EFT concepts for experimental analyses

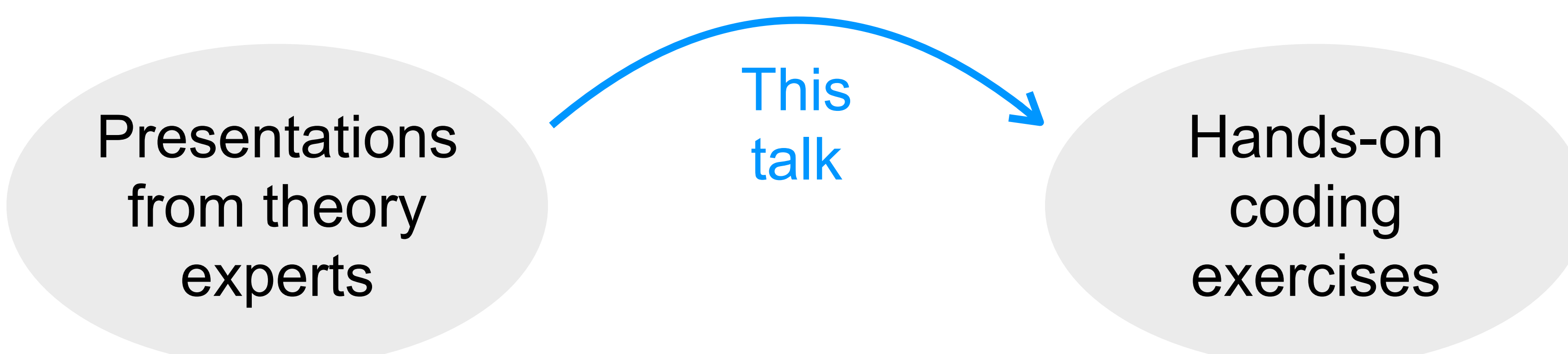
LPC EFT Workshop Tutorial Team:

Sapta Bhattacharya[5], Jennet Dickinson[2], Kelci Mohrman[4], Andrea Piccinelli[3], Nick Smith[2], Daniel Spitzbart[1]

[1]BU, [2]FNAL, [3]ND, [4]UF, [5]Wayne State

Goal of the talk

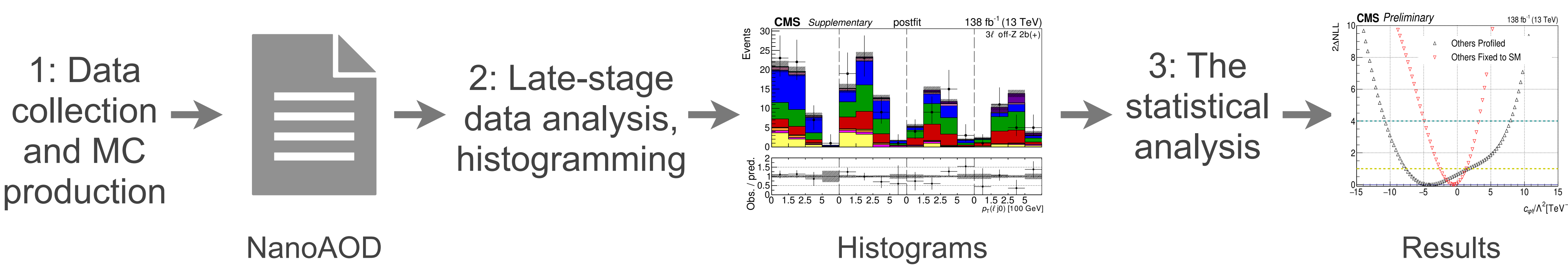
- Goal of this talk is to take what we learned from the theory talks, and discuss how it applies to doing an experimental EFT analysis
- I.e. we will try to bridge the gap between the preceding theory talks, and the following hands-on tutorial
- Aim to get a conceptual understanding of the code that we'll work though next



Introduction to experimental EFT

The big-picture experimental goal is to compare EFT prediction to data in order to extract confidence intervals for the WCs (Wilson Coefficients), involves three main steps:

1. Generate MC that incorporates the EFT into the prediction
2. Perform selection to obtain the events of interest, summarized in histogram objects
3. Perform a statistical analysis to compare the prediction to the observation and extract confidence intervals



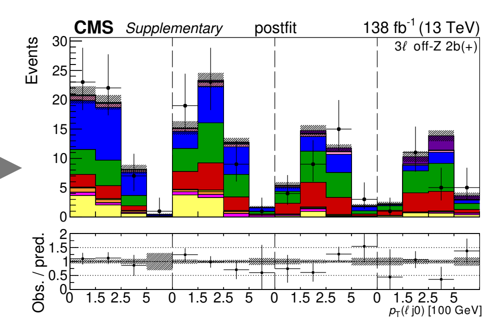
Outline for this talk

- Recap of EFT: What to know for an analysis
- Getting the prediction in terms of EFT (Step 1)
- EFT histogramming (Step 2)
- Extracting limits on EFT parameters (Step 3)

1: Data collection and MC production →

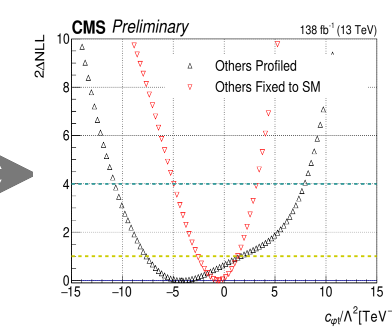
NanoAOD

→ 2: Late-stage data analysis, histogramming →



Histograms

→ 3: The statistical analysis →



Results

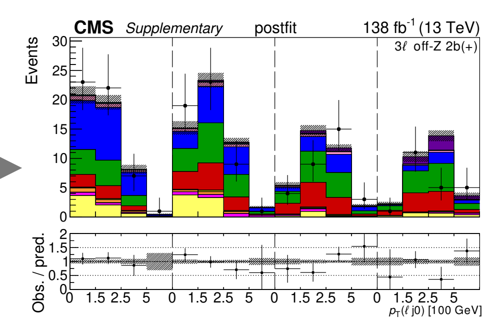
Outline for this talk

- Recap of EFT: What to know for an analysis
- Getting the prediction in terms of EFT (Step 1)
- EFT histogramming (Step 2)
- Extracting limits on EFT parameters (Step 3)

1: Data collection and MC production →

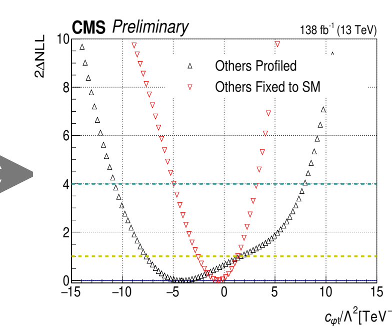
NanoAOD

→ 2: Late-stage data analysis, histogramming →



Histograms

→ 3: The statistical analysis →

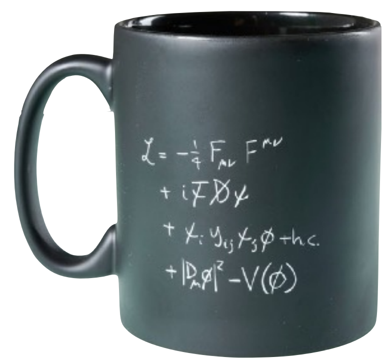


Results

Brief introduction to SM EFT*

SM is lowest order piece

$\mathscr{L}_{\mathrm{EFT}}$ =



\+ Higher order corrections (think Taylor series)

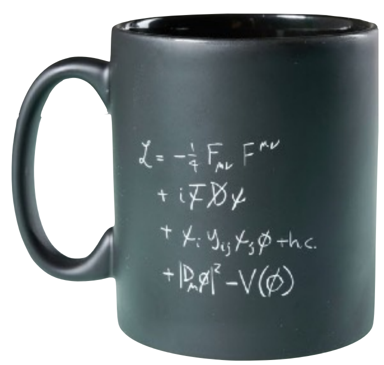
Kelci Mohrman, k.mohrman@ufl.edu

*There are many types of EFTs, but in this presentation, we will focus specifically on SMEFT

Brief introduction to SM EFT*

SM is lowest order piece

$$\mathscr{L}_{\text{EFT}} =$$



$$+ \sum_i \frac{c_i}{\Lambda}\mathscr{O}_i^{(5)} + \sum_i \frac{c_i}{\Lambda^2}\mathscr{O}_i^{(6)} + \dots$$

*There are many types of EFTs, but in this presentation, we will focus specifically on SMEFT

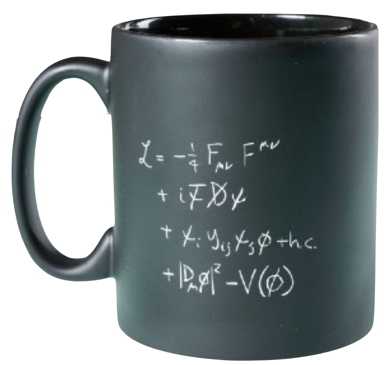
Brief introduction to SM EFT*

SM is lowest order piece

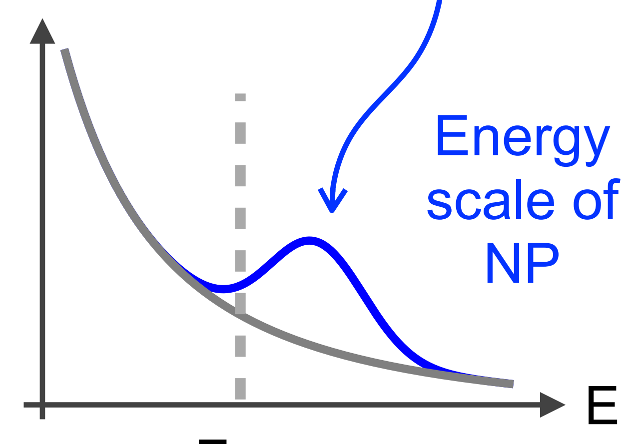
Wilson Coefficient (WC), strength of interaction

Operators are built of products of SM fields and their derivatives

$$\mathscr{L}_{\mathrm{EFT}} =$$



$$+ \sum_i \frac{c_i}{\Lambda}\mathcal{O}_i^{(5)} + \sum_i \frac{c_i}{\Lambda^2}\mathcal{O}_i^{(6)} + \dots$$



Note that if all WC=0, SM is recovered, so a nonzero WC is a sign of new physics!

*There are many types of EFTs, but in this presentation, we will focus specifically on SMEFT

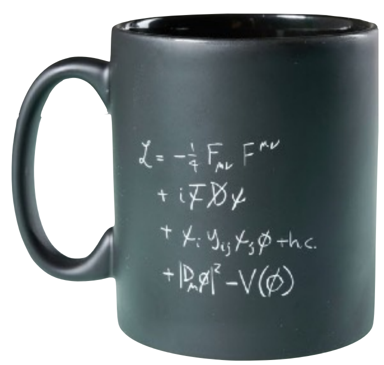
Brief introduction to SM EFT*

SM is lowest order piece

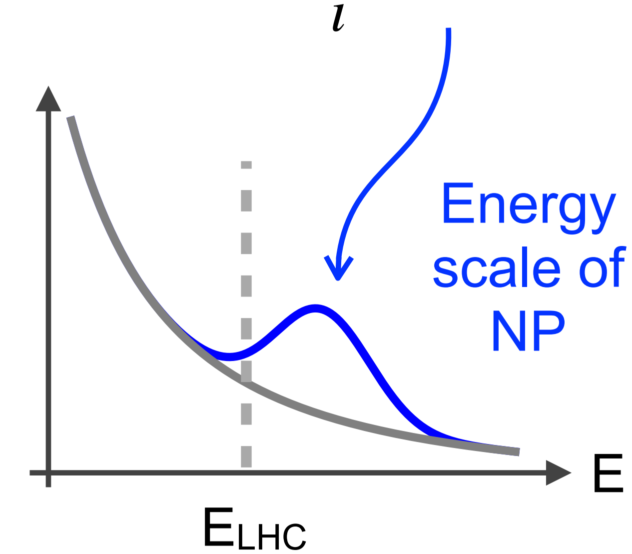
Wilson Coefficient (WC), strength of interaction

Operators are built of products of SM fields and their derivatives

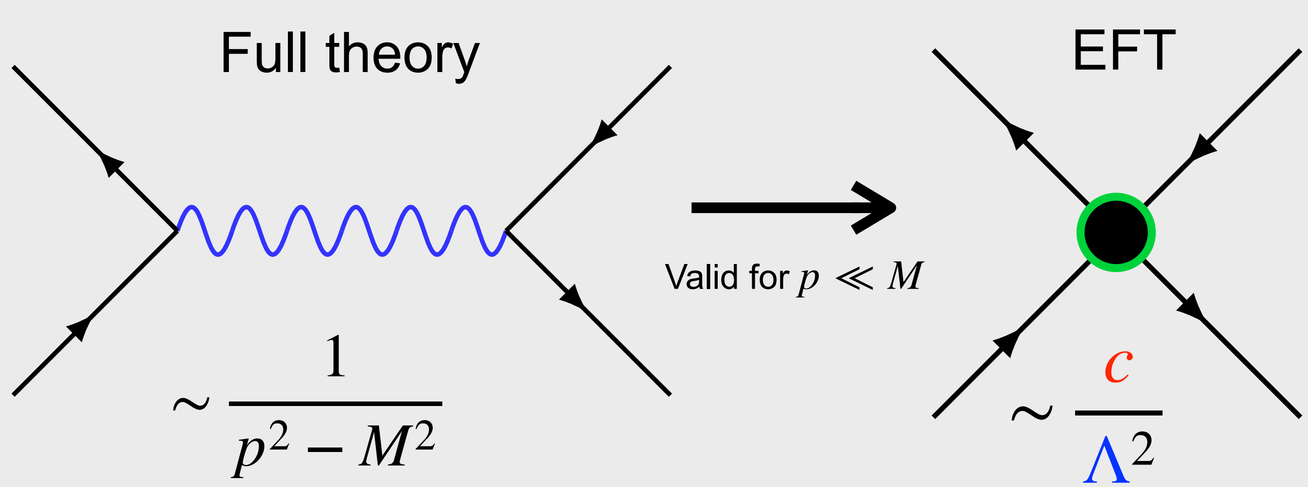
$$\mathscr{L}_{\mathrm{EFT}} =$$



$$+ \sum_i \frac{c_i}{\Lambda} \mathcal{O}_i^{(5)} + \sum_i \frac{c_i}{\Lambda^2} \mathcal{O}_i^{(6)} + \dots$$



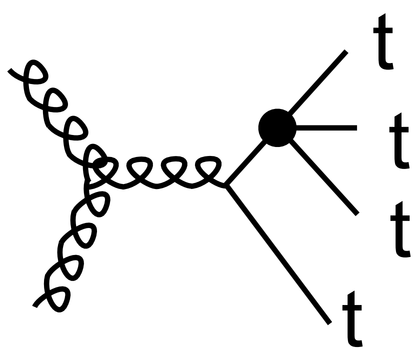
Example of EFT in action..



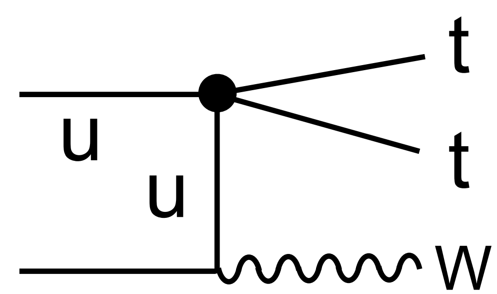
Note that if all WC=0, SM is recovered, so a nonzero WC is a sign of new physics!

*There are many types of EFTs, but in this presentation, we will focus specifically on SMEFT

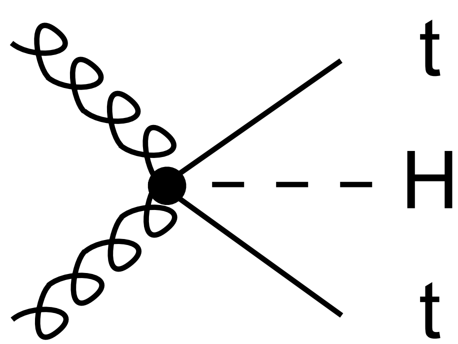
The EFT vertices can impact observables, where the strengths of the impacts are determined by the WCs that scale the vertices



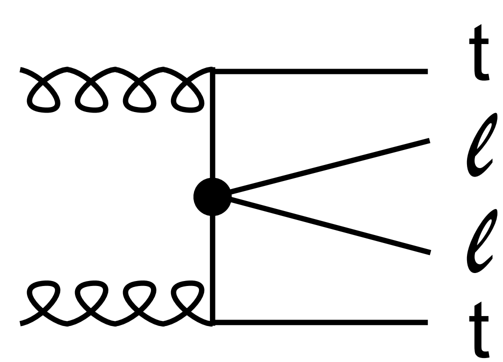
4 heavy quarks



2 heavy quarks and 2 light quarks



2 heavy quarks and bosons

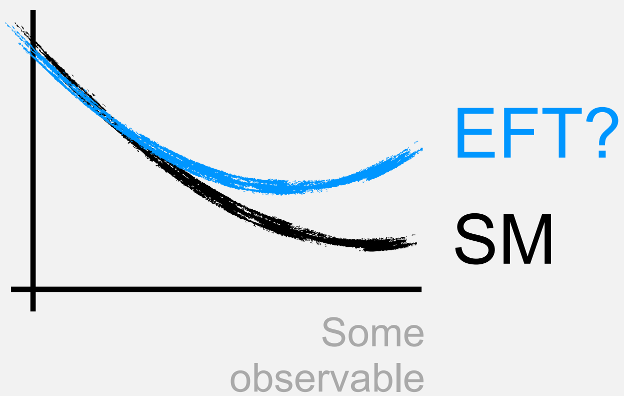


2 heavy quarks and 2 leptons

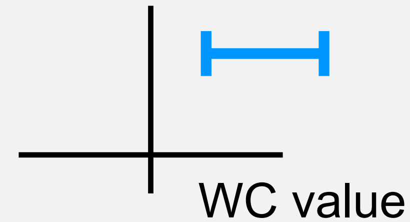
(A few example vertices shown here)

What: Compare prediction to data to find WC values

Why: Any non-zero WC would be new physics!

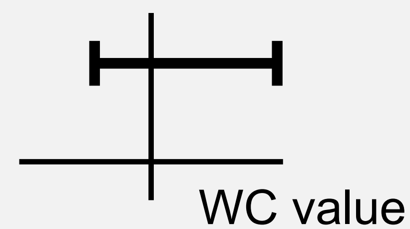


WC=0 NOT consistent with data



New physics!!

WC=0 consistent with data



Constraints and info on Λ probed

How?

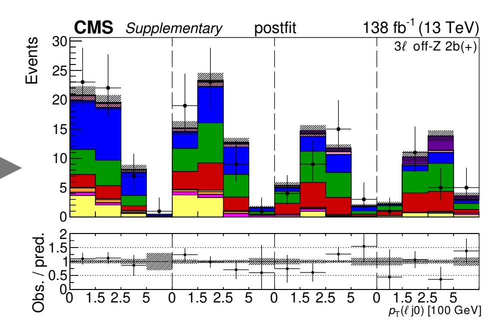
Parameterize some prediction in terms of the WCs

Compare observation to prediction and extract best fit values and corresponding uncertainties for the WCs

Outline for this talk

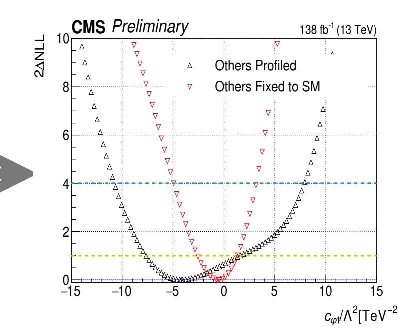
- Recap of EFT: What to know for an analysis
- Getting the prediction in terms of EFT (Step 1)
- EFT histogramming (Step 2)
- Extracting limits on EFT parameters (Step 3)

1: Data collection and MC production → NanoAOD → 2: Late-stage data analysis, histogramming →



Histograms

→ 3: The statistical analysis →

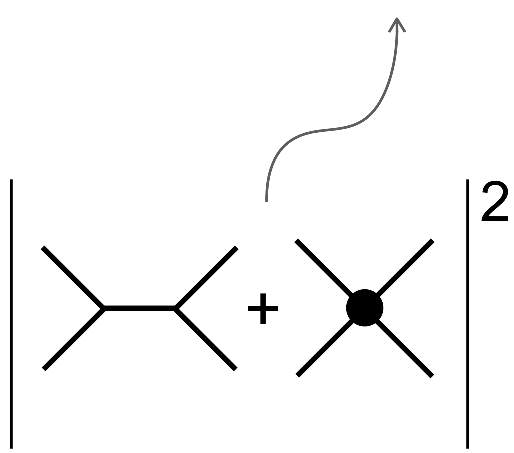


Results

How do observables depend on EFT? Let's start with σ

If the EFT is modeled linearly in amplitude, the cross section is an *n*-quadratic in terms of the WCs (where *n* is number of WCs)

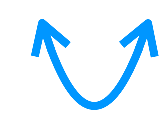
$$\sigma \propto |\mathscr{M}_{SM} + \frac{c_i}{\Lambda^2}\mathscr{M}_i|^2$$

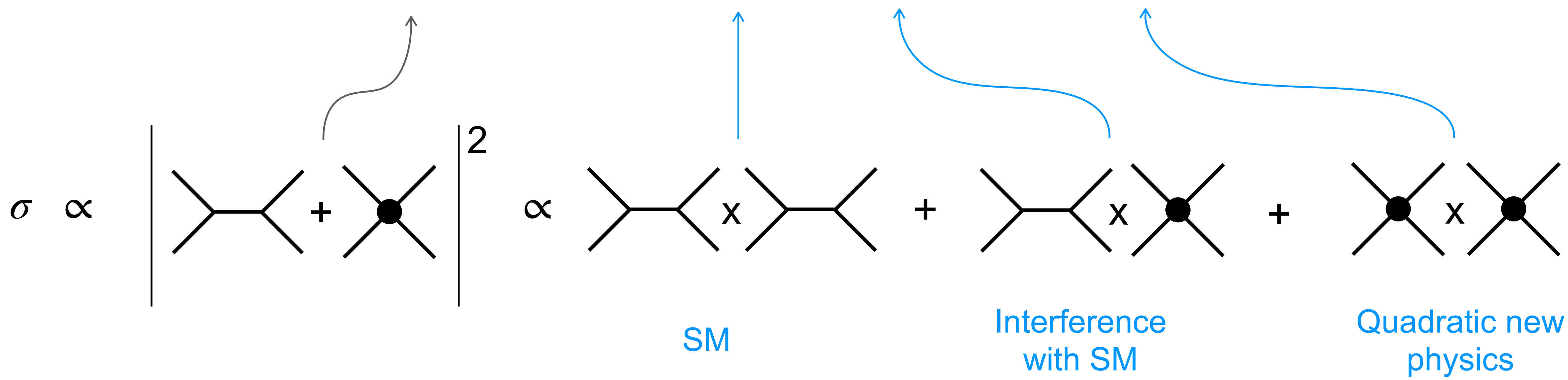
$\sigma \propto$ 

How do observables depend on EFT? Let's start with σ

If the EFT is modeled linearly in amplitude, the cross section is an *n*-quadratic in terms of the WCs (where *n* is number of WCs)

$$\sigma \propto \left| \mathcal{M}_{SM} + \frac{c_i}{\Lambda^2} \mathcal{M}_i \right|^2 \propto s_0 + s_i \frac{c_i}{\Lambda^2} + s_{ij} \frac{c_i}{\Lambda^2} \frac{c_j}{\Lambda^2} =$$

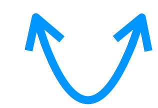


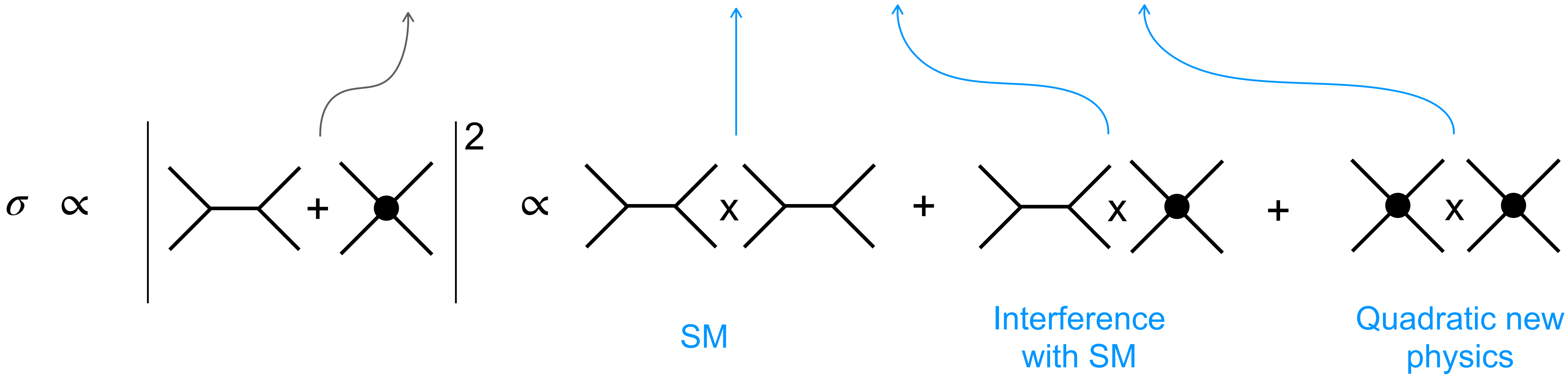


How do observables depend on EFT? Let's start with σ

If the EFT is modeled linearly in amplitude, the cross section is an *n*-quadratic in terms of the WCs (where *n* is number of WCs)

$$\sigma \propto \left| \mathcal{M}_{SM} + \frac{c_i}{\Lambda^2}\mathcal{M}_i \right|^2 \propto s_0 + s_i \frac{c_i}{\Lambda^2} + s_{ij}\frac{c_i}{\Lambda^2}\frac{c_j}{\Lambda^2} =$$

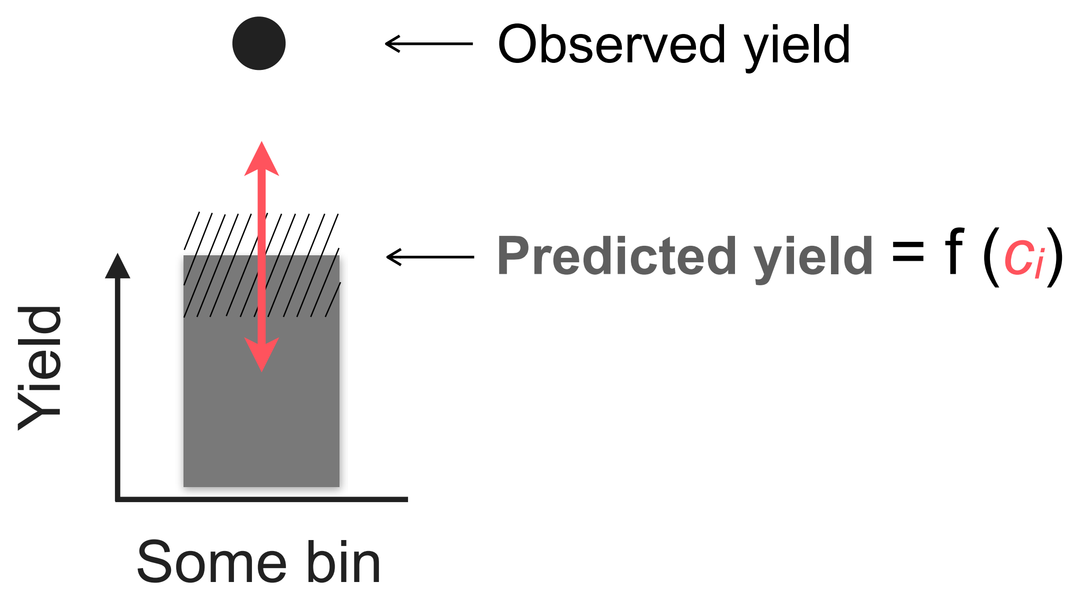




This holds for any cross section, inclusive or differential

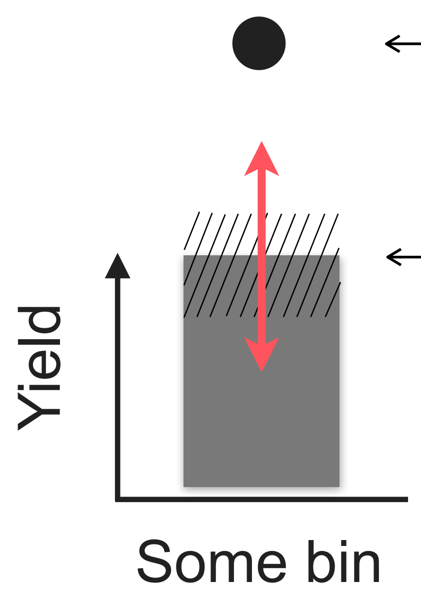
How to get a prediction as a function of EFT

1. Write the **prediction** in the observable bins as a function of WCs
2. Compare that to the observation to extract limits for the WCs

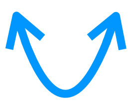


How to get a prediction as a function of EFT

1. Write the **prediction** in the observable bins as a function of WCs
2. Compare that to the observation to extract limits for the WCs

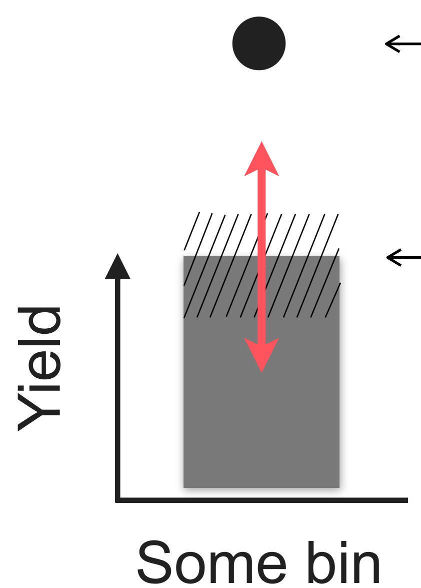


← Observed yield

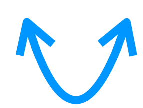
← **Predicted yield** $= f\,(c_i) = s_0 + s_i\, c_i + s_{ij}\, c_i\, c_j =$ 

How to get a prediction as a function of EFT

1. Write the **prediction** in the observable bins as a function of WCs
2. Compare that to the observation to extract limits for the WCs



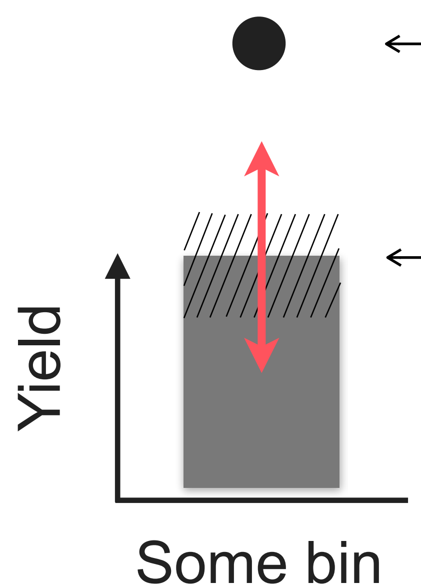
← Observed yield

← **Predicted yield** = $f(c_i)$ = $s_0 + s_i c_i + s_{ij} c_i c_j$ = 

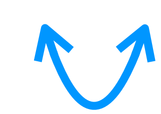
But we'd expect the dependence to vary from bin to bin

How to get a prediction as a function of EFT

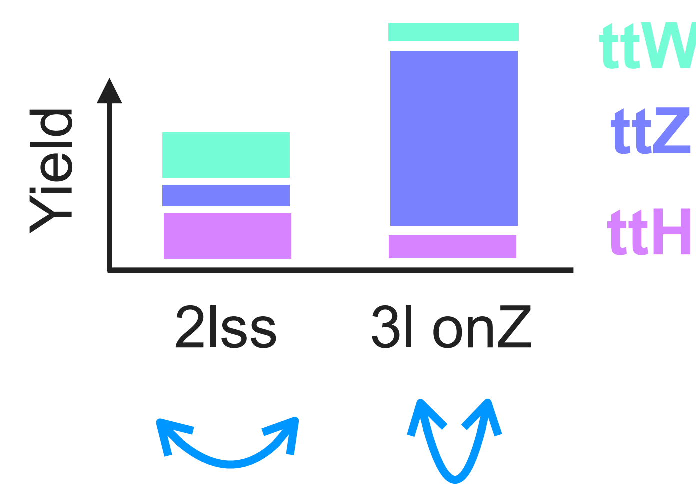
1. Write the **prediction** in the observable bins as a function of WCs
2. Compare that to the observation to extract limits for the WCs



← Observed yield

← **Predicted yield** = $f\,(c_i)$ = $s_0 + s_i\, c_i + s_{ij}\, c_i\, c_j$ = 

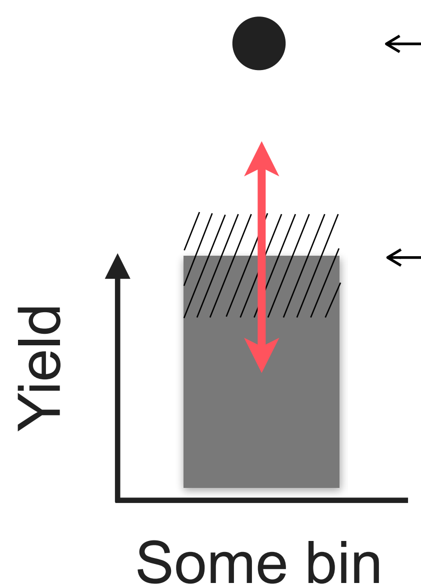
But we'd expect the dependence to vary from bin to bin



EFT dependence impacted by bin makeup

How to get a prediction as a function of EFT

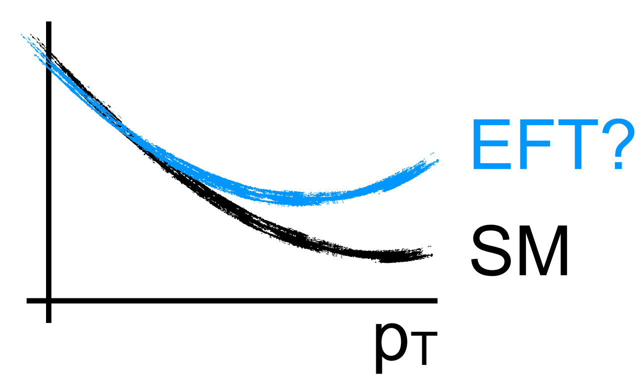
1. Write the **prediction** in the observable bins as a function of WCs
2. Compare that to the observation to extract limits for the WCs



← Observed yield

← **Predicted yield** = $f\,(c_i) = s_0 + s_i\, c_i + s_{ij}\, c_i\, c_j =$

But we'd expect the dependence to vary from bin to bin



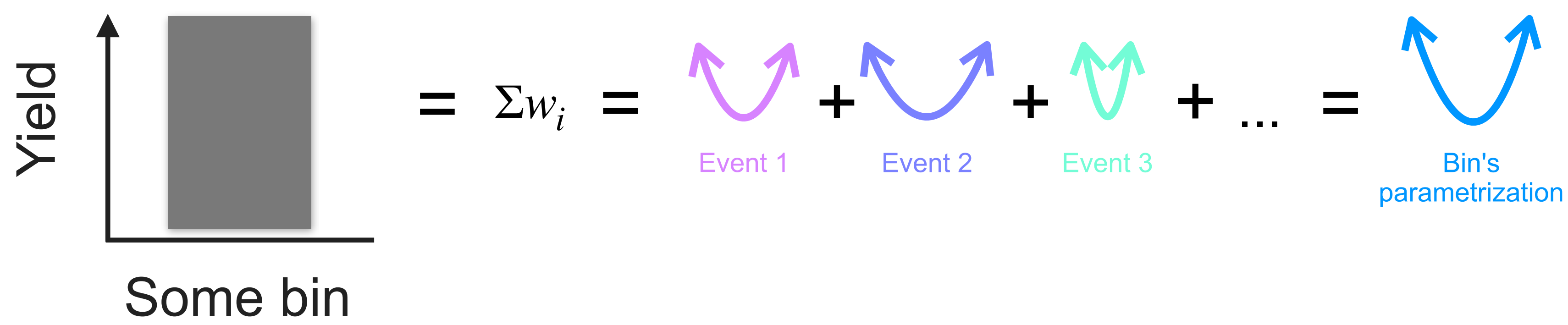
EFT dependence also impacted by kinematics

How do we find the quadratic parametrization for each bin's yield?

- The key is to parametrize the weight of each simulated event as a quadratic in terms of the WCs

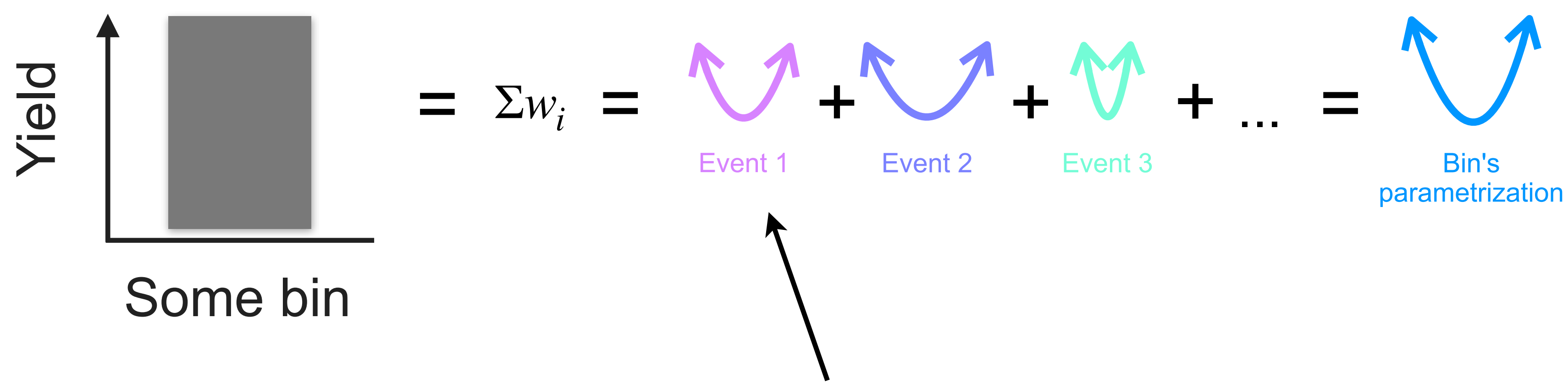
How do we find the quadratic parametrization for each bin's yield?

- The key is to parametrize the weight of each simulated event as a quadratic in terms of the WCs
- Can then find any arbitrary bin's yield as a function of the WCs by summing the quadratics of the events that fall in the bin



How do we find the quadratic parametrization for each bin's yield?

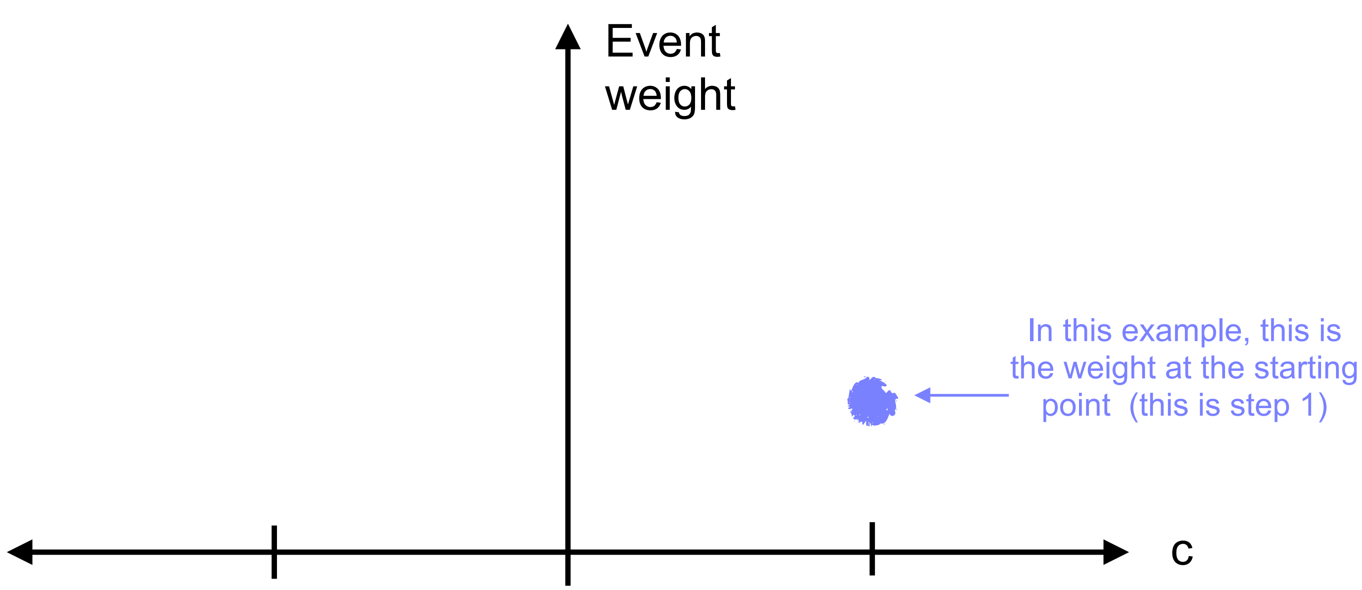
- The key is to parametrize the weight of each simulated event as a quadratic in terms of the WCs
- Can then find any arbitrary bin's yield as a function of the WCs by summing the quadratics of the events that fall in the bin



Rest of the section will explain how we can find these per-event quadratic parameterizations using **reweighting**

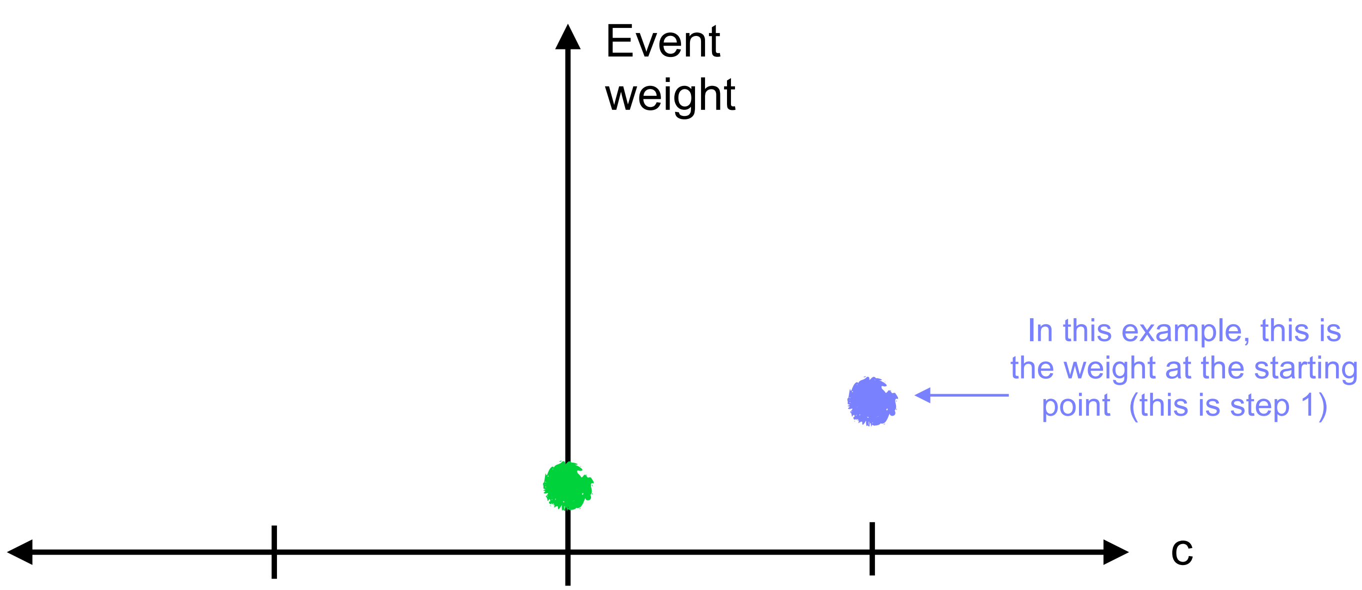
Extracting the quadratic dependence for a single event

- How do we find the quadratic dependence for each of the generated events?
- Use MG reweighting (will be introduced in the tutorial) e.g. for one single WC:
    1. Pick a "starting point" in the WC space, and MG generates an event (at some point in kinematic space) under the assumption of the given point in WC space (e.g. a "c=1" assumption)



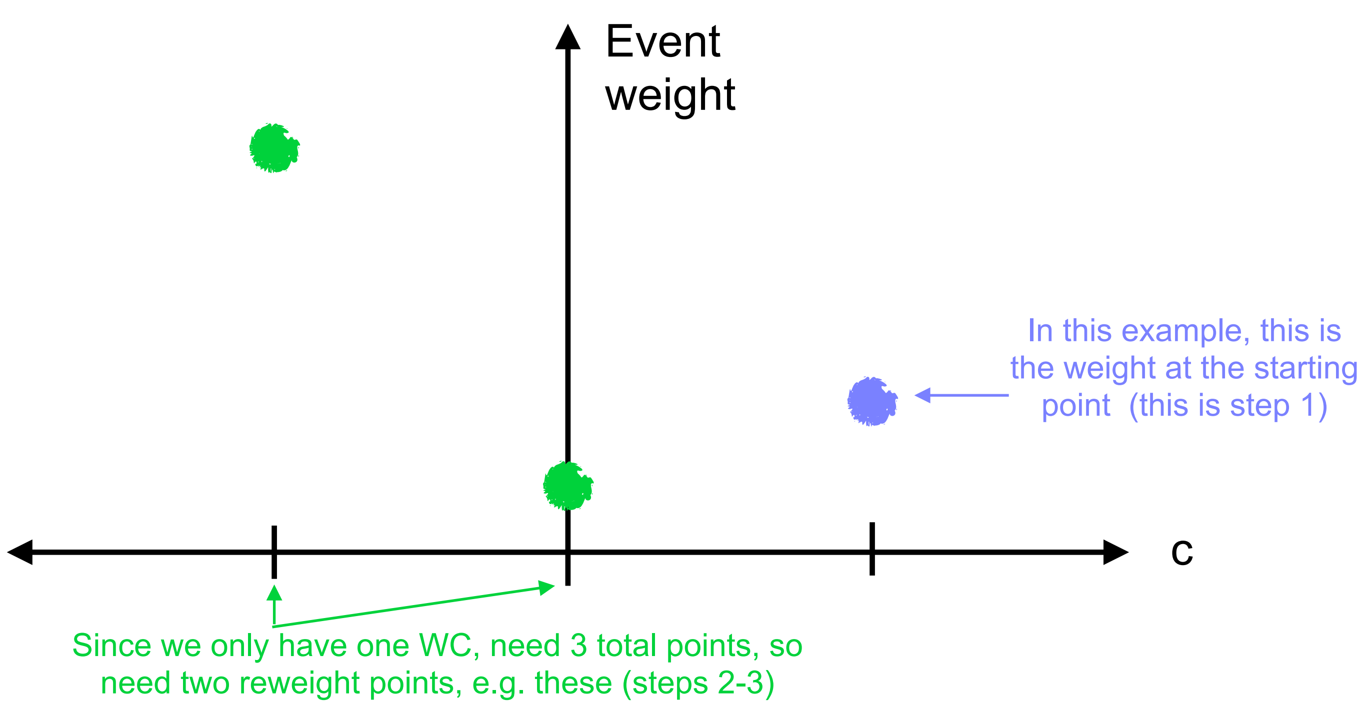
Extracting the quadratic dependence for a single event

- How do we find the quadratic dependence for each of the generated events?
- Use MG reweighting (will be introduced in the tutorial) e.g. for one single WC:
    1. Pick a "starting point" in the WC space, and MG generates an event (at some point in kinematic space) under the assumption of the given point in WC space (e.g. a "c=1" assumption)
    2. Ask MG "what would the weight of this event have been at a different point in the WC space?"



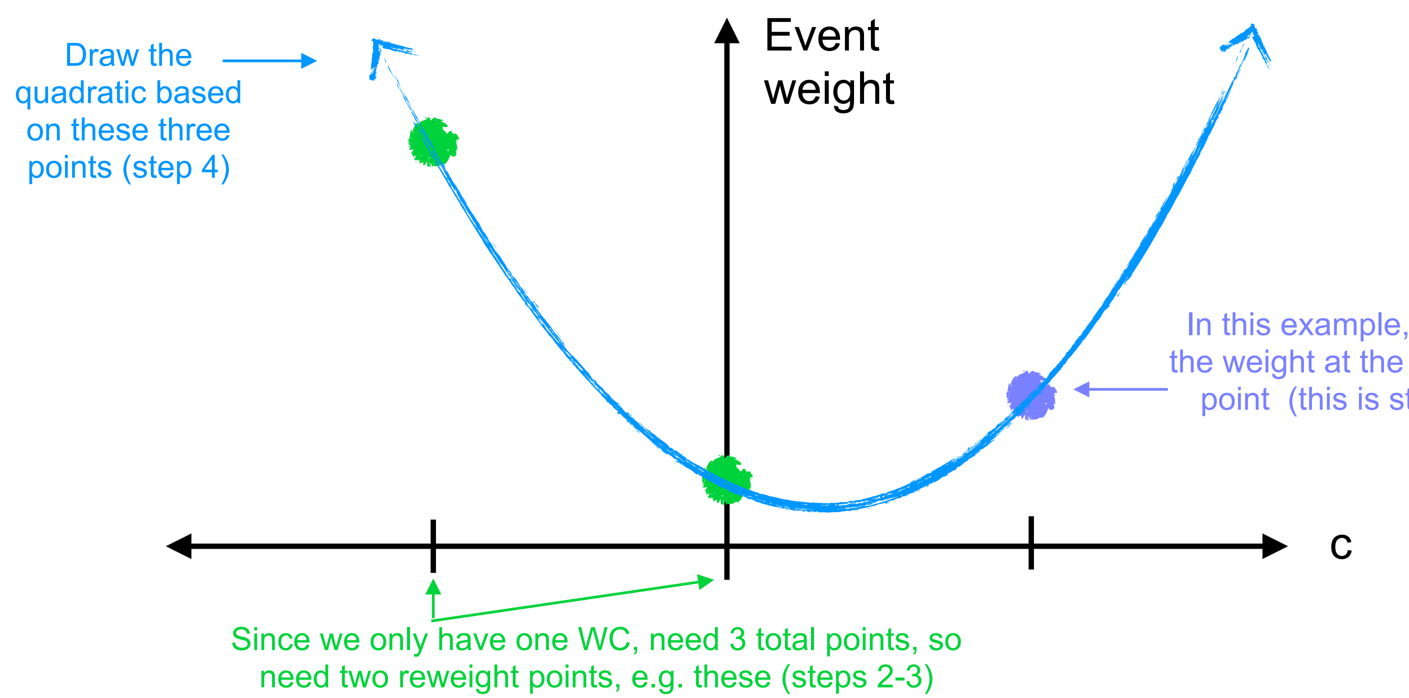
Extracting the quadratic dependence for a single event

- How do we find the quadratic dependence for each of the generated events?
- Use MG reweighting (will be introduced in the tutorial) e.g. for one single WC:
    1. Pick a "starting point" in the WC space, and MG generates an event (at some point in kinematic space) under the assumption of the given point in WC space (e.g. a "c=1" assumption)
    2. Ask MG "what would the weight of this event have been at a different point in the WC space?"
    3. Repeat step 2 for at least $((n+1)^2-(n+1))/2+n+1$ points in the WC space



Extracting the quadratic dependence for a single event

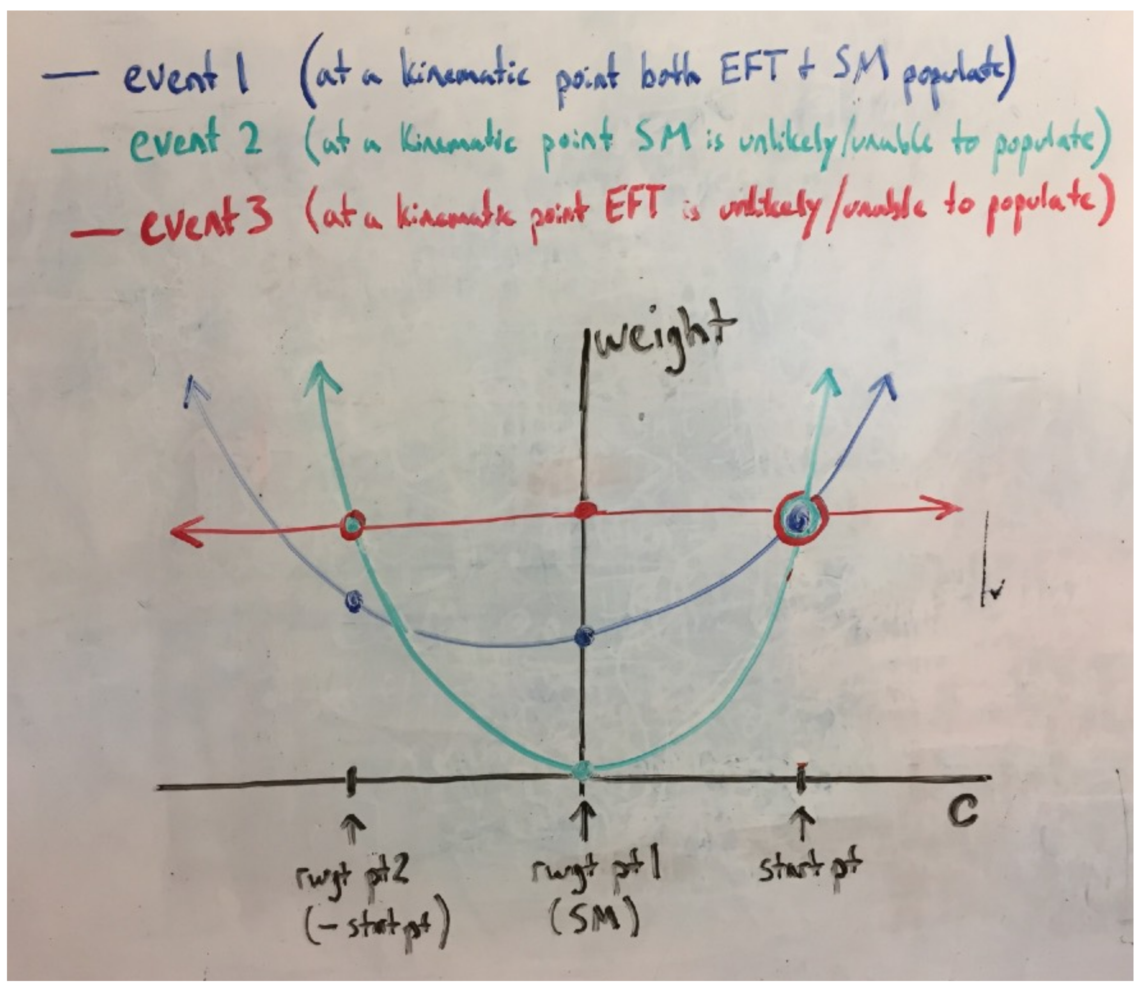
- How do we find the quadratic dependence for each of the generated events?
- Use MG reweighting (will be introduced in the tutorial) e.g. for one single WC:
  1. Pick a "starting point" in the WC space, and MG generates an event (at some point in kinematic space) under the assumption of the given point in WC space (e.g. a "c=1" assumption)
  2. Ask MG "what would the weight of this event have been at a different point in the WC space?"
  3. Repeat step 2 for at least $((n+1)^2 - (n+1))/2 + n + 1$ points in the WC space
  4. From the set of points in WC space and the associated weights, extract the quadratic parameterization



Why do different events have different quadratic shapes?

- Recall that MG will generate each event at a different kinematic point
- The kinematic point will be relatively less/more likely to be populated based on the theory assumption (i.e. at which point in WC space we are sitting)
- A complication to remember: Due to MG unweighting, the weight at the starting point will always be of the same magnitude (regardless of the differences in kinematics)

In this conceptual example, we're exploring different quadratic shapes we might see for three different simulated events →



This is somewhat difficult to conceptualize (at least for me) but remember that at the starting point, differences in probability due to different kinematics are conveyed by *how many events are generated at a given phase space point*, rather than by the *weight* of the given event at the given point in the space

Summary and some caveats

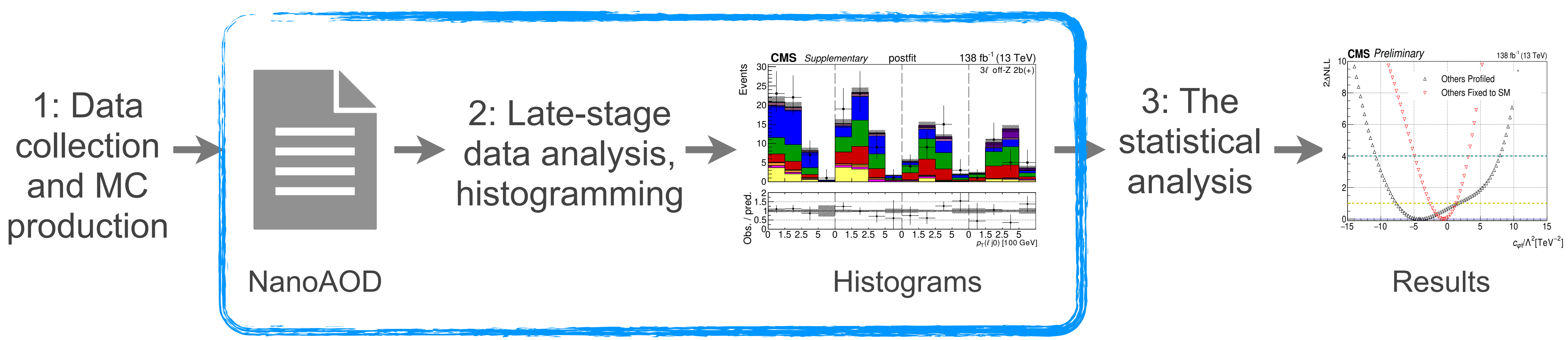
- Summary: If you have a sufficient number of reweighs points, you can extract the quadratic parametrization for each event's weight, which allows you to know the value of the event weight at any arbitrary point in the EFT space

- This can be a powerful approach for several reasons:
    - Allows essentially arbitrary regions in the EFT space to be probed with just a single sample
    - Allows the full effects of the EFT on kinematics to be accounted for
    - If the weights are carried through to detector level, allows EFT effects on acceptance/efficiency to be incorporated

- Caveats:
    - Vitally crucial to thoroughly validate the reweighted samples to ensure the sample can be consistently reweighted throughout the relevant EFT space
    - Important to explore the statistical power of the sample (highly non-uniform event weights degrade the statistical power)
    - Computationally challenging to produce samples with many WCs

Outline for this talk

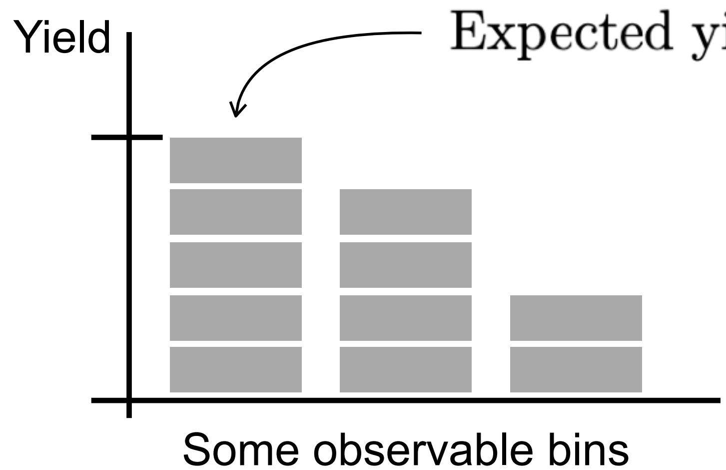
- Recap of EFT: What to know for an analysis
- Getting the prediction in terms of EFT (Step 1)
- EFT histogramming (Step 2)
- Extracting limits on EFT parameters (Step 3)



From regular histograms to “EFT aware” histograms

- Before we jump into EFT-aware histograms, let's start by recalling some concepts about "regular" histograms
- A regular histogram is essentially a list of bin values and corresponding bin edges
  - The value in each bin is just the sum of the weights of all of the events that pass the selection criteria for the given bin
  - To get the yield, need to normalize properly

“Regular” histogram



$$\text{Expected yield} = \sigma \mathcal{L} \frac{\sum_{\text{Pass}} w}{\sum_{\text{Gen}} w}$$

Normalization

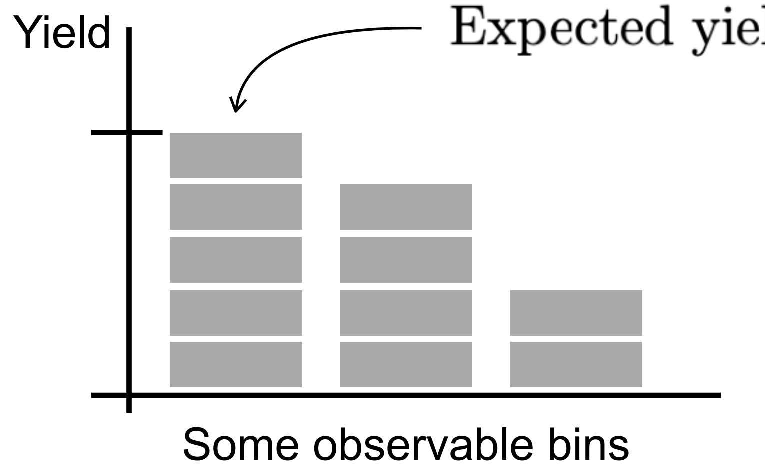
Sum of the weights of the events that pass the selection criteria for this bin

“Regular” histogram = [Value for bin 1 , Value for bin 2 , Value for bin 3]

From regular histograms to “EFT aware” histograms

- Before we jump into EFT-aware histograms, let's start by recalling some concepts about "regular" histograms
- A regular histogram is essentially a list of bin values and corresponding bin edges
  - The value in each bin is just the sum of the weights of all of the events that pass the selection criteria for the given bin
  - To get the yield, need to normalize properly

“Regular” histogram

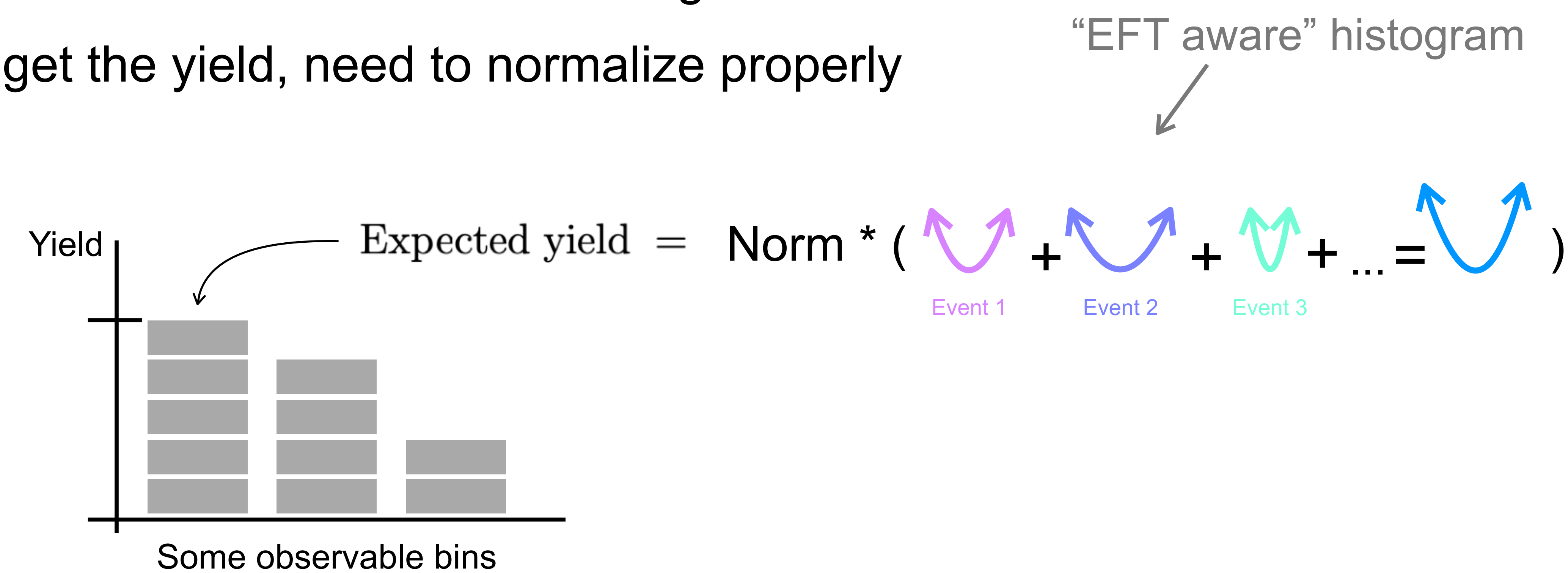


Expected yield = Norm * ($w_1 + w_2 + w_3 \ldots$)

“Regular” histogram = [Value for bin 1 , Value for bin 2 , Value for bin 3]

From regular histograms to “EFT aware” histograms

- Before we jump into EFT-aware histograms, let's start by recalling some concepts about "regular" histograms
- A regular histogram is essentially a list of bin values and corresponding bin edges
  - The value in each bin is just the sum of the weights of all of the events that pass the selection criteria for the given bin
  - To get the yield, need to normalize properly

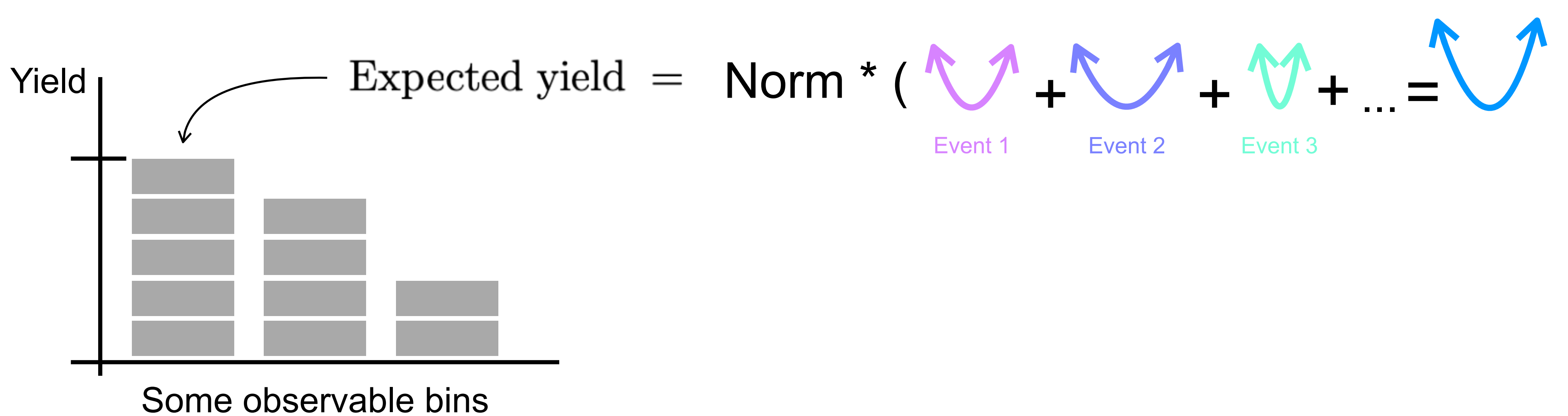


“Regular” histogram = [Value for bin 1 , Value for bin 2 , Value for bin 3]

From regular histograms to “EFT aware” histograms

- Before we jump into EFT-aware histograms, let's start by recalling some concepts about "regular" histograms
- A regular histogram is essentially a list of bin values and corresponding bin edges
  - The value in each bin is just the sum of the weights of all of the events that pass the selection criteria for the given bin
  - To get the yield, need to normalize properly

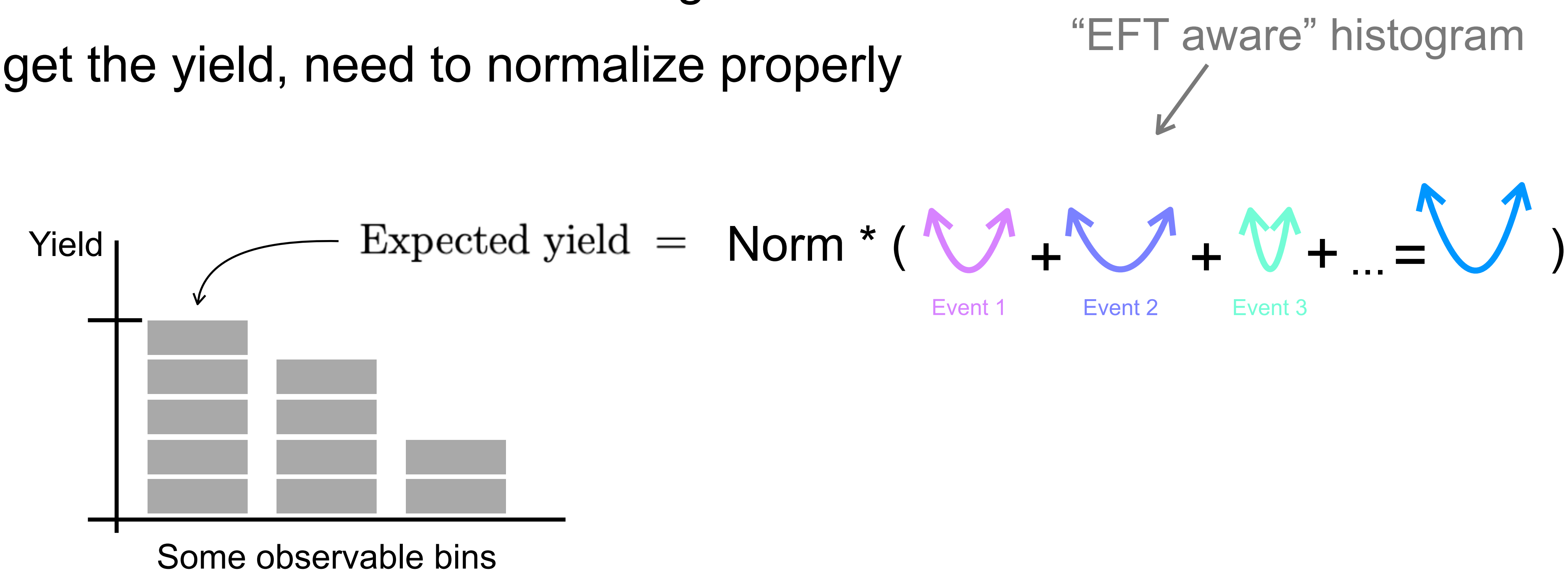
“EFT aware” histogram



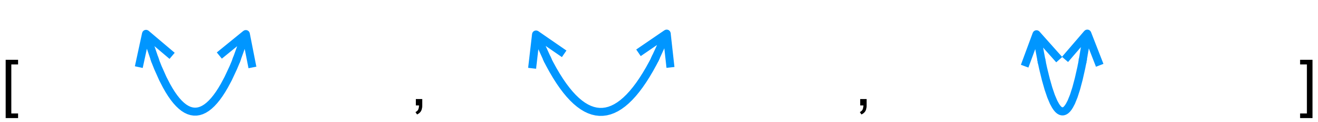
“EFT-aware” histogram = [Quadratic parameterization for bin 1 , Quadratic parameterization for bin 2 , Quadratic parameterization for bin 3]

From regular histograms to “EFT aware” histograms

- Before we jump into EFT-aware histograms, let's start by recalling some concepts about "regular" histograms
- A regular histogram is essentially a list of bin values and corresponding bin edges
  - The value in each bin is just the sum of the weights of all of the events that pass the selection criteria for the given bin
  - To get the yield, need to normalize properly

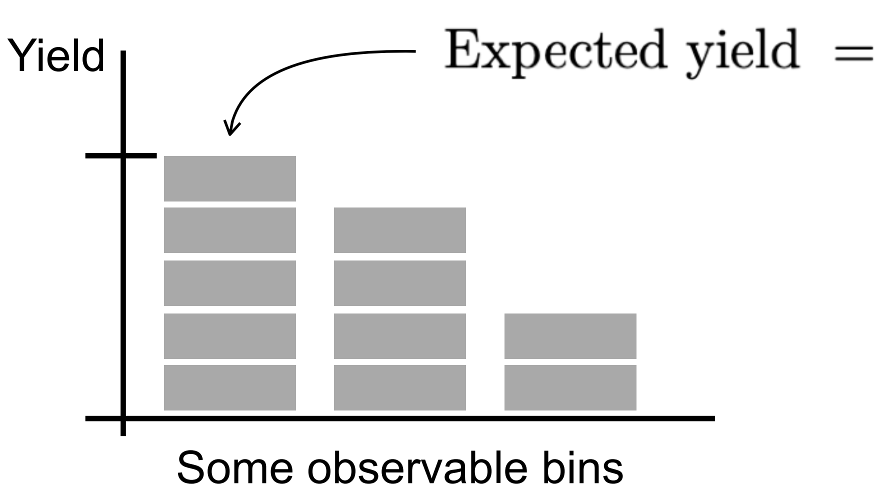


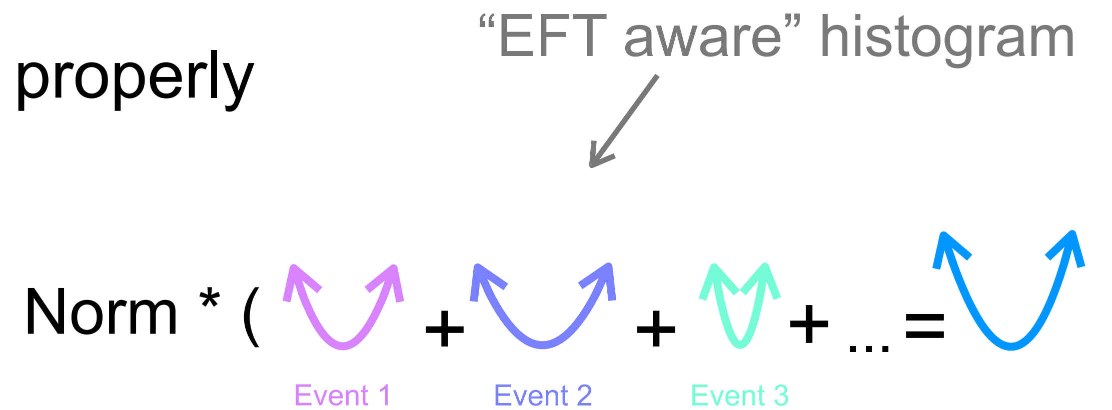
“EFT-aware” histogram = [



From regular histograms to “EFT aware” histograms

- Before we jump into EFT-aware histograms, let's start by recalling some concepts about "regular" histograms
- A regular histogram is essentially a list of bin values and corresponding bin edges
  - The value in each bin is just the sum of the weights of all of the events that pass the selection criteria for the given bin
  - To get the yield, need to normalize properly





Instead of storing the sum of weight values, EFT histograms store the *sum of the weight parameterizations* (in terms of the WCs), will in principle be different for each bin (since in principle the dependence is different for each event)

“EFT-aware” histogram = [

Practical considerations: Tools for EFT-aware histograms

- Now that we've talked about the concepts of EFT-aware histograms, let's discuss what this would look like in practice
- We know we need to store the quadratic parameterization for each bin
- But what really is the quadratic parameterization? Essentially it's just a list of terms, e.g. for two WCs:

$$\text{Quad parameterization} = s_0 + s_1 c_1 + s_2 c_1^2 + s_3 c_2 + s_4 c_1 c_2 + s_5 c_2^2$$

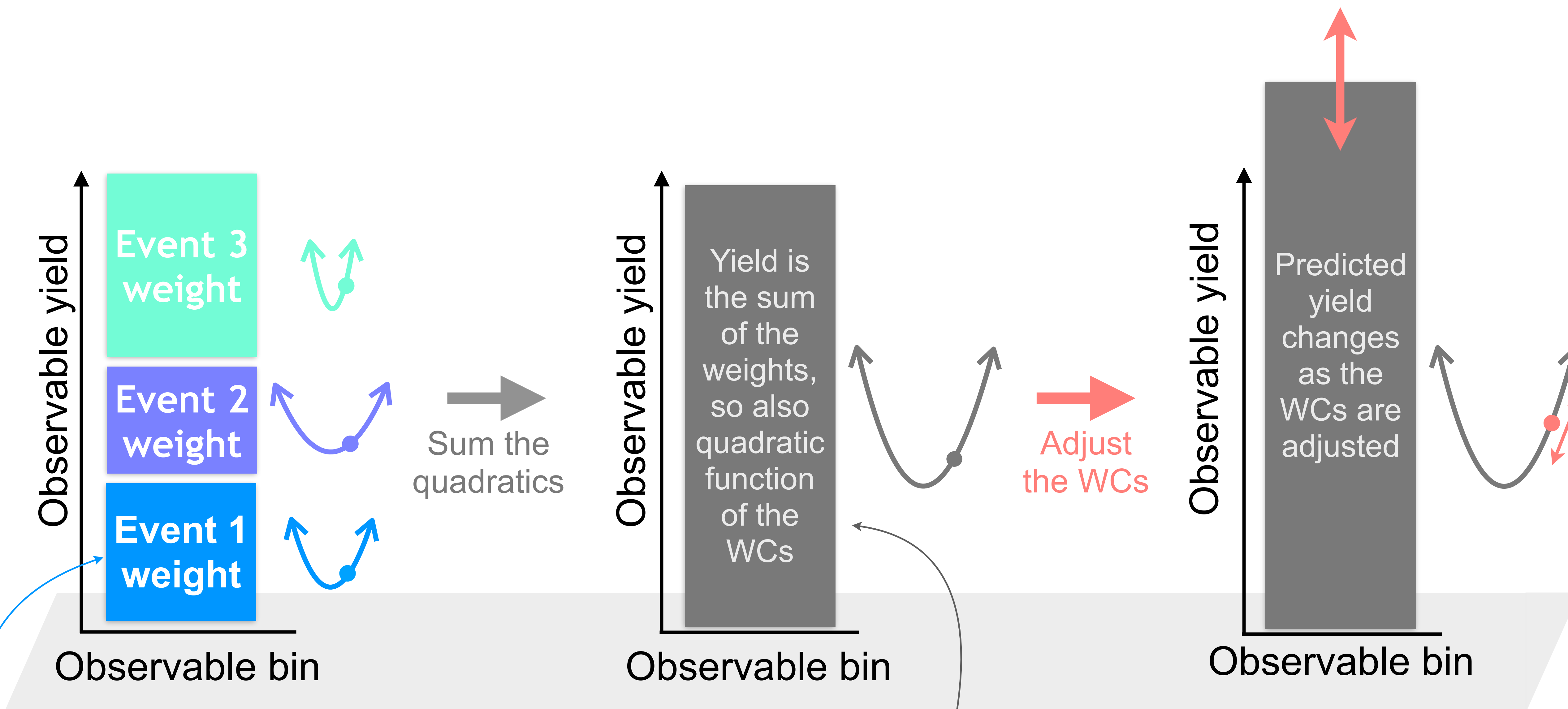
- The terms are essentially a structure constant (called "s" in the above) and the corresponding variables (i.e. the WCs denoted c_i)
- If we follow a convention for the order of the terms, we can just store the list of WCs $[c_1, c_2]$ and the structure constants $[s_0, s_1, s_2, s_3, s_4, s_5,]$ for each bin

See backup for discussion of ordering convention for structure constants

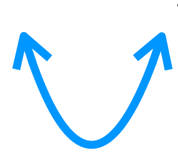
This is implemented in histEFT, which we will explore in the hands-on part coming next

Some history: TOP-19-001 developed EFT-aware "TH1EFT", then TOP-22-006 implemented new version on top of coffea hist and called it histEFT… but since coffee hist is now outdated, histEFT has recently been rewritten (by Ben Tovar of ND CCL) based on the scikit hep hist

Visualization of putting it all together (example with just one bin)

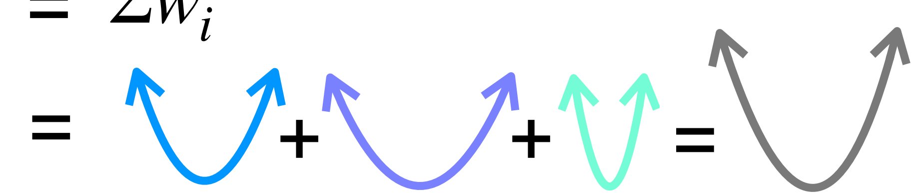


$$w_i = s_{0i} + \sum_j s_{1ij}\, c_j + \sum_j s_{2ij}\, c_j^2 + \sum_{j \neq k} s_{3ijk}\, c_j c_k$$

$=$ 

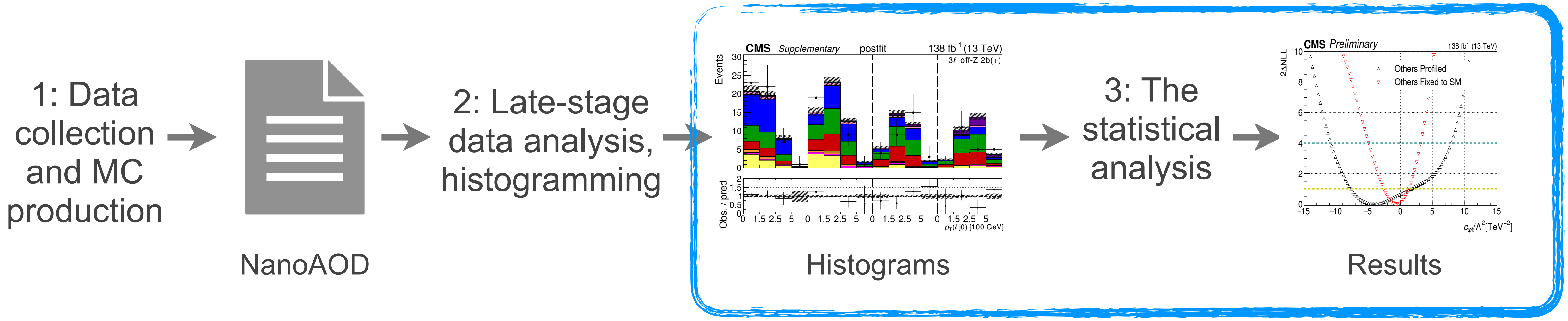
Obtain the fit coefficients for each event from MG reweighting

$$\text{Yield} = \Sigma w_i$$

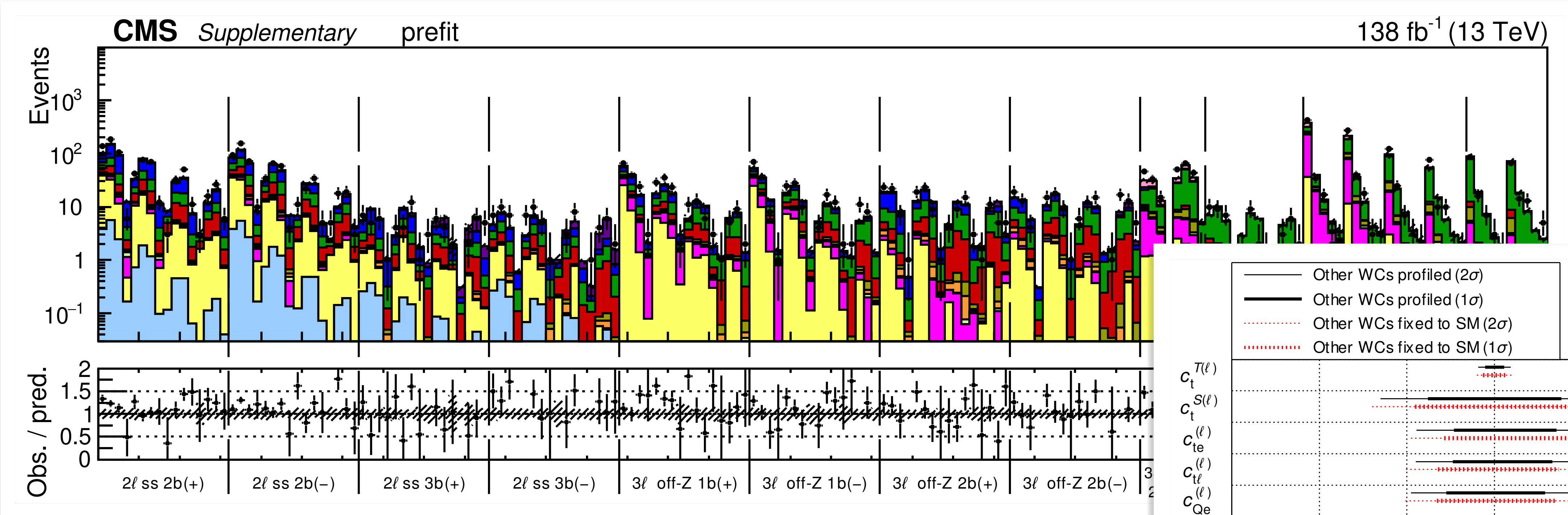


Outline for this talk

- Recap of EFT: What to know for an analysis
- Getting the prediction in terms of EFT (Step 1)
- EFT histogramming (Step 2)
- Extracting limits on EFT parameters (Step 3)

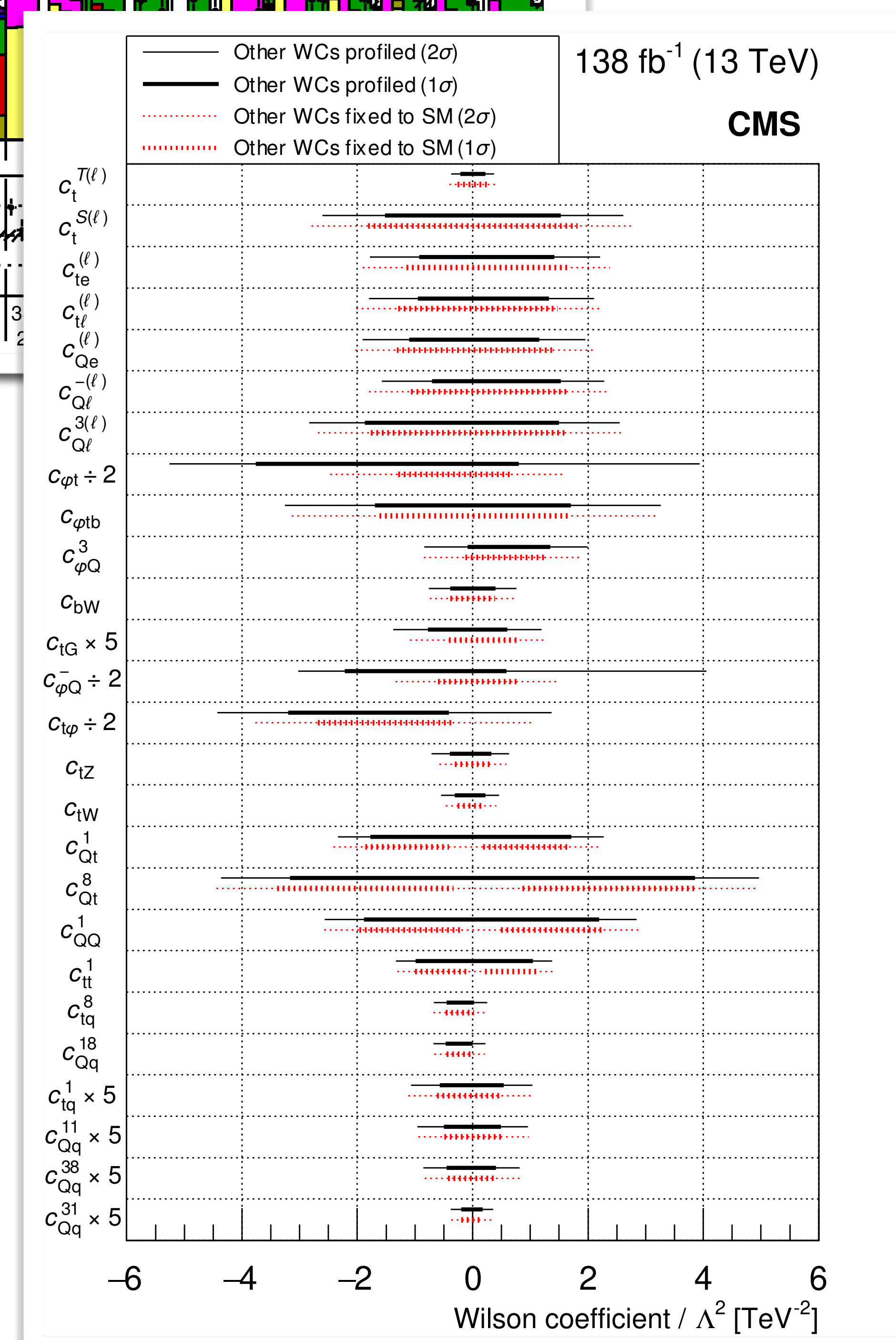


Statistical analysis



How to get from here…

To here:



Disclaimer: This section will be very brief, and only discuss one example (statistical treatment can vary by analysis)

The likelihood

- The likelihood characterizes the probability of measuring the observed number of events, given the theory i.e. $L = P(\text{data}|\text{theory})$
- Write the likelihood as a product over the N bins in the analysis, each treated as an independent Poisson measurement, with a mean corresponding to the predicted yield (which is a quadratic function of the WCs)
- We want to find the WC values that best agree with the data (i.e. that maximize the likelihood)

Product over the N bins in the analysis

n is the observed yield in the bin

m is the prediction in each bin, depends quadratically on the WCs θ, and the dependence on θ is different in every bin

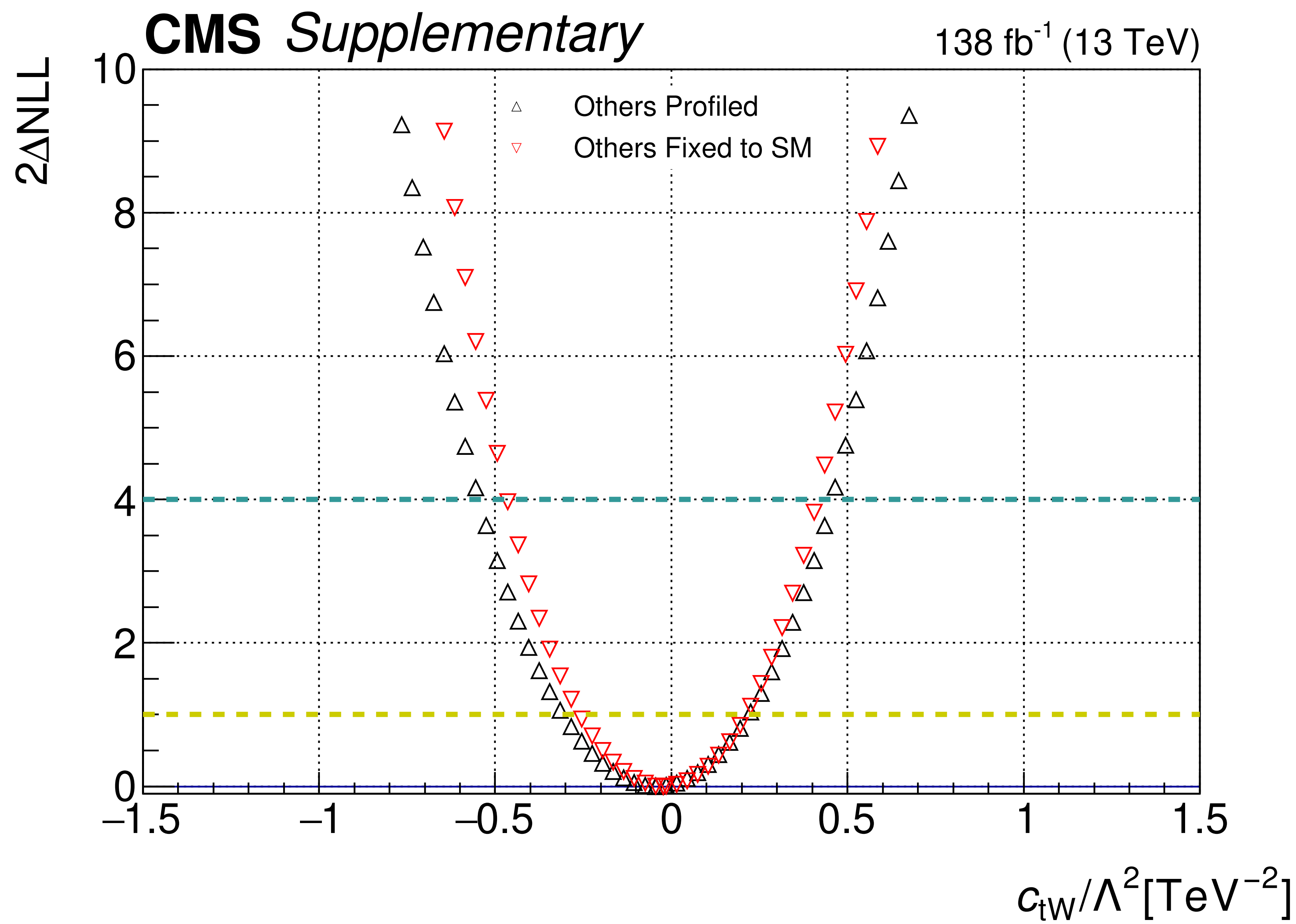
$$L = \prod_{i=1}^{N} \frac{m(\theta)_i^{n_i}}{n_i!} e^{-m(\theta)_i}$$

Understanding how the likelihood depends on the WCs

- We want to know how the likelihood depends on the WCs
- Ideally would scan across all WCs and map out the likelihood, but realistically this is too computationally expensive more more than a few WCs
    - E.g. for a recent CMS analysis (TOP-22-006) a back-of-the-envelope calculation indicated that even for a relatively sparse grid of 5 scan points in each direction and a relatively large amount of computing resources (10k CPU cores), it would take ~17 billion years to perform the scan
- Instead, we scan across one WC and *profile* the others (allowing them to float to their best fit value for the given scan point)
- This is computationally feasible, and allows us to understand the range of WC values that are consistent with the data

Extracting the confidence intervals

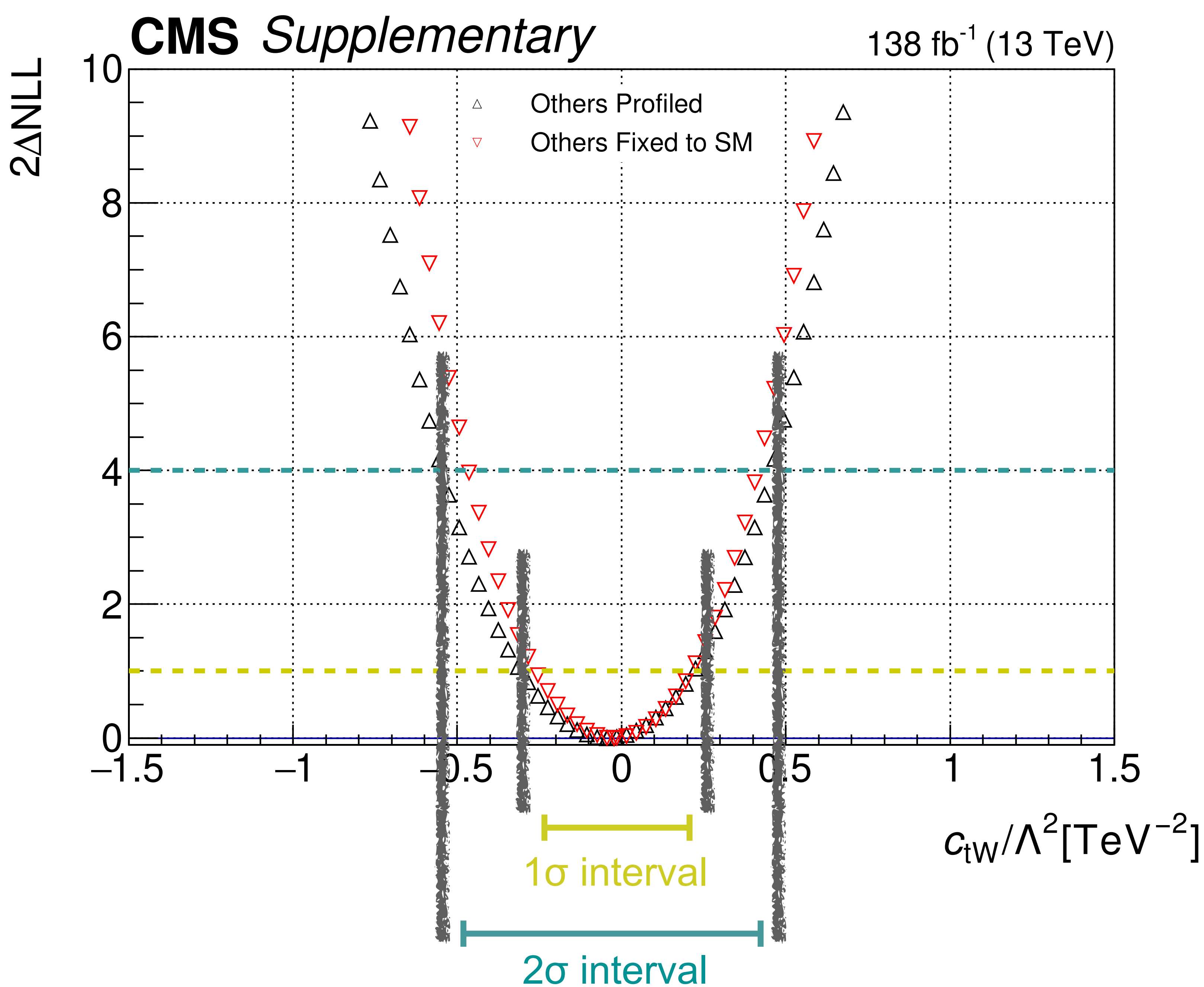
- For each WC, we scan across a range of values, profiling the other WCs
- We can then read off the best fit point and the one and two standard deviation confidence intervals from the scans



Since WC=0 is within the interval, this result is consistent with the SM

Extracting the confidence intervals

- For each WC, we scan across a range of values, profiling the other WCs
- We can then read off the best fit point and the one and two standard deviation confidence intervals from the scans



Since WC=0 is within the interval, this result is consistent with the SM

Recap of how the concepts fit into the workflow

Gridpack generation ← This is the step where we ask MG to find the weight at the set of reweight points

↓

MC event simulation ← Throughout the simulation steps in a general workflow (LHE->GEN->SIM->DIGI->RECO->MAOD->NAOD), we can just carry along the set of weights until the point where we want to extract the quadratic fit
In principle this can be done at any point in between the gridpack generation and histogramming, but in this tutorial we do it in the NanoGen step

↓

Histogramming ← Sum the quadratic parameterizations for all of the events passing the selection criteria for the given bins in order to find the parametrization for the predicted yield in the bin

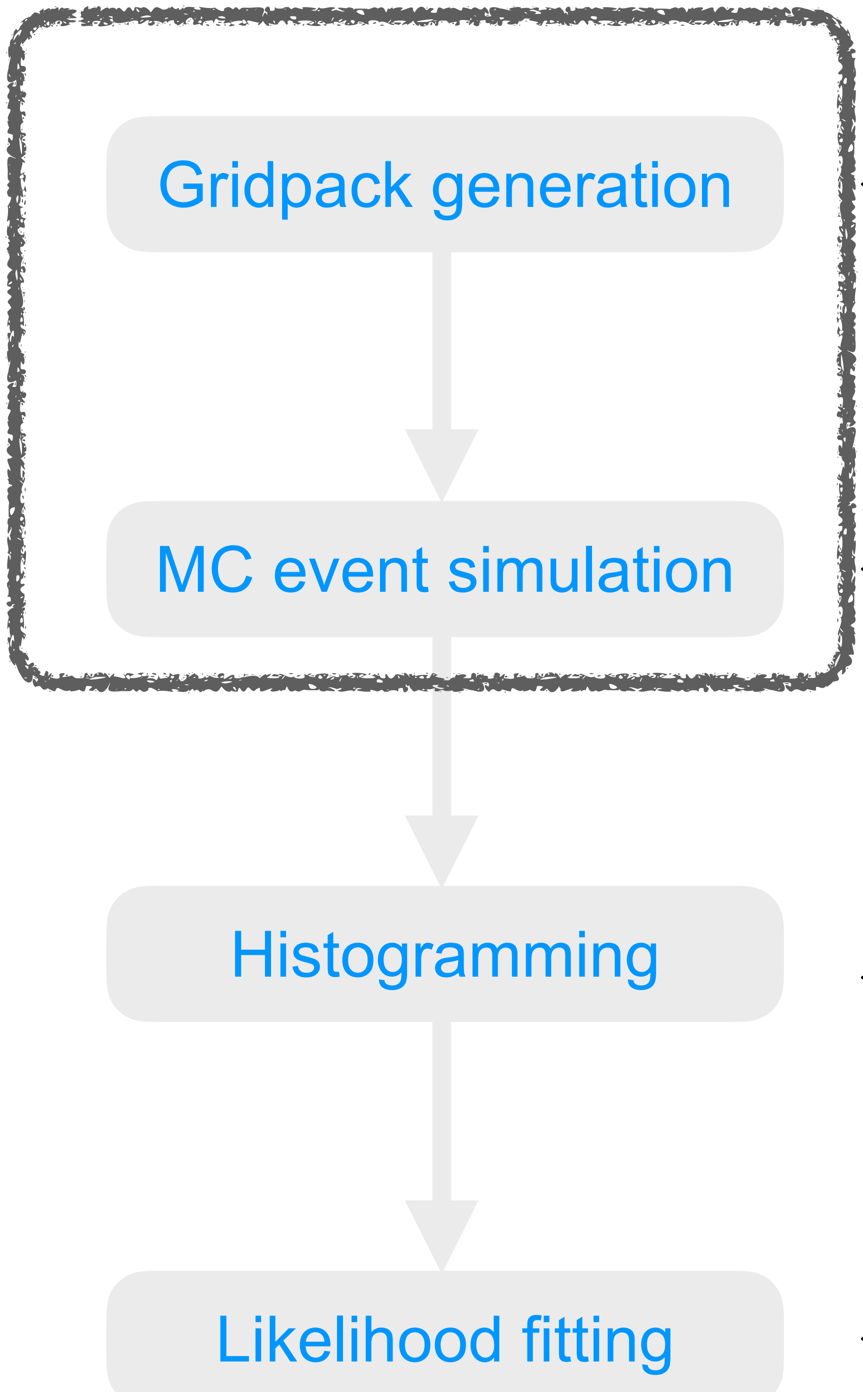
↓

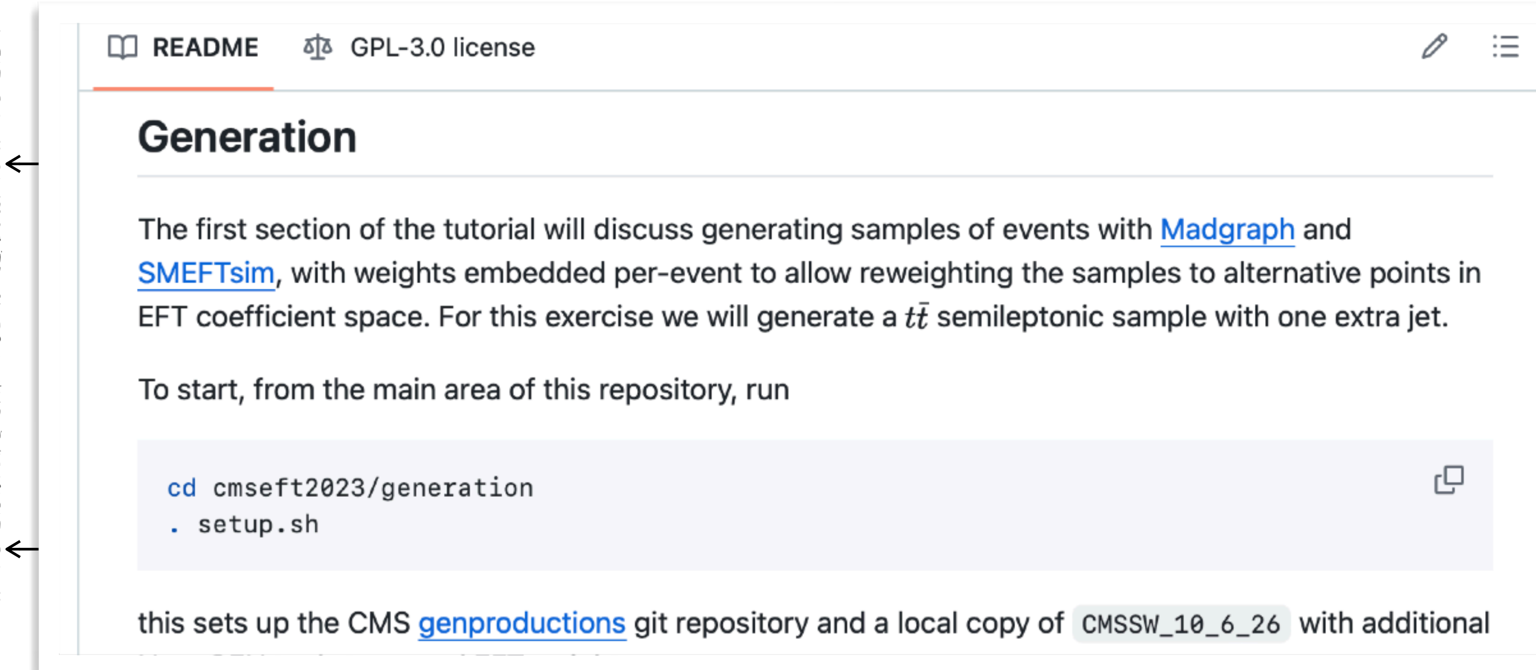
Likelihood fitting ← Compare the parametrized yields to observed data and extract best fit values and confidence intervals for WCs

Next up: Hands-on tutorial!

[Link to the repo for the tutorial]

Recap of how the concepts fit into the workflow





README GPL-3.0 license

Generation

The first section of the tutorial will discuss generating samples of events with Madgraph and SMEFTsim, with weights embedded per-event to allow reweighting the samples to alternative points in EFT coefficient space. For this exercise we will generate a $t\bar{t}$ semileptonic sample with one extra jet.

To start, from the main area of this repository, run

```
cd cmseft2023/generation
. setup.sh
```

this sets up the CMS genproductions git repository and a local copy of `CMSSW_10_6_26` with additional

gridpack generation and histogramming, but in this tutorial we do it in the NanoGen step

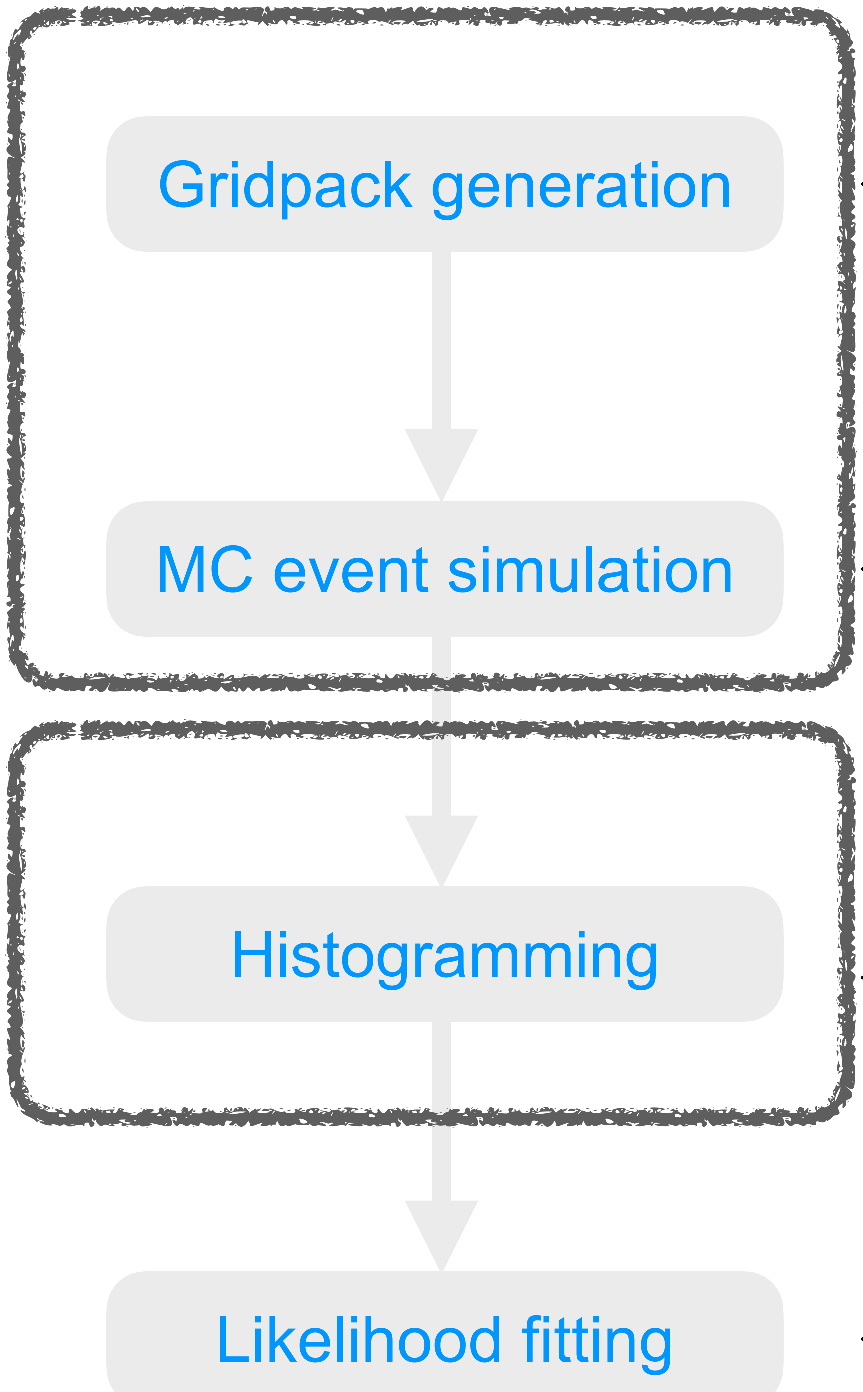
Sum the quadratic parameterizations for all of the events passing the selection criteria for the given bins in order to find the parametrization for the predicted yield in the bin

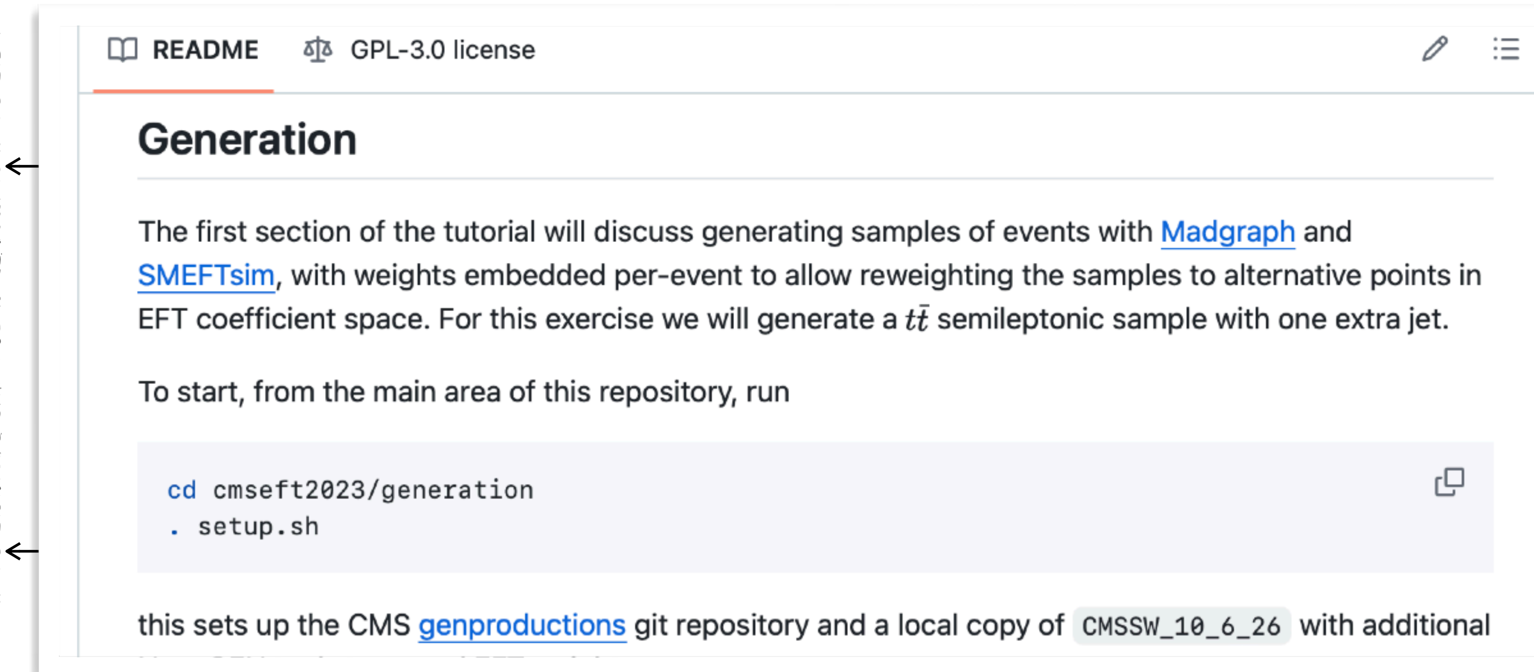
Compare the parametrized yields to observed data and extract best fit values and confidence intervals for WCs

Next up: Hands-on tutorial!

[Link to the repo for the tutorial]

Recap of how the concepts fit into the workflow




README GPL-3.0 license

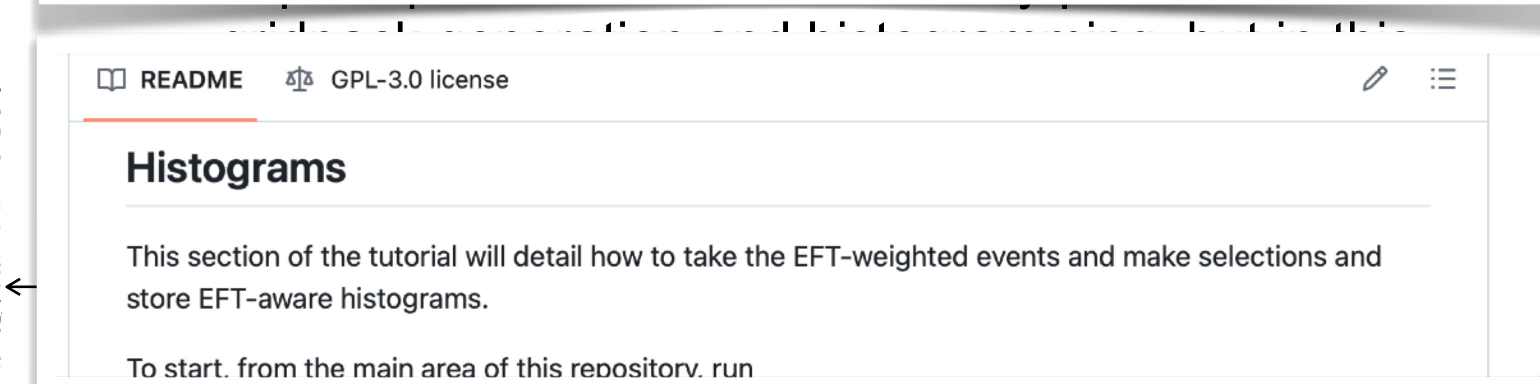
Generation

The first section of the tutorial will discuss generating samples of events with Madgraph and SMEFTsim, with weights embedded per-event to allow reweighting the samples to alternative points in EFT coefficient space. For this exercise we will generate a $t\bar{t}$ semileptonic sample with one extra jet.

To start, from the main area of this repository, run

```
cd cmseft2023/generation
. setup.sh
```

this sets up the CMS genproductions git repository and a local copy of `CMSSW_10_6_26` with additional


README GPL-3.0 license

Histograms

This section of the tutorial will detail how to take the EFT-weighted events and make selections and store EFT-aware histograms.

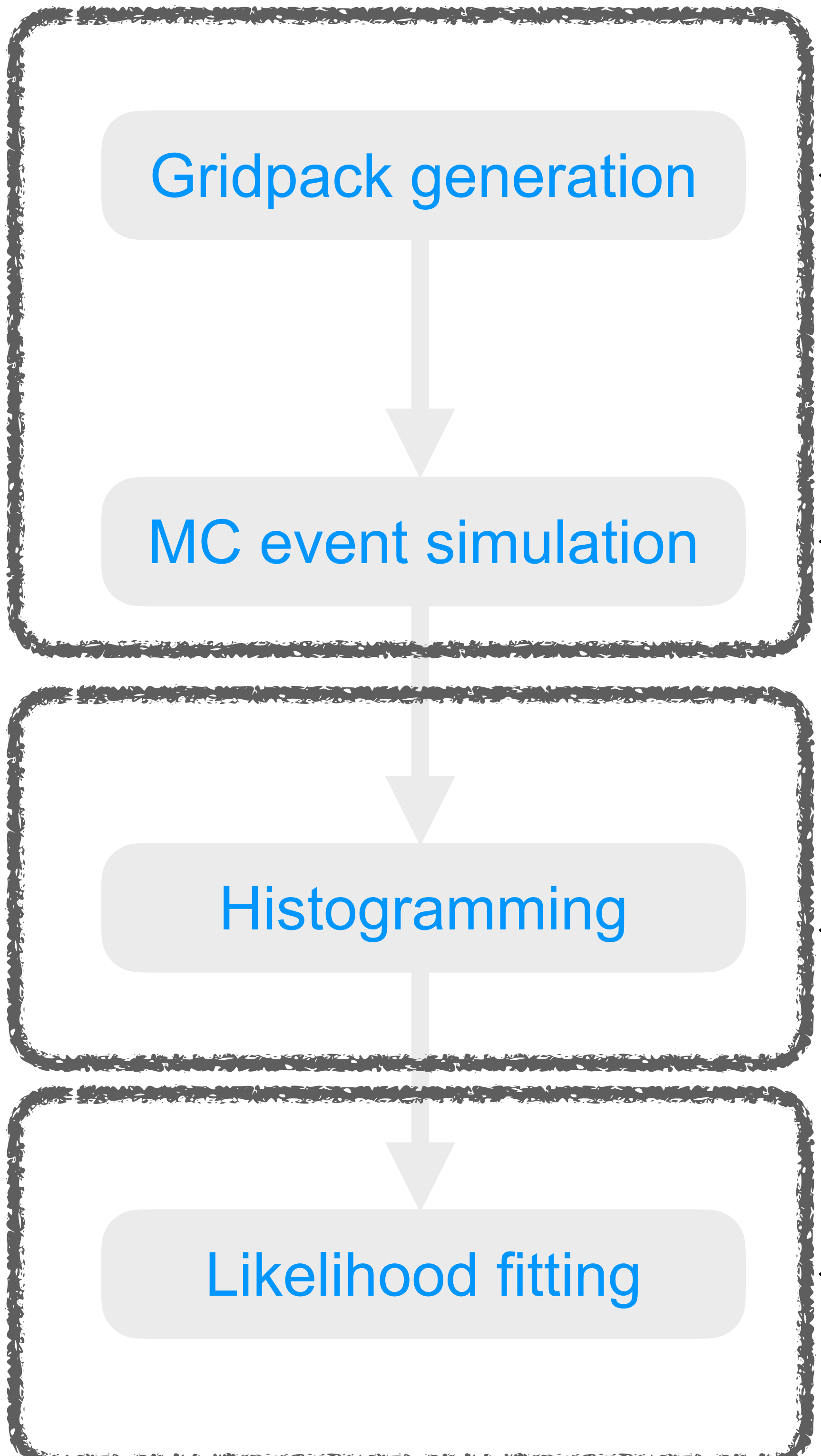
To start, from the main area of this repository, run

Compare the parametrized yields to observed data and extract best fit values and confidence intervals for WCs

Next up: Hands-on tutorial!

Link to the repo for the tutorial

Recap of how the concepts fit into the workflow



README | GPL-3.0 license

Generation

The first section of the tutorial will discuss generating samples of events with Madgraph and SMEFTsim, with weights embedded per-event to allow reweighting the samples to alternative points in EFT coefficient space. For this exercise we will generate a $t\bar{t}$ semileptonic sample with one extra jet.

To start, from the main area of this repository, run

```
cd cmseft2023/generation
. setup.sh
```

this sets up the CMS genproductions git repository and a local copy of `CMSSW_10_6_26` with additional

README | GPL-3.0 license

Histograms

This section of the tutorial will detail how to take the EFT-weighted events and make selections and store EFT-aware histograms.

To start, from the main area of this repository, run

README | GPL-3.0 license

Statistics

This section of the tutorial will demonstrate how to build a model from the template histograms and run fits using the Combine tool.

Next up: Hands-on tutorial!

Link to the repo for the tutorial

Backup

Advantageous vs more challenging aspects of the direct approach

Challenging ← → Advantageous

Analysis preservation/longevity

Reinterpretations

Need to produce detector-level EFT simulations

↗ These challenging aspects for direct approaches are generally advantages of the indirect approach

More information available → potential for more sensitivity

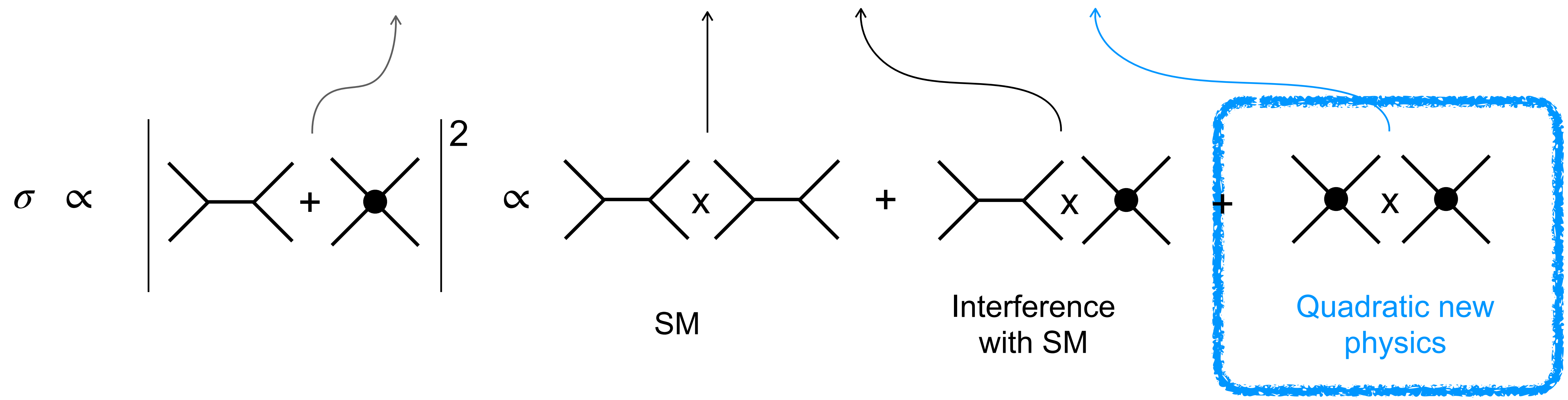
Can handle final states with complicated admixtures of processes all affected differently by EFT

Account for all relevant correlations

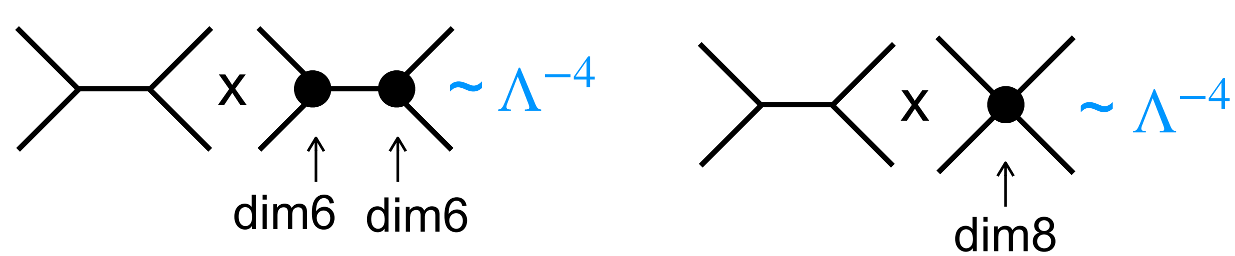
How do observables depend on EFT? Let's start with σ

If the EFT is modeled linearly in amplitude, the cross section is an *n*-quadratic in terms of the WCs (where *n* is number of WCs)

$$\sigma \propto \left| \mathcal{M}_{SM} + \frac{c_i}{\Lambda^2} \mathcal{M}_i \right|^2 \propto s_0 + s_i \frac{c_i}{\Lambda^2} + s_{ij} \frac{c_i}{\Lambda^2} \frac{c_j}{\Lambda^2} = \text{⤻}$$

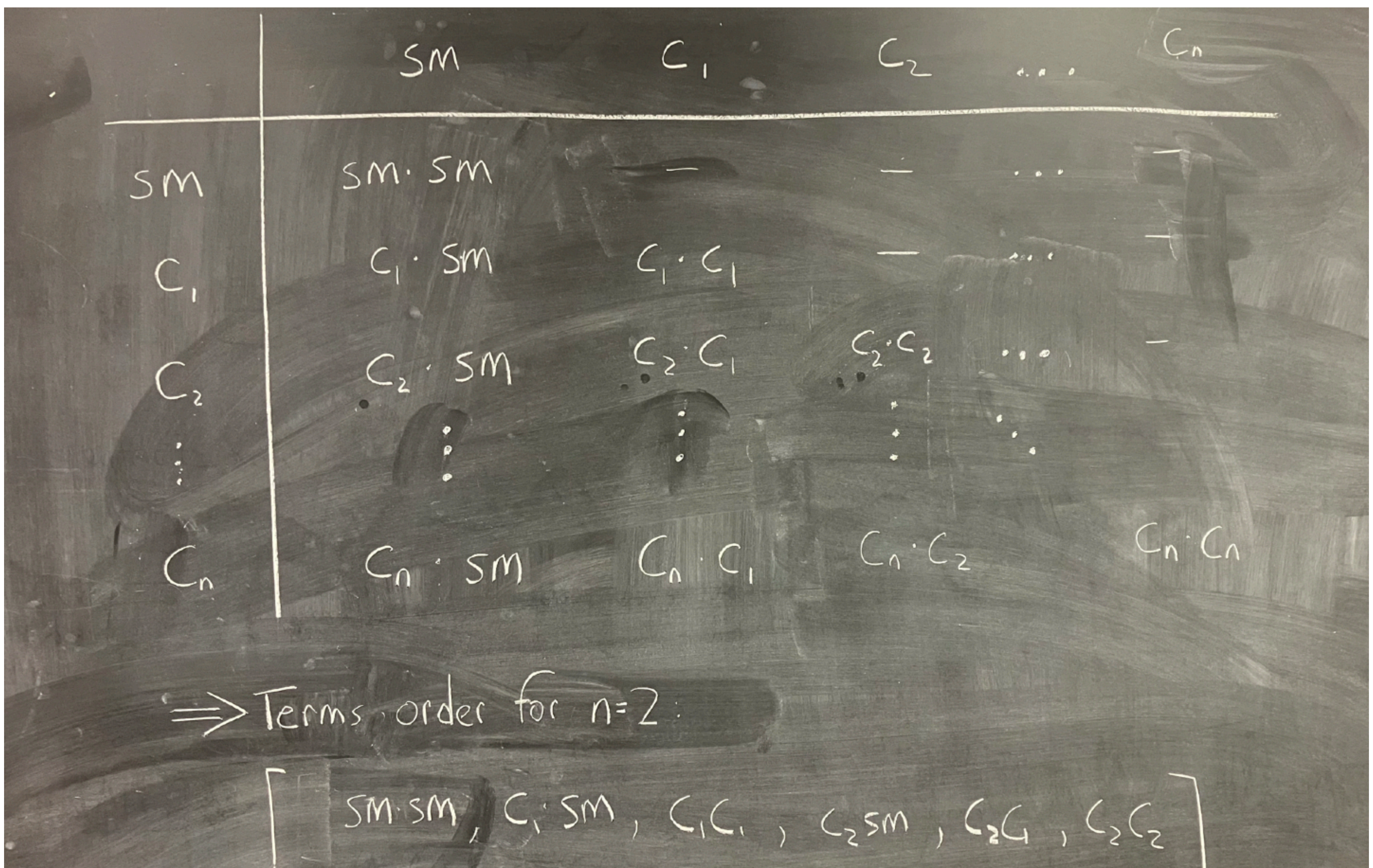


Other contributions at same Λ^{-4} order as quad piece



Term ordering convention for histEFT

- For histEFT, the term ordering convention follows the order of the lower triangle of an (n+1)x(n+1) matrix, where n is the number of WCs, and the order of the WCs is assumed to be [sm, c1, c2, …, cn]
- Thus, if you know the WC order, you can reconstruct the quadratic parametrization from the list of terms



Some technical considerations: Normalization of EFT-aware histograms

- Usually you don't want to use the normalization straight from your generated sample (usually for EFT samples this is LO)
- Want to normalize to the best available theory cross section, as usual
- Usually achieve this normalization by dividing summing the parameterizations for all all generated events, then reweighting to the SM* (i.e. the SM prediction for the total cross section, denoted w(SM))
- After dividing by the w(SM), the constant term in your quadratic parameterization is 1, so after scaling by the lumi and the NLO xsec, the constant piece is the SM predicted yield

$$\text{Expected yield } (\vec{c}) \;=\; \sigma_{SM}\, \mathcal{L}\; \frac{\sum_{\text{Pass}} w(\vec{c})}{\sum_{\text{Gen}} w(SM)},$$

* Note: This normalization approach is not possible in the case when the SM prediction for your sample is 0 (e.g. for FCNC samples, a different normalization approach is required)

Extracting the quadratic dependence, a toy example

- Let's say we have just two WCs, called c_1 and c_2
- We thus need 6 reweight points: $((n+1)^2 - (n+1))/2 + n + 1 \ |_{n=2} \ = \ 6$
- Let's say we run MG and get the following weights at the following points: ⟶

c1	c2	weight
0	0	1.000
0	1	0.909
5	0	1.403
5	10	0.721
-5	10	0.333
-10	10	0.418
10	10	1.194

(Notice that we have one more point than we need! This will let us make sure that this shape indeed looks quadratic)

- What we want to find are the structure constants (let's call them $\vec{s}$), given the set of reweight points and weights, i.e.: $\mathbf{A}\,\vec{s} = \vec{w}$

$$A = \begin{bmatrix} 1 & (c_1)_0 & (c_2)_0 & (c_1^2)_0 & (c_2^2)_0 & (c_1c_2)_0 \\ 1 & (c_1)_1 & (c_2)_1 & (c_1^2)_1 & (c_2^2)_1 & (c_1c_2)_1 \\ 1 & (c_1)_2 & (c_2)_2 & (c_1^2)_2 & (c_2^2)_2 & (c_1c_2)_2 \\ 1 & (c_1)_3 & (c_2)_3 & (c_1^2)_3 & (c_2^2)_3 & (c_1c_2)_3 \\ 1 & (c_1)_4 & (c_2)_4 & (c_1^2)_4 & (c_2^2)_4 & (c_1c_2)_4 \\ 1 & (c_1)_5 & (c_2)_5 & (c_1^2)_5 & (c_2^2)_5 & (c_1c_2)_5 \\ 1 & (c_1)_6 & (c_2)_6 & (c_1^2)_6 & (c_2^2)_6 & (c_1c_2)_6 \end{bmatrix}, \quad \vec{s} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix}, \quad \vec{w} = \begin{bmatrix} w_0(c_1,c_2) \\ w_1(c_1,c_2) \\ w_2(c_1,c_2) \\ w_3(c_1,c_2) \\ w_4(c_1,c_2) \\ w_5(c_1,c_2) \\ w_6(c_1,c_2) \end{bmatrix}^{\mathrm{T}}$$

Extracting the quadratic dependence, a toy example

- Let's plug in the numbers from our seven reweight points and find the $\vec{s}$ that minimizes $||\vec{w} - \mathbf{A}\vec{s}||$ using numpy.linalg.lstsq

```
import numpy as np

w = [ 1.000, 0.909, 1.403, 0.721, 0.333, 0.418, 1.194]
A = [
     [1.0,   0.0,  0.0,   (0.0)**2,  (0.0)**2,   (0.0)*(0.0)  ],
     [1.0,   0.0,  1.0,   (0.0)**2,  (1.0)*2,    (0.0)*(1.0)  ],
     [1.0,   5.0,  0.0,   (5.0)**2,  (0.0)**2,   (5.0)*(0.0)  ],
     [1.0,   5.0, 10.0,   (5.0)**2, (10.0)**2,   (5.0)*(10.0) ],
     [1.0,  -5.0, 10.0,  (-5.0)**2, (10.0)**2,  (-5.0)*(10.0) ],
     [1.0, -10.0, 10.0, (-10.0)**2, (10.0)**2, (-10.0)*(10.0) ],
     [1.0,  10.0, 10.0,  (10.0)**2, (10.0)**2,  (10.0)*(10.0) ],
]

s, resid, _, _ = np.linalg.lstsq(A,w,rcond=None)
```

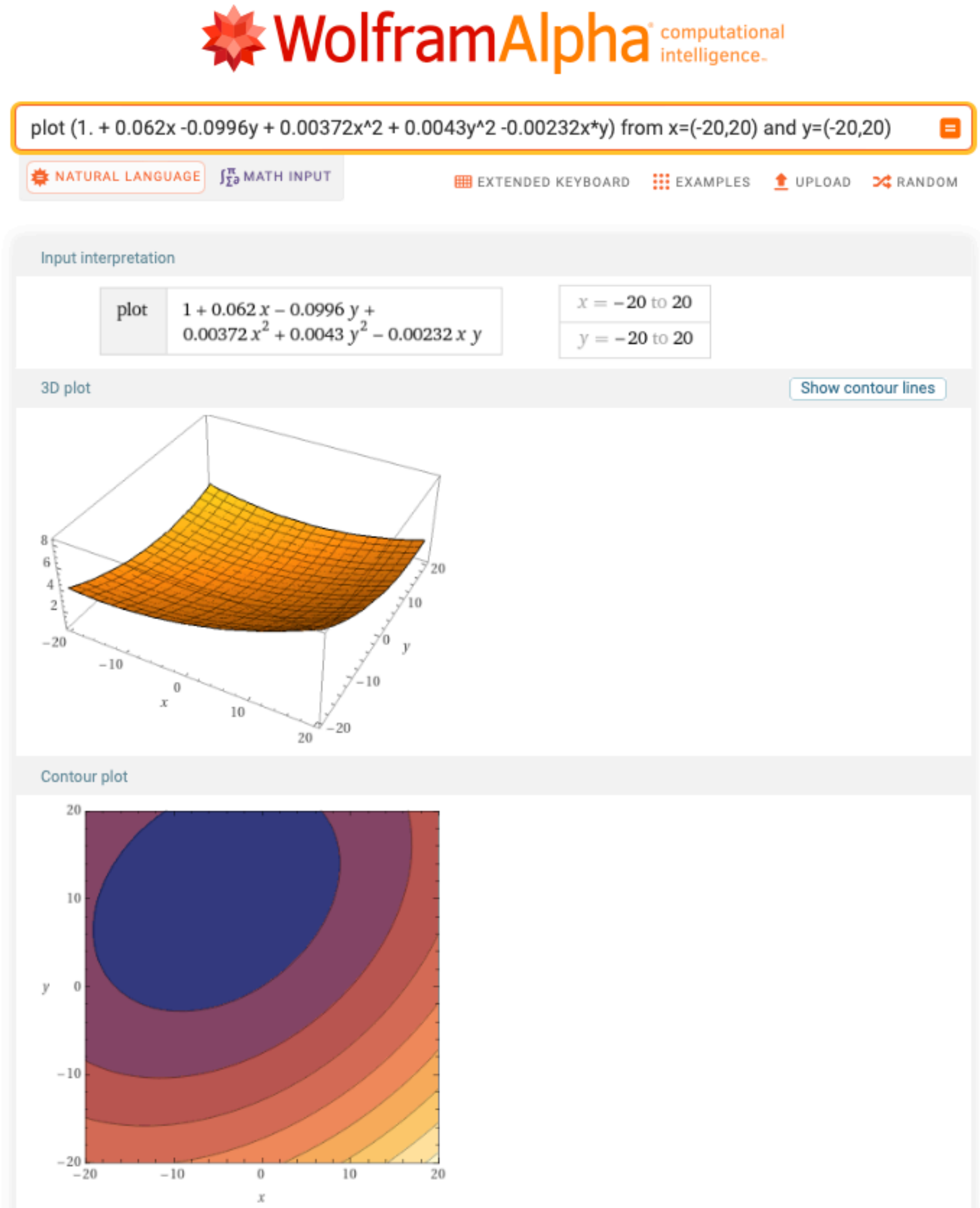
And the sum of the squared residuals is just 1.15168414e-30, not too big :)

- We find that: `s = [ 1.0, 0.062, -0.0996, 0.00372, 0.0043, -0.00232]`

- This means our quadratic dependence of the weight on WCs is thus:

$$w(c_1, c_2) = 1 + 0.062\,c_1 - 0.0996\,c_2 + 0.00372\,c_1^2 + 0.0043\,c_2^2 - 0.00232\,c_1 c_2$$

Just for fun →


WolframAlpha computational intelligence.
plot (1. + 0.062x -0.0996y + 0.00372x^2 + 0.0043y^2 -0.00232x*y) from x=(-20,20) and y=(-20,20)
NATURAL LANGUAGE
MATH INPUT
EXTENDED KEYBOARD
EXAMPLES
UPLOAD
RANDOM
Input interpretation
plot
$1 + 0.062\,x - 0.0996\,y + 0.00372\,x^2 + 0.0043\,y^2 - 0.00232\,x\,y$
$x = -20$ to 20
$y = -20$ to 20
3D plot
Show contour lines
Contour plot
x
y