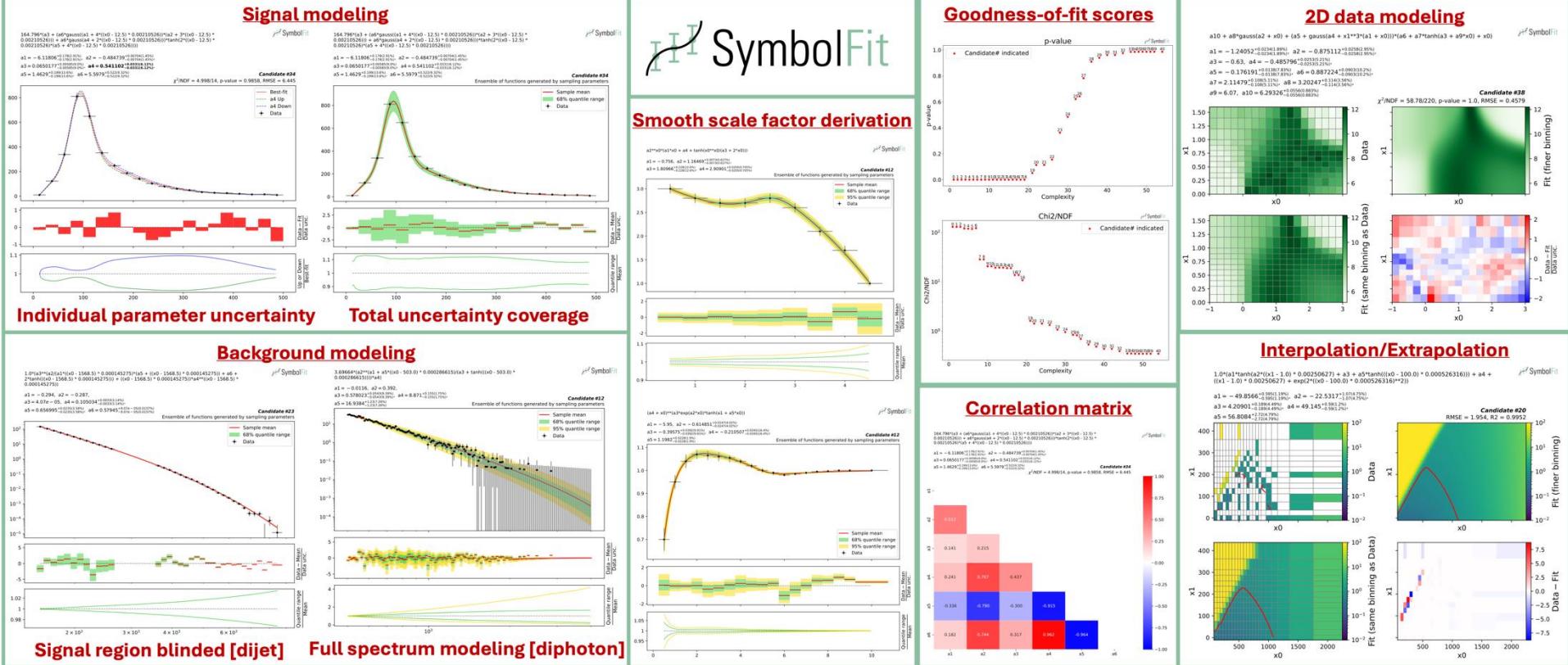
⁶Massachusetts Institute of Technology, USA

⁷Institute for Artificial Intelligence and Fundamental Interactions, USA

⁸Institute of Physics Belgrade, Serbia

E-mail: ho.fung.tsoi@cern.ch

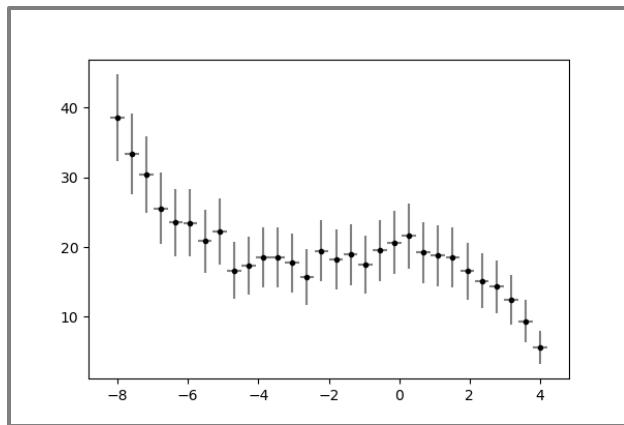
Abstract. We introduce **SymbolFit**, a framework that automates parametric modeling by using symbolic regression to perform a machine-search for functions that fit the data while simultaneously providing uncertainty estimates in a single run. Traditionally, constructing a parametric model to accurately describe binned data has been a manual and iterative process, requiring an adequate functional form to be determined before the fit can be performed. The main challenge arises when the appropriate functional forms cannot be derived from first principles, especially when there is no underlying true closed-form function for the distribution. In this work, we develop a framework that automates and streamlines the process by utilizing symbolic regression, a machine learning technique that explores a vast space of candidate functions without requiring a predefined functional form because the functional form itself is treated as a trainable parameter, making the process far more efficient and effortless than traditional regression methods. We demonstrate the framework in high-energy physics experiments at the CERN Large Hadron Collider (LHC) using five real proton-proton collision datasets from new physics searches, including background modeling in resonance searches for high-mass dijet, trijet, paired-dijet, diphoton, and dimuon events. We show that our framework can flexibly and efficiently generate a wide range of candidate functions that fit a nontrivial distribution well using a simple fit configuration that varies only by random seed, and that the same fit configuration, which defines a vast function space, can also be applied to distributions of different shapes, whereas achieving a comparable result with traditional methods would have required extensive manual effort.



Overview

- Problem statement
- Symbolic regression
- SymbolFit API
- Examples

Problem statement

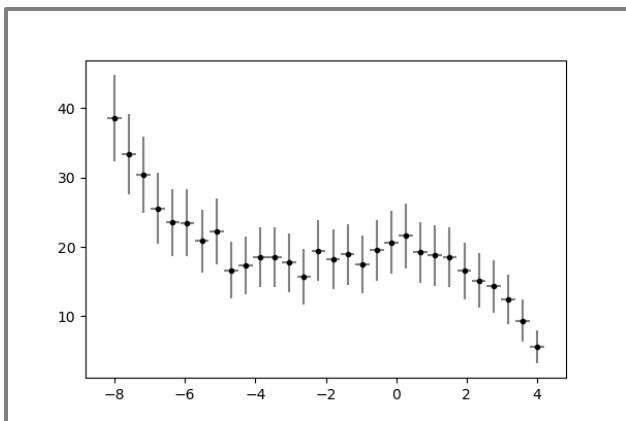


Given a distribution, how to derive a function to smoothly model it?

Applications in HEP

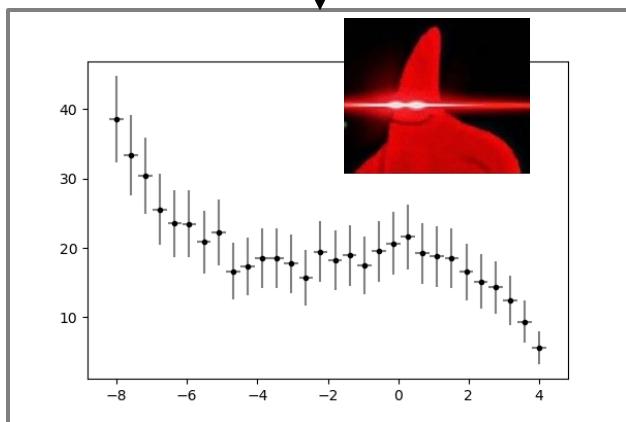
- Sig/bkg modeling for unbinned analyses (e.g. $h \rightarrow \gamma\gamma$ discovery, dijet searches,...)
- Smooth data/MC scale factor derivation from binned measurements (e.g. trigger SFs)
- Smooth transfer factor derivation for data-driven bkg modeling (e.g. ABCD method)

The traditional way – feel the curve, fit the pain



The traditional way – feel the curve, fit the pain

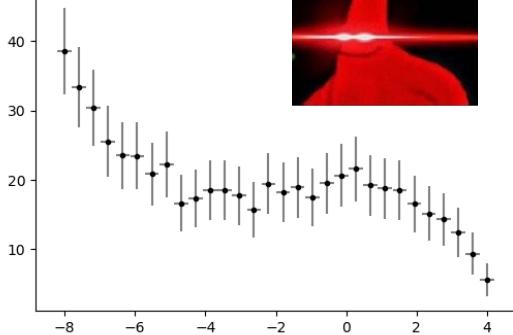
Interrogate the distribution



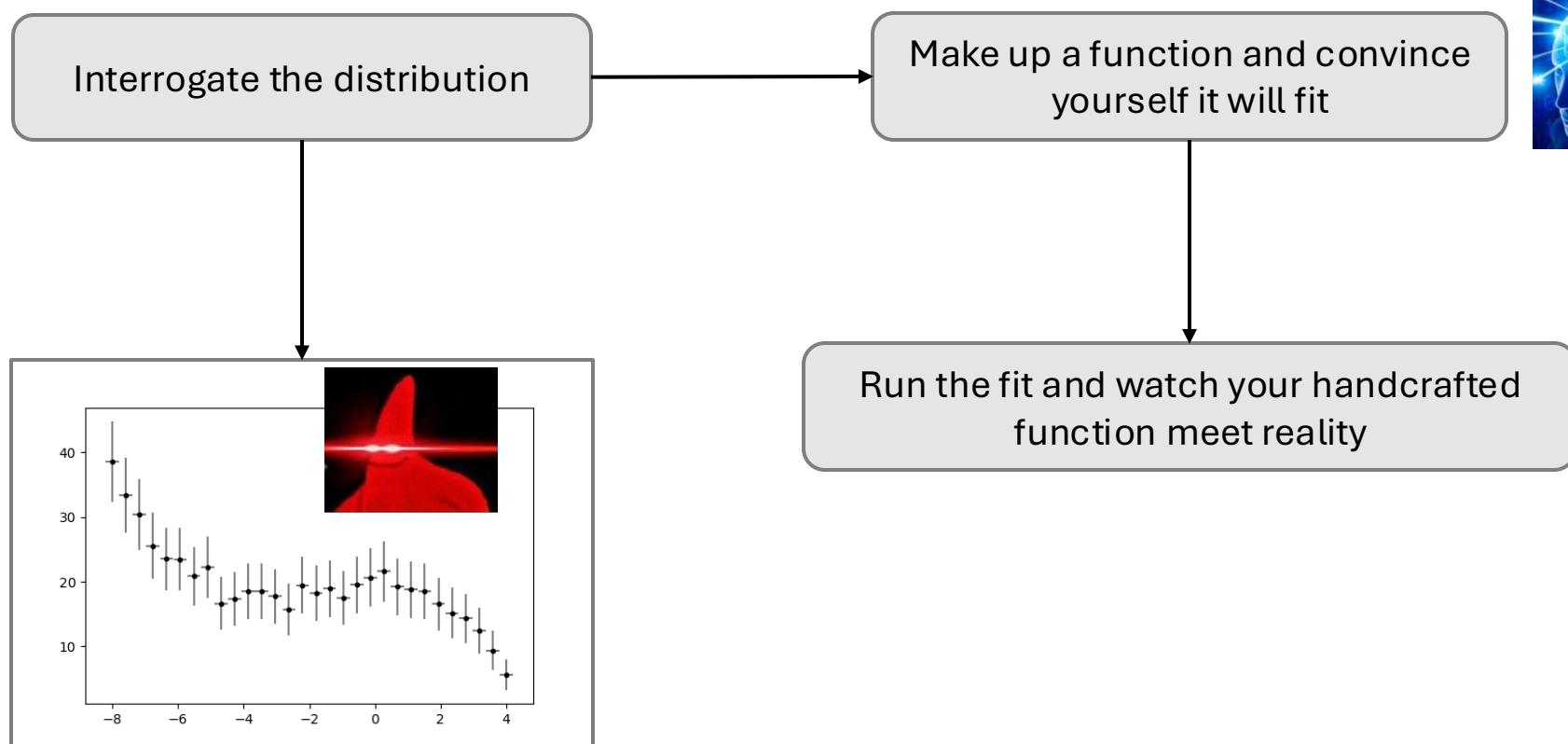
The traditional way – feel the curve, fit the pain

Interrogate the distribution

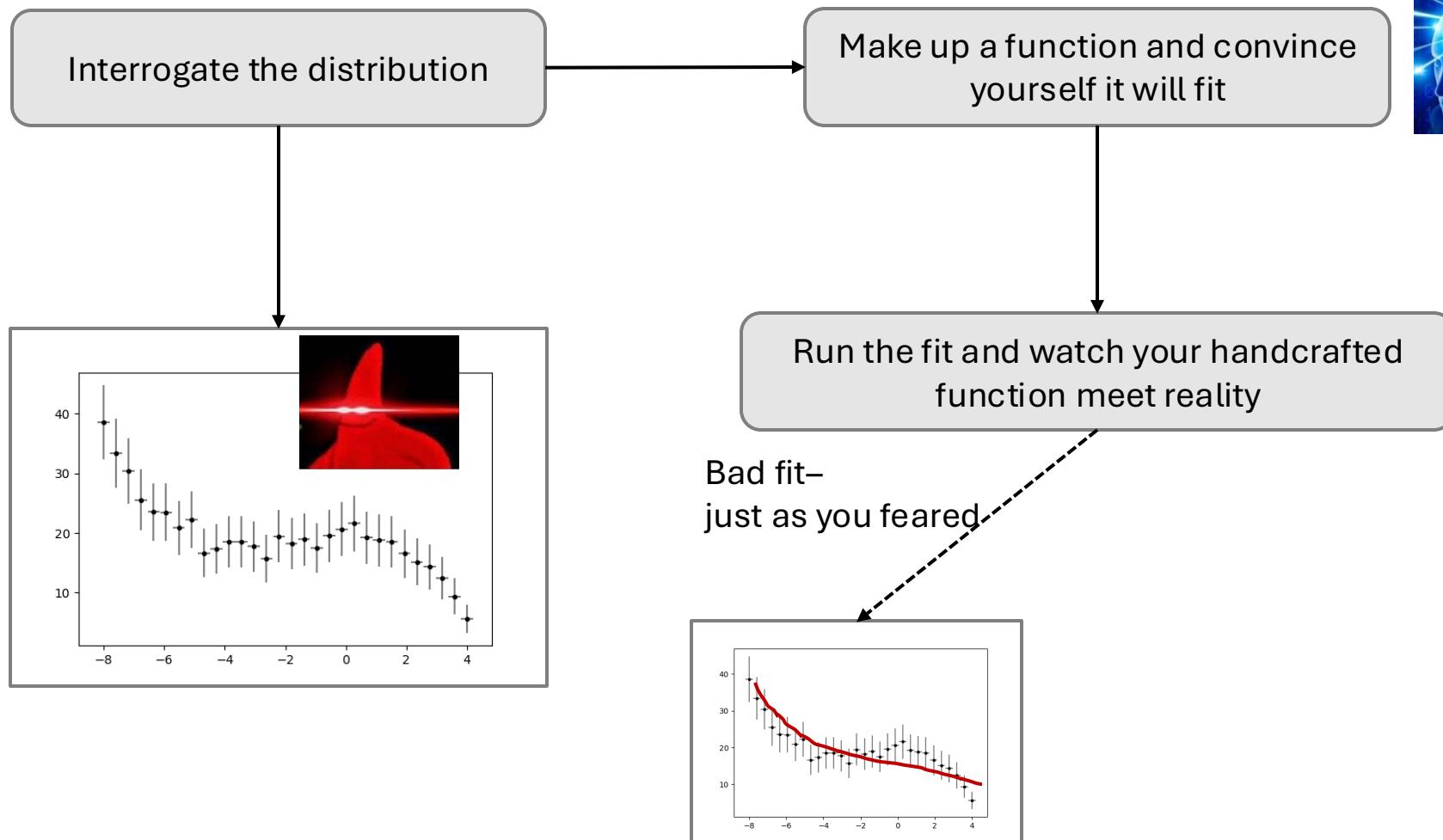
Make up a function and convince
yourself it will fit



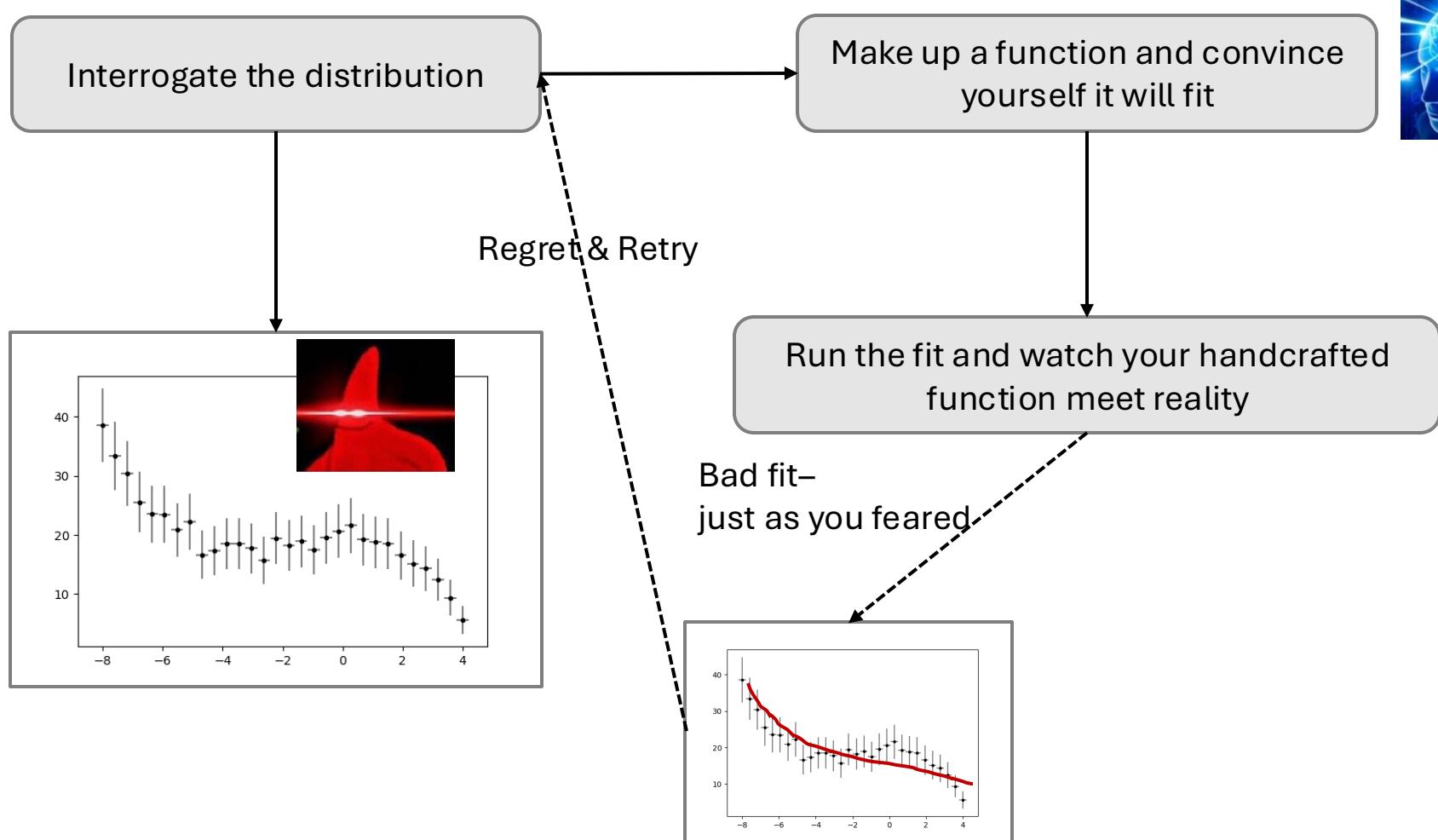
The traditional way – feel the curve, fit the pain



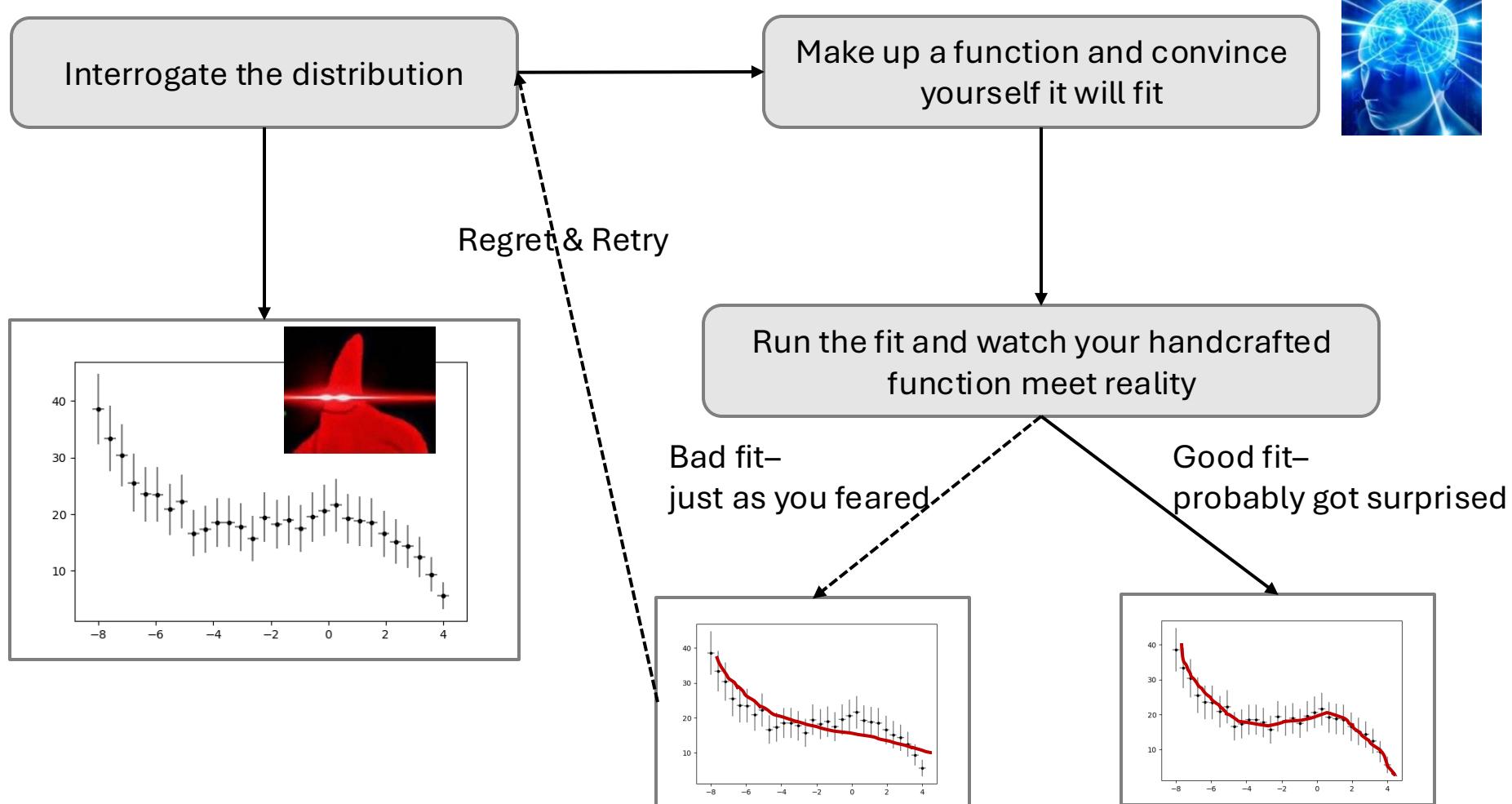
The traditional way – feel the curve, fit the pain



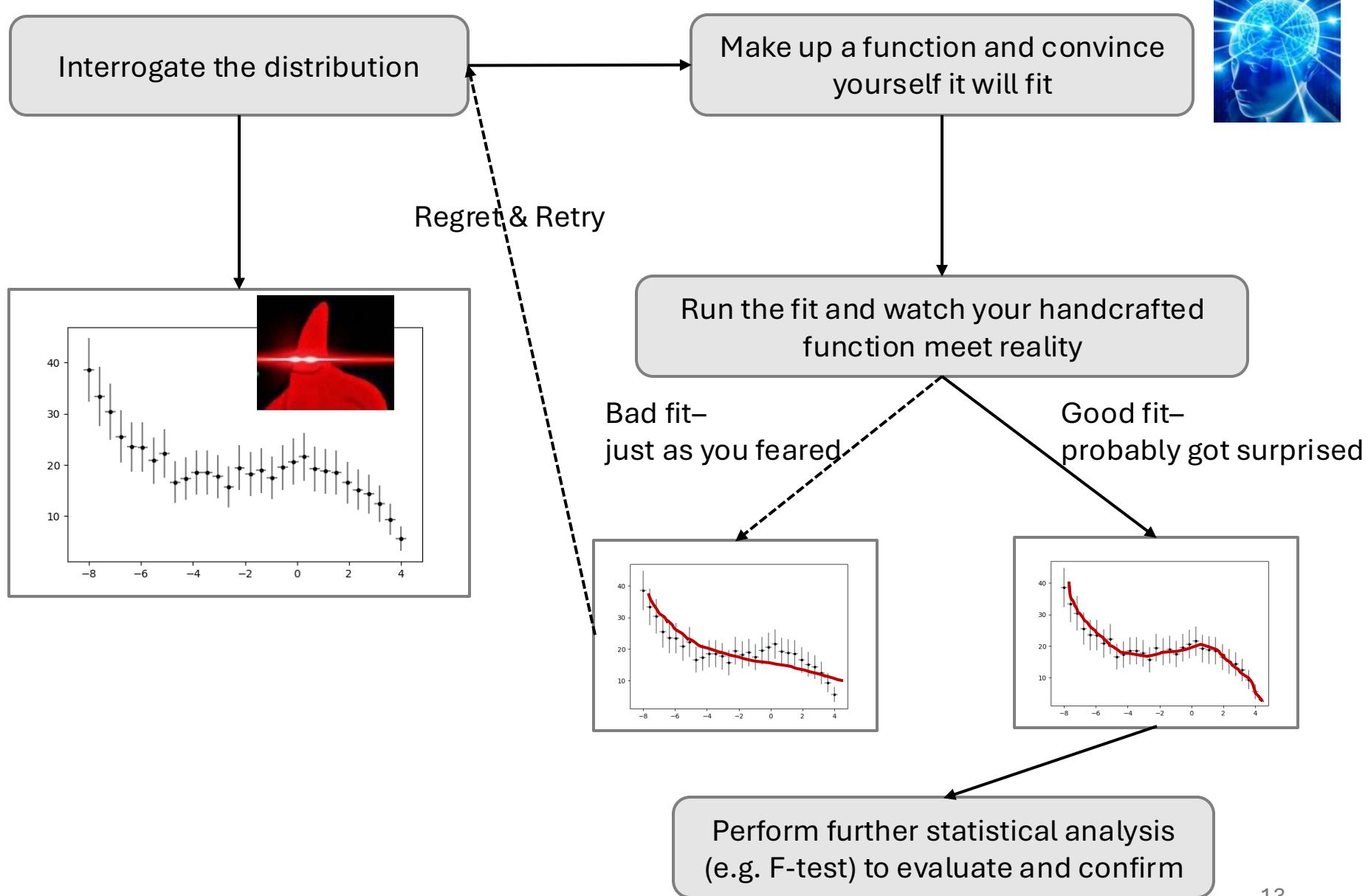
The traditional way – feel the curve, fit the pain



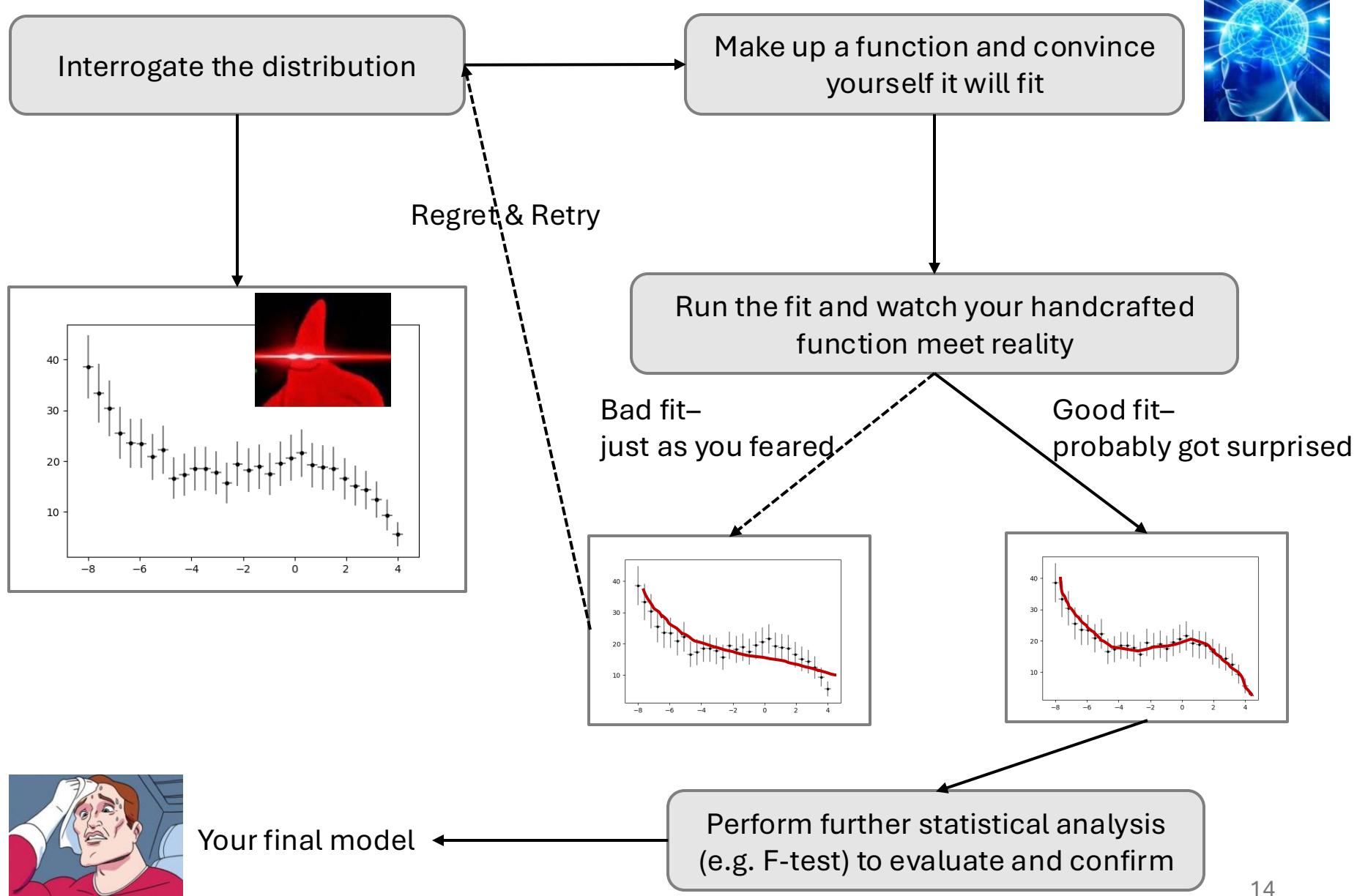
The traditional way – feel the curve, fit the pain



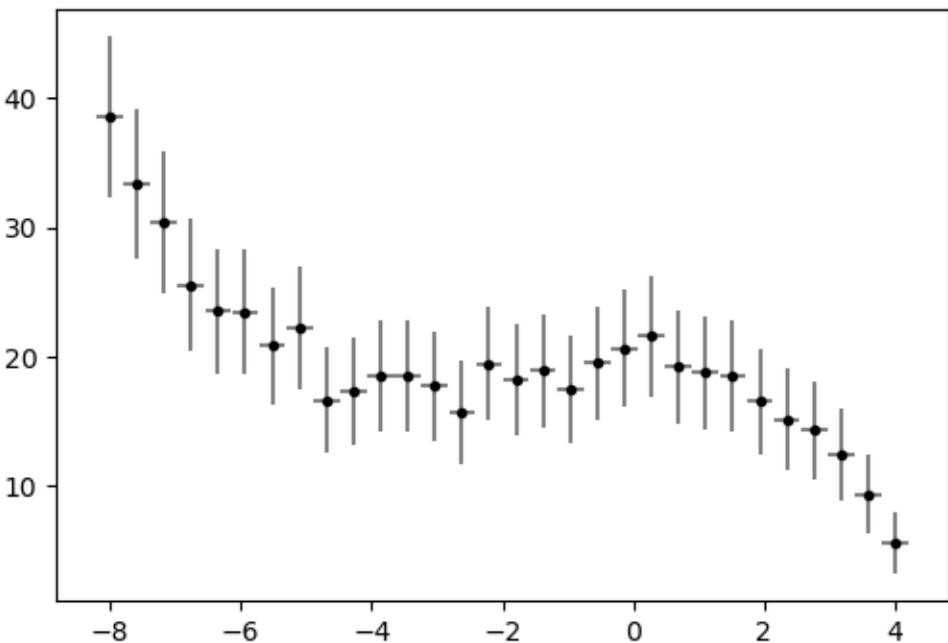
The traditional way – feel the curve, fit the pain



The traditional way – feel the curve, fit the pain

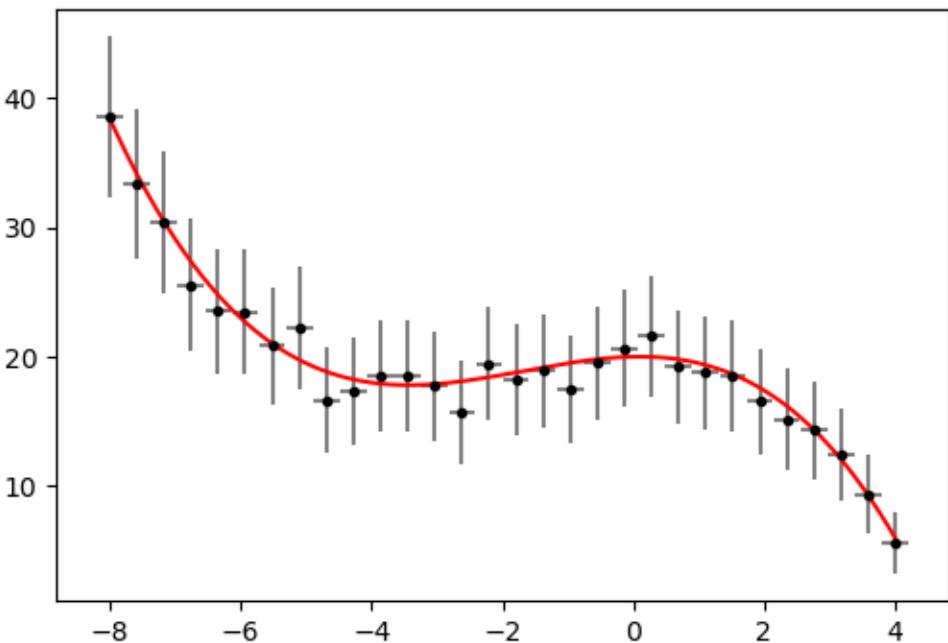


The traditional way – feel the curve, fit the pain



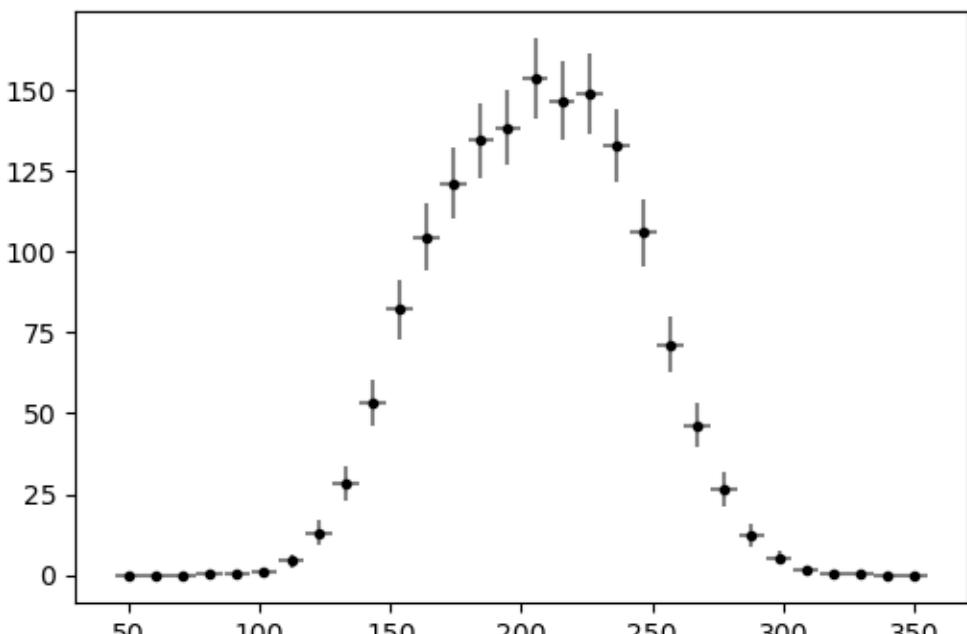
- A simple enough distribution.
- Guess a polynomial template should be sufficient to fit it.

The traditional way – feel the curve, fit the pain



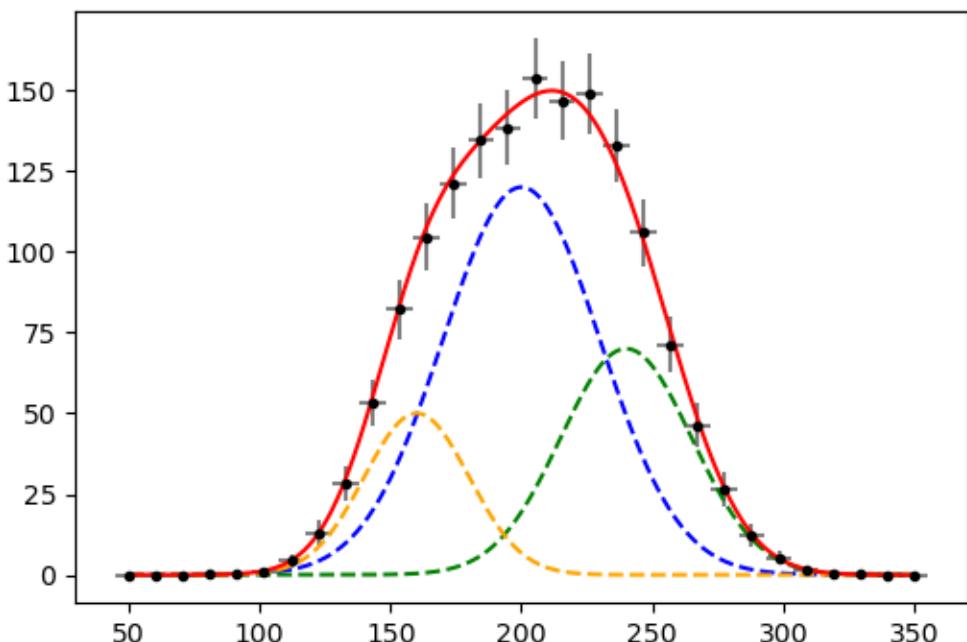
- Indeed, a 3rd order polynomial is sufficient!

The traditional way – feel the curve, fit the pain



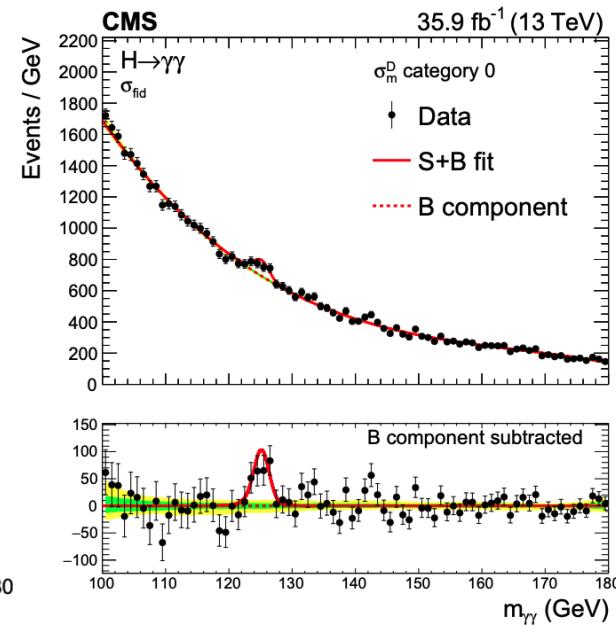
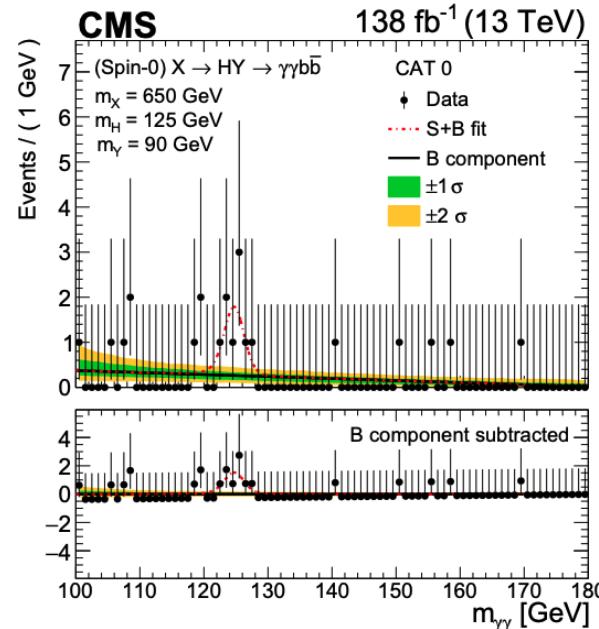
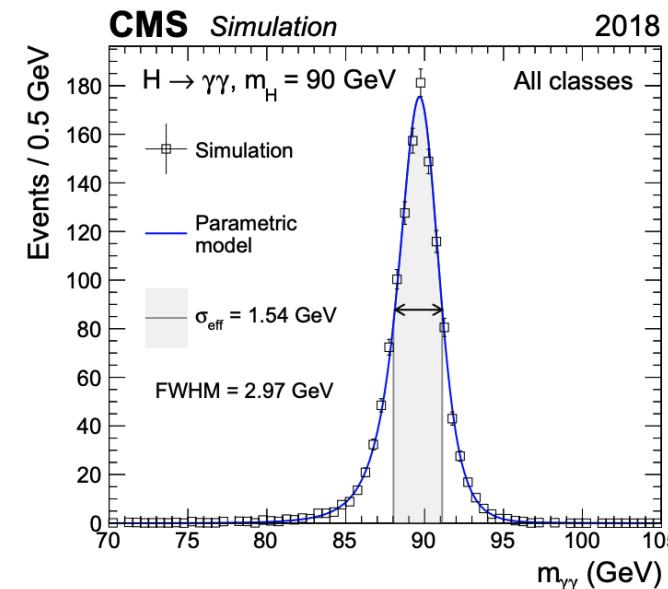
- How about this?
- Looks like a Gaussian, but not exactly when look closer, as it has a wider width...
- Natural guess: multiple Gaussians stacked together.

The traditional way – feel the curve, fit the pain



- Let's try sum of Gaussians.
- This is actually a sum of three Gaussians with different yield/mean/variance.

The traditional way – feel the curve, fit the pain



[2405.18149](#)

“...is modeled empirically with a sum of between two and four Gaussian functions...”

“...the number of Gaussian functions is determined using an F-test...”

[2310.01643](#)

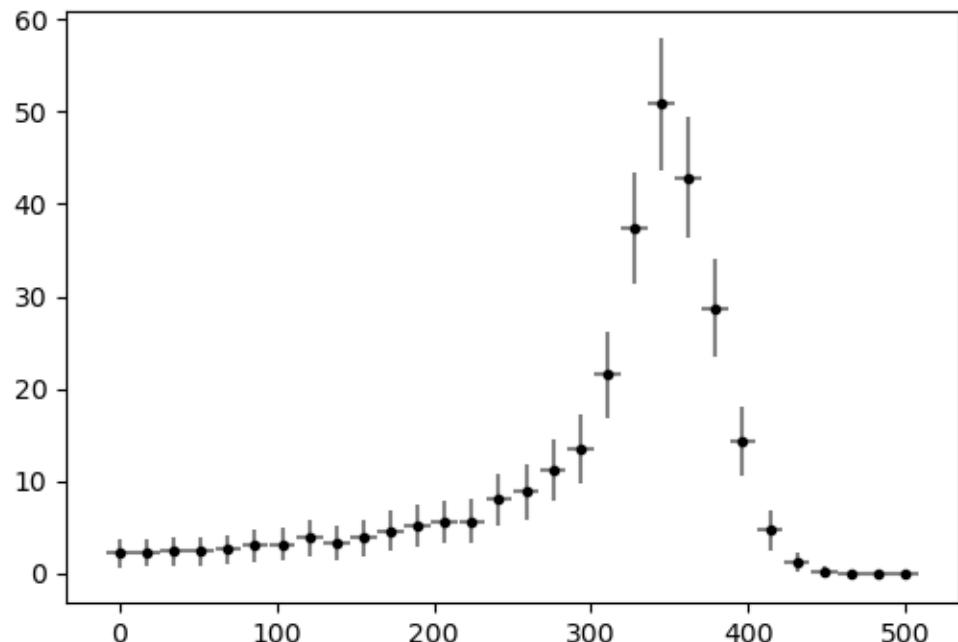
“...The $m_{\gamma\gamma}$ distribution is parametrized using the sum of up to five Gaussian functions...”

[1807.03825](#)

“...The model is built as a fit to a sum of up to five Gaussian distributions of the simulated invariant mass shape...”

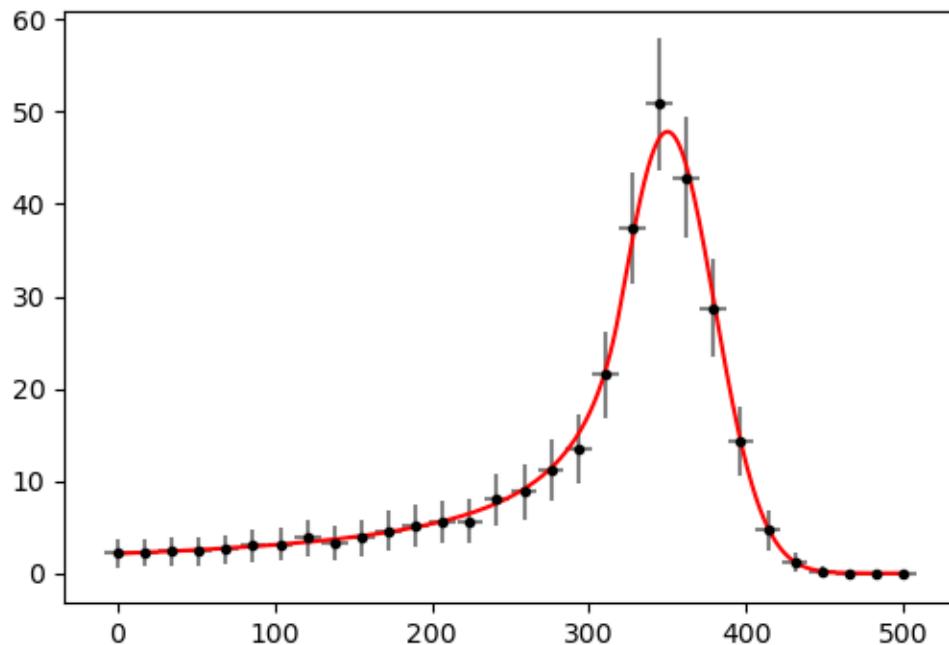
- Sum of Gaussians – a common template to model peaked signals in CMS/ATLAS analyses.
- It’s ~all about luck, as it is trial and error.

The traditional way – feel the curve, fit the pain



- How about this?
- Looks like a Gaussian but with an exponential tail.

The traditional way – feel the curve, fit the pain



- This function even earned its own name.
- But the same could have been done *effortlessly* with symbolic regression!

“The Crystal Ball function, named after the Crystal Ball Collaboration (hence the capitalized initial letters), is a probability density function commonly used to model various lossy processes in high-energy physics. It consists of a Gaussian core portion and a power-law low-end tail, below a certain threshold. The function itself and its first derivative are both continuous.” -- [wiki](#)

The Crystal Ball function is given by:

$$f(x; \alpha, n, \bar{x}, \sigma) = N \cdot \begin{cases} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right), & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot (B - \frac{x-\bar{x}}{\sigma})^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leq -\alpha \end{cases}$$

where

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right),$$

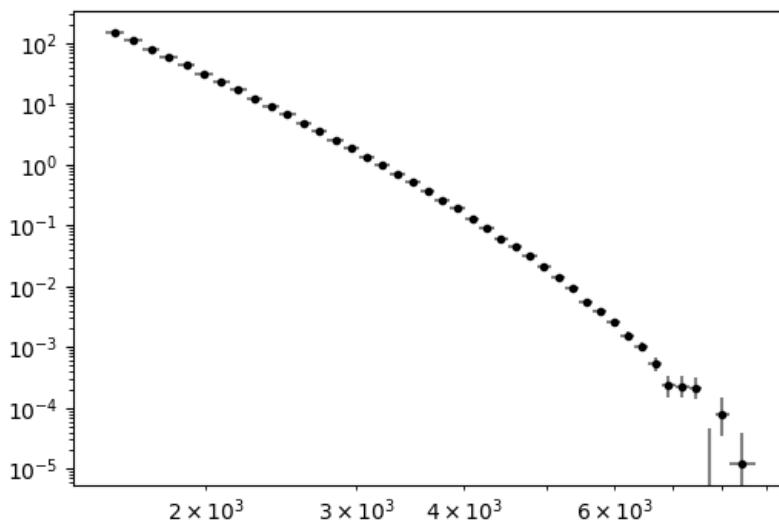
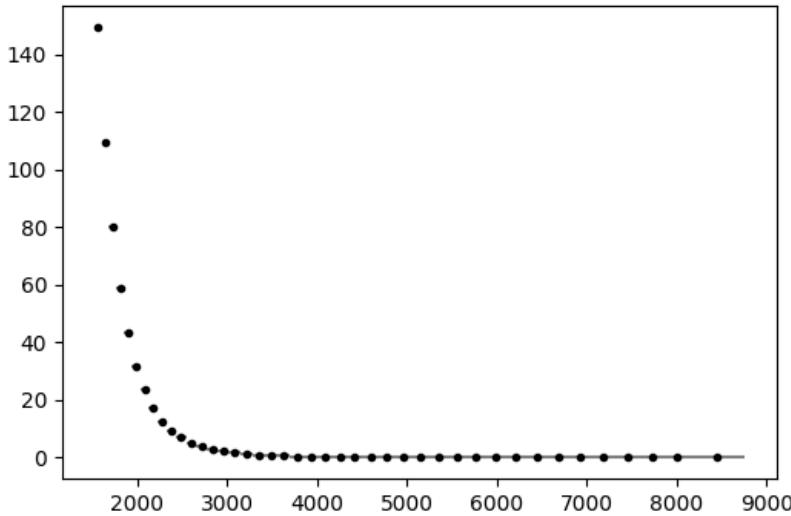
$$B = \frac{n}{|\alpha|} - |\alpha|,$$

$$N = \frac{1}{\sigma(C + D)},$$

$$C = \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot \exp\left(-\frac{|\alpha|^2}{2}\right),$$

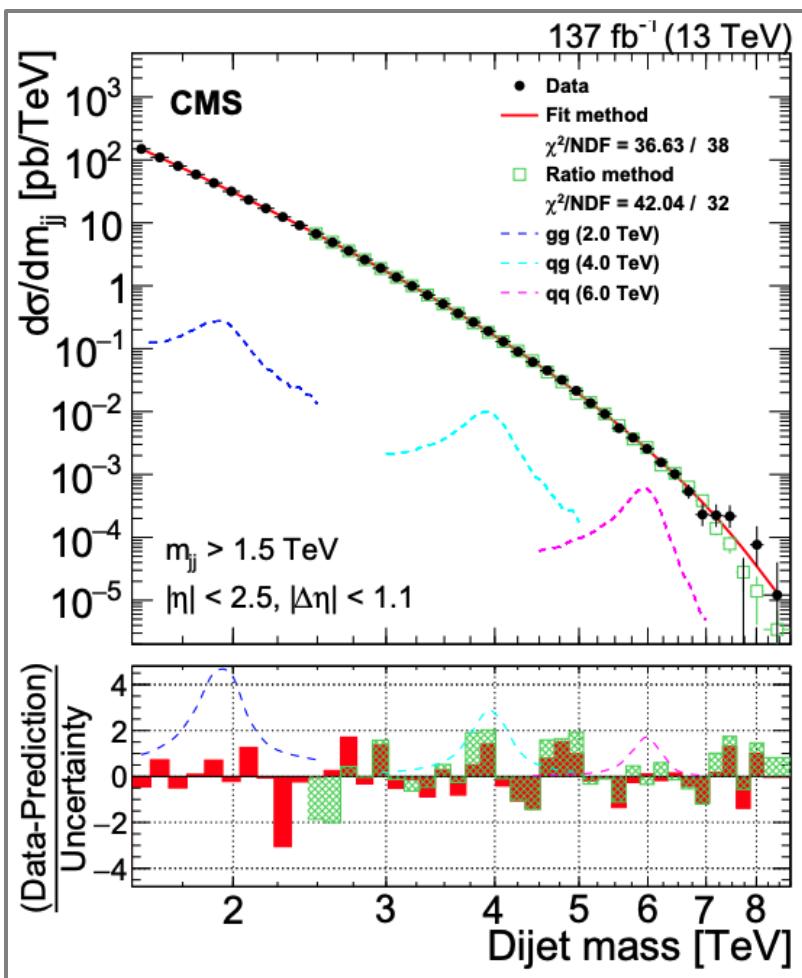
$$D = \sqrt{\frac{\pi}{2}} \left(1 + \operatorname{erf}\left(\frac{|\alpha|}{\sqrt{2}}\right)\right).$$

The traditional way – feel the curve, fit the pain



- How about the CMS dijet spectrum?
[1911.03947](https://arxiv.org/abs/1911.03947)
- Looks like a steeply falling shape, a simple exponential is not sufficient...

The traditional way – feel the curve, fit the pain



[1911.03947](#)

5 Background prediction methods

In the fit method, utilized here and in previous dijet resonance searches [17, 19–32, 50], the main background in the SR coming from QCD is parametrized with an empirical function of the form

$$\frac{d\sigma}{dm_{jj}} = \frac{P_0(1-x)^{P_1}}{x^{P_2+P_3 \ln(x)}}, \quad (1)$$

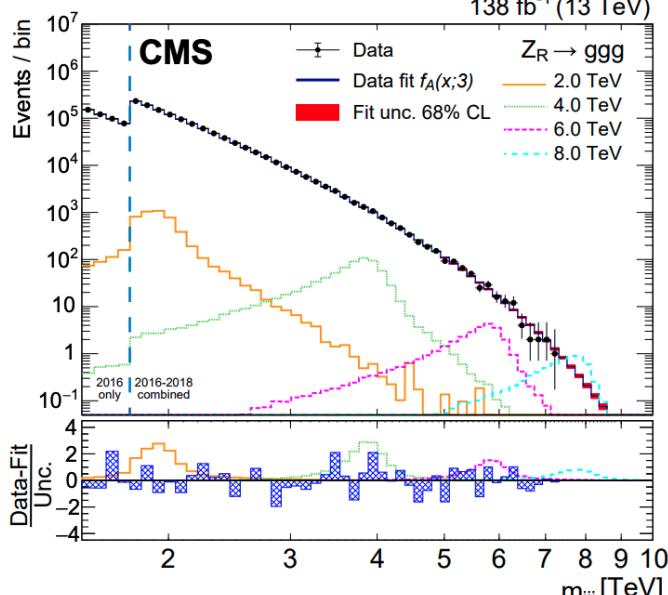
where $x = m_{jj}/\sqrt{s}$, and P_0 , P_1 , P_2 , and P_3 are four free parameters. The search for resonances

- They invented a bunch of various “dijet functions” empirically...
- A new one will need to be invented every time if the old ones do not work for new data anymore.

[2206.09997](#)

larger event samples. These are the modified dijet function (ModDijet-5p) given by $d\sigma/dm = p_0(1 - x^{1/3})^{p_1}/(x^{p_2+p_3 \log x + p_4 \log^2 x})$, the dijet function (Dijet-5p) given by $d\sigma/dm = p_0(1 - x)^{p_1}/(x^{p_2+p_3 \log x + p_4 \log^2 x})$, and the power-law times exponential function (PowExp-5p) given by $d\sigma/dm = p_0 e^{-p_1 x - p_2 x^2 - p_3 x^3}/x^{p_4}$, where $x = m/\sqrt{s}$, with m being the four-jet mass.

The traditional way – feel the curve, fit the pain



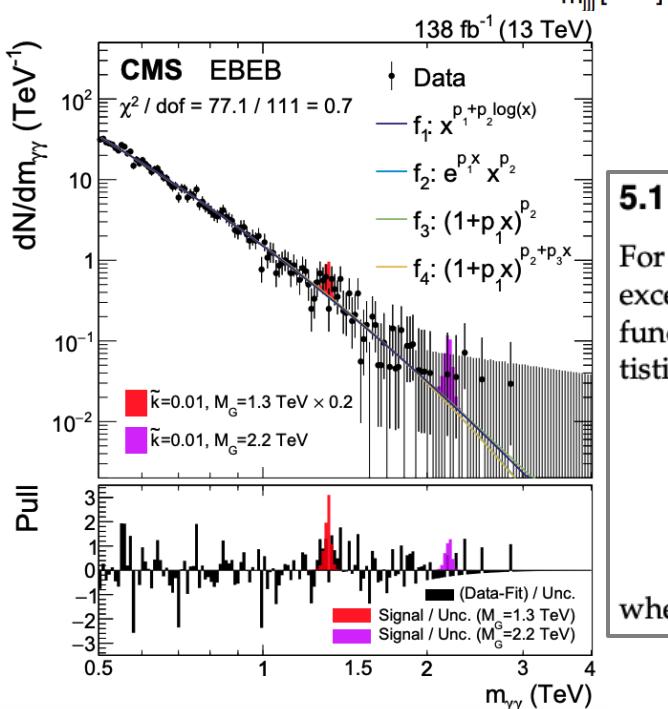
CMS High-mass trijet search

[2310.14023](#)

The background, dominantly arising from the QCD multijet process, is modeled by fitting the m_{jjj} distribution with smoothly falling, empirical functions that have been used in previous searches [19, 20]. Three families of functions are considered:

$$\begin{aligned} f_A(x; N) &= p_0 \frac{(1-x)^{p_1}}{x^{\sum_{i=2}^N p_i \log^{i-2}(x)}}, \\ f_B(x; N) &= p_0 \frac{e^{-p_1 x}}{x^{\sum_{i=2}^N p_i \log^{i-2}(x)}}, \\ f_C(x; N) &= p_0 x^{\sum_{i=1}^N p_i \log^{i-1}(x)}, \end{aligned} \quad (1)$$

where $x = m_{jjj} / \sqrt{s}$, p_i ($i = 0, 1, \dots, N$) are free parameters, and N is the order of the fit function, which is determined using a Fisher F-test [48]. It is found that the third order yields the



CMS High-mass diphoton search

[2405.09320](#)

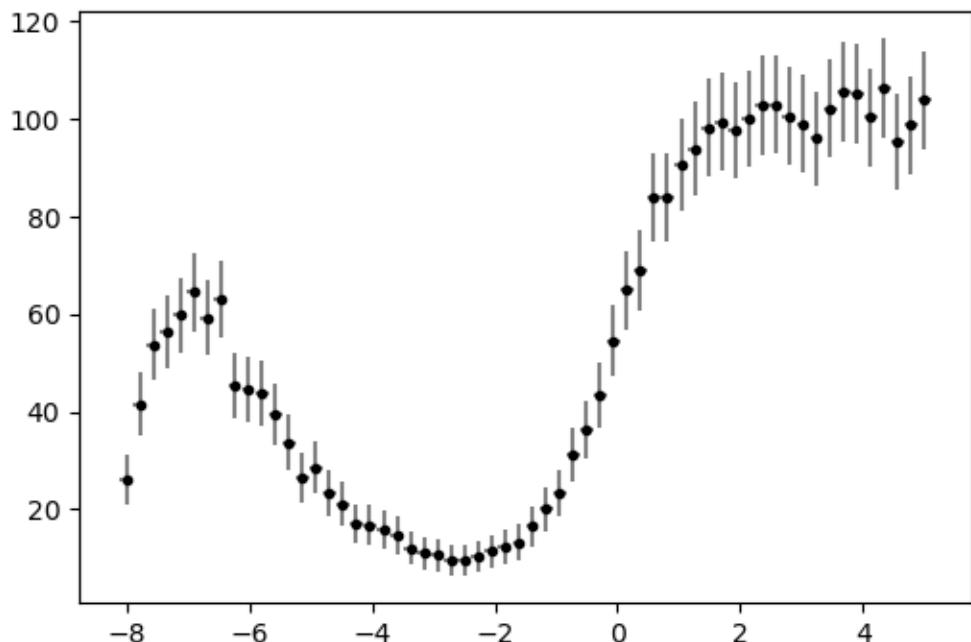
5.1 Resonant search background

For the resonant signal search, parametrized functions are fit to the $m_{\gamma\gamma}$ spectrum. Localized excesses in the form of a peak are distinguishable from the smoothly-falling background. Four functions of the dimensionless variable $x = m_{\gamma\gamma} / \sqrt{s}$ are given equal consideration in the statistical analysis:

$$\begin{aligned} f_1(x) &= p_0 x^{p_1 + p_2 \log(x)}, \\ f_2(x) &= p_0 e^{p_1 x} x^{p_2}, \\ f_3(x) &= p_0 (1 + x p_1)^{p_2}, \\ f_4(x) &= p_0 (1 + x p_1)^{p_2 + p_3 x}, \end{aligned} \quad (4)$$

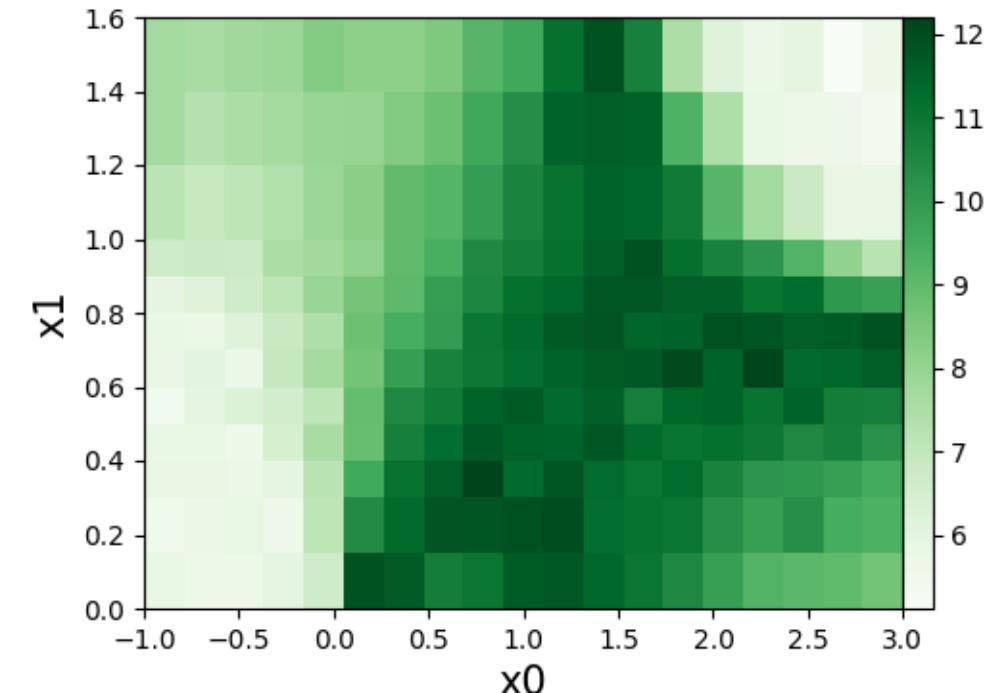
where p_i are unconstrained nuisance parameters in the hypothesis test. During the likelihood

The traditional way – feel the curve, fit the pain



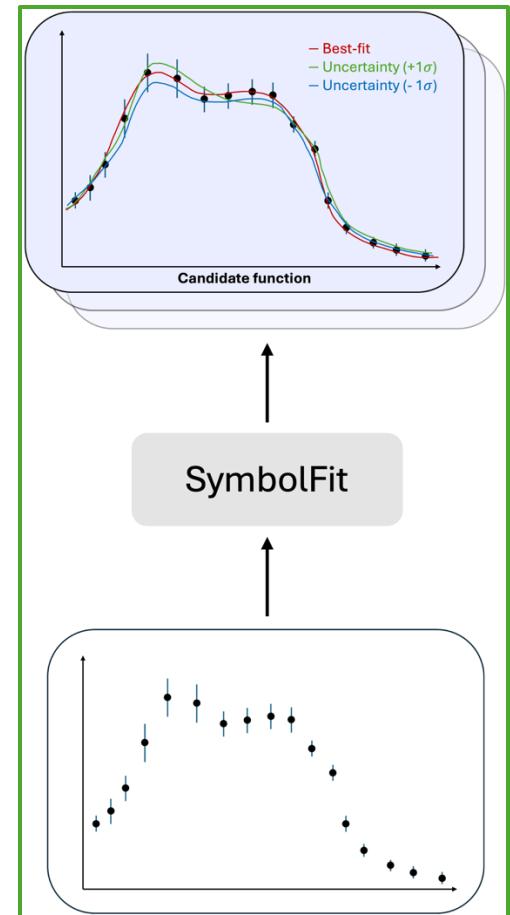
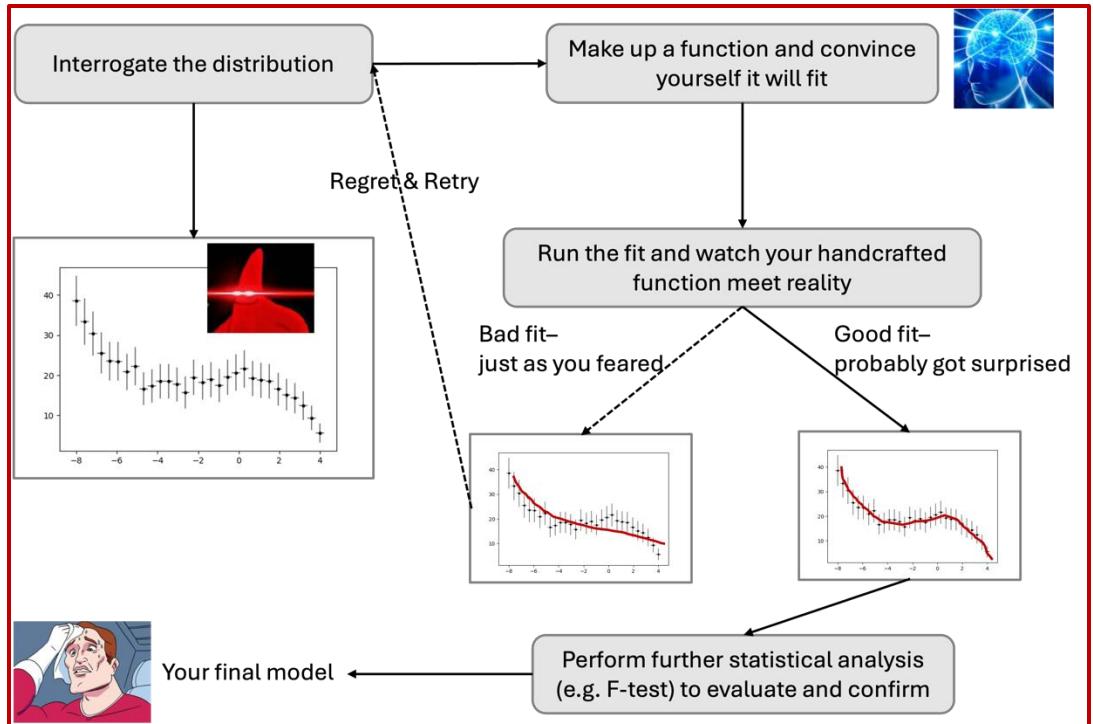
- How about this?
- No clue until spending hours or even longer with trial and error...

The traditional way – feel the curve, fit the pain



- How about this?
- No clue until spending hours or even longer with trial and error...

Now you can make the choice!

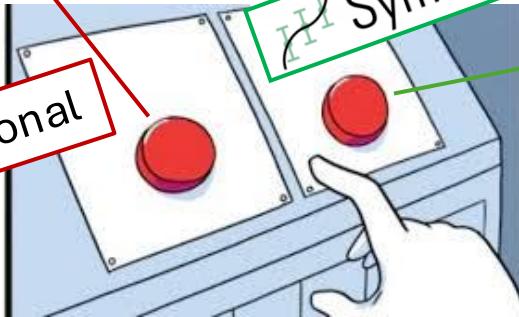


Manual way

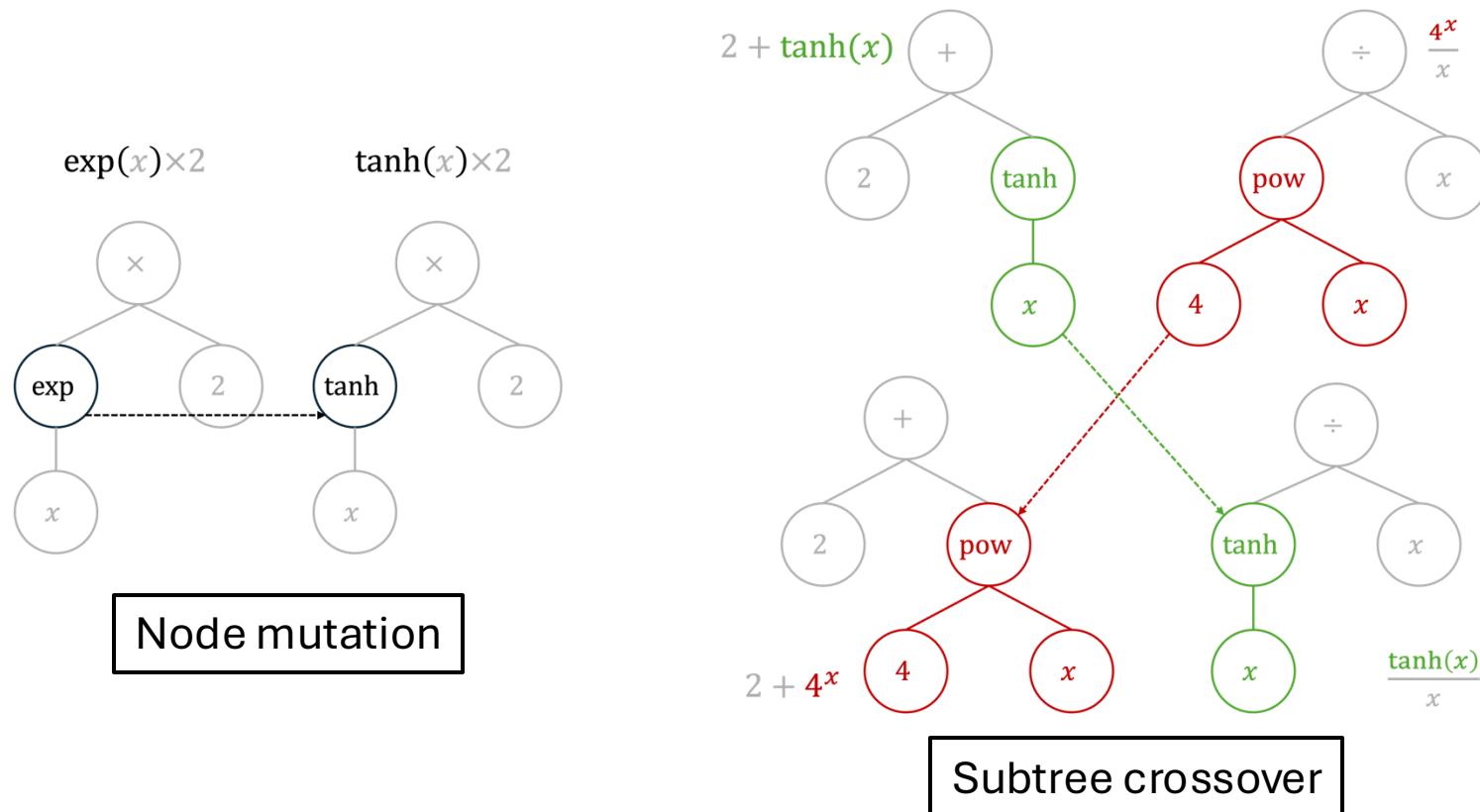
Traditional

SymbolFit

Automatic way



What is symbolic regression– A function ATM spitting out guesses at data until something sticks



- *Machine-search* for closed-form functions to fit data.
- Functional form is itself a “trainable parameter”.
- A vast function space is being searched using, e.g., genetic programming.
- **An exact functional form does not need to be known a priori!**

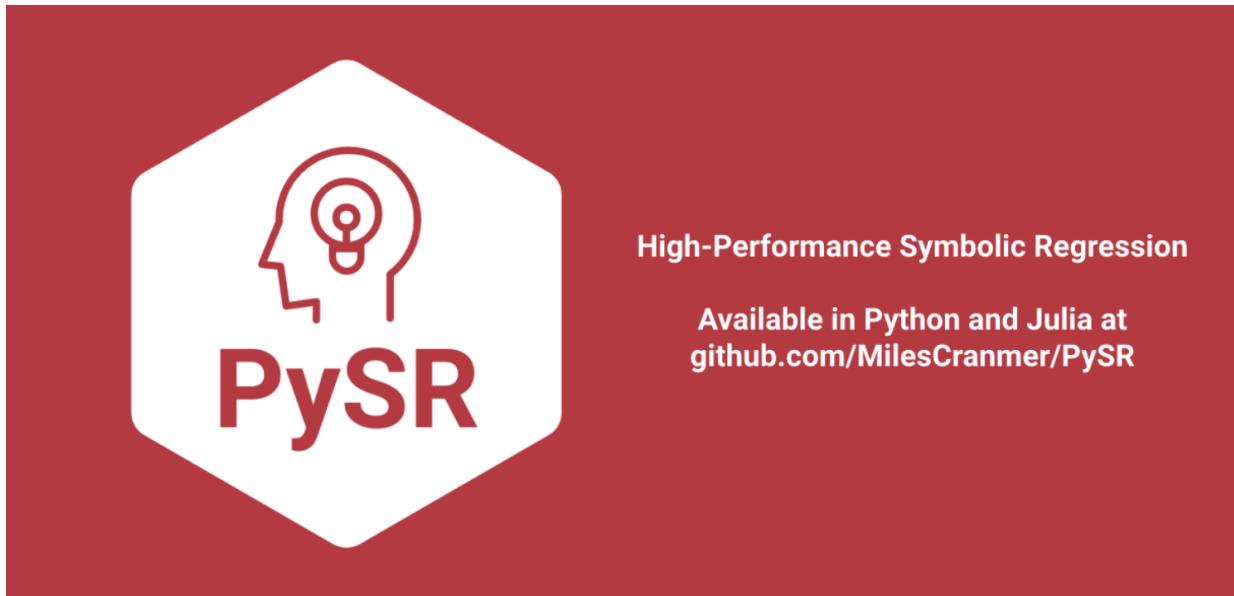
What is symbolic regression–

A function ATM spitting out guesses at data until something sticks

PySR and SymbolicRegression.jl

See source animation from PySR [here](#)

PySR: a symbolic regression algorithm



Why PySR?

PySR is an open-source tool for *Symbolic Regression*: a machine learning task where the goal is to find an interpretable symbolic expression that optimizes some objective.

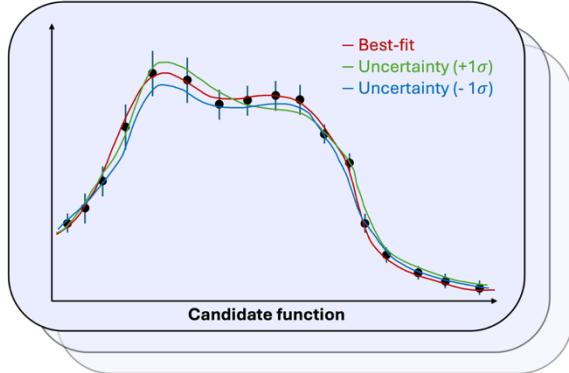
Over a period of several years, PySR has been engineered from the ground up to be (1) as high-performance as possible, (2) as configurable as possible, and (3) easy to use. PySR is developed alongside the Julia library [SymbolicRegression.jl](#), which forms the powerful search engine of PySR. The details of these algorithms are described in the [PySR paper](#).

What is SymbolFit?

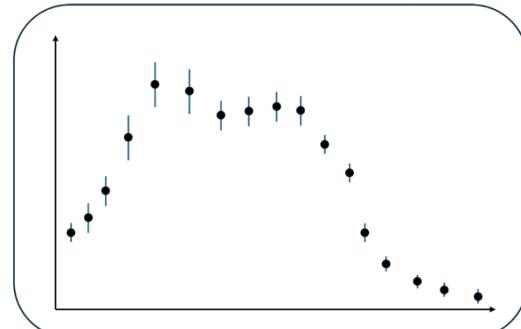


Output

(Multiple candidate functions, each with uncertainty measure)



SymbolFit



Input

(Without a pre-defined functional form)



Automatic parametric modeling with symbolic regression

python machine-learning automation interpolation symbolic-regression

- A python API that interfaces PySR & LMFIT to streamline and automate parametric modeling:
 1. Function forms searching [PySR]
 2. Dedicated re-optimization of functions [LMFIT]
 3. Uncertainty modeling [LMFIT]
 4. Evaluation and plotting [standalone]

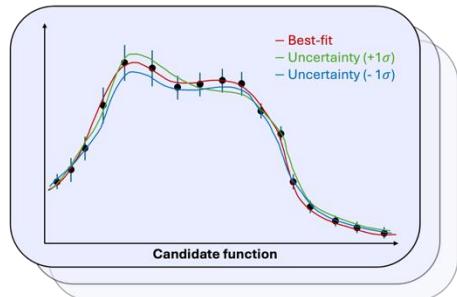
All in one go!

- Note: SR algorithms do not inherently estimate uncertainties, and numerical constants in functions are sometimes sub-optimal.

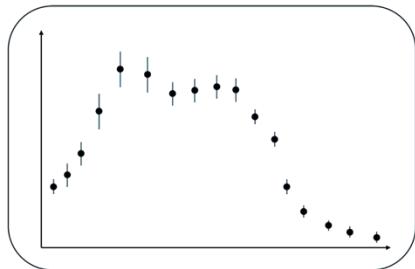
Internal steps of SymbolFit

Output

(Multiple candidate functions, each with uncertainty measure)

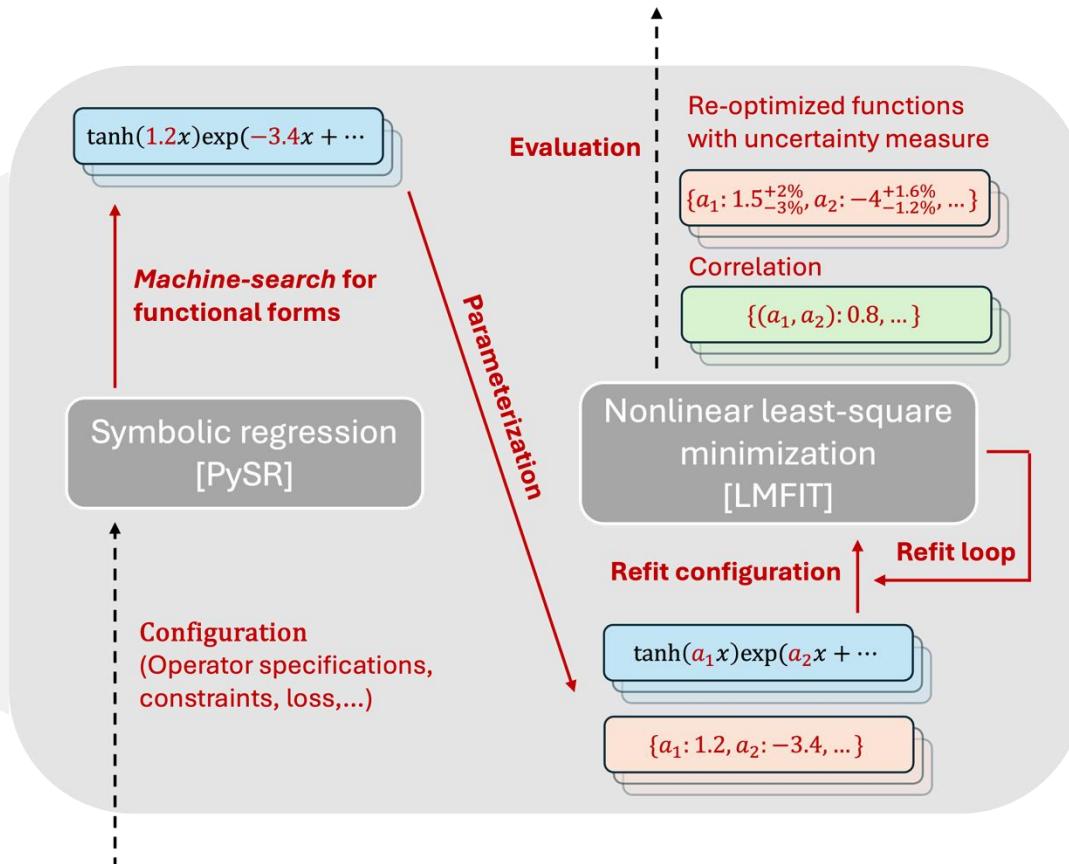


SymbolFit



Input

(Without a pre-defined functional form)



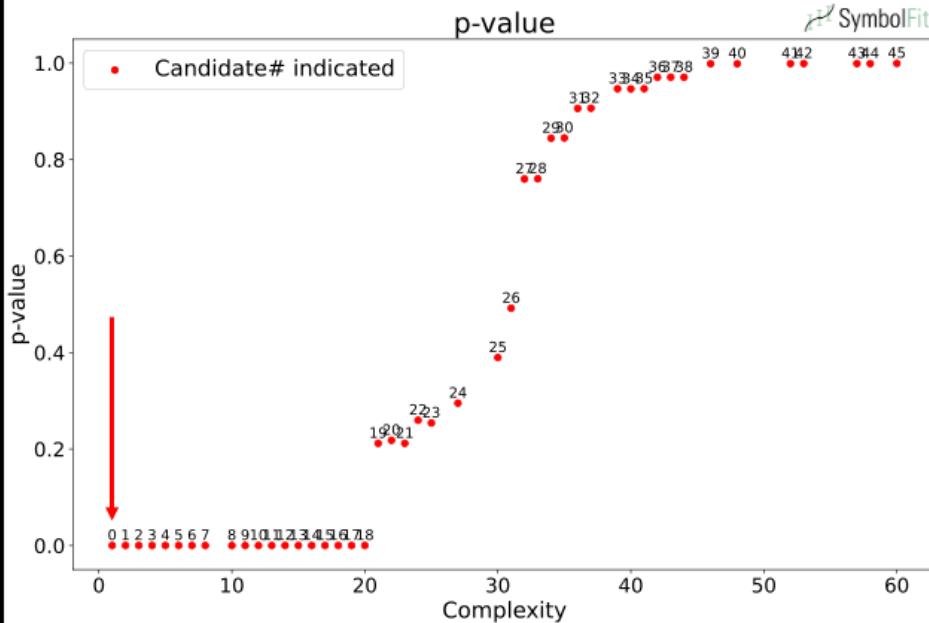
Convergence of candidate functions in a fit

164.796*(a1)

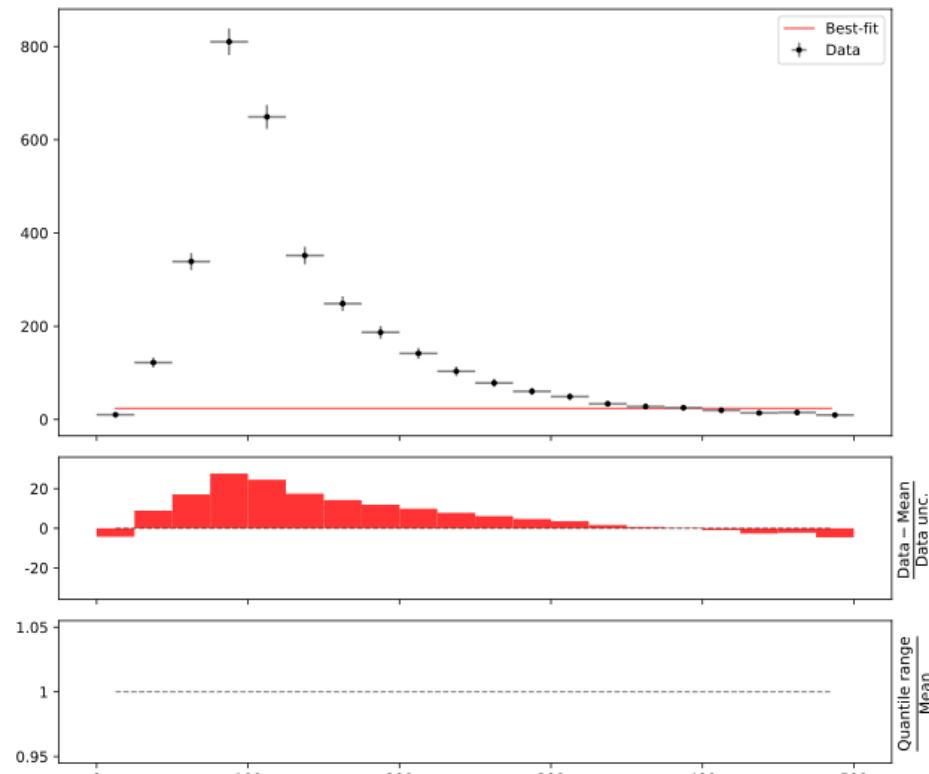
^ Candidate function in parameterized form {a₁, a₂, a₃, ...}

A total of 46 candidate functions were generated from a single fit, each with uncertainty estimates.

 SymbolFit



Total uncertainty coverage for each candidate function (68% quantile range obtained by sampling parameters using the covariance matrix).



See the animation [here](#)

Function space configuration

```
1 from pysr import PySRegressor
2
3 pysr_config = PySRegressor(
4     model_selection = "accuracy",
5     timeout_in_seconds = 60*100,
6     niterations = 200,
7     maxsize = 80,
8     binary_operators = [
9         "+", "*", "/", "^"
10        ],
11     unary_operators = [
12         "exp",
13         "tanh",
14     ],
15     nested_constraints = {
16         "exp": {"exp": 0, "tanh": 0, "*": 2, "/": 1, "^": 1},
17         "tanh": {"exp": 0, "tanh": 0, "*": 2, "/": 1, "^": 1},
18         "*": {"exp": 1, "tanh": 1, "*": 2, "/": 1, "^": 1},
19         "^": {"exp": 1, "tanh": 1, "*": 2, "/": 1, "^": 0},
20         "/": {"exp": 1, "tanh": 1, "*": 2, "/": 0, "^": 1},
21     },
22     loss="loss(y, y_pred, weights) = (y - y_pred)^2 * weights",
23 )
```

Flexibility of symbolic regression

- In fact, this single function space configuration can be re-used to simultaneously search for different functions that fit well the CMS dijet, pair-dijet, trijet, diphoton, and more spectra!

Function space configuration (template spec.)

```
1  from pysr import PySRRRegressor, TemplateExpressionSpec
2
3  expression_spec = TemplateExpressionSpec(
4      'p[1] * f(x/13000) ^ g(log(x/13000))',
5      expressions = ['f', 'g'],
6      variable_names = ['x'],
7      parameters = {'p': 1}
8  )
9
10 pysr_config = PySRRRegressor(
11     expression_spec = expression_spec,
12     model_selection = 'accuracy',
13     niterations = 200,
14     maxsize = 40,
15     binary_operators = ['+', '*'],
16     elementwise_loss='loss(y, y_pred, weights) = (y - y_pred)^2 * weights',
17 )
```

User can impose desired structure on the final expressions, which can potentially help narrowing down the function search space when domain knowledge is present.

- For example, one can constrain to search for dijet functions of the form $f(x)^g(\log(x))$, where f and g are functions being searched for. One can further constrain to allow only + and * operators, restricting $f(x)$ to be a polynomial of x and $g(\log(x))$ to be a polynomial of $\log(x)$.

SymbolFit output files per fit

Each fit generates tens of candidate functions. Their details, ranking, uncertainty coverage, goodness-of-fit, data comparison etc. are all auto-generated and saved to:

1. `candidates.csv`: saves all candidate functions and evaluations in a csv table.
 2. `candidates_compact.csv`: saves a reduced version for essential information without intermediate results.
 3. `candidates.pdf`: plots all candidate functions (1D/2D only for now) with associated uncertainties one by one for fit quality evaluation.
 4. `candidates_sampling.pdf`: plots all candidate functions (1D only for now) with total uncertainty coverage generated by sampling parameters.
 5. `candidates_gof.pdf`: plots the goodness-of-fit scores.
 6. `candidates_correlation.pdf`: plots the correlation matrices for the parameters of the candidate functions.
-
- See output files from an example fit at github [here](#)

Examples

	Candidate function (after ROF)	# param.	χ^2/NDF (before ROF)	χ^2/NDF (after ROF)	p-value (after ROF)
SR model 1	$(570x(x(-0.423 \exp(2x) + \exp(x)) + x) + 149) \times (0.00328 + 0.0304 \tanh(x))^{4.87x}$	5	400.5 / 30	29.21 / 30	0.507
SR model 2	$(145(0.958 + x)^{\tanh(-0.711+4.32x)} + 145 \tanh(x)) \times (0.00591 + 0.0522 \tanh(x))^{5.48x}$	5	103.8 / 30	29.91 / 30	0.47
SR model 3	$\text{pow}(149(0.0101x + 0.0101 \tanh(0.171 + 0.724x)), x + (2.38x \tanh(-0.71 + x) + 2.39) \tanh(x) + \tanh(x))$	5	214.8 / 30	30.93 / 30	0.419

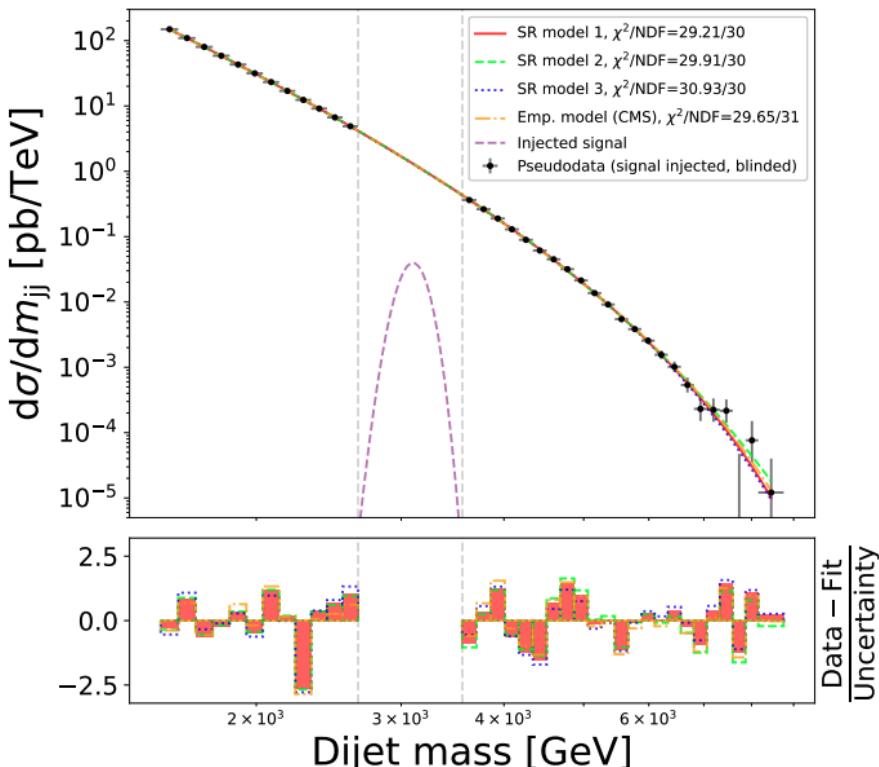


Figure 9: Pseudodata of the dijet spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. 2) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

- SymbolFit easily generated multiple candidate functions of very different forms that are all comparable to the CMS empirical “dijet function”

Traditional way

5 Background prediction methods

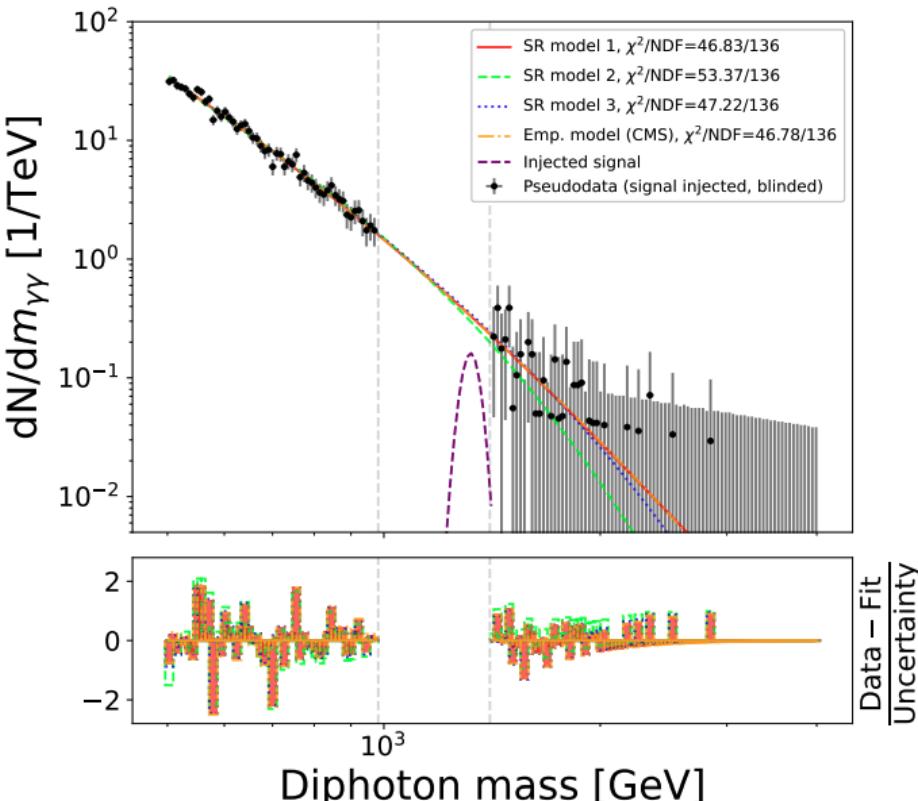
In the fit method, utilized here and in previous dijet resonance searches [17, 19–32, 50], the main background in the SR coming from QCD is parametrized with an empirical function of the form

$$\frac{d\sigma}{dm_{jj}} = \frac{P_0(1-x)^{P_1}}{x^{P_2+P_3 \ln(x)}}, \quad (1)$$

where $x = m_{jj}/\sqrt{s}$, and P_0 , P_1 , P_2 , and P_3 are four free parameters. The search for resonances

Examples

	Candidate function (after ROF)	# param.	χ^2/NDF (before ROF)	χ^2/NDF (after ROF)	p-value (after ROF)
SR model 1	$33.3(0.017 + 0.177x)^{6.93x}$	3	$47.12 / 136 = 0.3465$	$46.83 / 136 = 0.3443$	1.0
SR model 2	$35.0 \exp(-16.0x^{0.834})$	3	$63.44 / 136 = 0.4665$	$53.37 / 136 = 0.3925$	1.0
SR model 3	$0.0226 \exp(0.975x)(0.158 + x)^{-3.96}$	3	$48.94 / 136 = 0.3598$	$47.22 / 136 = 0.3472$	1.0



- SymbolFit easily generated multiple candidate functions of very different forms that are all comparable to the CMS empirical “diphoton function”.

Traditional way

5.1 Resonant search background

For the resonant signal search, parametrized functions are fit to the $m_{\gamma\gamma}$ spectrum. Localized excesses in the form of a peak are distinguishable from the smoothly-falling background. Four functions of the dimensionless variable $x = m_{\gamma\gamma}/\sqrt{s}$ are given equal consideration in the statistical analysis:

$$\begin{aligned} f_1(x) &= p_0 x^{p_1+p_2 \log(x)}, \\ f_2(x) &= p_0 e^{p_1 x} x^{p_2}, \\ f_3(x) &= p_0 (1 + x p_1)^{p_2}, \\ f_4(x) &= p_0 (1 + x p_1)^{p_2+p_3 x}, \end{aligned} \quad (4)$$

where p_i are unconstrained nuisance parameters in the hypothesis test. During the likelihood

Figure A5: Pseudodata of the diphoton spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. A2) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

Examples

	Candidate function (after ROF)	# param.	χ^2/NDF (before ROF)	χ^2/NDF (after ROF)	p-value (after ROF)
SR model 1	$(1.08 \times 10^{-5}) \tanh(x) / ((0.165 + x) \times \exp(x^2(-1.96 + 4x))^{\tanh(1.17x^2)})$	3	$50.46 / 26 = 1.941$	$49.04 / 26 = 1.886$	0.00408
SR model 2	$\exp(x(-10.8 + x)) / (-0.261x \tanh(x \times (-10.9 + x)) + 0.165 + \tanh(x))$	4	$39.49 / 25 = 1.58$	$33.15 / 25 = 1.326$	0.1273
SR model 3	$0.0554^{-0.622+4.03x} (0.568 \tanh(2x) + (0.00302 \exp(x) / (1.92 + x))^x)$	4	$38.4 / 25 = 1.536$	$37.31 / 25 = 1.492$	0.05395

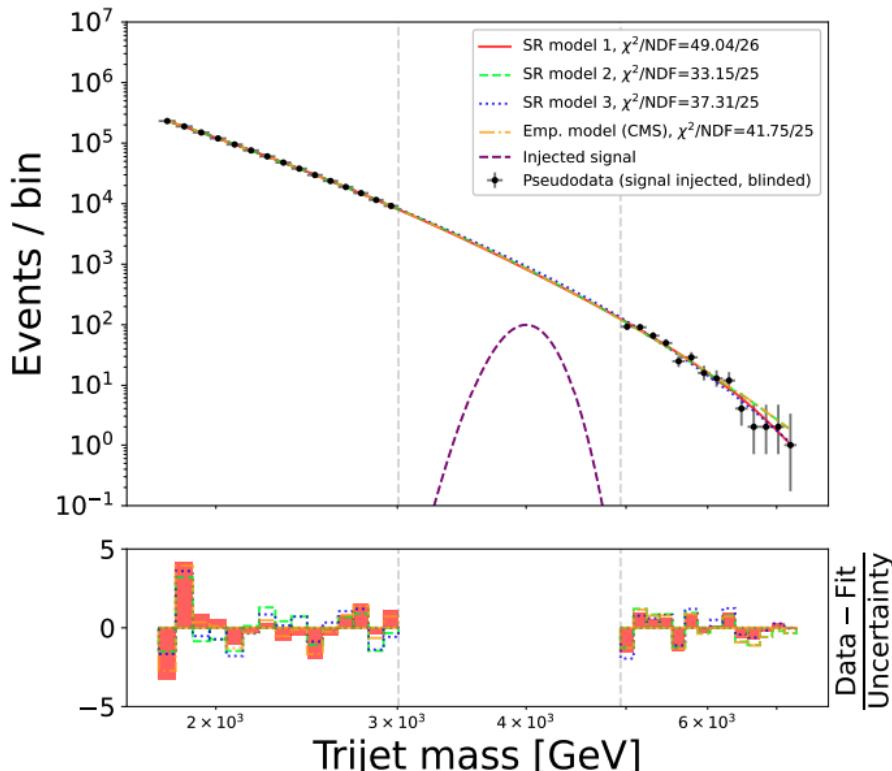


Figure A9: Pseudodata of the trijet spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. A4) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

- SymbolFit easily generated multiple candidate functions of very different forms that are all comparable to the CMS empirical “trijet function”.

Traditional way

The background, dominantly arising from the QCD multijet process, is modeled by fitting the m_{jj} distribution with smoothly falling, empirical functions that have been used in previous searches [19, 20]. Three families of functions are considered:

$$\begin{aligned} f_A(x; N) &= p_0 \frac{(1-x)^{p_1}}{x^{\sum_{i=2}^N p_i \log^{i-2}(x)}}, \\ f_B(x; N) &= p_0 \frac{e^{-p_1 x}}{x^{\sum_{i=2}^N p_i \log^{i-2}(x)}}, \\ f_C(x; N) &= p_0 x^{\sum_{i=1}^N p_i \log^{i-1}(x)}, \end{aligned} \quad (1)$$

where $x = m_{jj} / \sqrt{s}$, p_i ($i = 0, 1, \dots, N$) are free parameters, and N is the order of the fit function, which is determined using a Fisher F-test [48]. It is found that the third order yields the

Examples

	Candidate function (after ROF)	# param.	χ^2/NDF (before ROF)	χ^2/NDF (after ROF)	p-value (after ROF)
SR model 1	$(1.13 \times 10^{-5}x)^{1.28x}/(0.143x^x + 1.63x)$	3	$47.95 / 34 = 1.41$	$39.07 / 34 = 1.149$	0.2524
SR model 2	$6.98x^{0.857x}(x + \exp(x))^{-11.8}$	3	$47.36 / 34 = 1.393$	$39.83 / 34 = 1.171$	0.2267
SR model 3	$((6.52 \times 10^{-5} + 0.000378x)\tanh(0.641+3x))^{x+\tanh(x)}/\tanh(0.145+x)$	3	$71.24 / 34 = 2.095$	$35.57 / 34 = 1.046$	0.3942

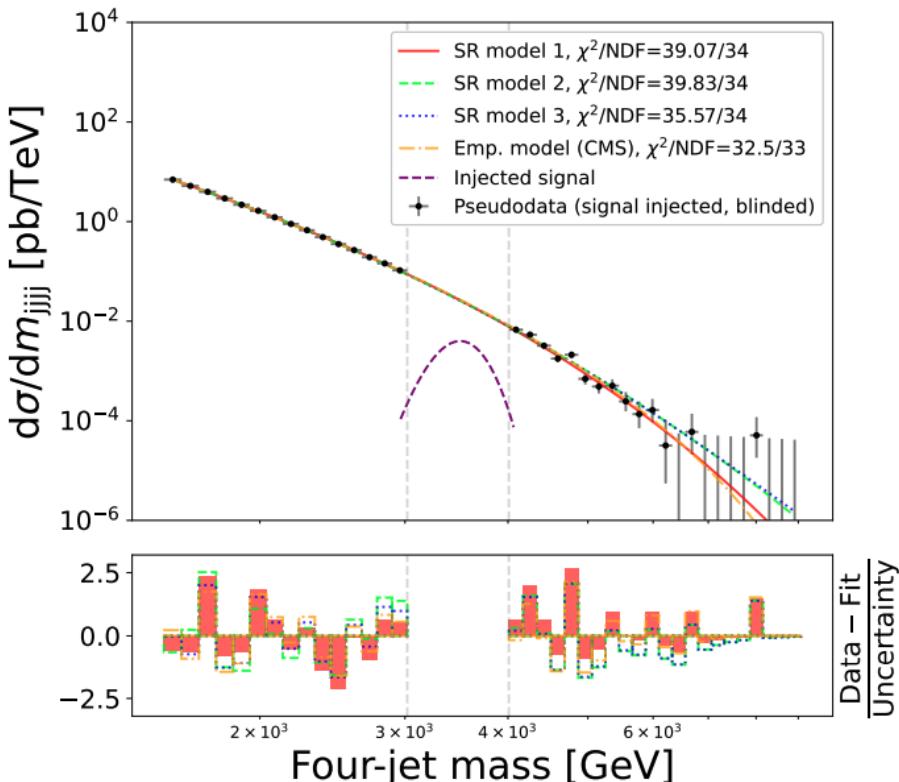


Figure A13: Pseudodata of the four-jet spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. A6) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

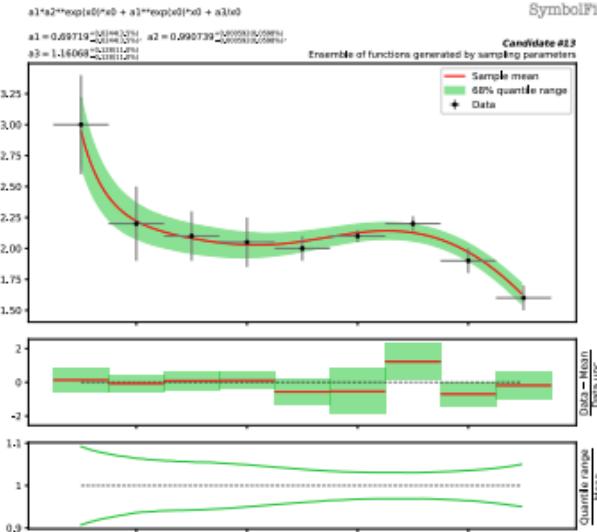
- SymbolFit easily generated multiple candidate functions of very different forms that are all comparable to the CMS empirical “dijet function”.

Traditional way

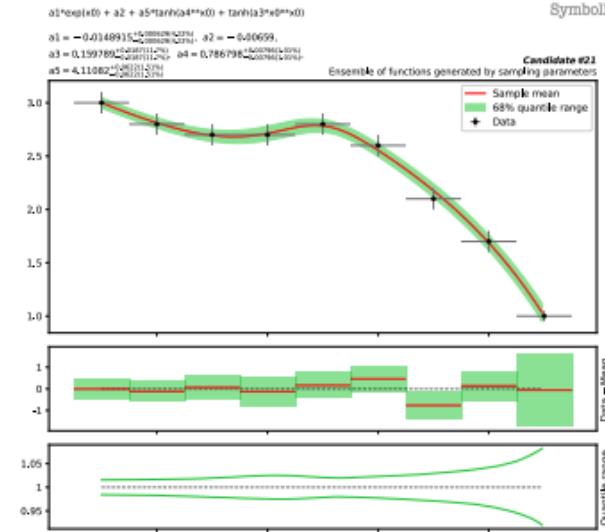
larger event samples. These are the modified dijet function (ModDijet-5p) given by $d\sigma/dm = p_0(1 - x^{1/3})^{p_1}/(x^{p_2+p_3 \log x + p_4 \log^2 x})$, the dijet function (Dijet-5p) given by $d\sigma/dm = p_0(1 - x)^{p_1}/(x^{p_2+p_3 \log x + p_4 \log^2 x})$, and the power-law times exponential function (PowExp-5p) given by $d\sigma/dm = p_0 e^{-p_1 x - p_2 x^2 - p_3 x^3}/x^{p_4}$, where $x = m/\sqrt{s}$, with m being the four-jet mass.

CMS paired-dijet [arxiv:2206.09997](https://arxiv.org/abs/2206.09997)

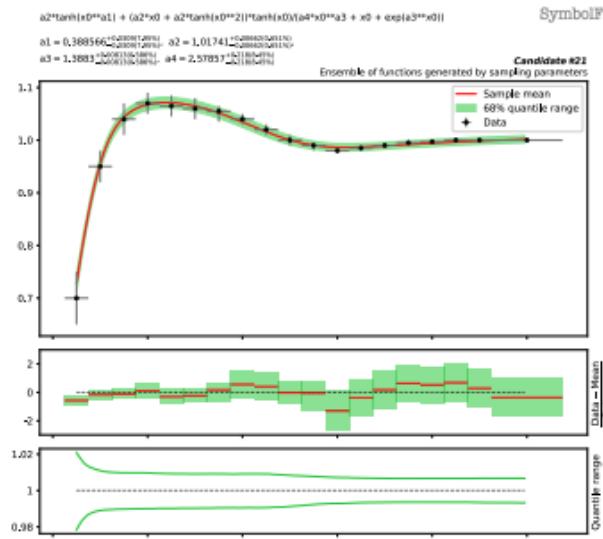
Examples



(a) Candidate function #13 for Toy Dataset 2a.



(b) Candidate function #21 for Toy Dataset 2b.



(c) Candidate function #21 for Toy Dataset 2c.

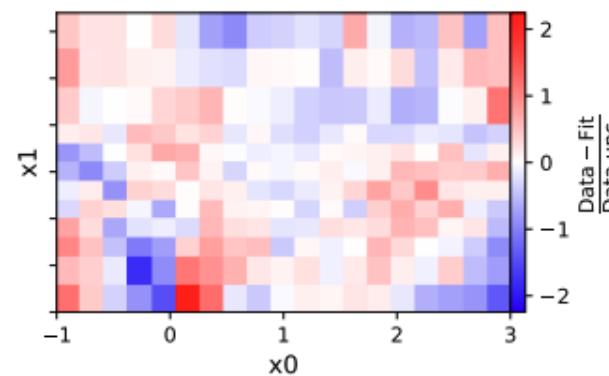
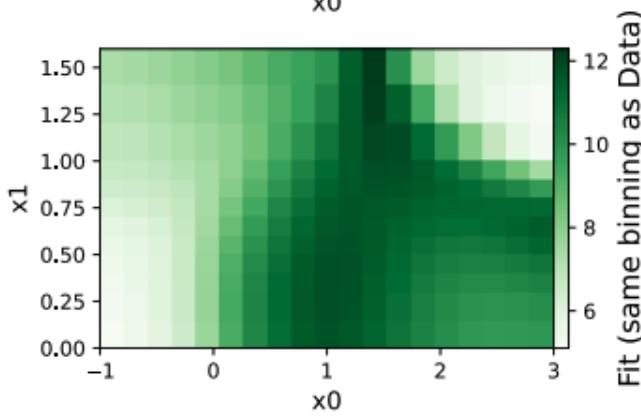
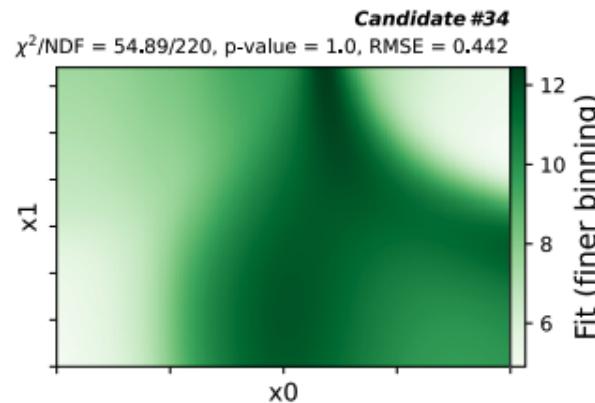
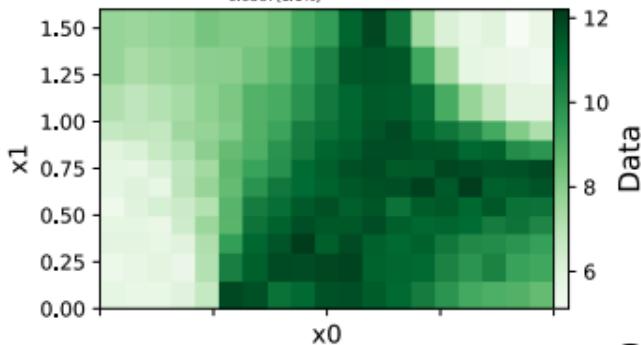
- Some “scale-factor-like” examples.

Examples

$a7*gauss(a2 + x0) + a9 + x1 + (a4 + gauss(a3 + x1**3*(a1 + x0)))*(a5*x0 + a5*tanh(a8*x0) + a6*gauss(x1))$

SymbolFit

$a1 = -1.2547^{+0.0236(1.88\%)}_{-0.0236(1.88\%)}, a2 = -0.907136^{+0.0209(2.3\%)}_{-0.0209(2.3\%)}$
 $a3 = -0.48079^{+0.0224(4.66\%)}_{-0.0224(4.66\%)}, a4 = -0.228784^{+0.0135(5.9\%)}_{-0.0135(5.9\%)}$
 $a5 = 1.47714^{+0.0371(2.51\%)}_{-0.0371(2.51\%)}, a6 = 2.34806^{+0.143(6.09\%)}_{-0.143(6.09\%)}$
 $a7 = 3.52694^{+0.105(2.98\%)}_{-0.105(2.98\%)}, a8 = 4.66,$
 $a9 = 5.1566^{+0.0567(1.1\%)}_{-0.0567(1.1\%)}$



- Works well for 2D data.
- Generally no problem for data with several variables.

Figure 16: Candidate function #34 for Toy Dataset 3c (see Tab. 4). The parameterized form of this function is shown at the top of the figure, along with the best-fit values and associated uncertainties. Upper left: the binned data being fitted. Lower left: the candidate function plotted with the same binning as the fitting data. Upper right: the candidate function plotted with a finer binning. Lower right: the residual error, $\frac{\text{Data} - \text{Fit}}{\text{Uncertainty}}$, in units of the data uncertainty.

Summary

- Recap: SymbolFit is a python API that interfaces PySR & LMFIT to streamline and automate parametric modeling using symbolic regression:
 1. Function forms searching [PySR]
 2. Dedicated re-optimization of functions [LMFIT]
 3. Uncertainty modeling [LMFIT]
 4. Evaluation and plotting [standalone]
- When in doubt, feel free to reach out to me, I will be very happy to clarify and assist in your use case 😊

 [hftsoi/symbolfit](#)

Automatic parametric modeling with symbolic regression

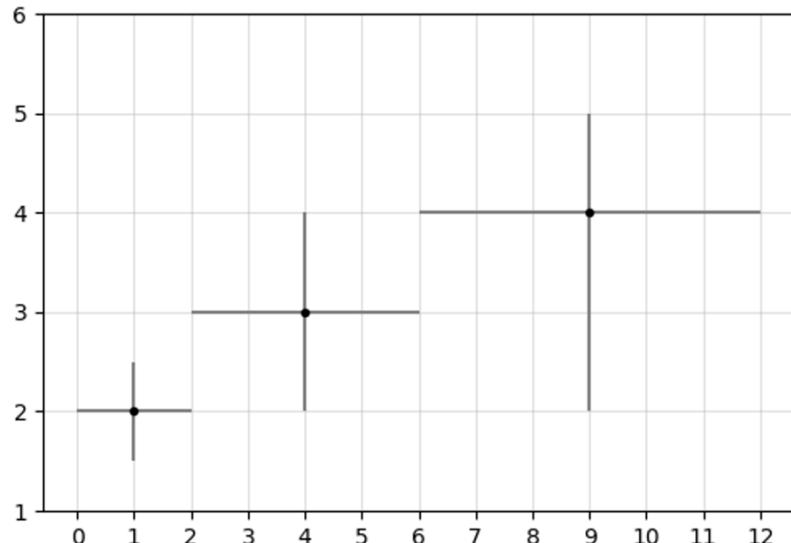
[python](#) [machine-learning](#) [automation](#) [interpolation](#) [symbolic-regression](#)

● Python · ⭐ 42 · Updated 23 days ago

[!\[\]\(735ab89543c6171e29ffb46ef228365d_img.jpg\) Unstar](#)

Backup

Input data format (1d)



`x = [1, 4, 9]`

`y = [2, 3, 4]`

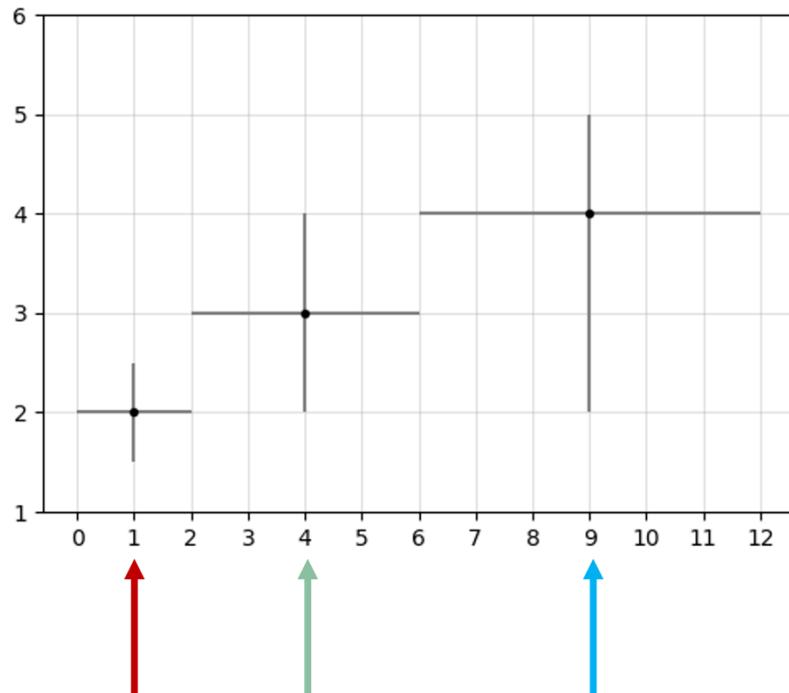
`y_up = [0.5, 1, 1]`

`y_down = [0.5, 1, 2]`

`bin_widths_1d = [2, 4, 6]`

See docs [here](#)

Input data format (1d)



x = [1, 4, 9]

y = [2, 3, 4]

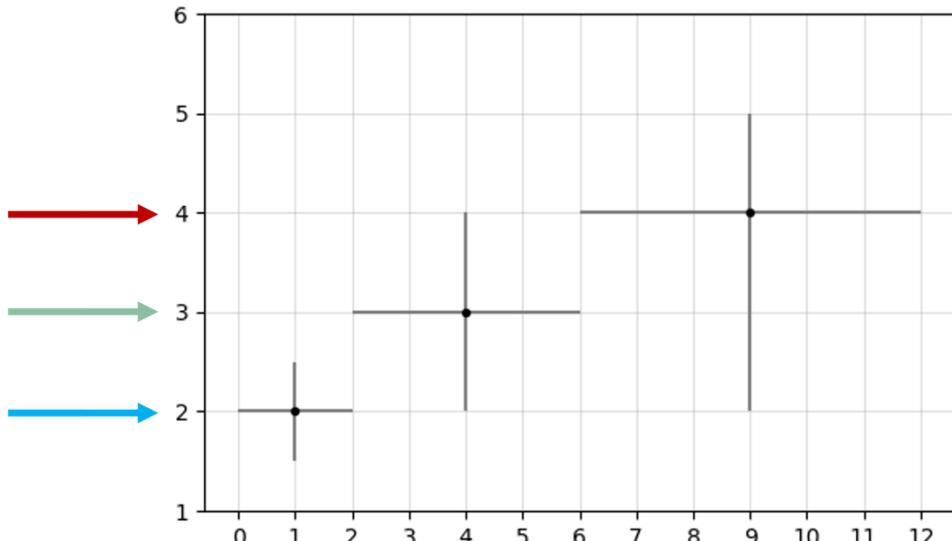
y_up = [0.5, 1, 1]

y_down = [0.5, 1, 2]

bin_widths_1d = [2, 4, 6]

See docs [here](#)

Input data format (1d)



$x = [1, 4, 9]$

$\mathbf{y} = [\color{red}{2}, \color{green}{3}, \color{blue}{4}]$

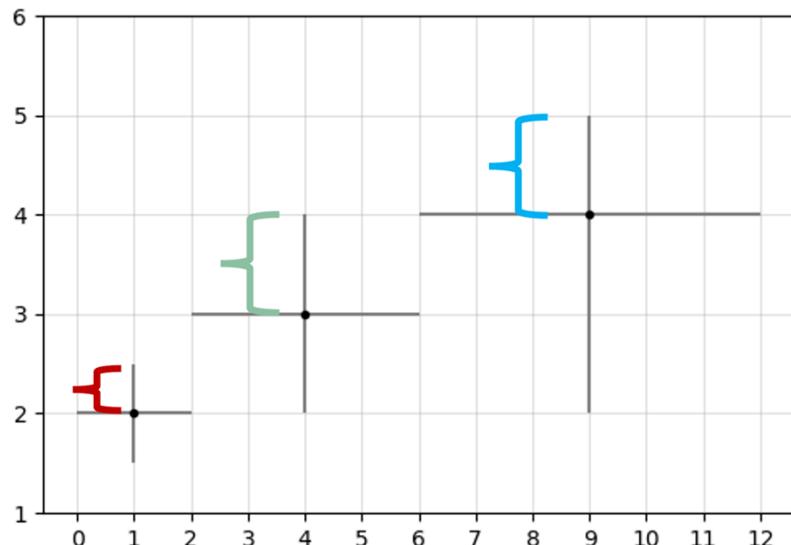
$y_{\text{up}} = [0.5, 1, 1]$

$y_{\text{down}} = [0.5, 1, 2]$

$\text{bin_widths_1d} = [2, 4, 6]$

See docs [here](#)

Input data format (1d)



$x = [1, 4, 9]$

$y = [2, 3, 4]$

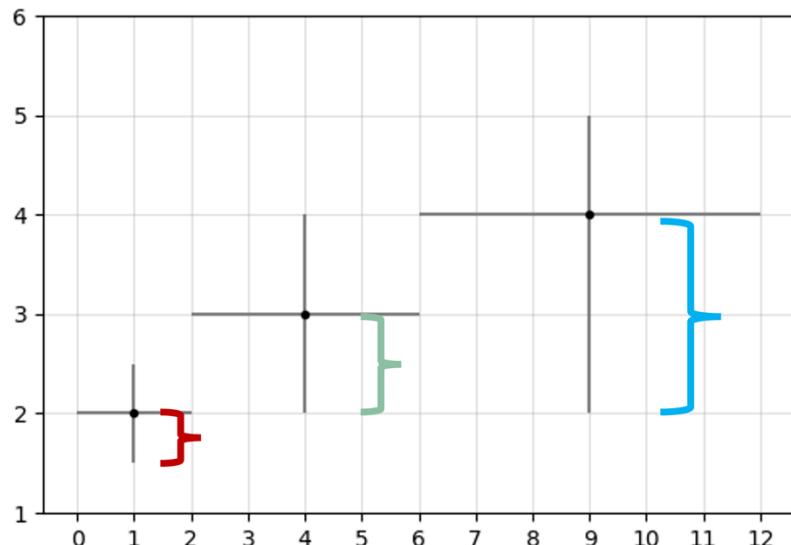
$y_{\text{up}} = [0.5, 1, 1]$

$y_{\text{down}} = [0.5, 1, 2]$

$\text{bin_widths_1d} = [2, 4, 6]$

See docs [here](#)

Input data format (1d)



$x = [1, 4, 9]$

$y = [2, 3, 4]$

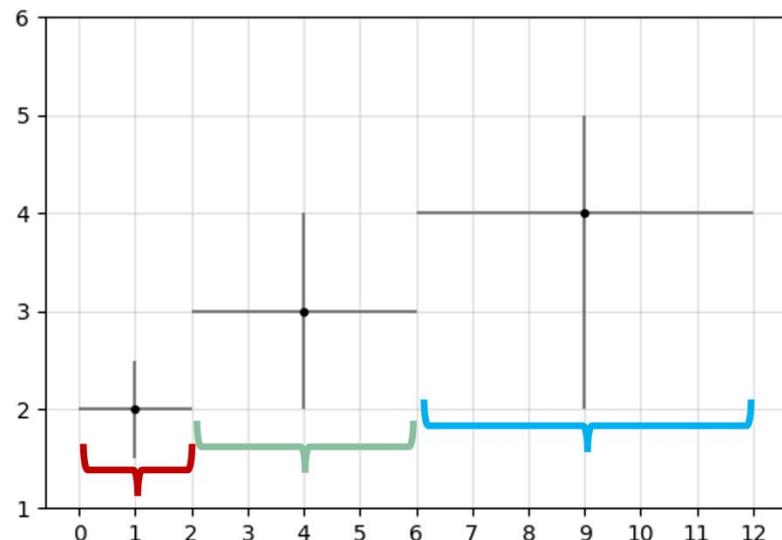
$y_{\text{up}} = [0.5, 1, 1]$

$y_{\text{down}} = [0.5, 1, 2]$

$\text{bin_widths_1d} = [2, 4, 6]$

See docs [here](#)

Input data format (1d)



`x = [1, 4, 9]`

`y = [2, 3, 4]`

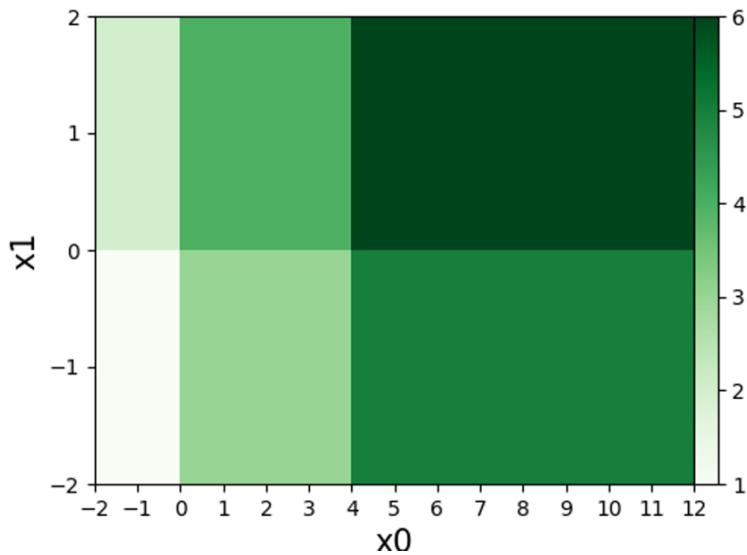
`y_up = [0.5, 1, 1]`

`y_down = [0.5, 1, 2]`

`bin_widths_1d = [2, 4, 6]`

See docs [here](#)

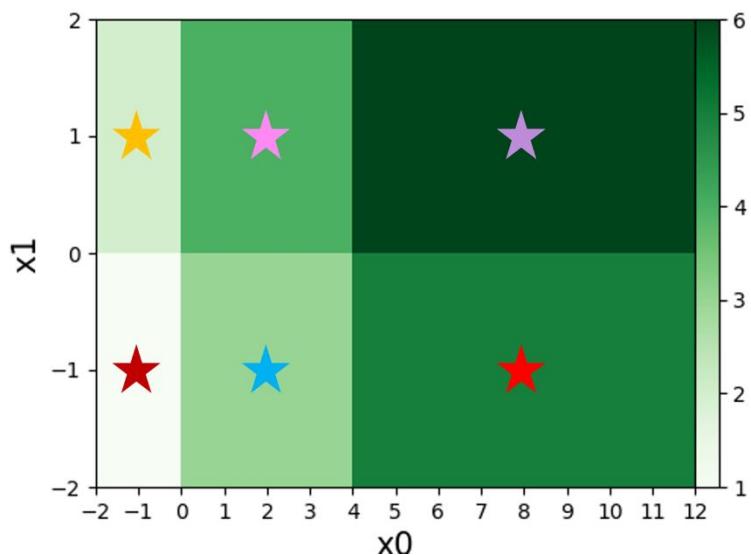
Input data format (2d)



```
x = [[-1, -1], [-1, 1], [2, -1], [2, 1], [8, -1], [8, 1]]  
bin_edges_2d = [  
    [-2, 0, 4, 12],  
    [-2, 0, 2]  
]  
y = np.array([1, 2, 3, 4, 5, 6])  
y_up = np.sqrt(y)  
y_down = np.sqrt(y)
```

See docs [here](#)

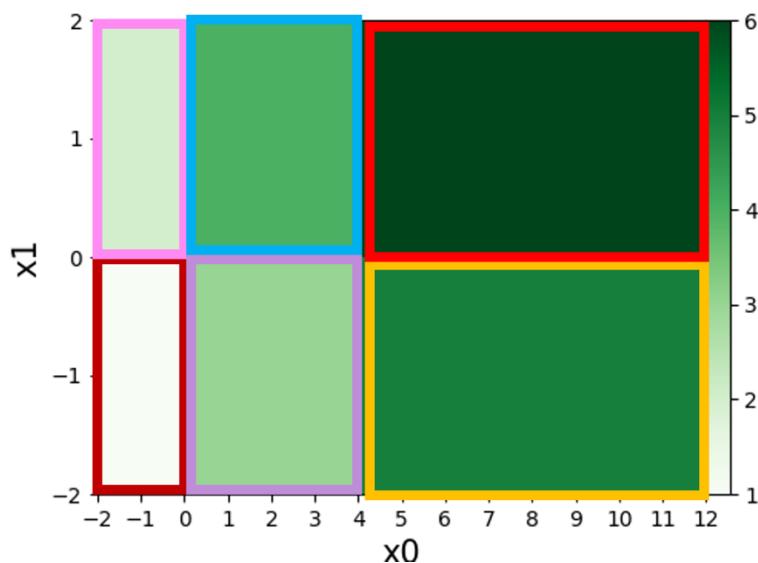
Input data format (2d)



```
x = [[-1, -1], [-1, 1], [2, -1], [2, 1], [8, -1], [8, 1]]  
bin_edges_2d = [  
    [-2, 0, 4, 12],  
    [-2, 0, 2]  
]  
y = np.array([1, 2, 3, 4, 5, 6])  
y_up = np.sqrt(y)  
y_down = np.sqrt(y)
```

See docs [here](#)

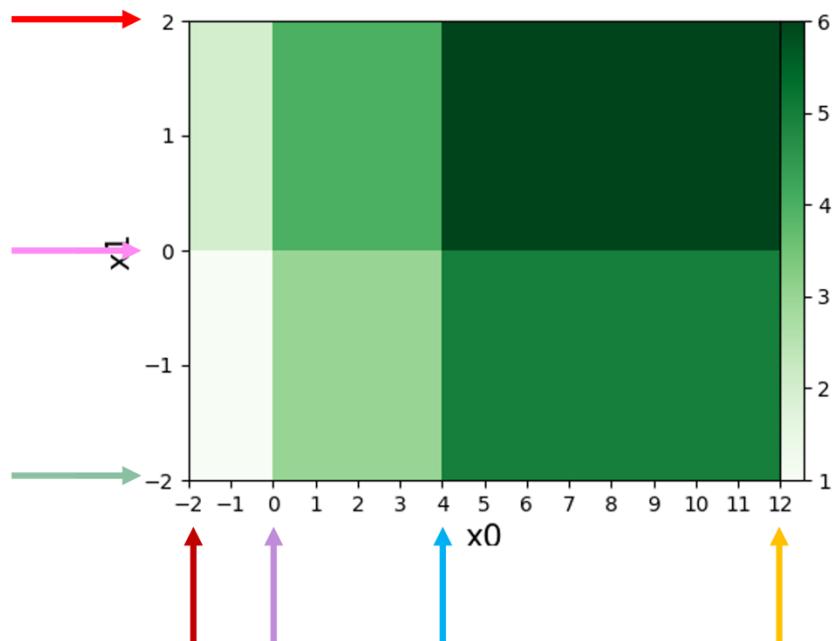
Input data format (2d)



```
x = [[-1, -1], [-1, 1], [2, -1], [2, 1], [8, -1], [8, 1]]  
bin_edges_2d = [  
    [-2, 0, 4, 12],  
    [-2, 0, 2]  
]  
y = np.array([1, 2, 3, 4, 5, 6])  
y_up = np.sqrt(y)  
y_down = np.sqrt(y)
```

See docs [here](#)

Input data format (2d)



```
x = [[-1, -1], [-1, 1], [2, -1], [2, 1], [8, -1], [8, 1]]  
bin_edges_2d = [  
    [-2, 0, 4, 12],  
    [-2, 0, 2]  
]  
y = np.array([1, 2, 3, 4, 5, 6])  
y_up = np.sqrt(y)  
y_down = np.sqrt(y)
```

See docs [here](#)