SymbolFit: Automatic parametric modeling with symbolic regression

Ho Fung Tsoi

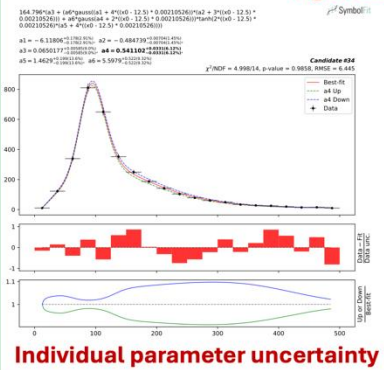
Mar 23, 2025

arxiv:2411.09851

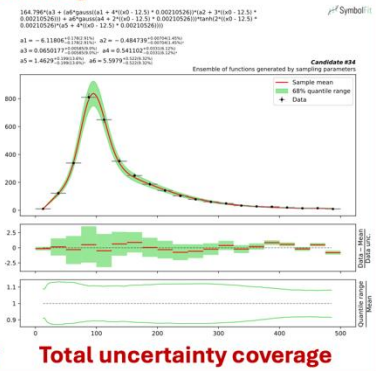
github:hftsoi/symbolfit

symbolfit.readthedocs.io

Signal modeling

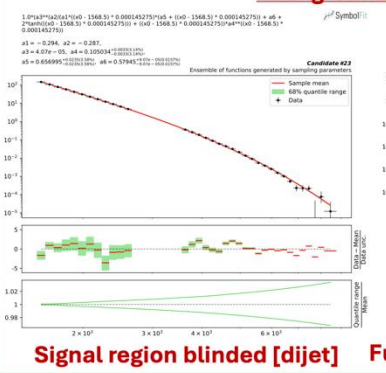


Individual parameter uncertainty

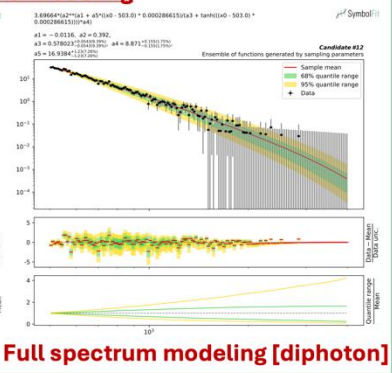


Total uncertainty coverage

Background modeling



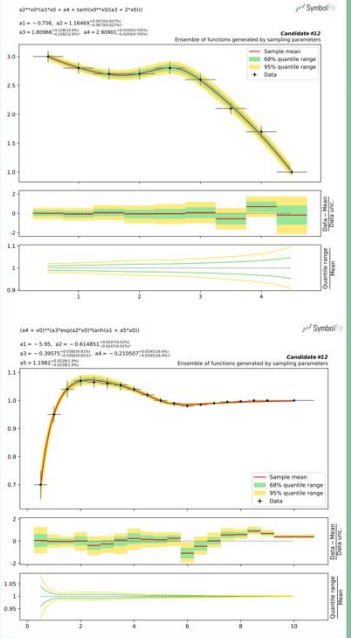
Signal region blinded [dijet]



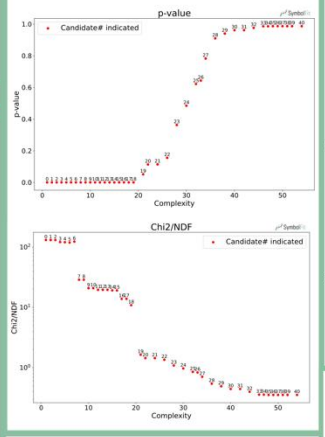
Full spectrum modeling [diphoton]

SymbolFit

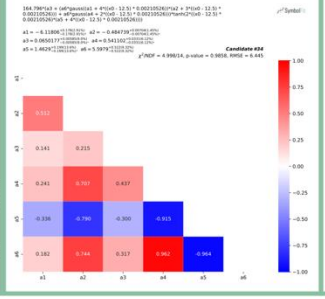
Smooth scale factor derivation



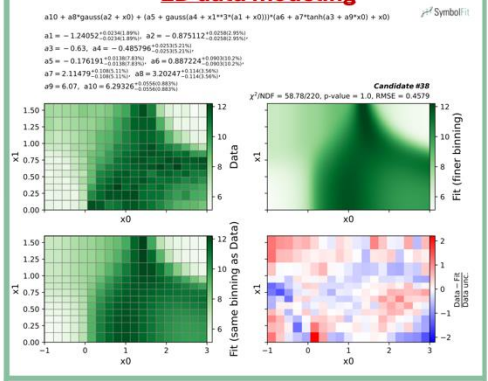
Goodness-of-fit scores



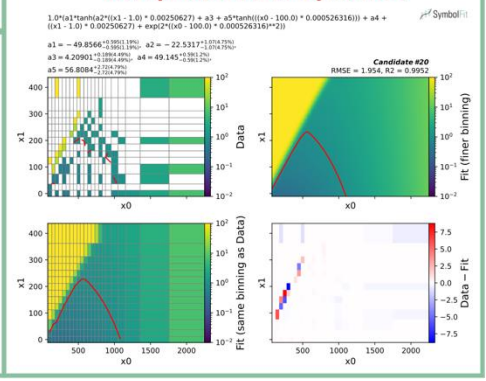
Correlation matrix



2D data modeling



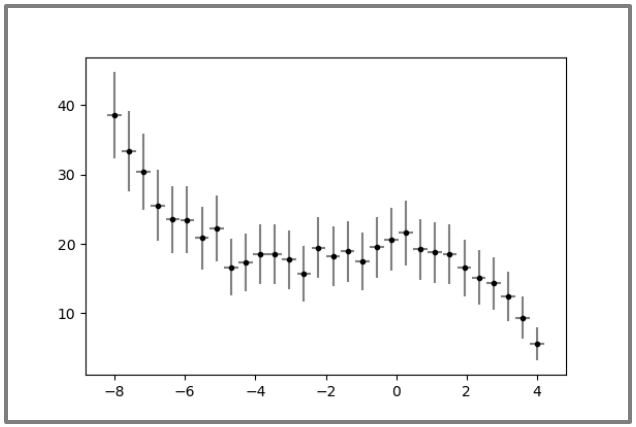
Interpolation/Extrapolation



Overview

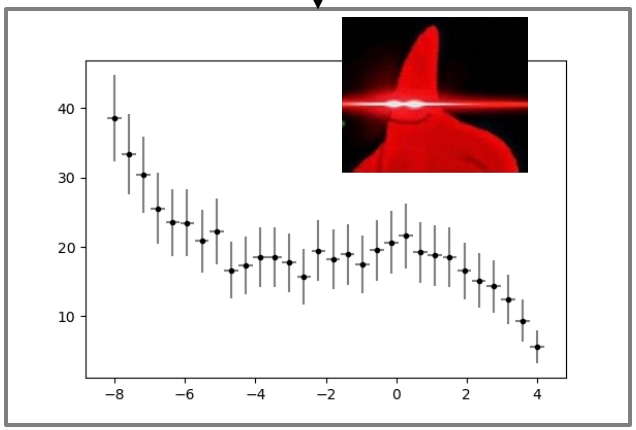
- Problem to be addressed
- What is symbolic regression
- SymbolFit framework
- A few examples

Traditional way of parametric modeling



Traditional way of parametric modeling

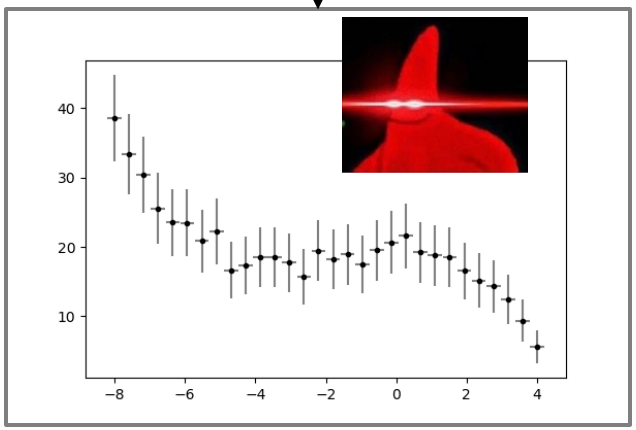
Stare at the distribution shape



Traditional way of parametric modeling

Stare at the distribution shape

Think hard and guess an adequate functional form as fit template

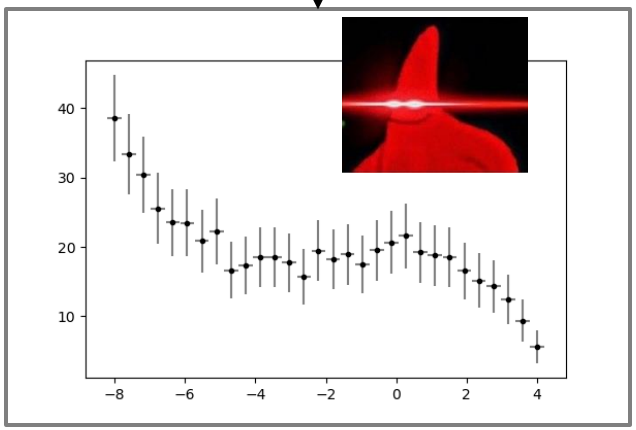


Traditional way of parametric modeling

Stare at the distribution shape

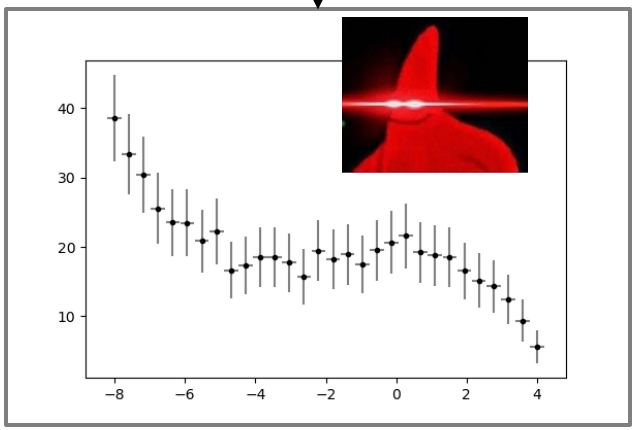
Think hard and guess an adequate functional form as fit template

Perform fit per function hyperparameter



Traditional way of parametric modeling

Stare at the distribution shape



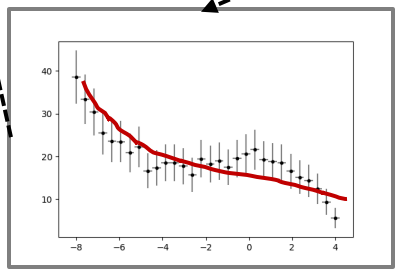
Think hard and guess an adequate functional form as fit template

Perform fit per function hyperparameter

Compare the fitted functions with data

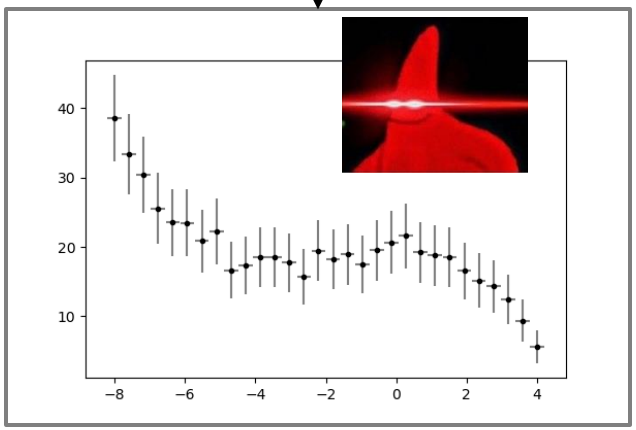
Bad fit?

Iterate again



Traditional way of parametric modeling

Stare at the distribution shape



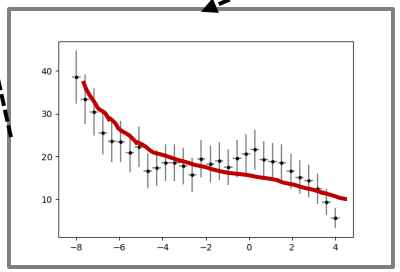
Think hard and guess an adequate functional form as fit template

Iterate again

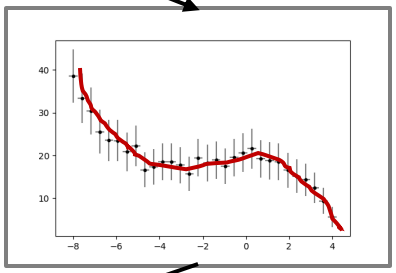
Perform fit per function hyperparameter

Compare the fitted functions with data

Bad fit?



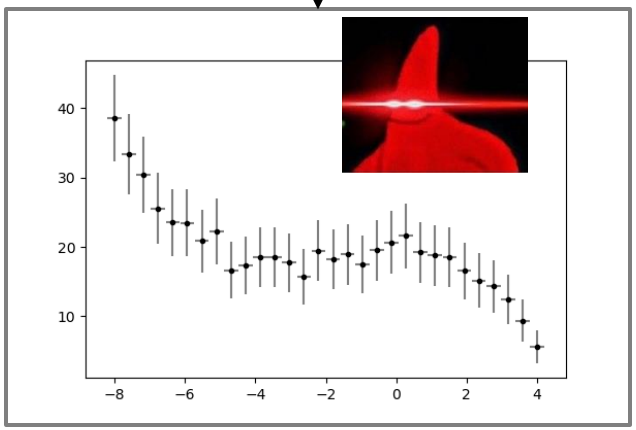
Good fit? (finally)



Perform statistical analysis such as F-test to fix the final function hyperparameter

Traditional way of parametric modeling

Stare at the distribution shape



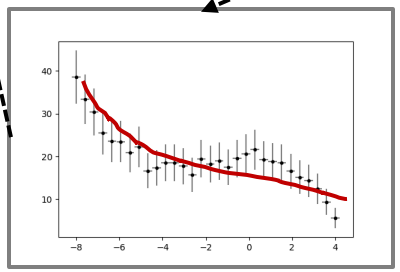
Think hard and guess an adequate functional form as fit template

Iterate again

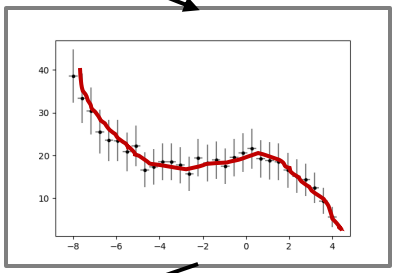
Perform fit per function hyperparameter

Compare the fitted functions with data

Bad fit?



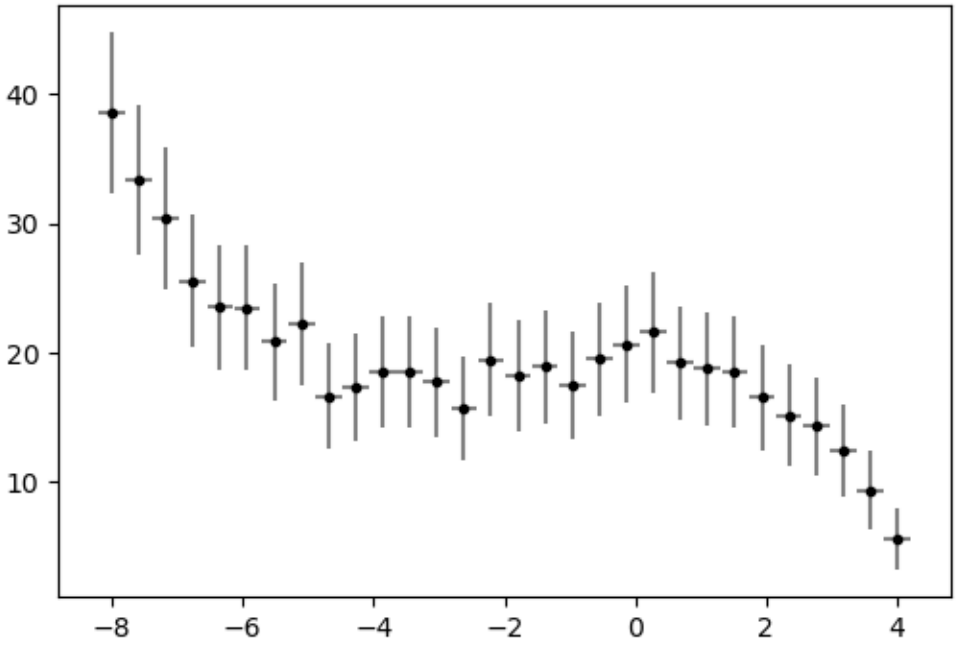
Good fit? (finally)



Perform statistical analysis such as F-test to fix the final function hyperparameter

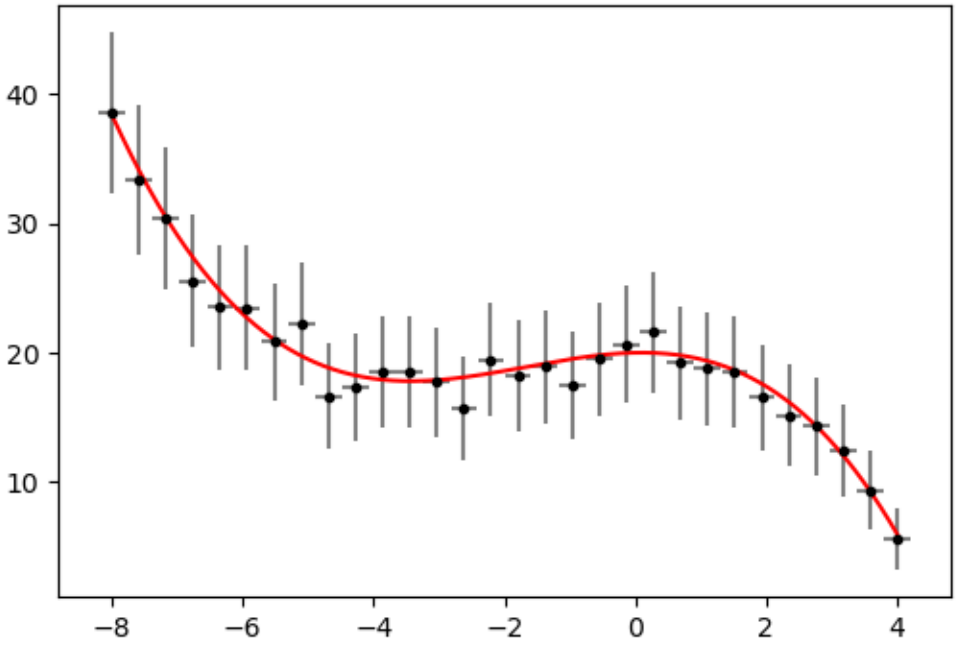
Final function as your model

Traditional way of parametric modeling



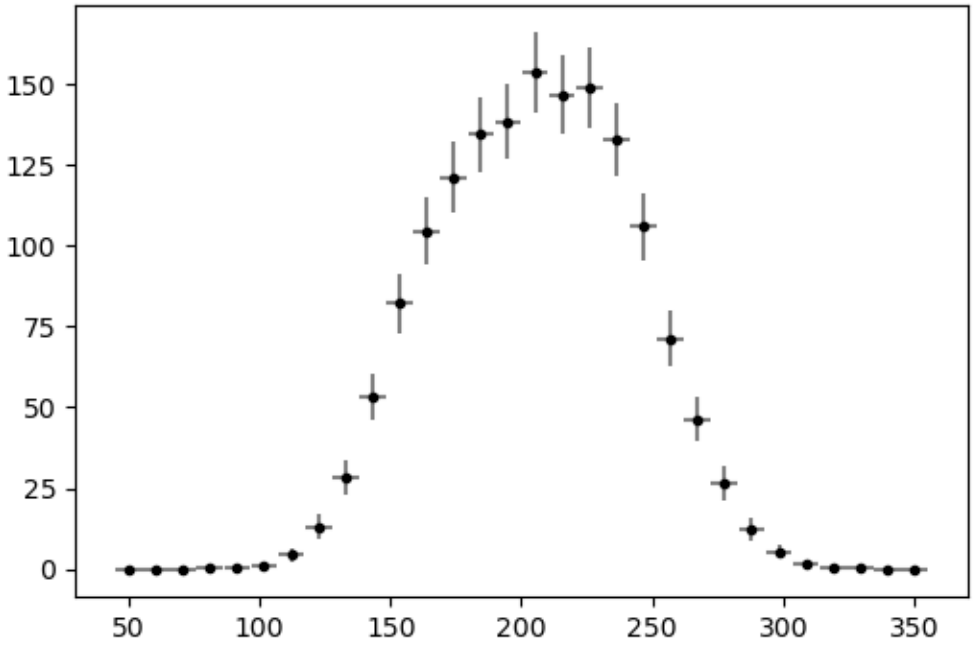
- A simple enough distribution.
- Guess a polynomial template should be sufficient to fit it.

Traditional way of parametric modeling



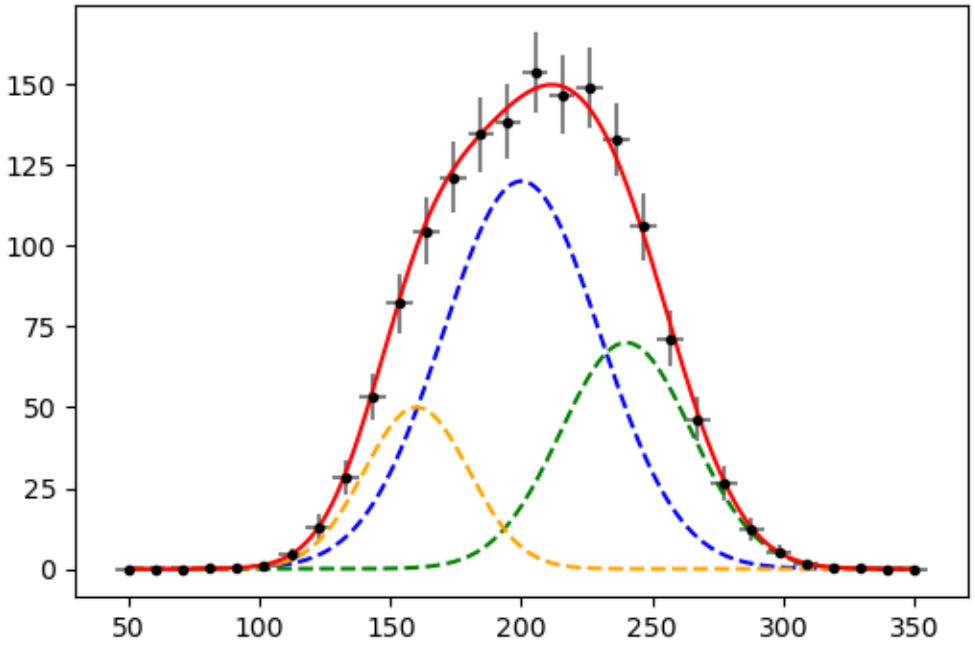
- Indeed, a 3rd order polynomial is sufficient!

Traditional way of parametric modeling



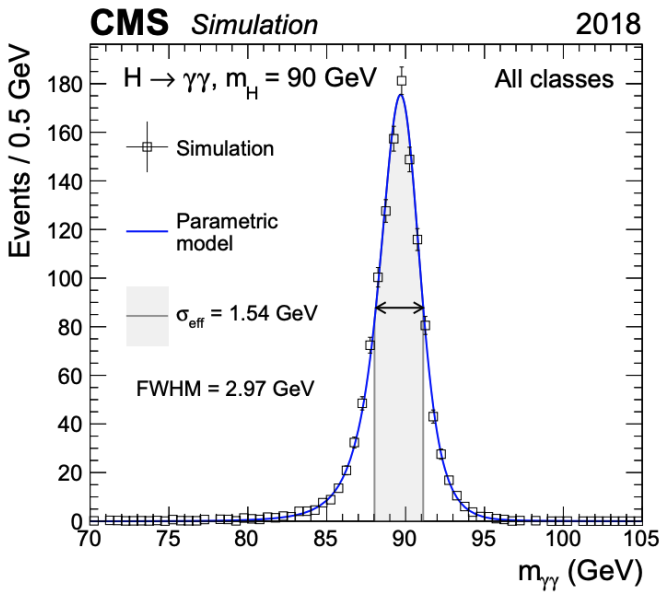
- How about this?
- Looks like a Gaussian, but not exactly if look closer, as it has a wider peak…
- Natural guess: multiple Gaussians stacked together.

Traditional way of parametric modeling



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

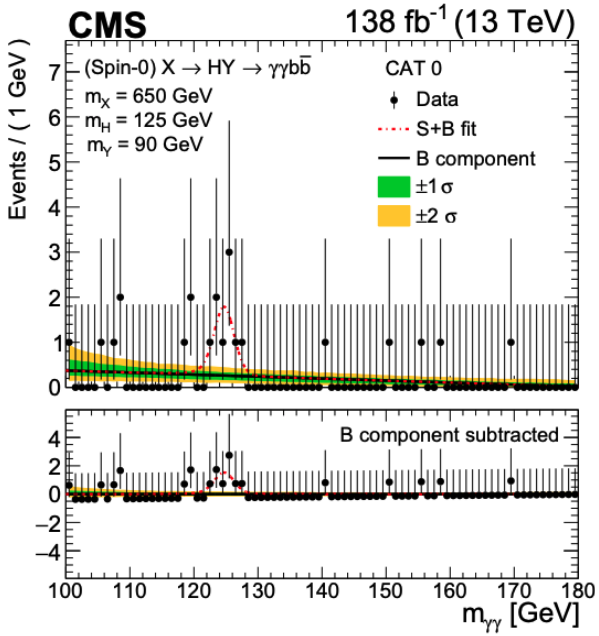
Traditional way of parametric modeling



arxiv:2405.18149

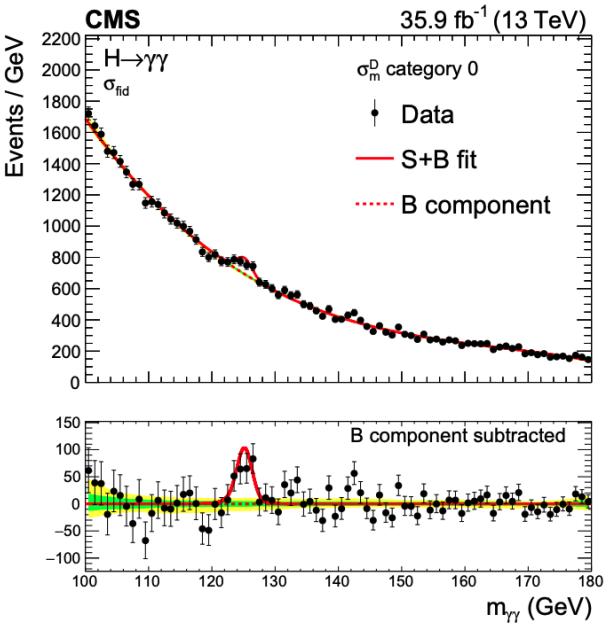
"...is modeled empirically with a sum of between two and four Gaussian functions..."

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



arxiv:2310.01643

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

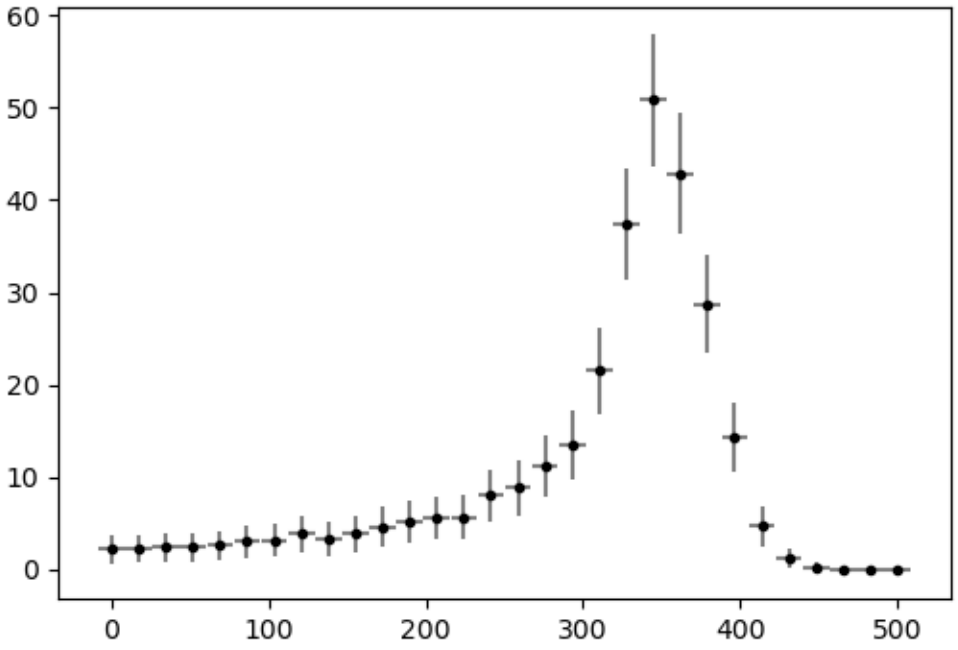


arxiv:1807.03825

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

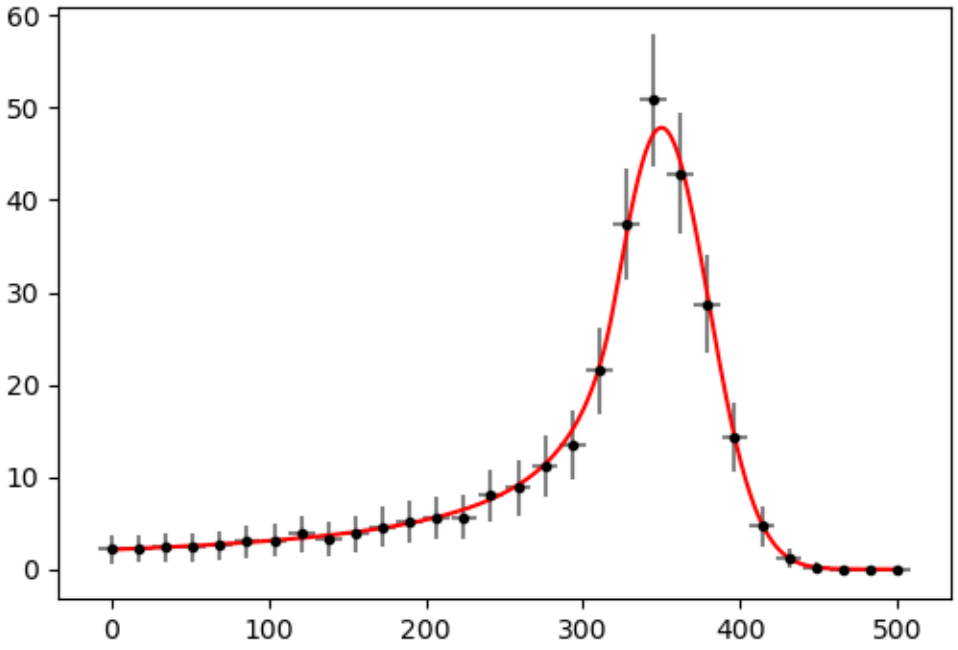
- Sum of Gaussians – a common template to model peaked signals in CMS/ATLAS analyses.
- It's about luck, as it may not always work well!

Traditional way of parametric modeling



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

Traditional way of parametric modeling



- 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:

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

where

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

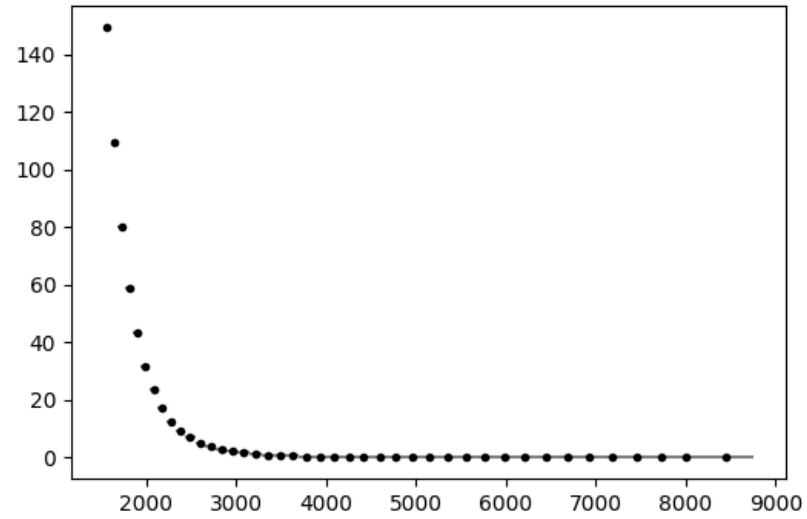
$$B = \frac{n}{|\alpha|} - |\alpha|,$$

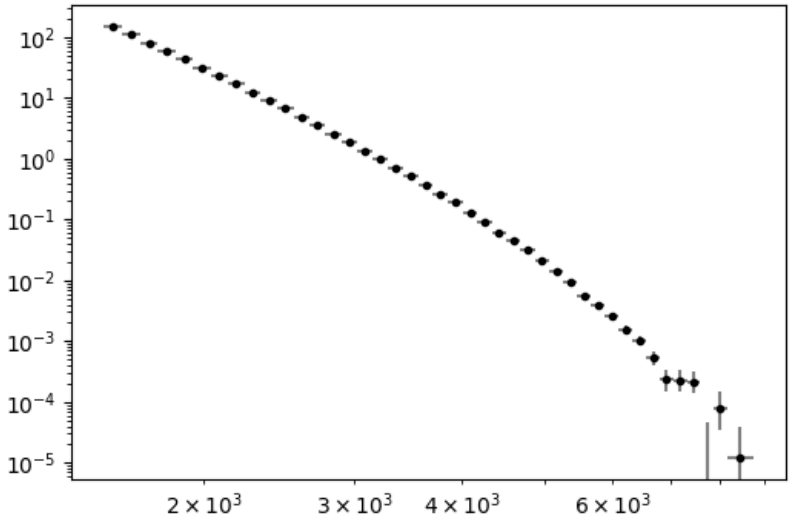
$$N = \frac{1}{\sigma(C+D)},$$

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

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

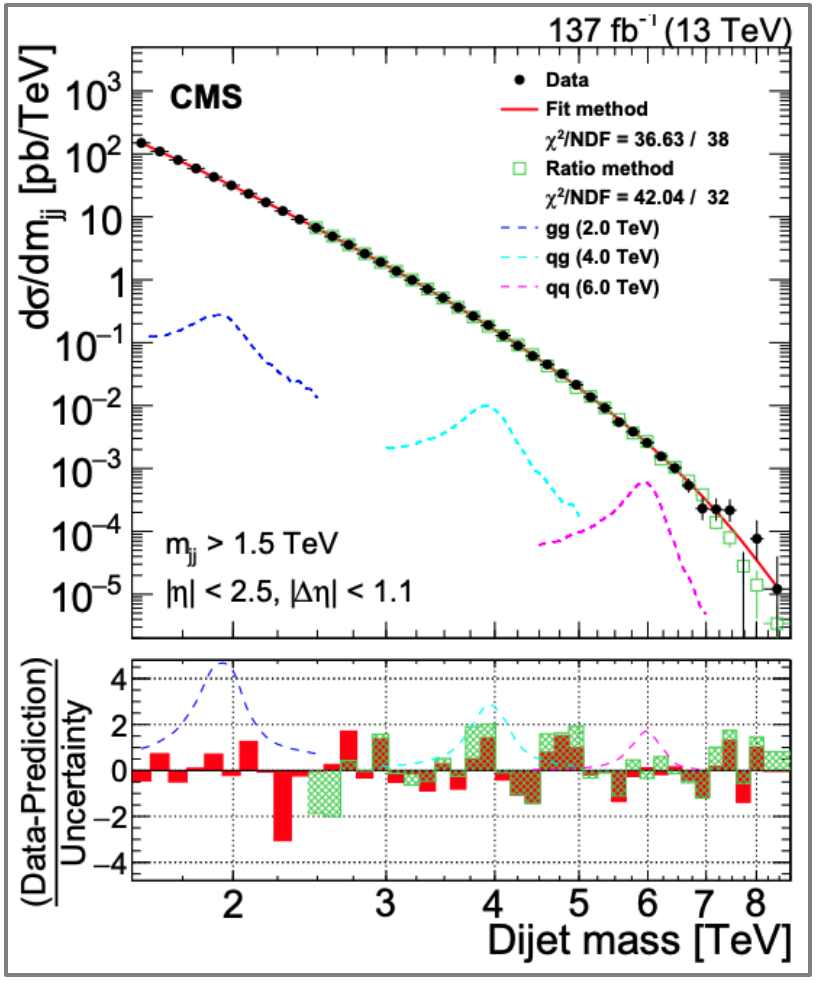
Traditional way of parametric modeling





- How about this, the CMS dijet spectrum? arxiv:1911.03947
- Looks like a steeply falling shape, a simple exponential is not sufficient…

Traditional way of parametric modeling



arxiv: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

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

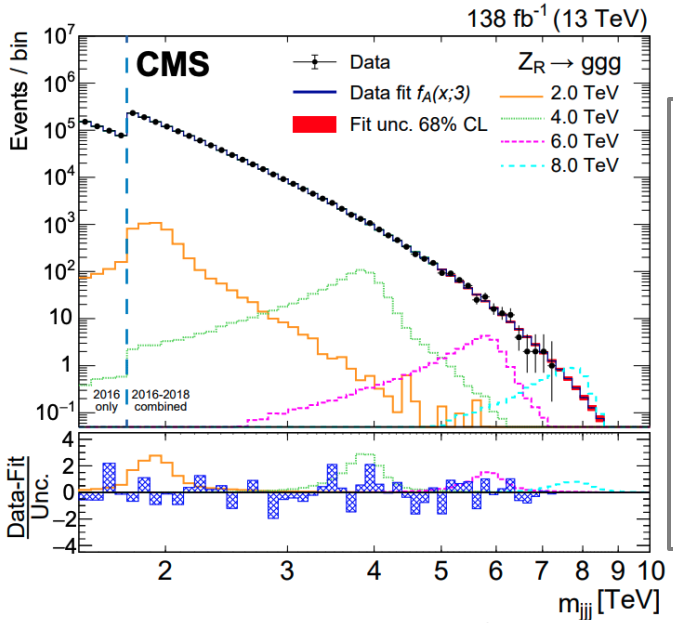
where $x = m_{jj}/\sqrt{s}$, and P_0, P_1, P_2, and P_3 are four free parameters. The search for resonances

- CMS 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 current data anymore.

arxiv:2206.09997

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

Traditional way of parametric modeling



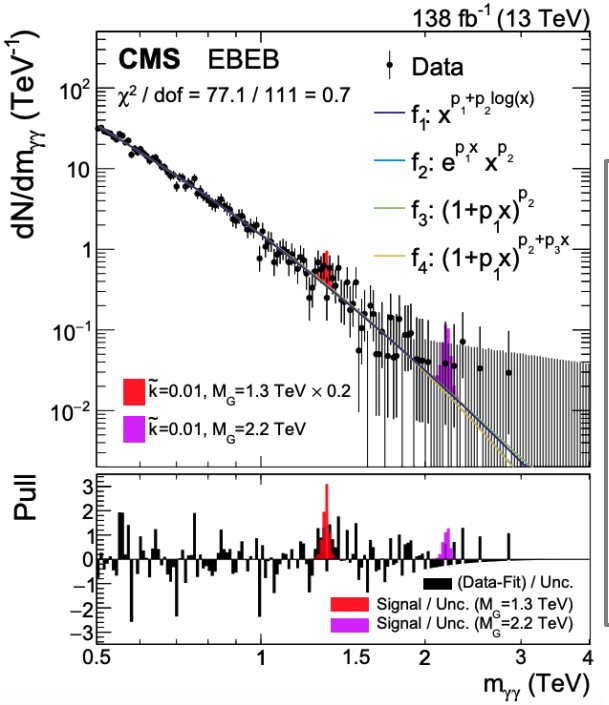
CMS High-mass trijet search

arxiv: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:

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

where $x = m_{jjj}/\sqrt{s}$, p_i $(i = 0, 1, \ldots, N)$ 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

arxiv:2405.09320

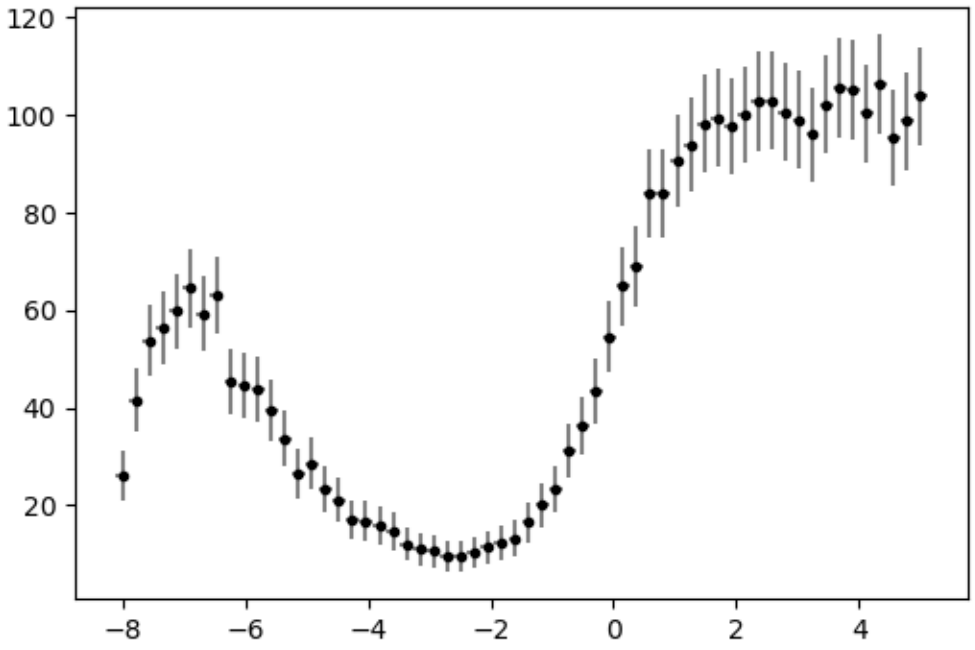
5.1 Resonant search background

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:

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

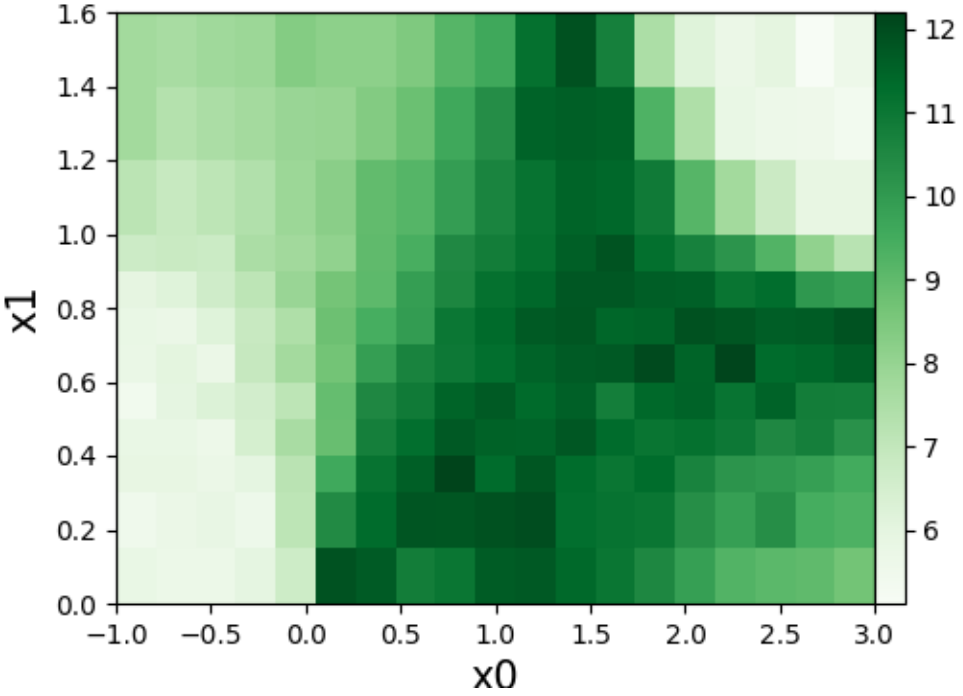
where p_i are unconstrained nuisance parameters in the hypothesis test. During the likelihood

Traditional way of parametric modeling



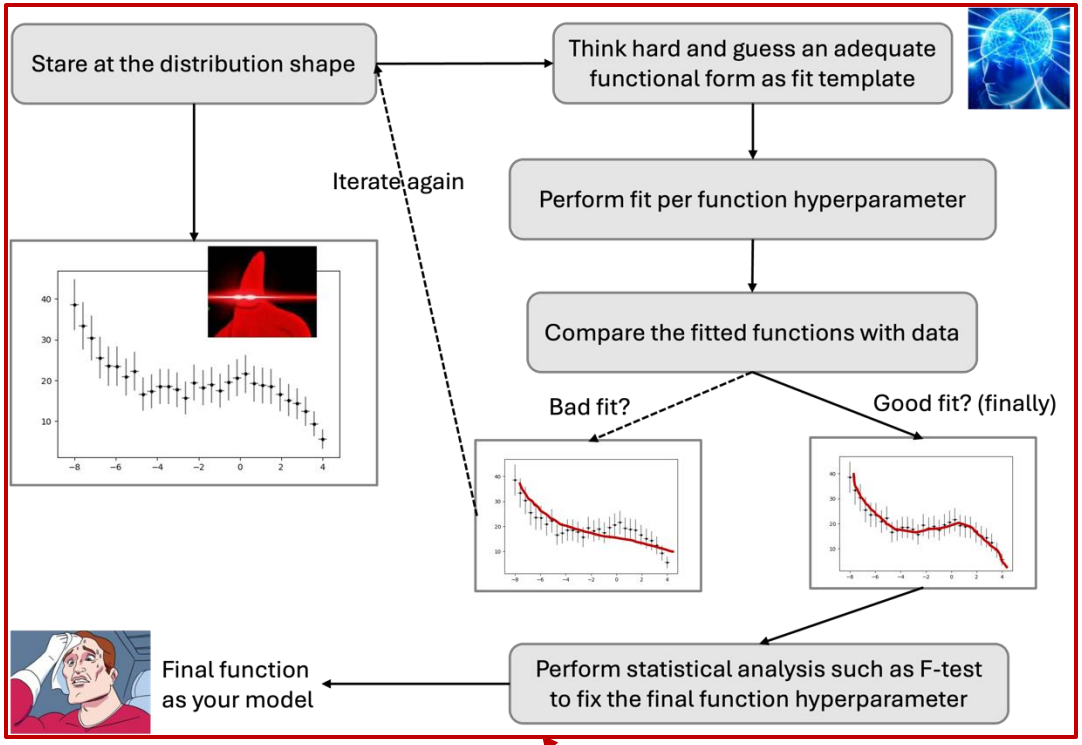
- How about this?
- No clue until spending hours or even longer with trial and error…

Traditional way of parametric modeling

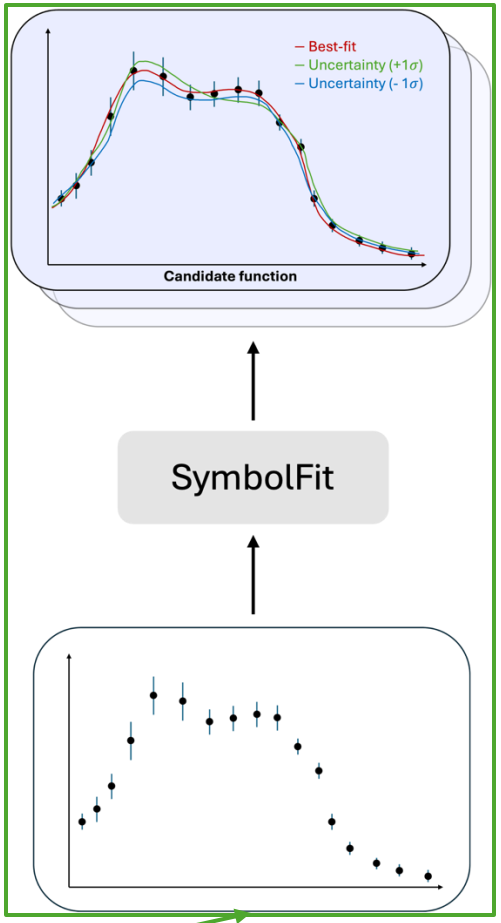


- How about this?
- No clue until spending hours or even longer with trial and error…

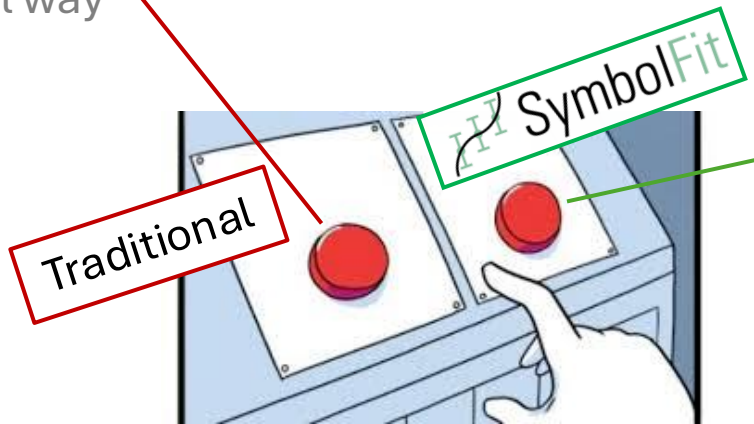
Now time to make the switch



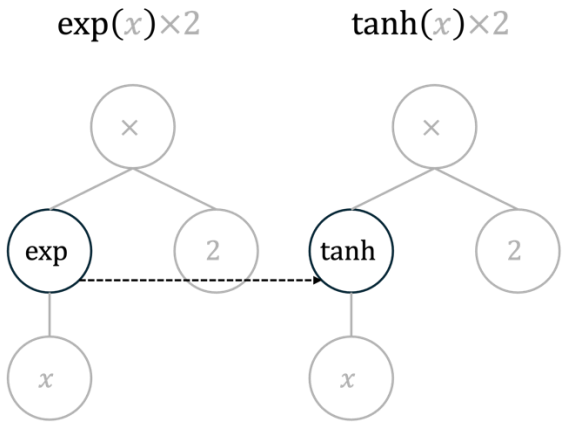
Manual way



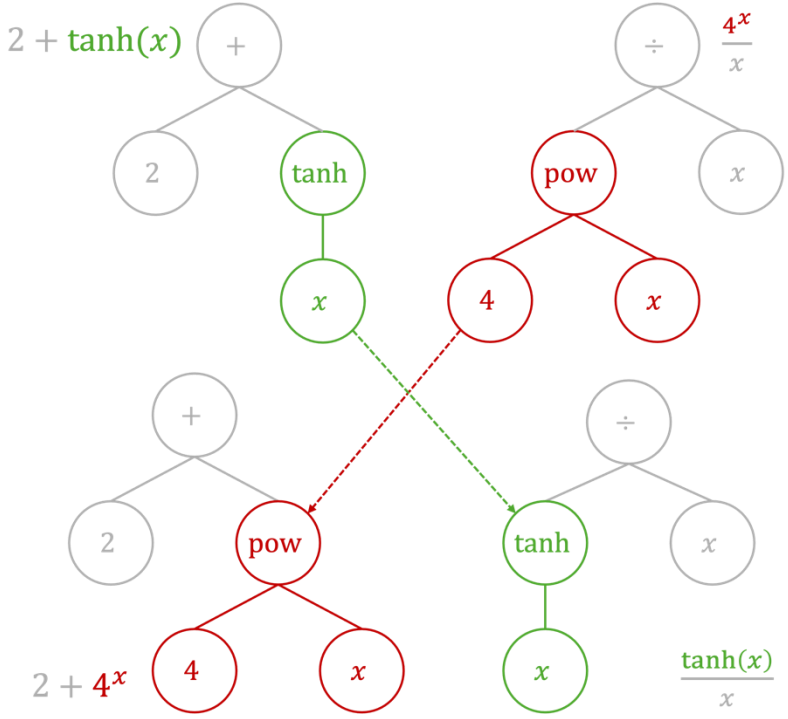
Automatic way



What is symbolic regression?



Node mutation



Subtree crossover

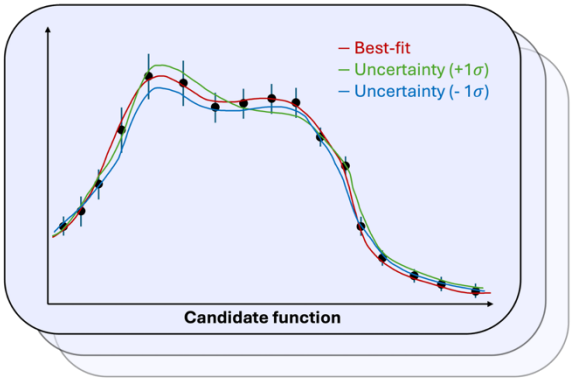
- *Machine-search* for closed-form functions to fit data.
- Functional form is itself a “trainable parameter”.
- A vast function space is being searched using, e.g., genetic programming.
- An exact functional form does not need to be known a priori at all!

What is symbolic regression?

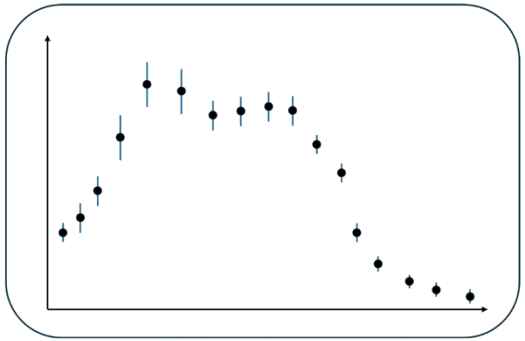
See the animation here

What is SymbolFit?

Output
(Multiple candidate functions, each with uncertainty measure)



SymbolFit



Input
(*Without a pre-defined functional form*)

hftsoi/symbolfit

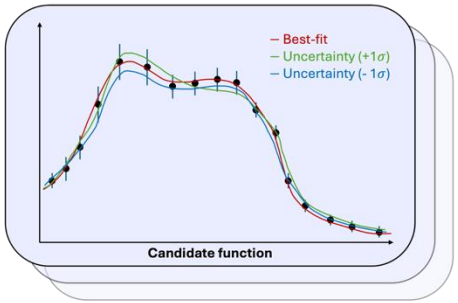
Automatic parametric modeling with symbolic regression

python | machine-learning | automation | interpolation | symbolic-regression

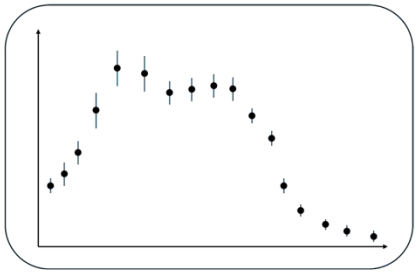
- A high-level python API
- Automates parametric modeling
  - Simple, flexible, and highly configurable
  - Functional forms are searched by machine
  - Functions are then re-optimized
  - Uncertainty estimates are provided
  - Evaluations and plotting are done automatically
- All in one go!

Internal steps of SymbolFit

Output
(Multiple candidate functions, each with uncertainty measure)



SymbolFit



Input
(*Without a pre-defined functional form*)

$\tanh(1.2x)\exp(-3.4x + \cdots$

***Machine-search* for functional forms**

Symbolic regression [PySR]

Configuration
(Operator specifications, constraints, loss,...)

Evaluation

Parameterization

Re-optimized functions with uncertainty measure

$\{a_1: 1.5^{+2\%}_{-3\%}, a_2: -4^{+1.6\%}_{-1.2\%}, \ldots\}$

Correlation

$\{(a_1, a_2): 0.8, \ldots\}$

Nonlinear least-square minimization [LMFIT]

Refit configuration

Refit loop

$\tanh(a_1 x)\exp(a_2 x + \cdots$

$\{a_1: 1.2, a_2: -3.4, \ldots\}$

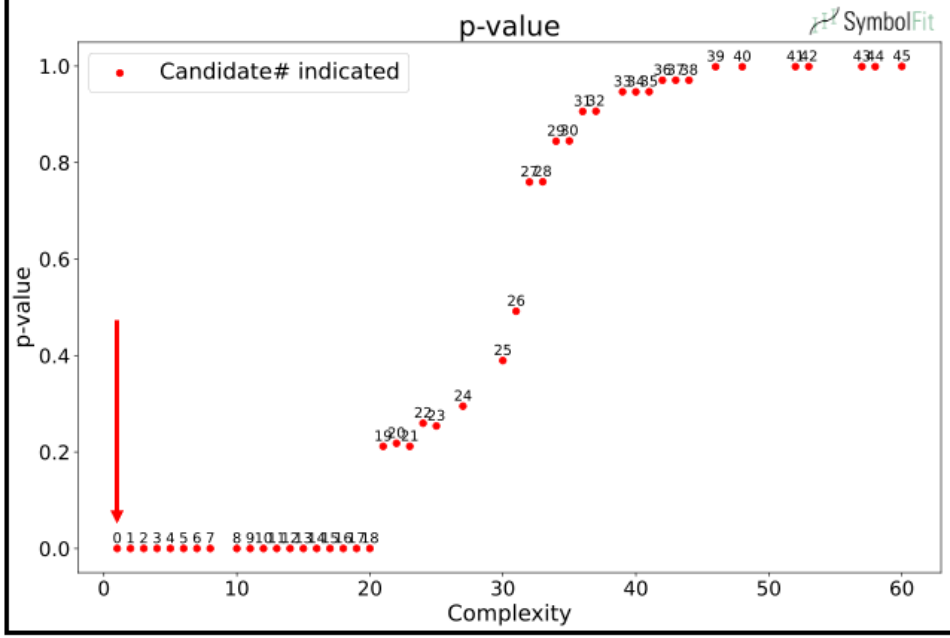
Convergence of candidate functions in a fit

164.796*(a1)

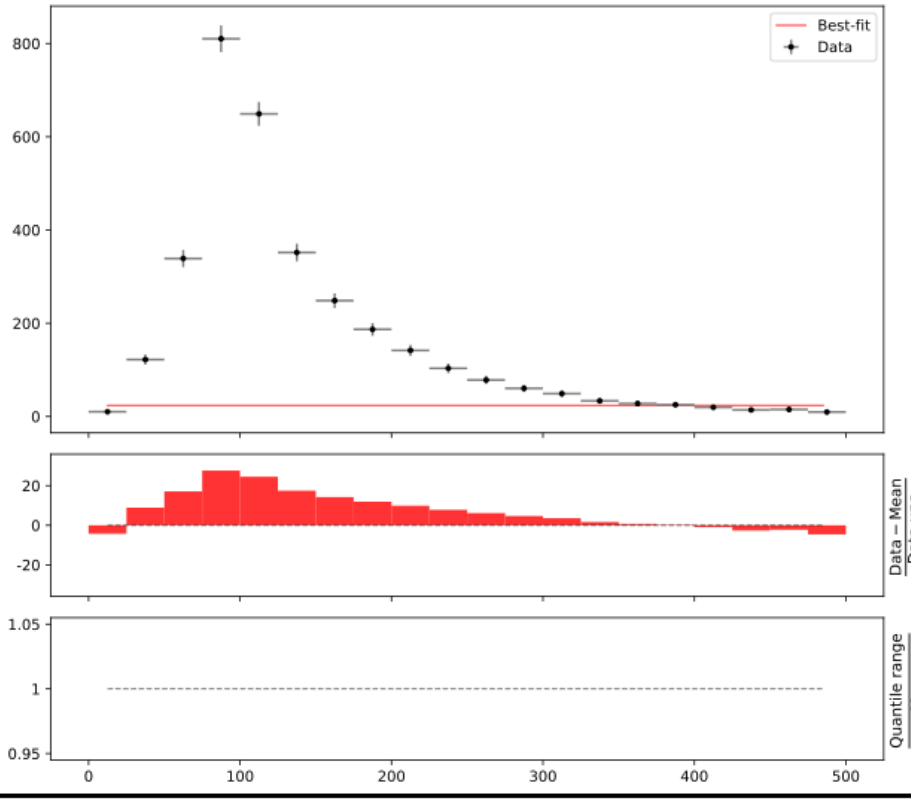
^ Candidate function in parameterized form {a1, a2, a3, ...}

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

```
from pysr import PySRRegressor

pysr_config = PySRRegressor(
    model_selection = "accuracy",
    timeout_in_seconds = 60*100,
    niterations = 200,
    maxsize = 80,
    binary_operators = [
        "+", "*", "/", "^"
                    ],
    unary_operators = [
        "exp",
        "tanh",
    ],
    nested_constraints = {
        "exp":    {"exp": 0, "tanh": 0, "*": 2, "/": 1, "^": 1},
        "tanh":   {"exp": 0, "tanh": 0, "*": 2, "/": 1, "^": 1},
        "*":      {"exp": 1, "tanh": 1, "*": 2, "/": 1, "^": 1},
        "^":      {"exp": 1, "tanh": 1, "*": 2, "/": 1, "^": 0},
        "/":      {"exp": 1, "tanh": 1, "*": 2, "/": 0, "^": 1},
    },
    loss="loss(y, y_pred, weights) = (y - y_pred)^2 * weights",
)
```

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!

See docs here and here

Function space configuration (template spec.)

```
from pysr import PySRRegressor, TemplateExpressionSpec

expression_spec = TemplateExpressionSpec(
    'p[1] * f(x/13000) ^ g(log(x/13000))',
    expressions = ['f', 'g'],
    variable_names = ['x'],
    parameters = {'p': 1}
)

pysr_config = PySRRegressor(
    expression_spec = expression_spec,
    model_selection = 'accuracy',
    niterations = 200,
    maxsize = 40,
    binary_operators = ['+', '*'],
    elementwise_loss='loss(y, y_pred, weights) = (y - y_pred)^2 * weights',
)
```

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).

See docs here and 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	$\mathrm{pow}(149(0.0101x+0.0101\tanh(0.171+0.724x)), x+(2.38x\tanh(-0.71+x)+2.39)\tanh(x)+\tanh(x))$	5	214.8 / 30	30.93 / 30	0.419

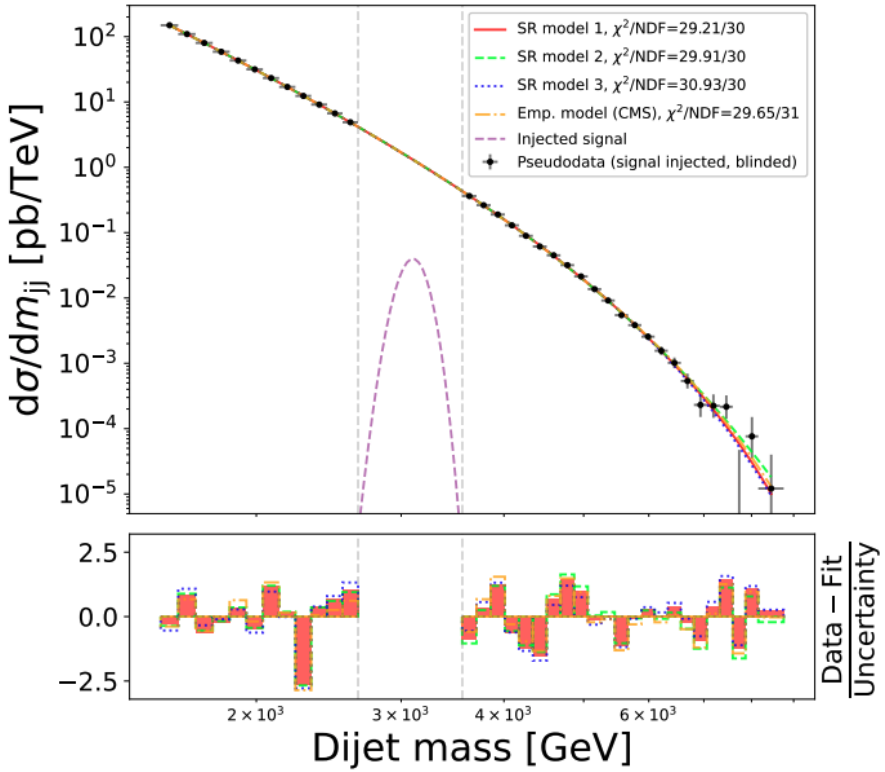


Figure 9: Pseudodata of the dijet spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. 2) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

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

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

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

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

CMS dijet arxiv:1911.03947

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

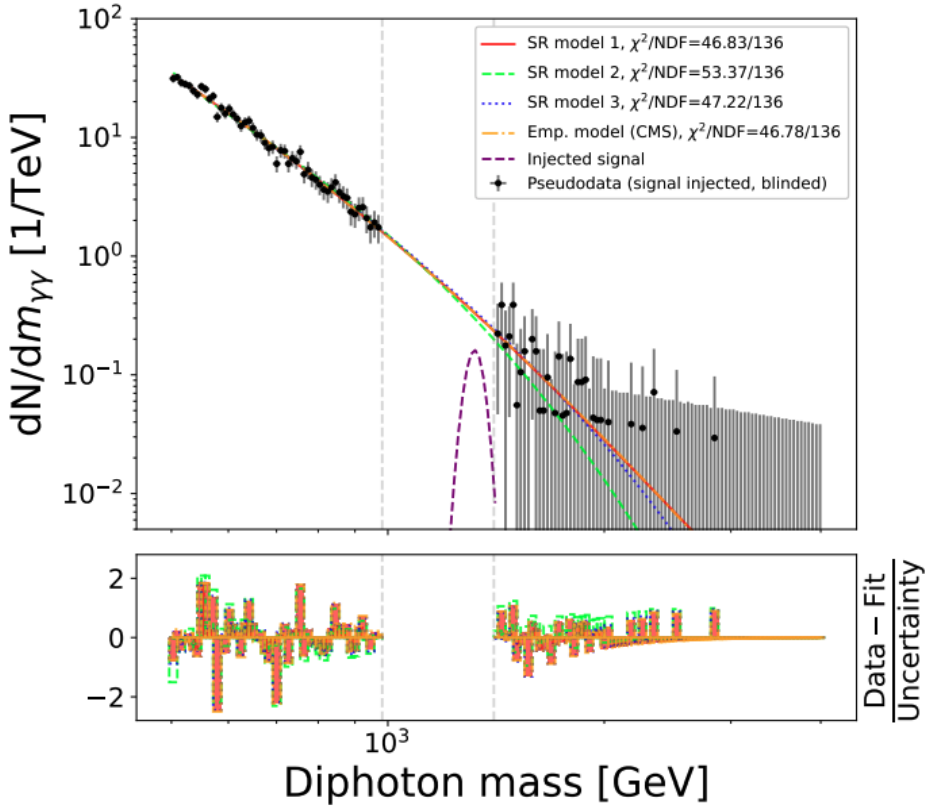


Figure A5: Pseudodata of the diphoton spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. A2) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

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

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:

$$\begin{aligned} f_1(x) &= p_0 x^{p_1+p_2\log(x)}, \\ f_2(x) &= p_0 e^{p_1 x} x^{p_2}, \\ f_3(x) &= p_0(1+xp_1)^{p_2}, \\ f_4(x) &= p_0(1+xp_1)^{p_2+p_3 x}, \end{aligned} \quad (4)$$

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

CMS diphoton arxiv:2405.09320

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

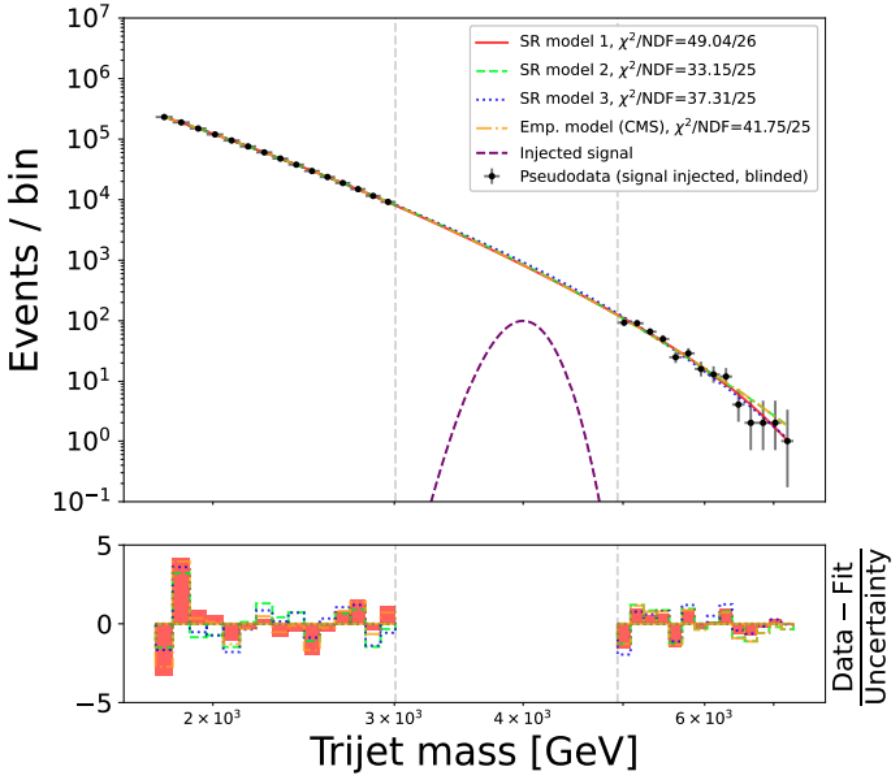


Figure A9: Pseudodata of the trijet spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. A4) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

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

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:

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

where $x = m_{jjj}/\sqrt{s}$, p_i $(i = 0, 1, \ldots, N)$ 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 trijet arxiv:2310.14023

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

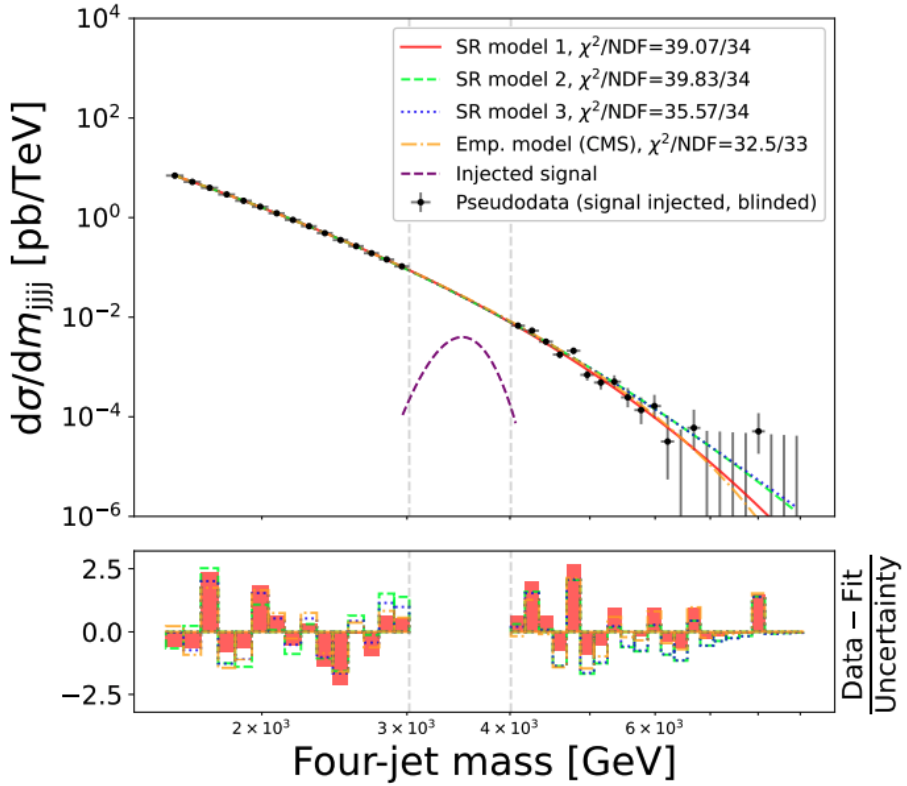


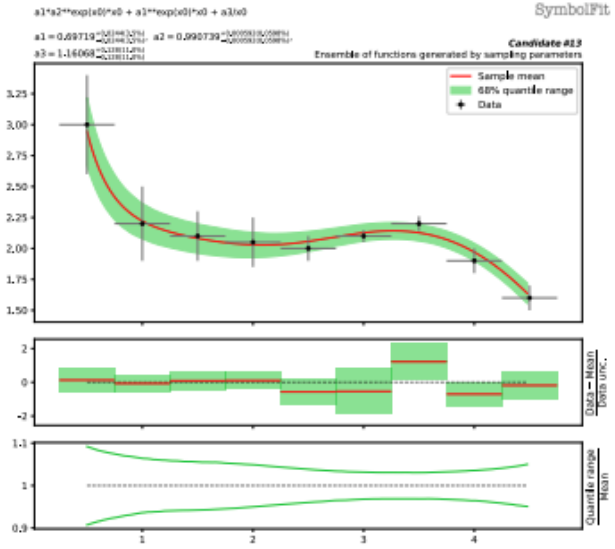
Figure A13: Pseudodata of the four-jet spectrum with the injected signal shown in the blinded signal region. The three SR models (see Tab. A6) are compared against the empirical model used by CMS. The lower panel shows the residual error per bin, measured in units of the data uncertainty.

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

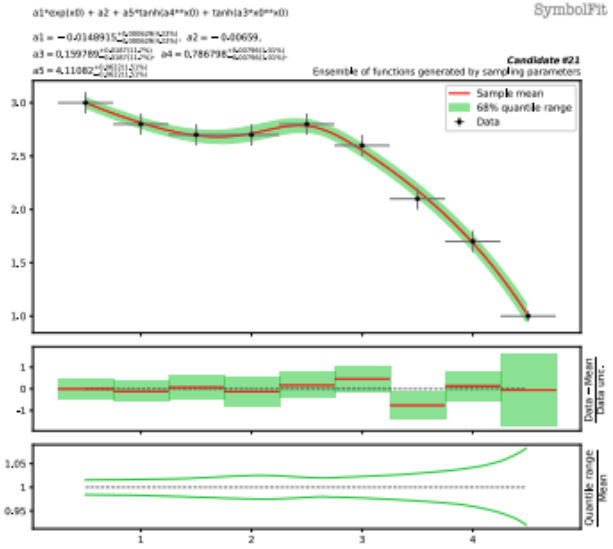
larger event samples. These are the modified dijet function (ModDijet-5p) given by $d\sigma/dm = p_0(1-x^{1/3})^{p_1}/(x^{p_2+p_3\log x+p_4\log^2 x})$, the dijet function (Dijet-5p) given by $d\sigma/dm = p_0(1-x)^{p_1}/(x^{p_2+p_3\log x+p_4\log^2 x})$, and the power-law times exponential function (PowExp-5p)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

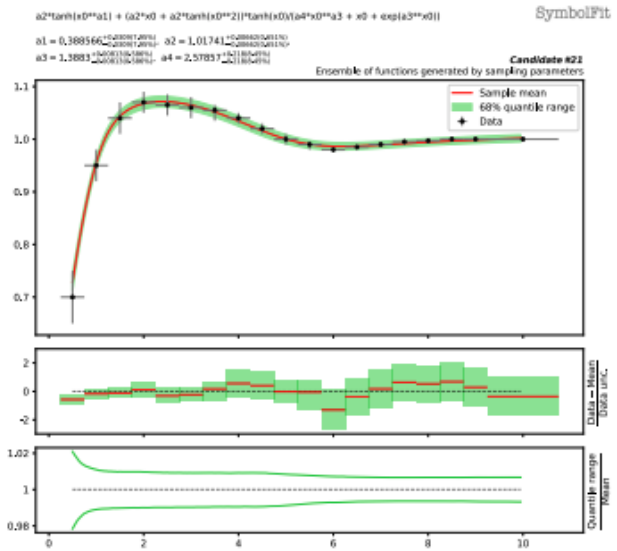
Examples



(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

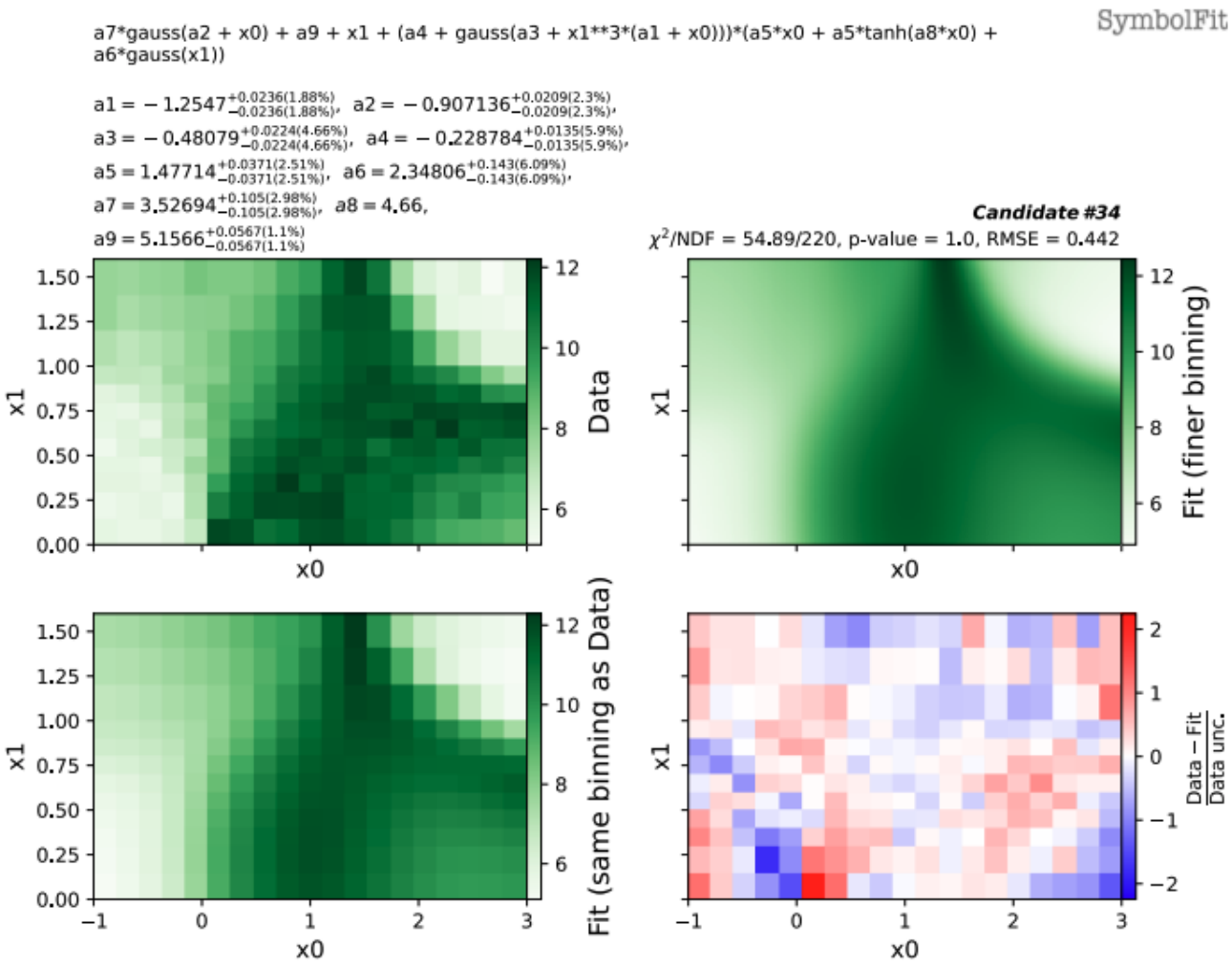
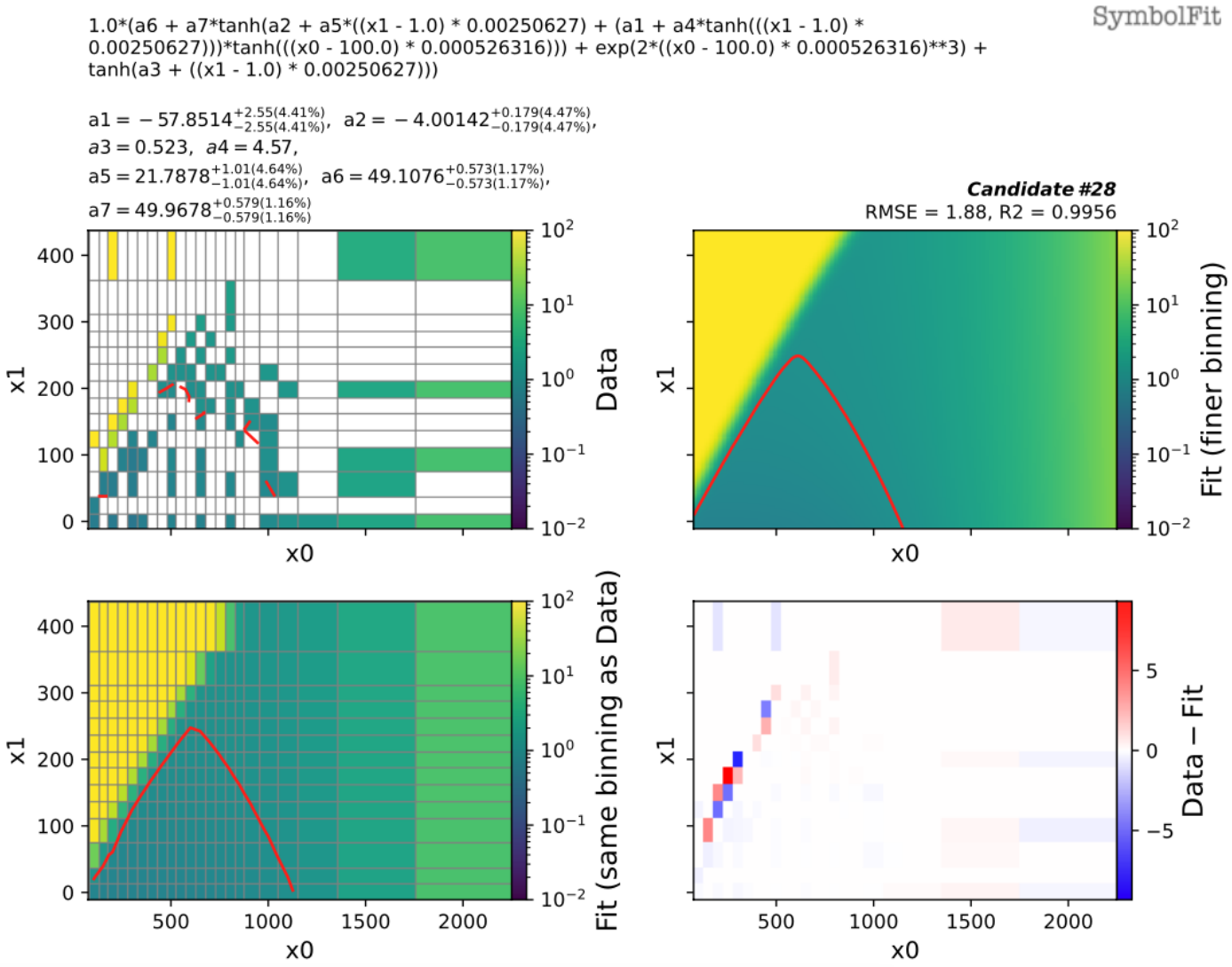


Figure 16: Candidate function #34 for Toy Dataset 3c (see Tab. 4). The parameterized form of this function is shown at the top of the figure, along with the best-fit values and associated uncertainties. Upper left: the binned data being fitted. Lower left: the candidate function plotted with the same 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.

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

Extrapolation of sparse data



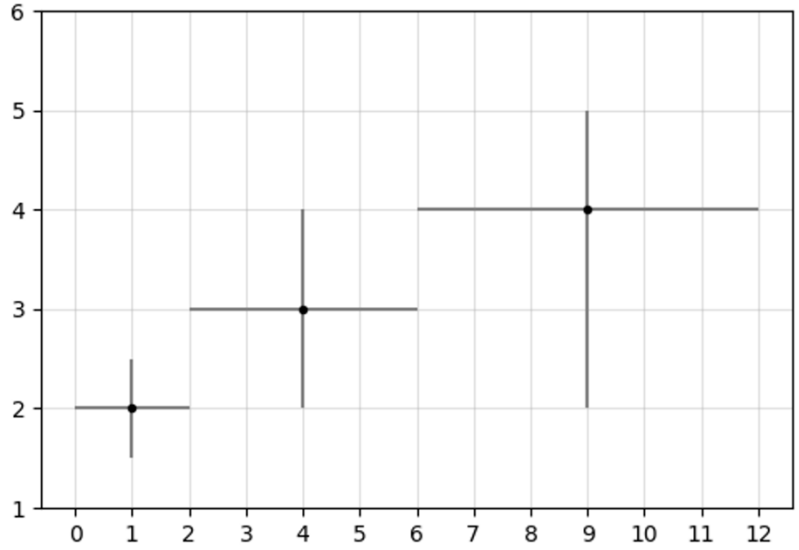
SUSY limit plots often have limited signal mass points.

- SymbolFit can find smooth functions to smoothen these.

Applicable to 2D scale factors with limited stats as well.

Backup

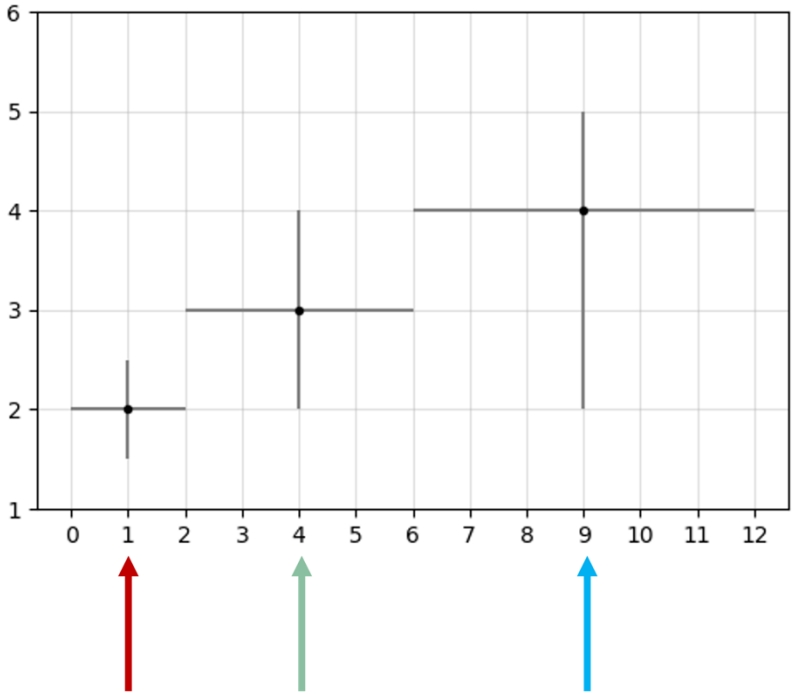
Input data format (1d)



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

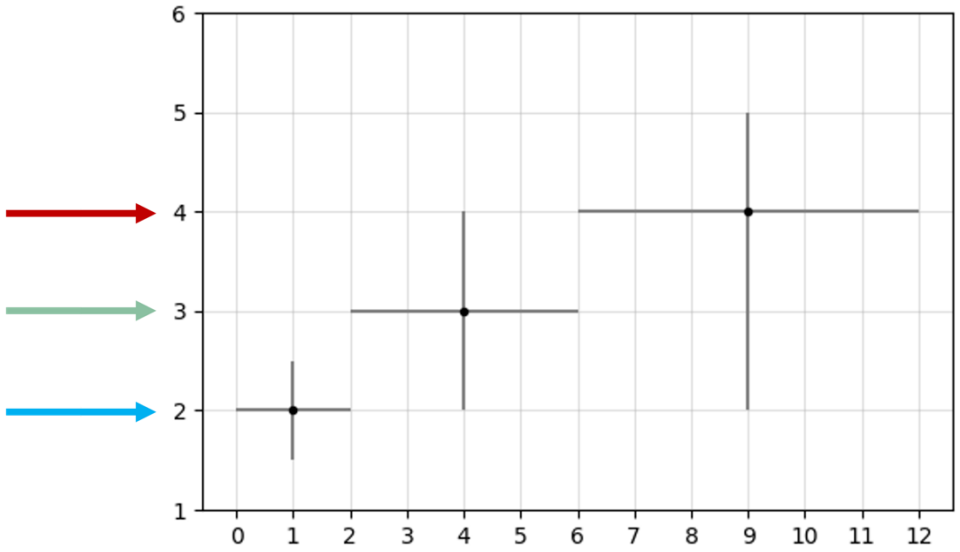
Input data format (1d)



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]

y = [2, 3, 4]

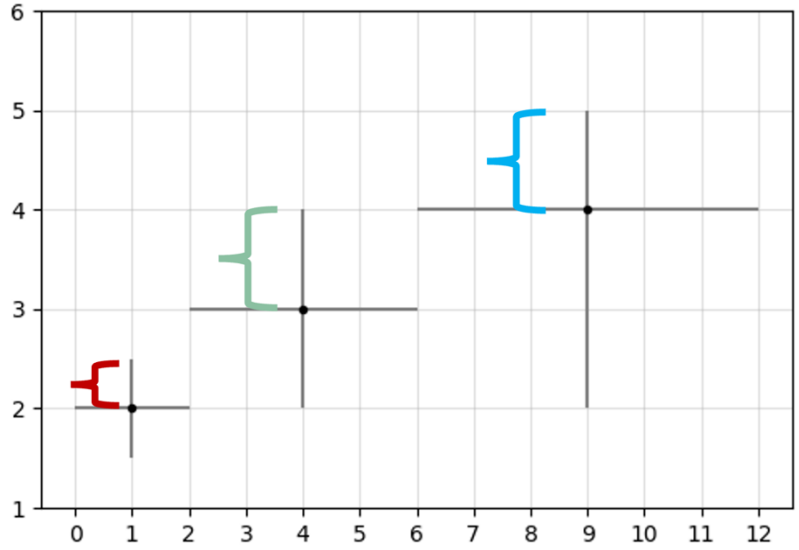
y_up = [0.5, 1, 1]

y_down = [0.5, 1, 2]

bin_widths_1d = [2, 4, 6]

See docs here

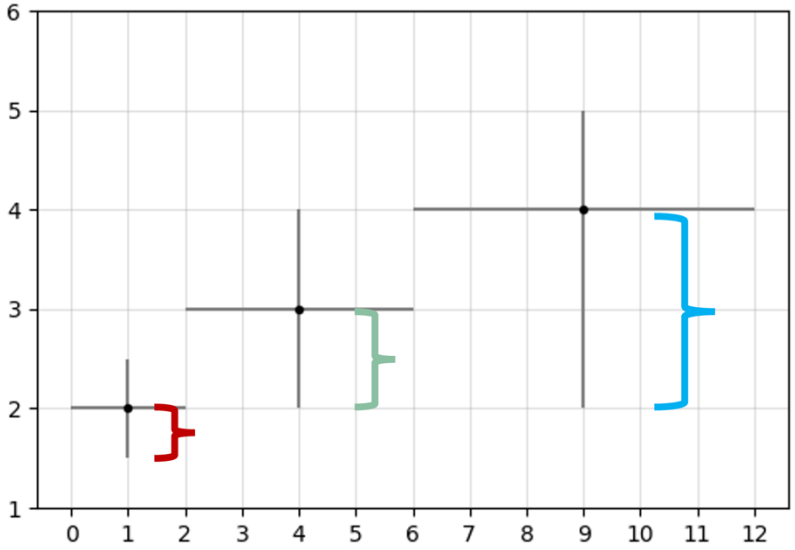
Input data format (1d)



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

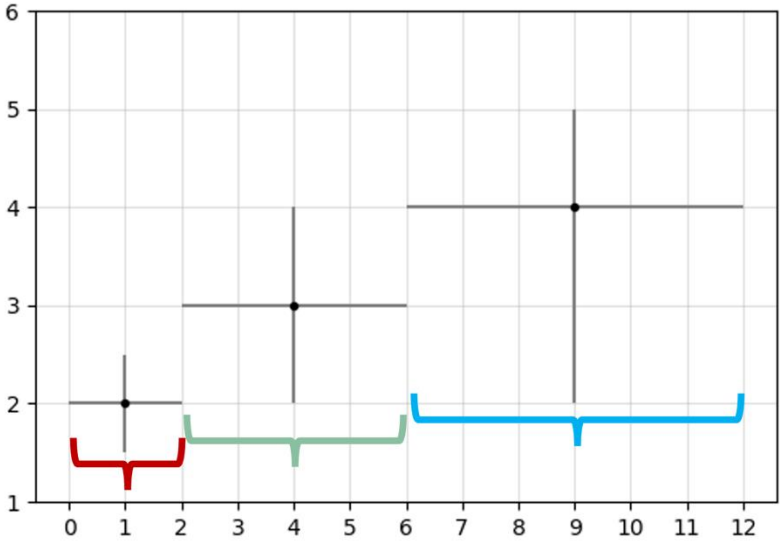
Input data format (1d)



x = [1, 4, 9]
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

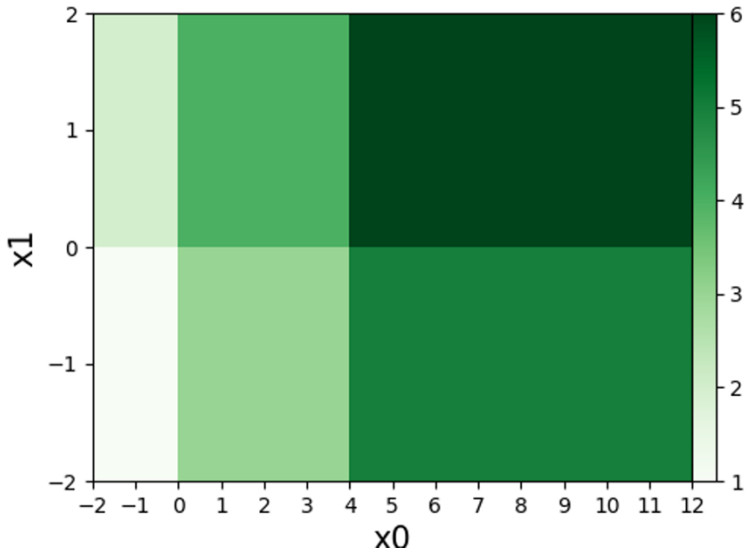
Input data format (1d)



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

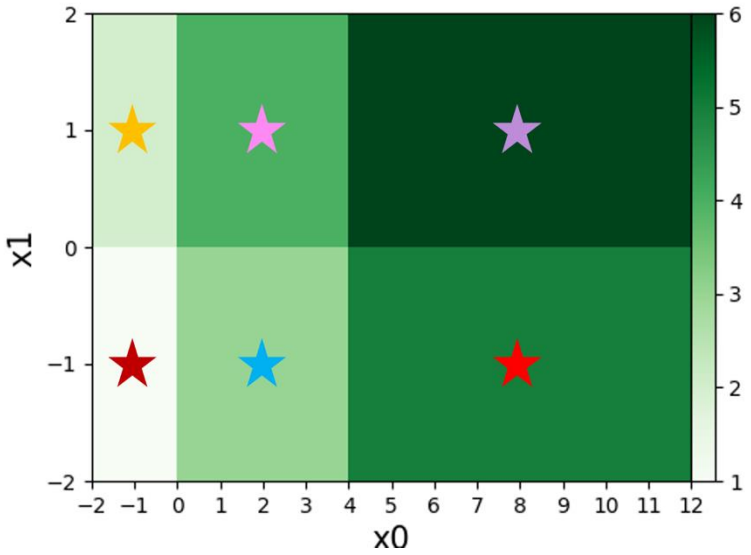
Input data format (2d)




```
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

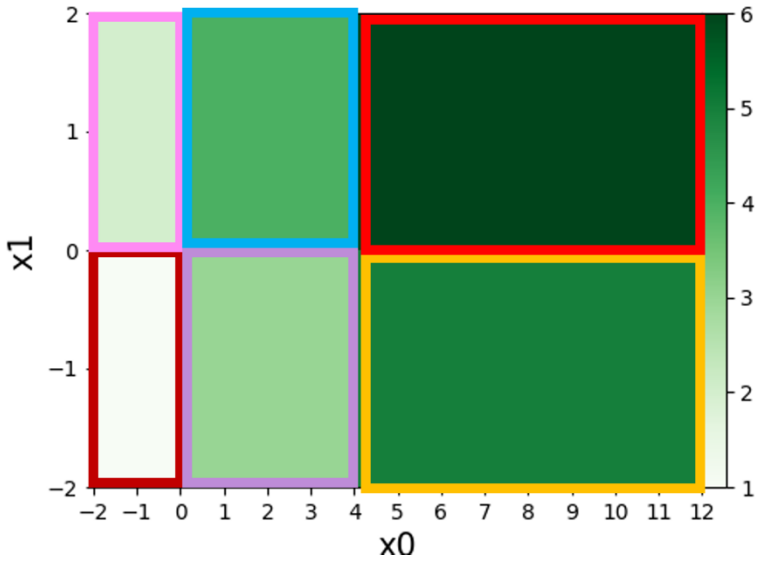
Input data format (2d)




```
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

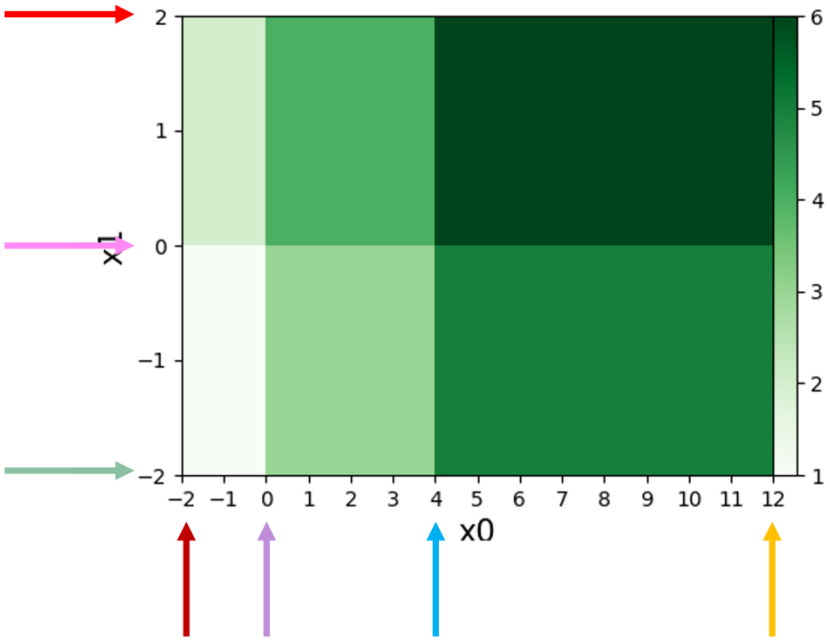
Input data format (2d)




```
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