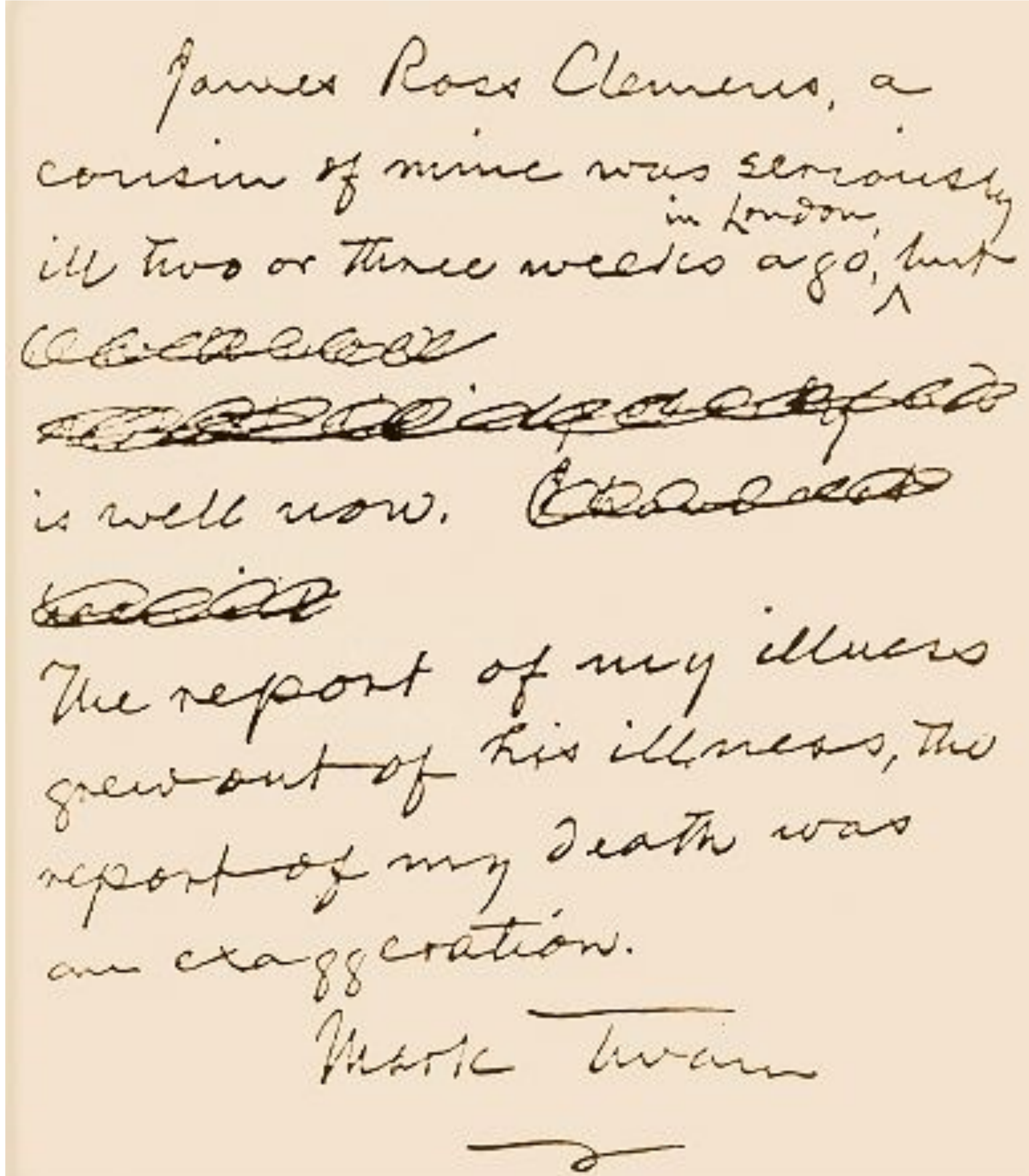

James Ross Clemens, a
cousin of mine was seriously
ill two or three weeks ago, in London, but
is well now.
The report of my illness
grew out of his illness, the
report of my death was
an exaggeration.
Mark Twain

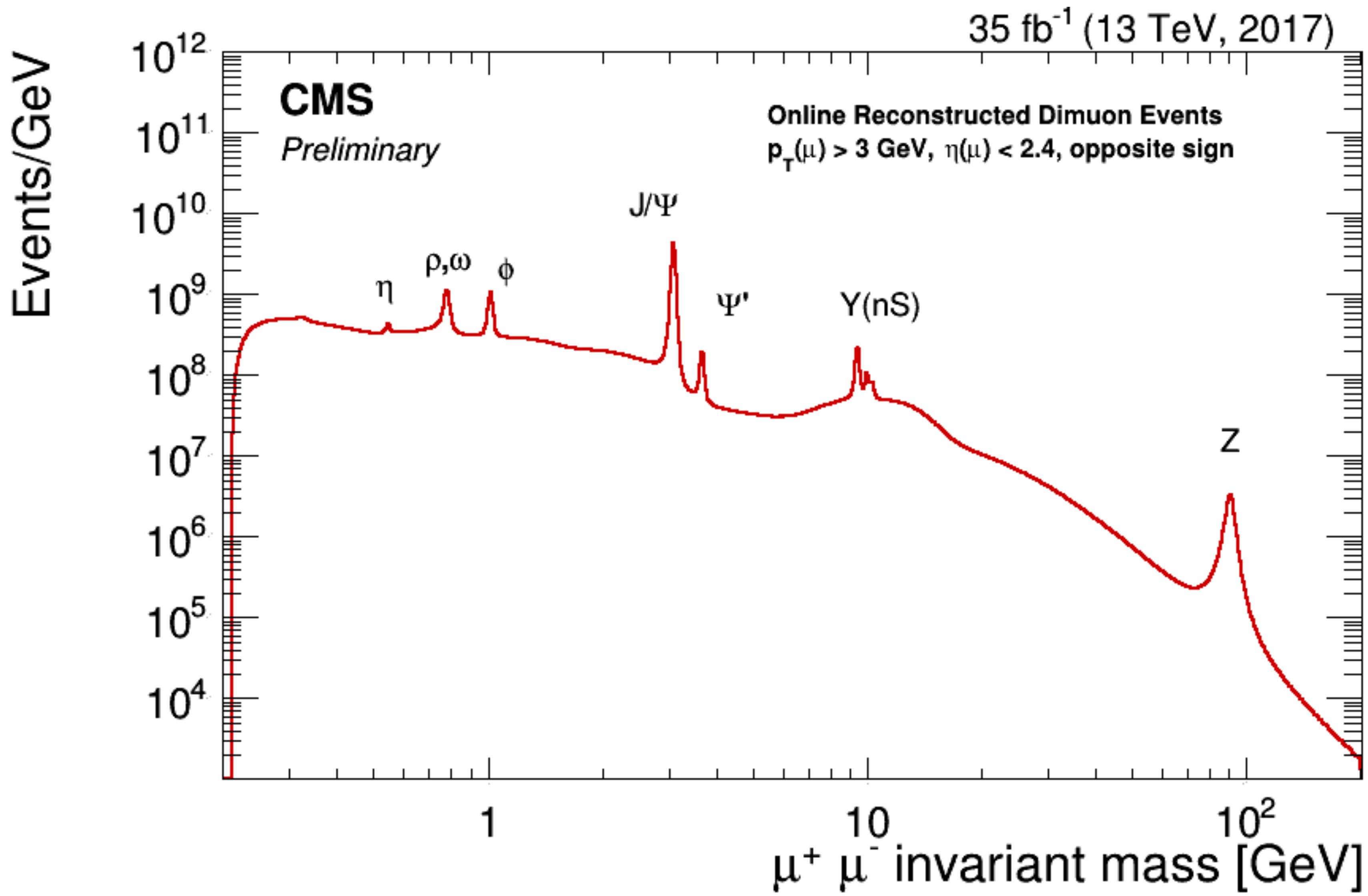
# BSM Experimental Searches 2

Zach Marshall (LBNL), CERN-Fermilab HCP Summer School, 1 September 2021


- We talked about *picking a model or signature*
- Parameters and assumptions
- And a bit about trigger and reconstruction
- Now we move on to *designing your search*

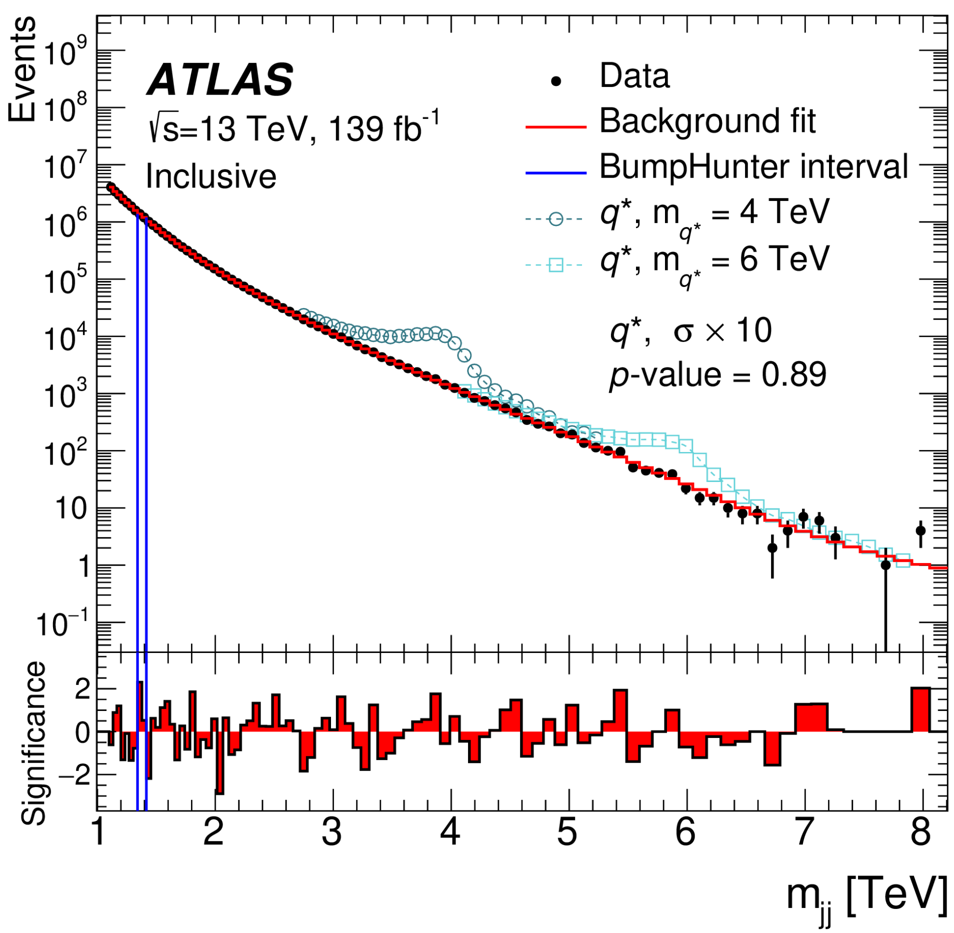
# What kind of search is it?

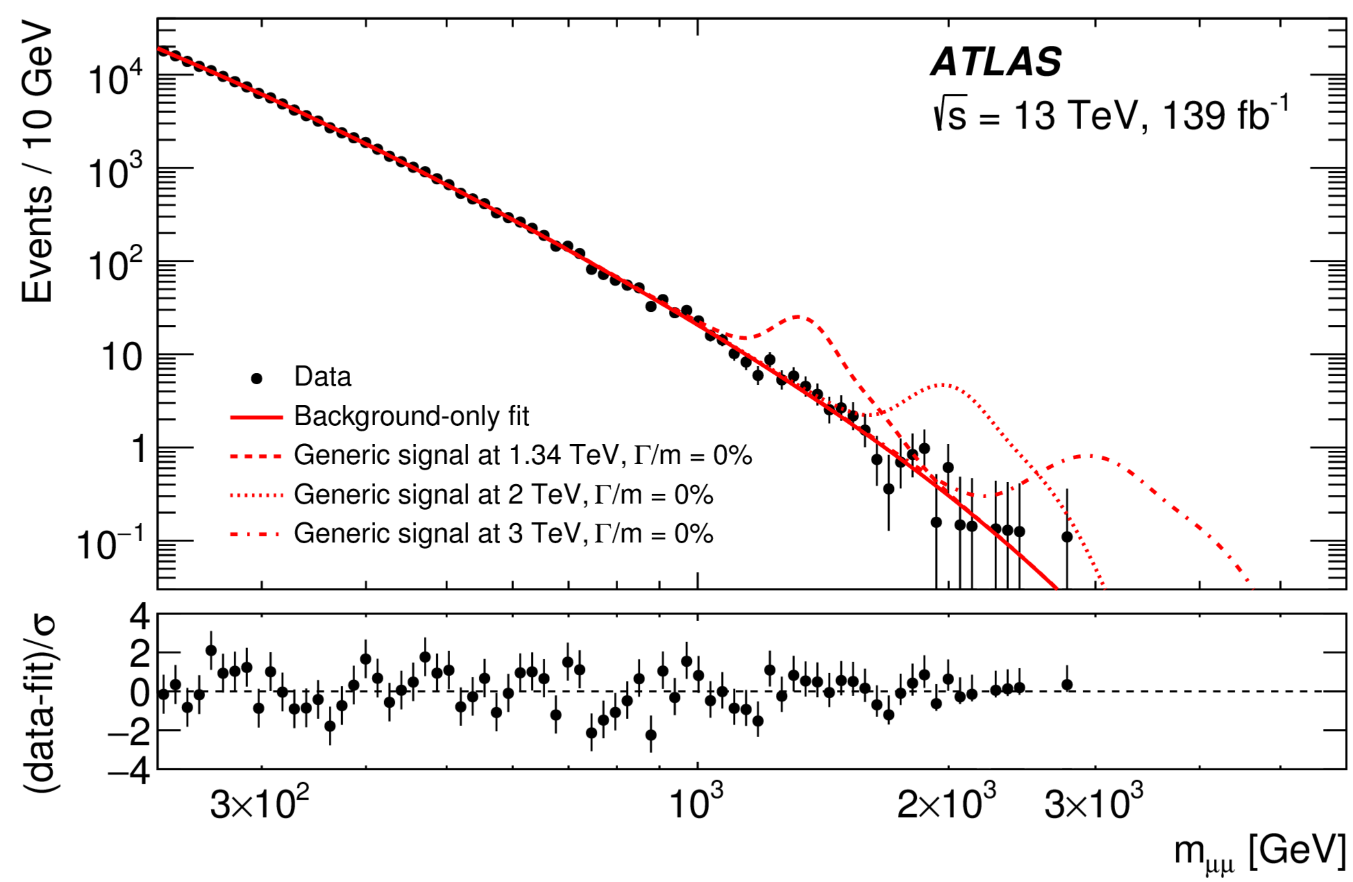
- Once you’re pretty sure your signal will be recorded and reconstructed, it’s time to decide how you’re going to look for it!
- Let’s start with an “easy” one: a **bump hunt**



# Hunting Bumps

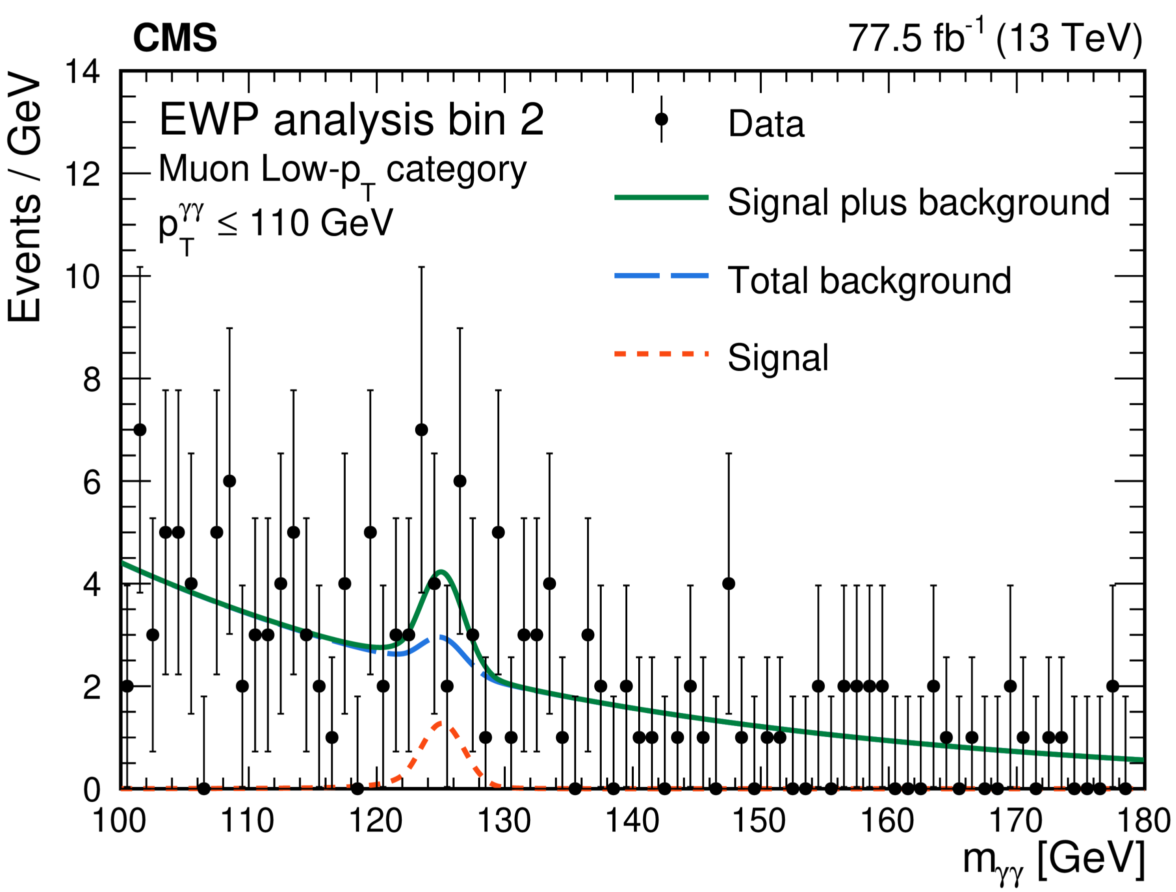
- There are only a few questions when designing a bump hunt:
  - In what observable is the bump? E.g. invariant mass of what?
  - How peaked is the bump (the narrower the better)?
  - Do you need any other cuts to make the bump show up?
- You're almost immediately ready for statistical analysis!

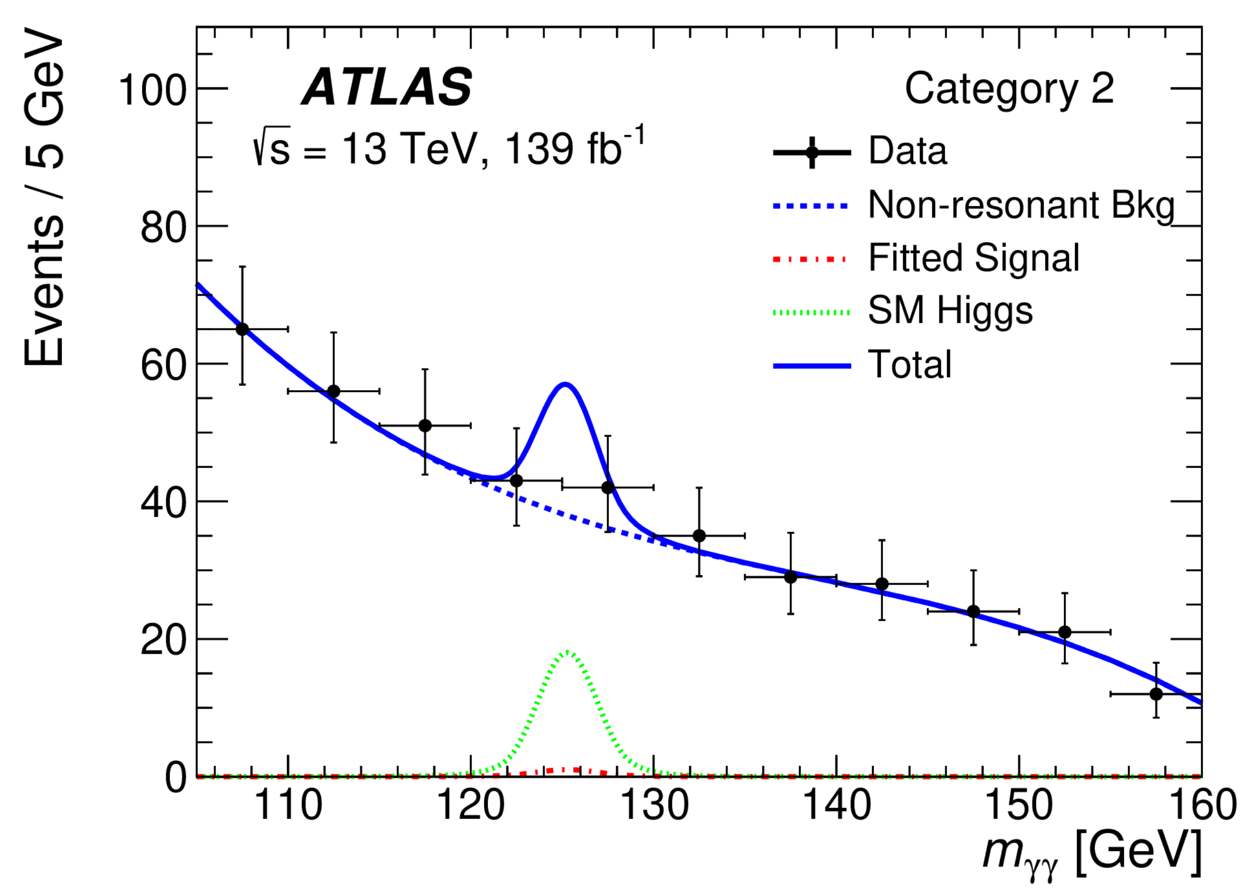


# Hunting Bumps

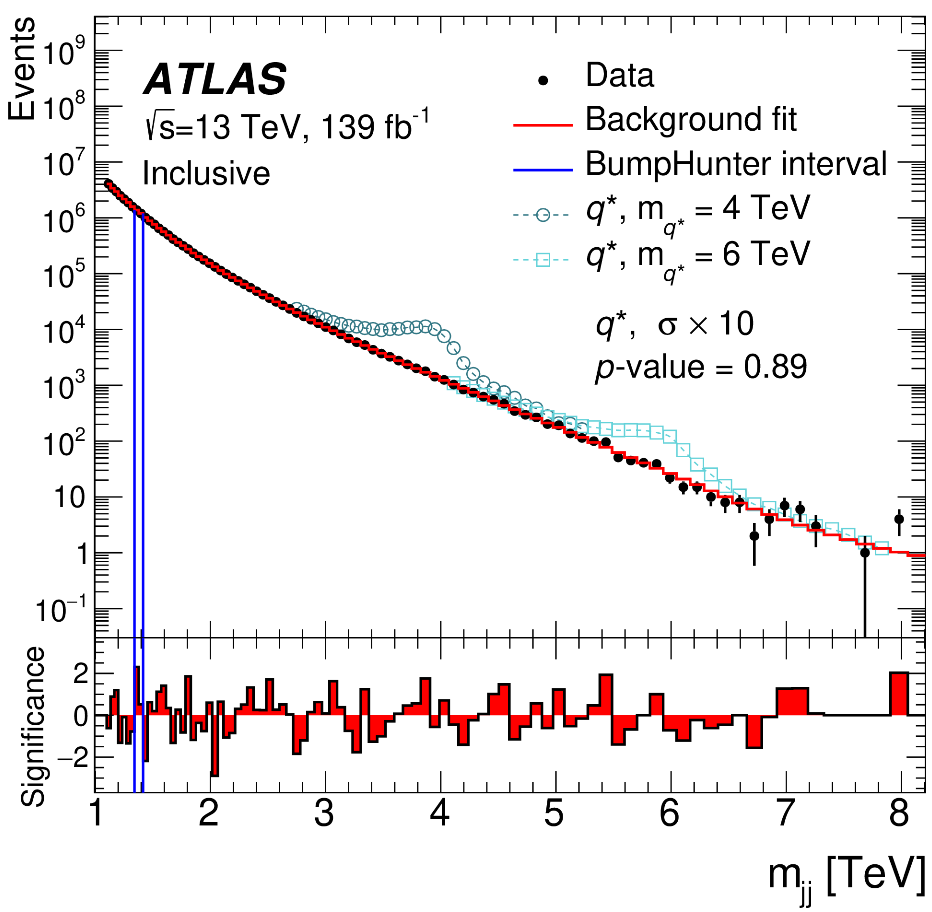
- Generally, bump hunts are easier than many other search setups
- There is a lot of good technology to use
  - Fit functions, statistical tools, etc
- If you can *turn your search into a bump hunt*, do it!
  - Conceptually, this is where the "R-Jigsaw variables" came from

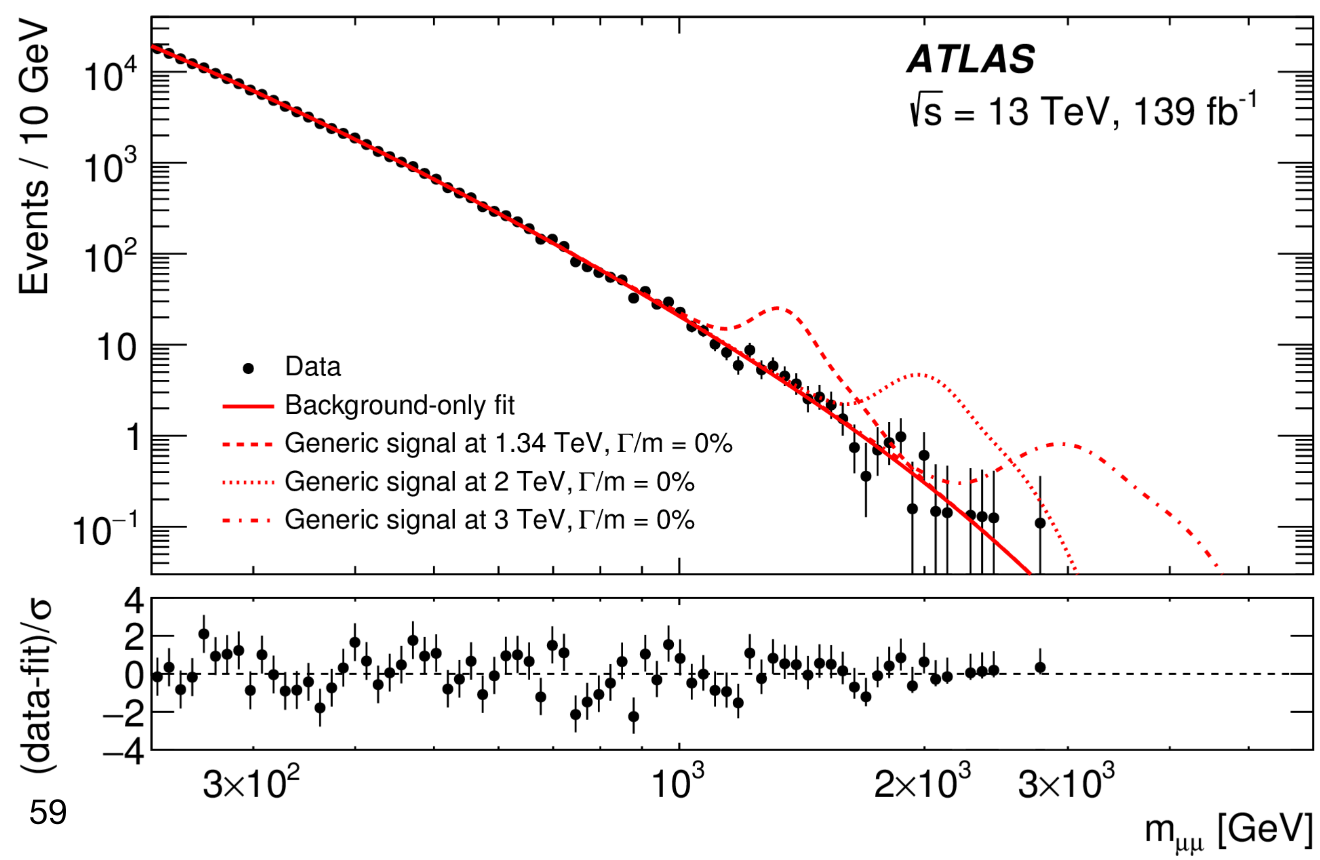


# Basic Steps for Hunting Bumps

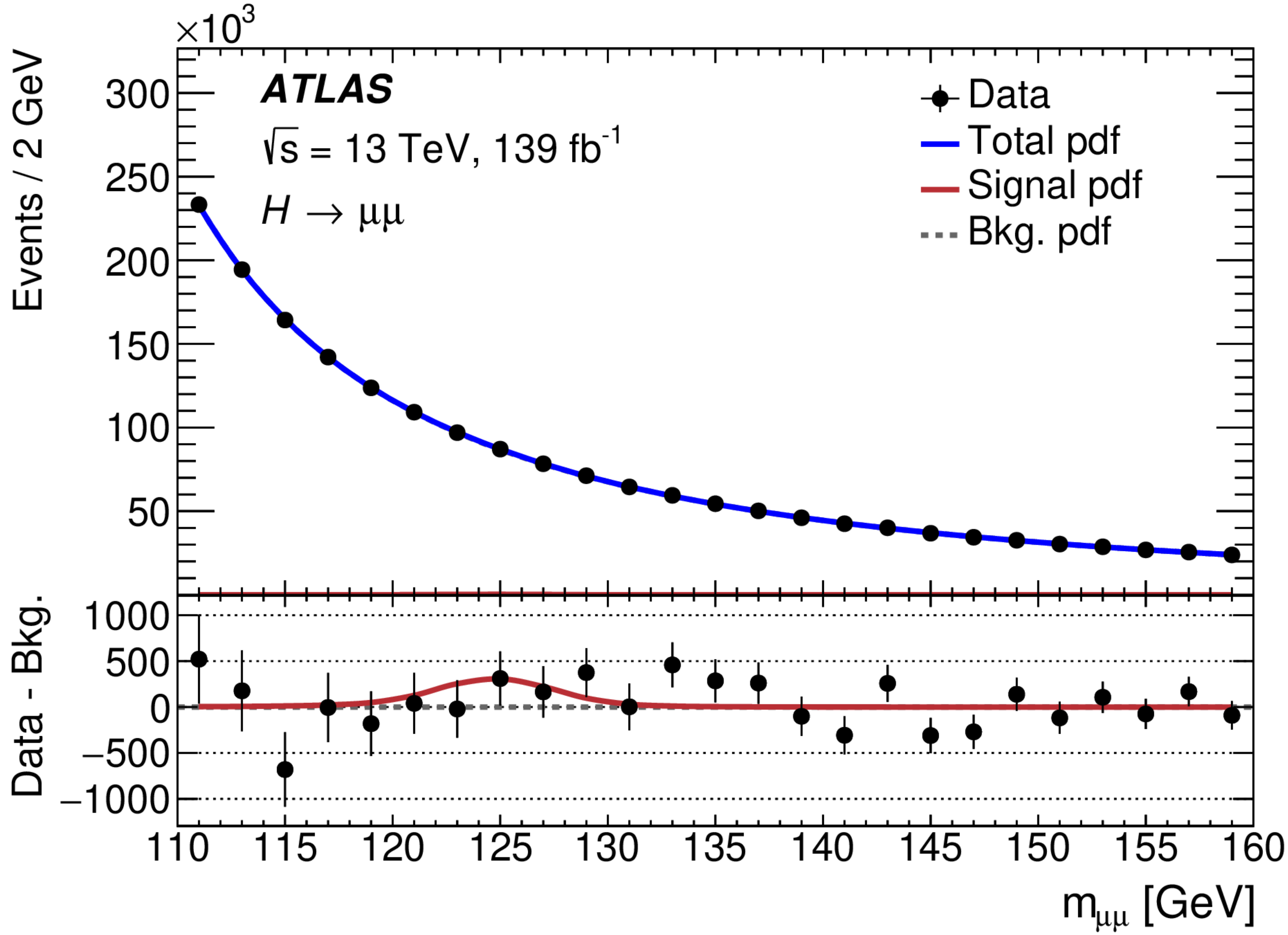
- Do you have one dominant background? If so, use a function!
    - Nature is usually (mostly) smooth, so your background should be too
- Pick a window to look in based on your signal width
    - The distribution outside the window is what you'll estimate that background with!
- Slide the window along the distribution and look for signal!



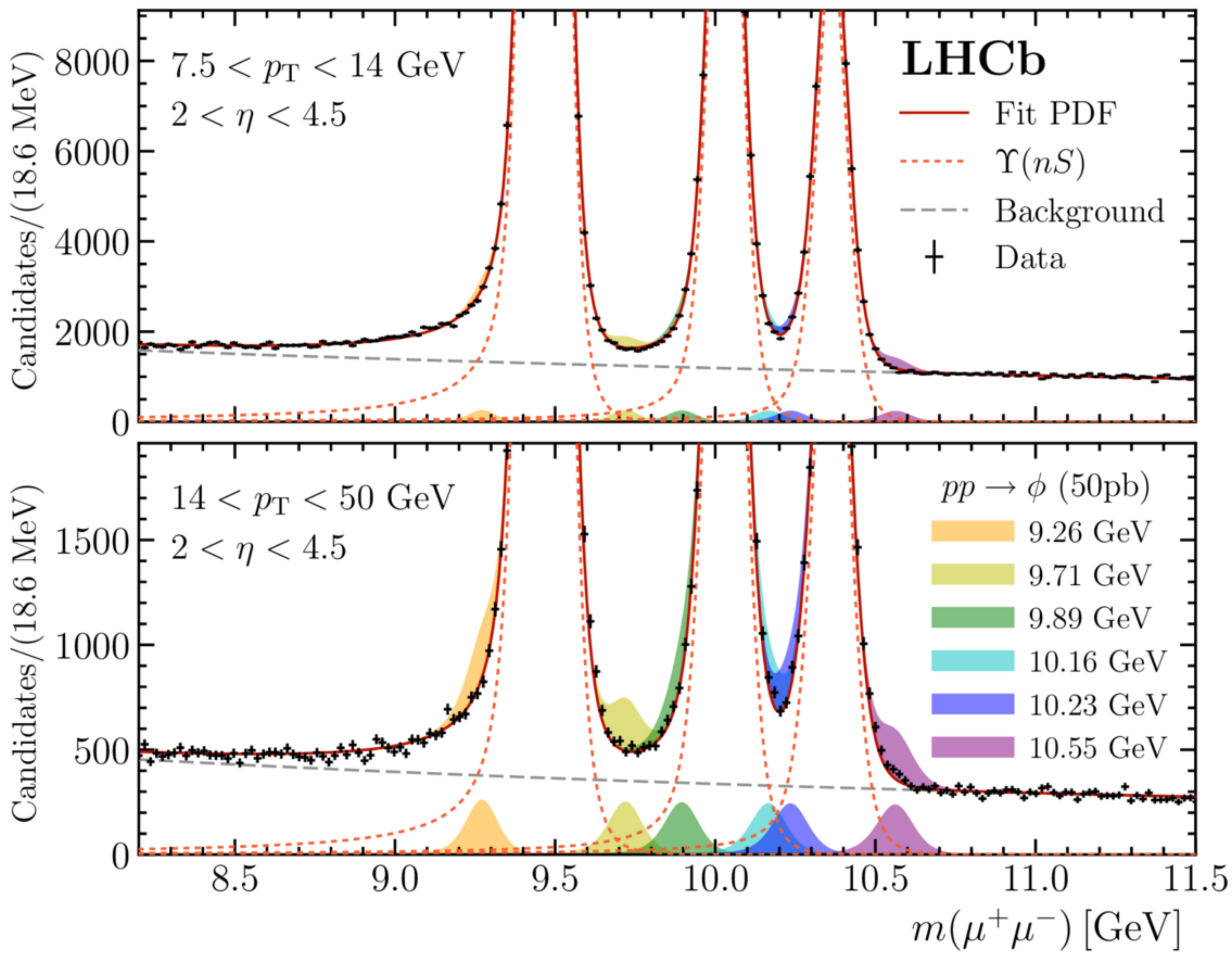
# Bump Hunts: Hard Mode

- If you have a *lot of background*, expect to spend time estimating biases in your background estimate ("spurious signal")



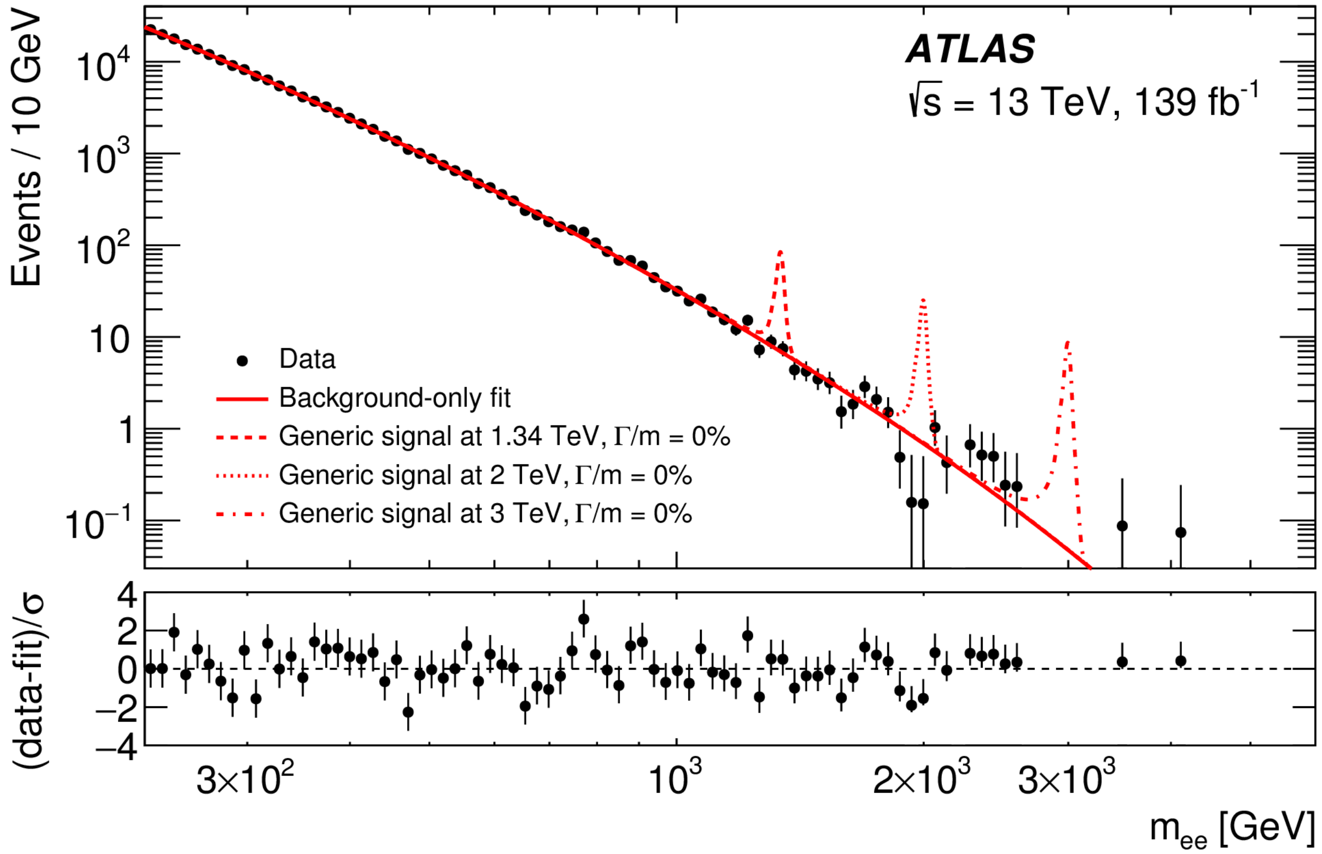
# Bump Hunts: Hard Mode

- If you have *bumpy backgrounds*, expect to spend a lot of time understanding those other structures (and setting up functions)

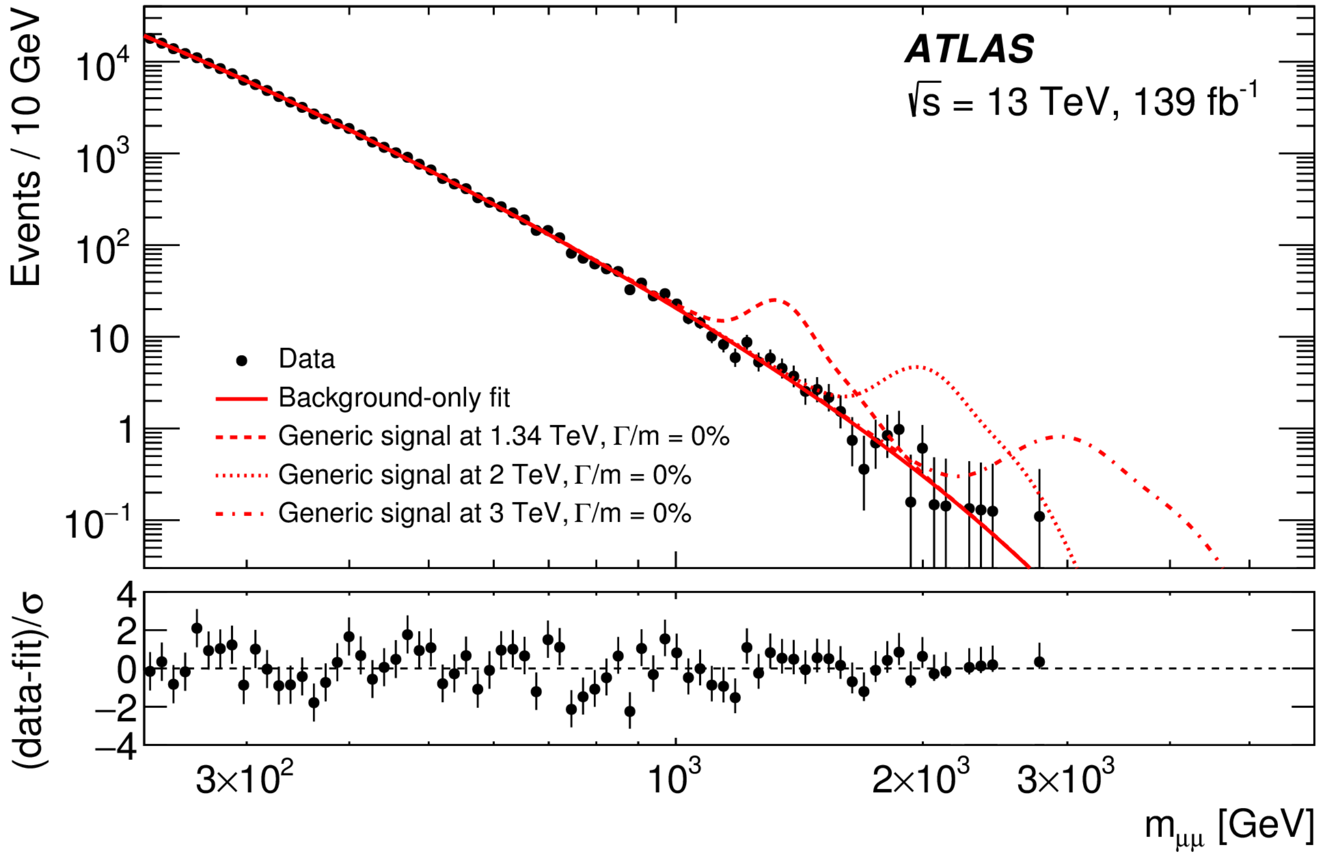


# Bump Hunts: Hard Mode

- If your signal appears in several places, remember it may not always look the same!



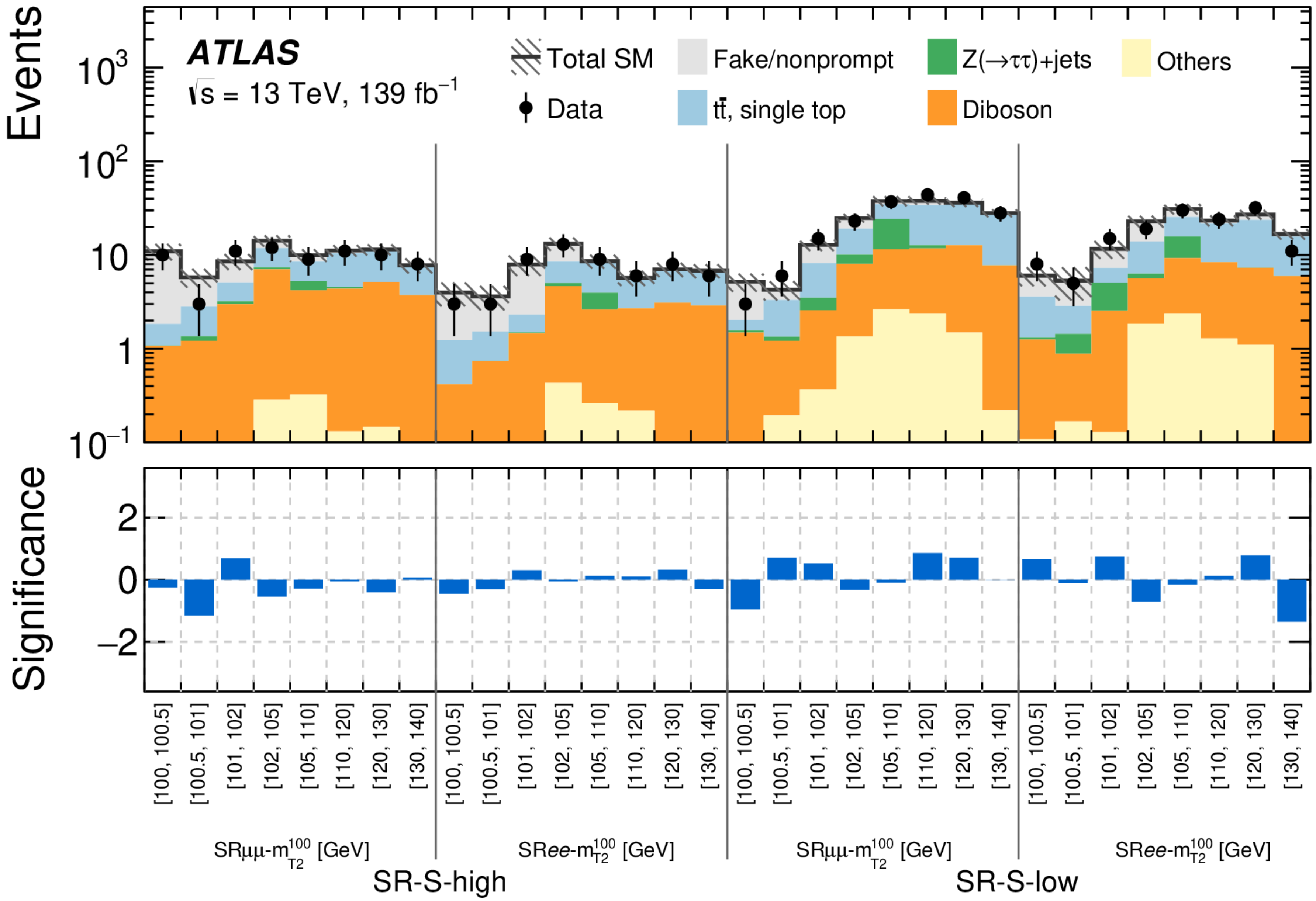

Calorimeter measurement




Tracker measurement

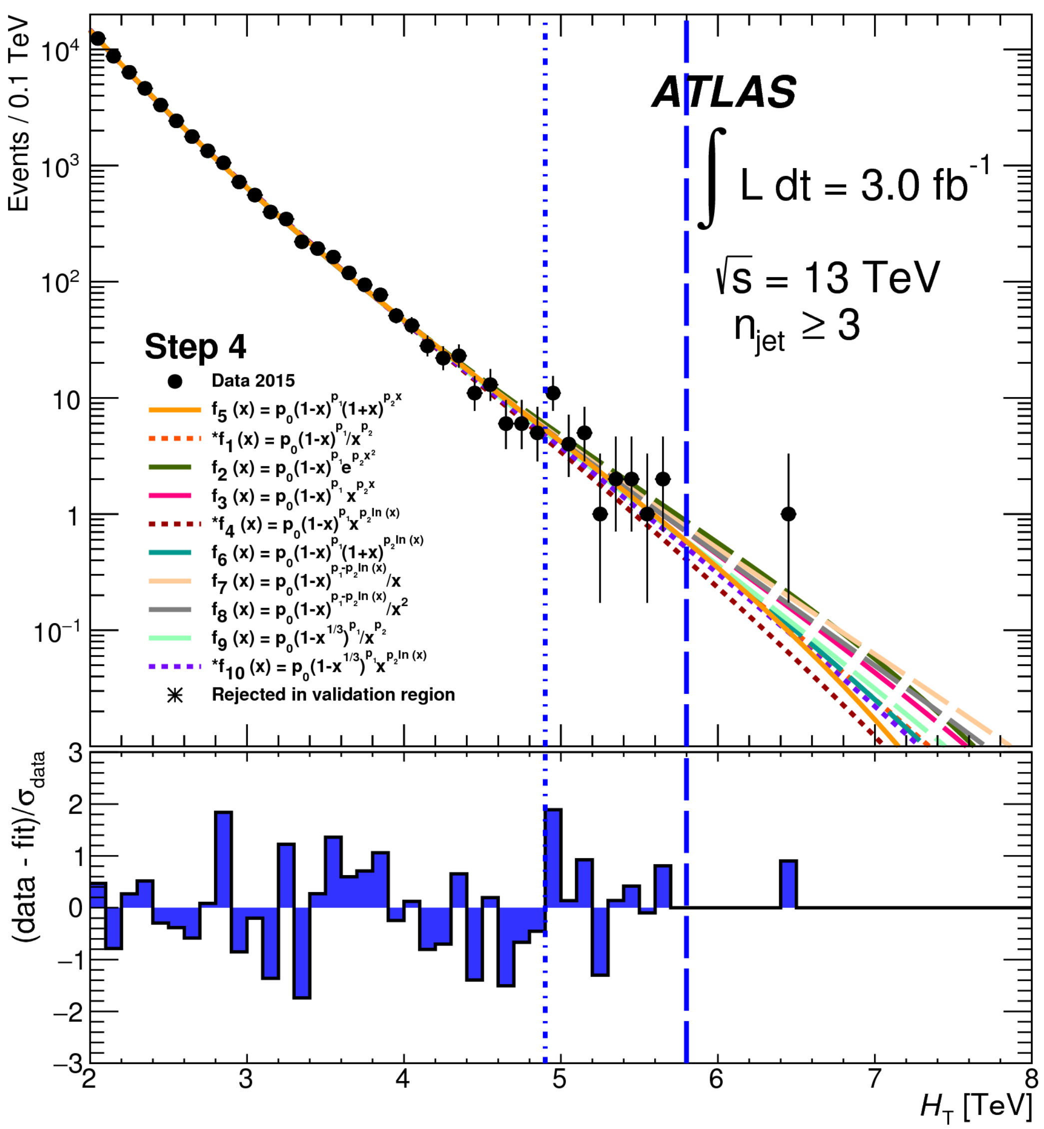
# Bump Hunts: Hard Mode

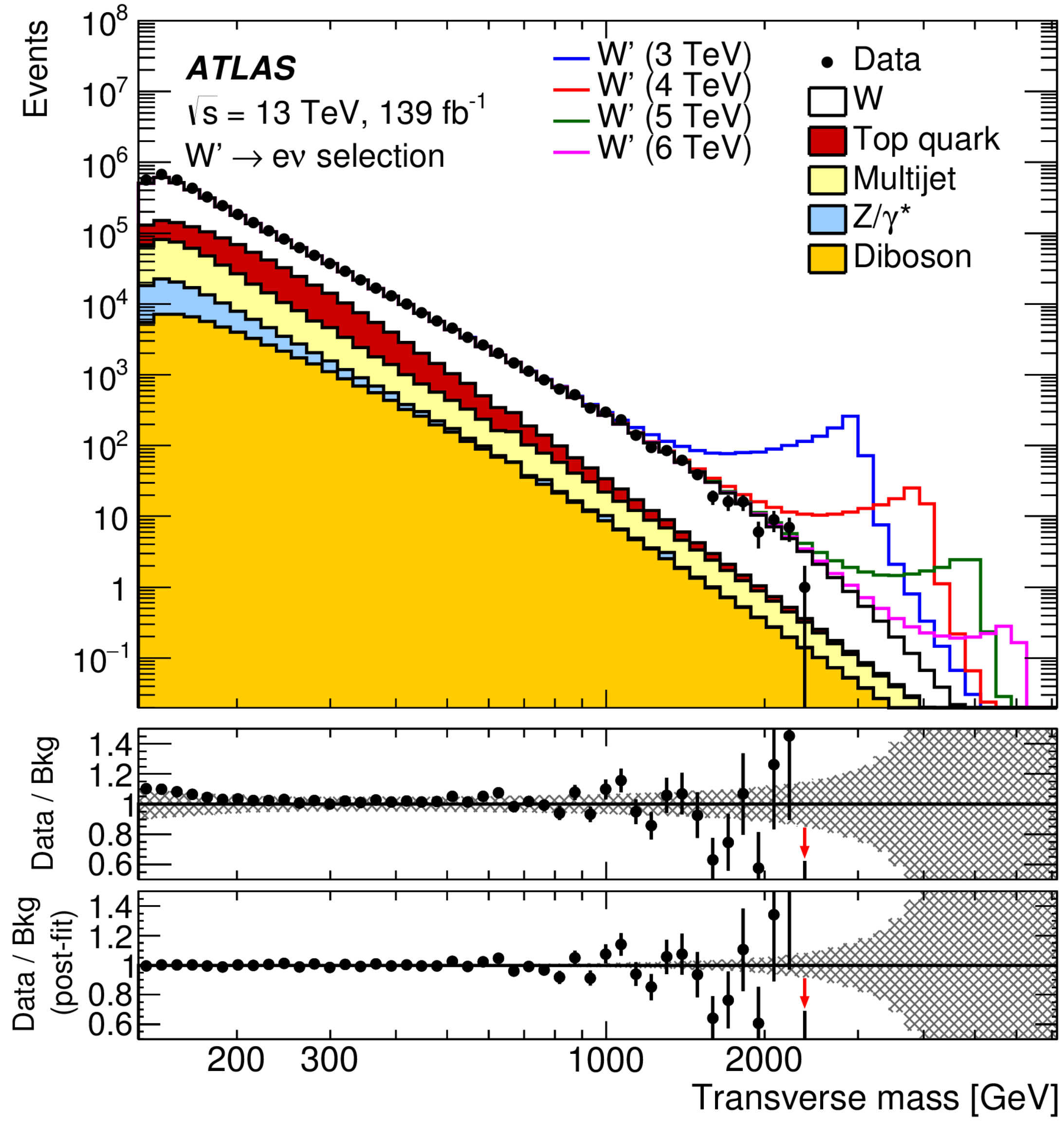
- If you have *many different backgrounds*, then you might just be giving up on the function entirely… more on that in a few min!



# Into the Tail

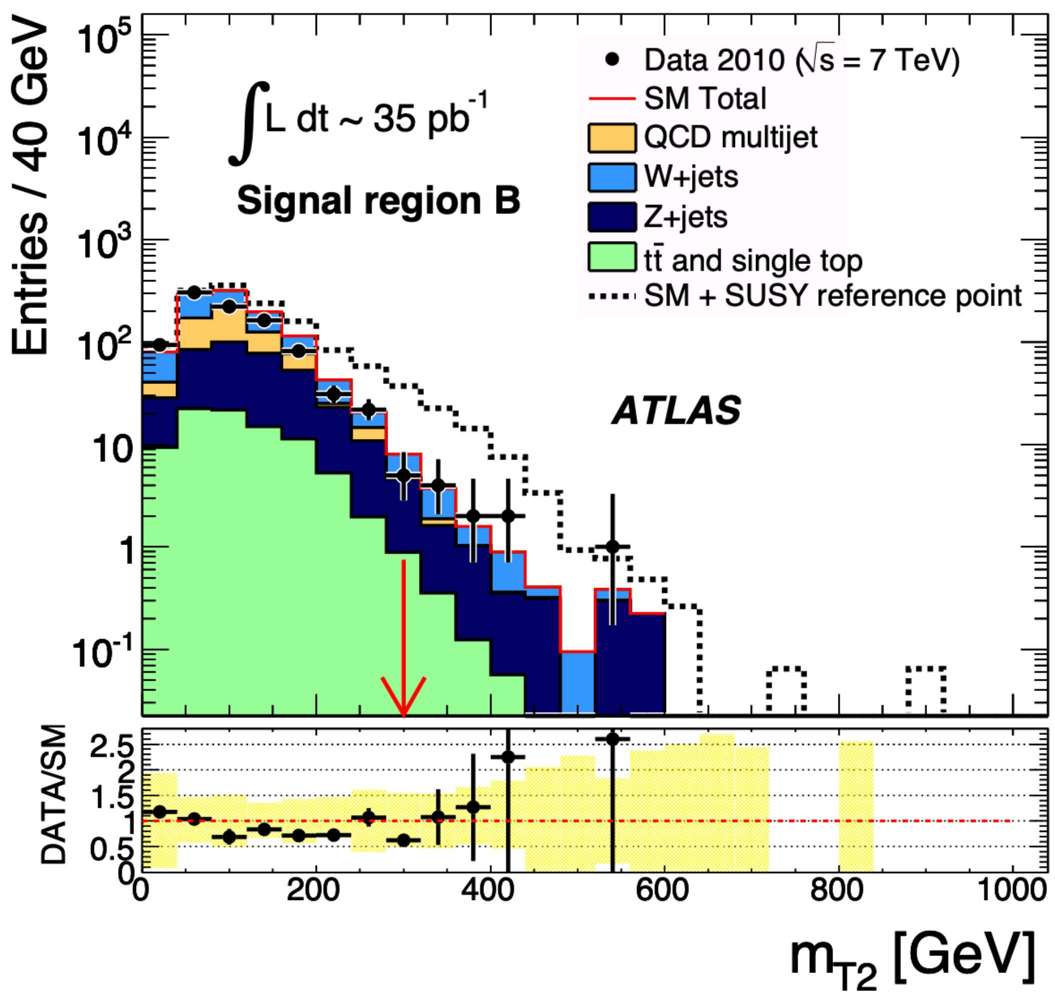
- At the highest end of the observable spectrum, a bump hunt becomes a *tail search* (also often known as *cut-and-count*)

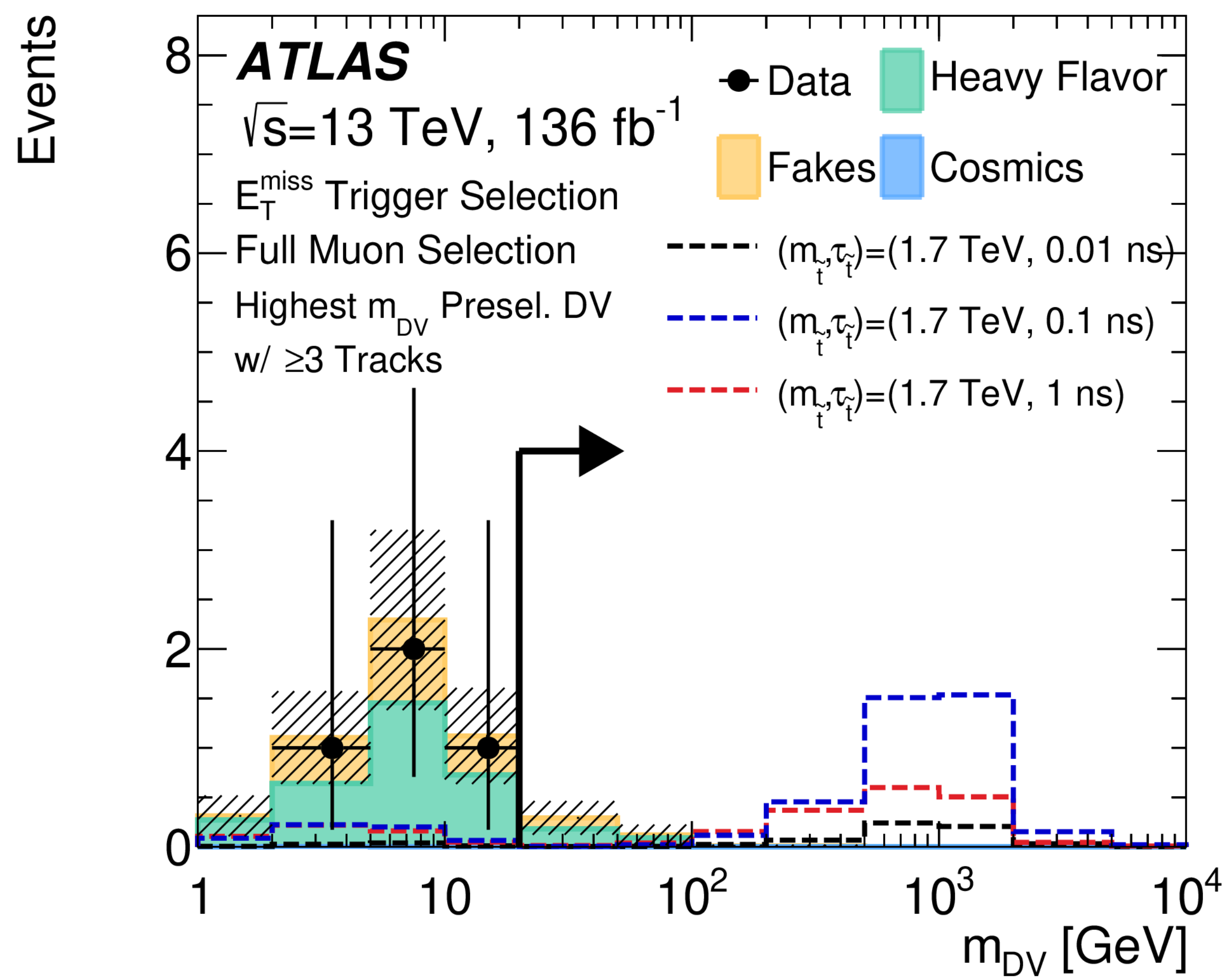


# Into the Tail

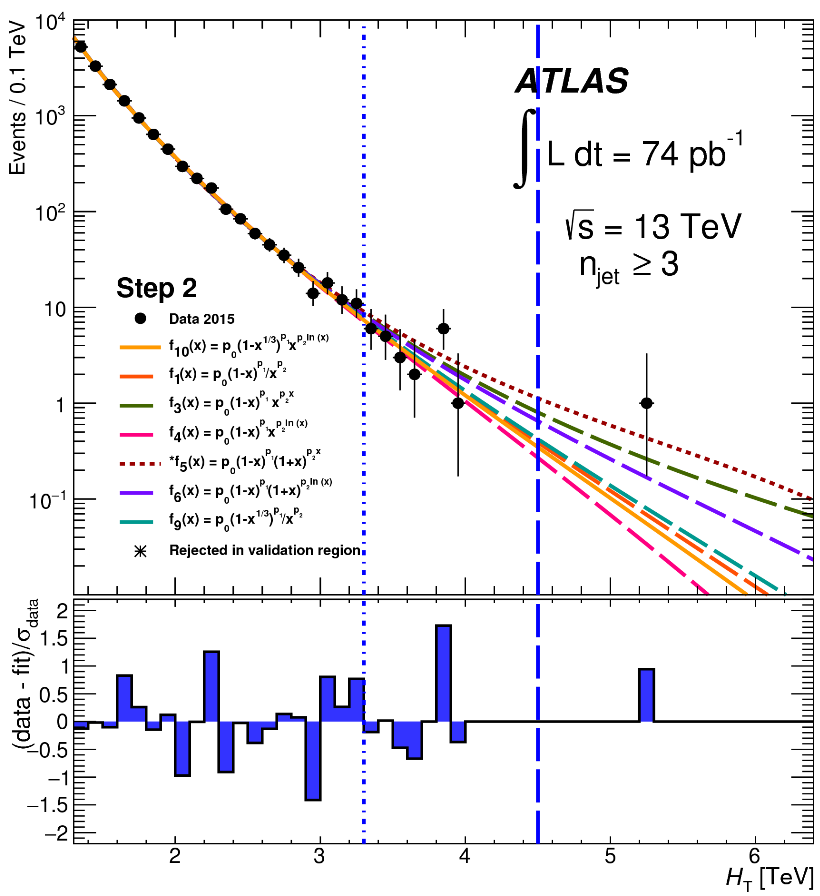
- The simplest version of a tail search is a “cut and count” search
  - We did a *lot* of these in the early days of the LHC
- These are *super* common, particularly for weird searches
  - If your signal is reasonably easy to isolate and doesn’t “bump”, this is the way to go!

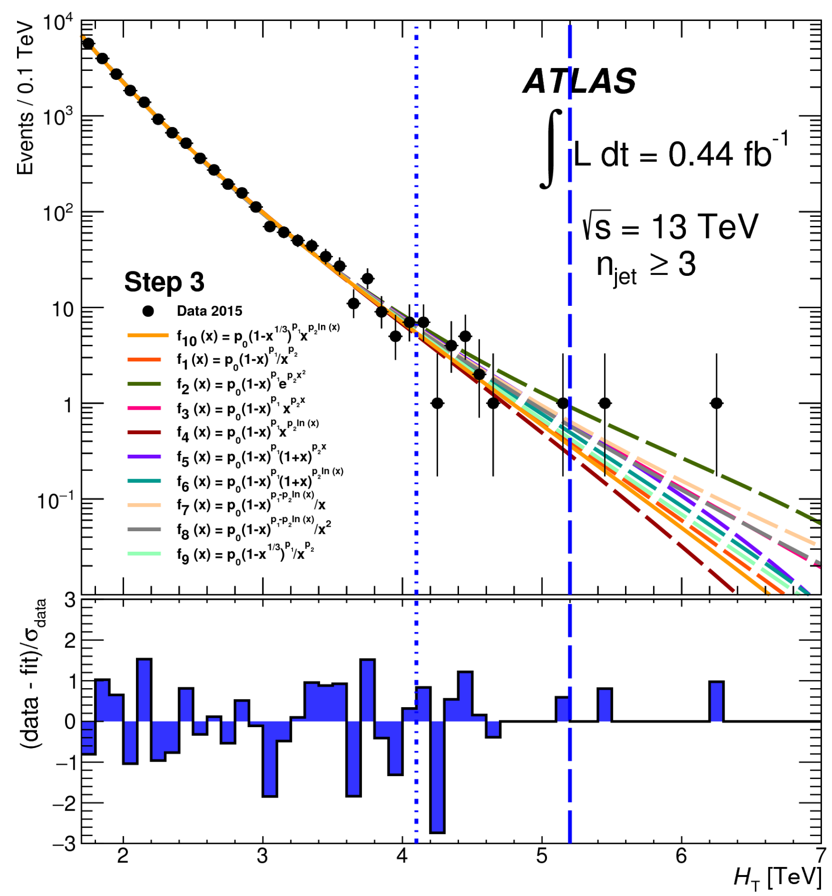


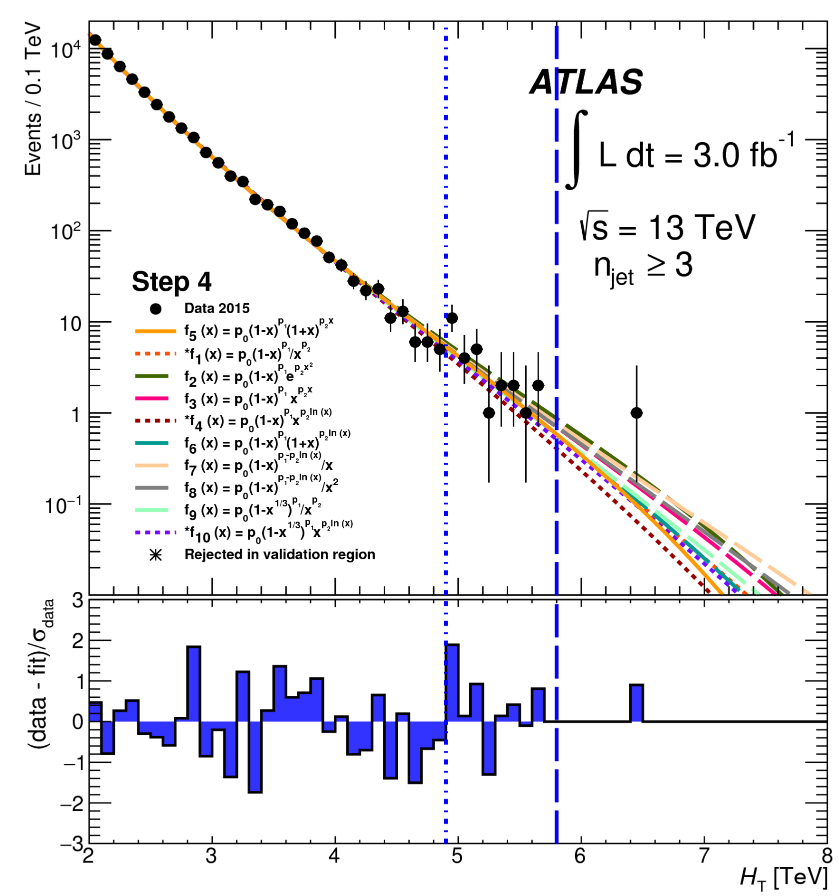



# Tails with Fits

- For this particular example, just add up the jet masses:
  - Fit some of the data, check for signal
  - Add more data, fit higher masses, check for more signal
  - Repeat until you're done

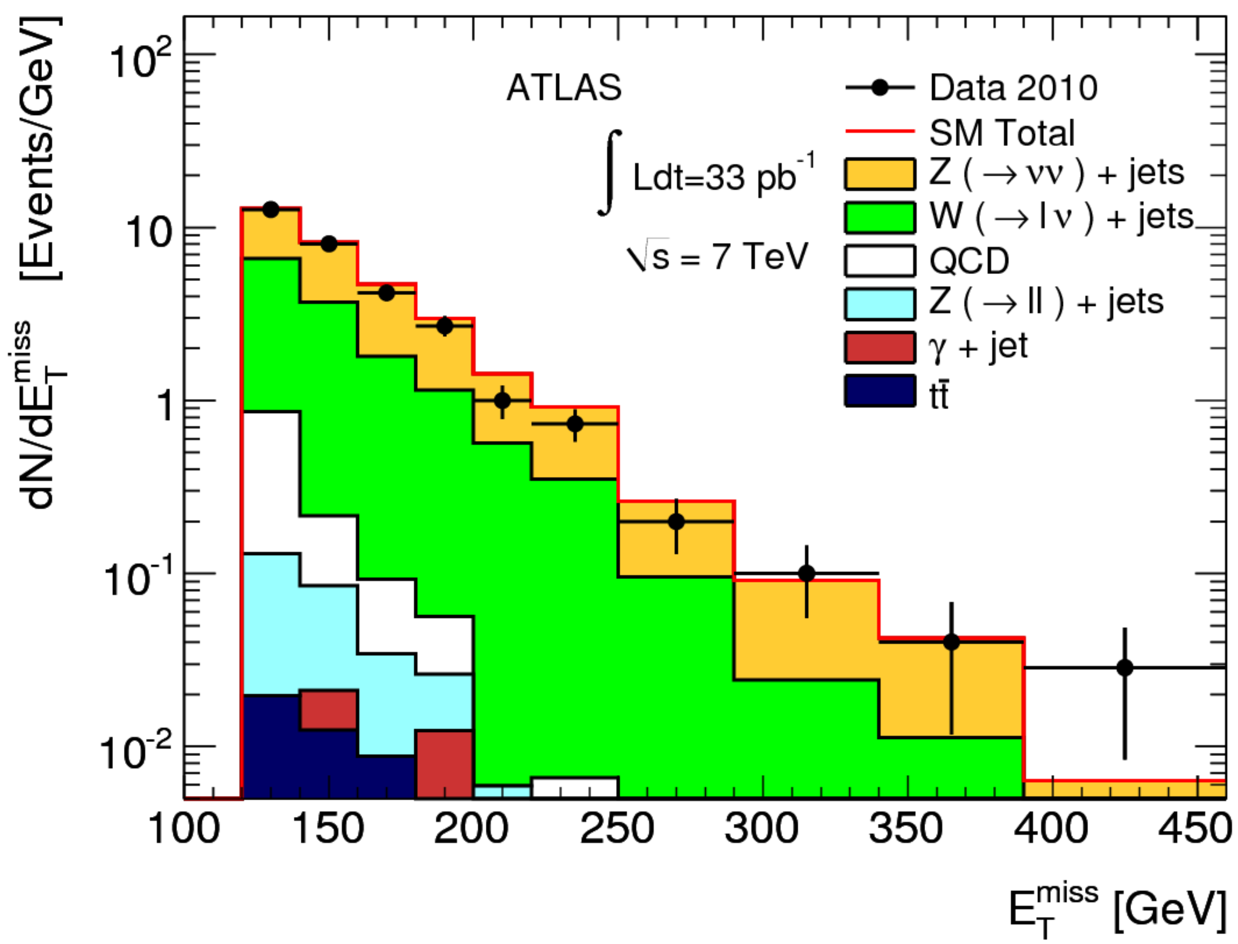


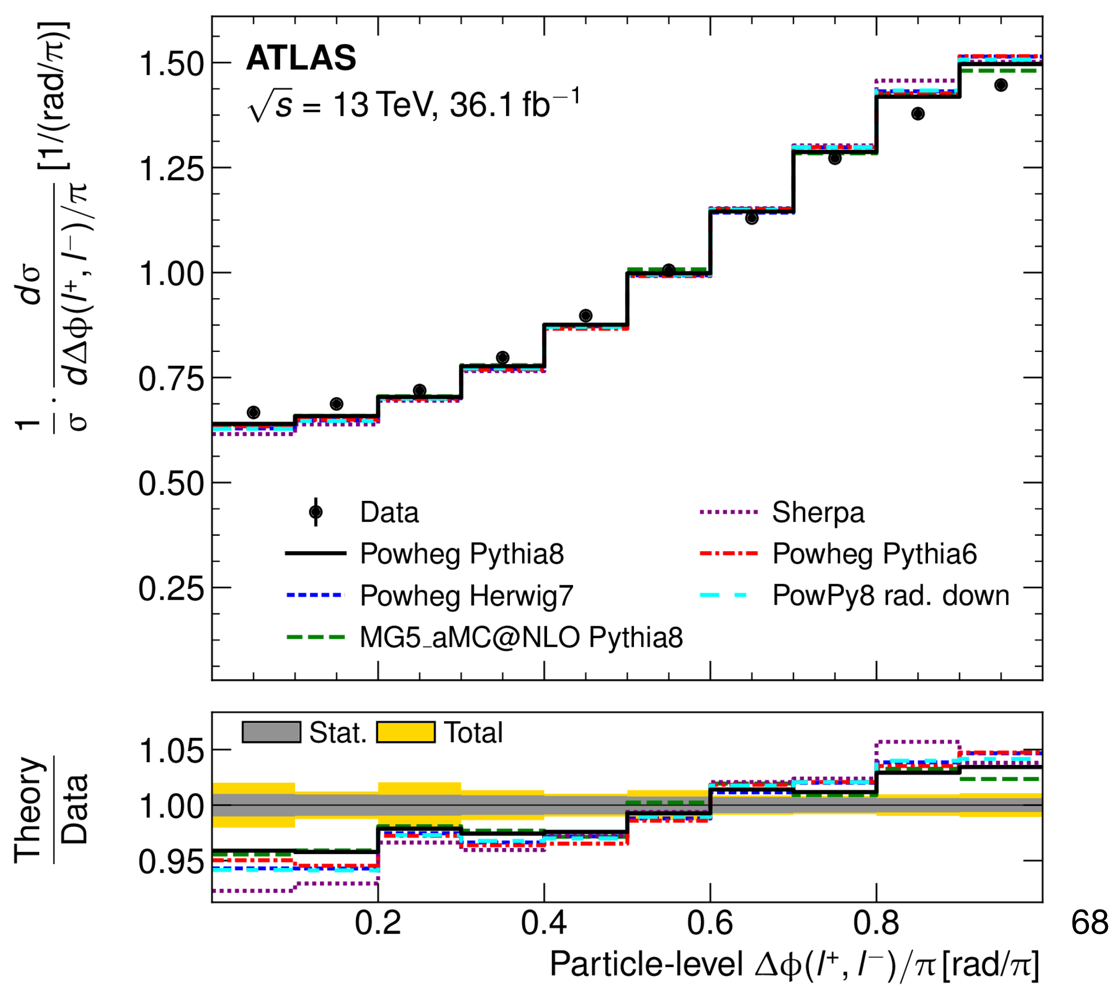
# Tails can be tricky

- In tails, it's common to have a mix of events that can't be fit with a single function
- Fitting many functions becomes complicated: what forms do you use? How do they relate?
- The solution is clear: use Monte Carlo simulation! (with validation)



# With apologies to Rikkert

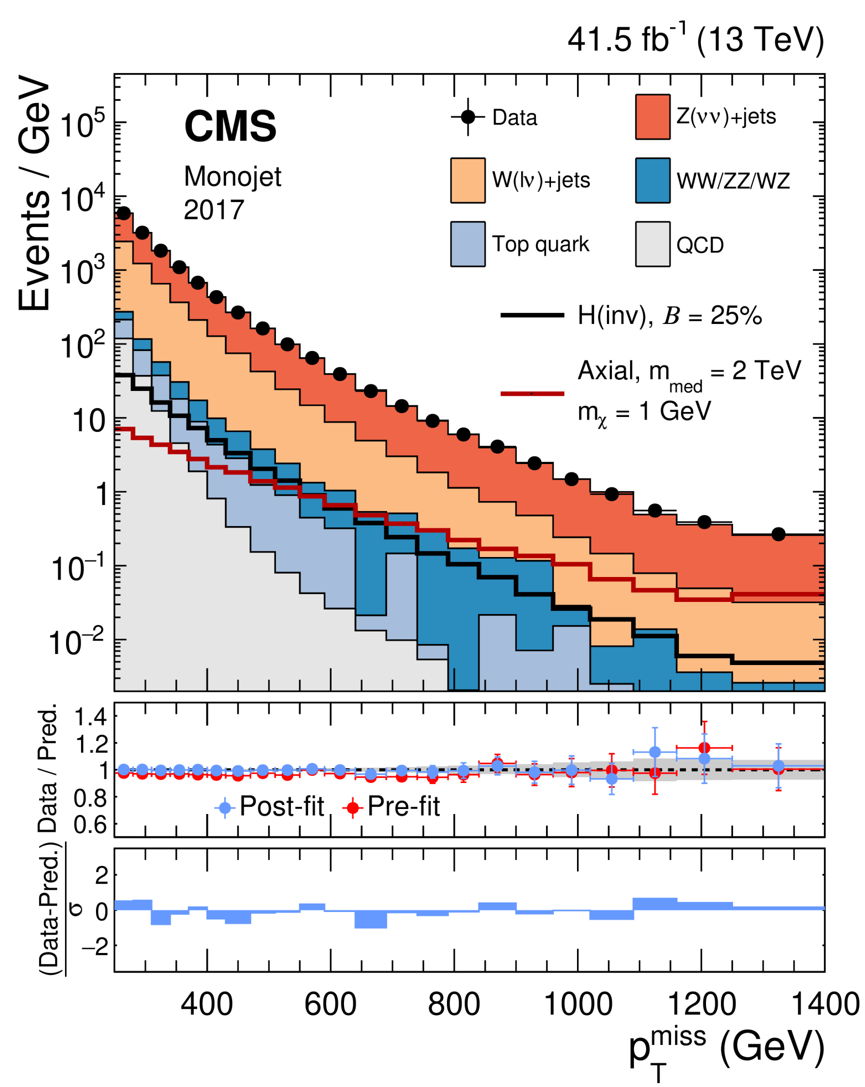
- The problem is that sometimes Monte Carlo simulation is terrible
  - To be fair, we often search regions of phase space that haven't yet been measured (in fact that's kinda the point)
- So what can we do to get a **good enough** background estimate to find new physics without waiting years for higher-order predictions, tuning, etc?

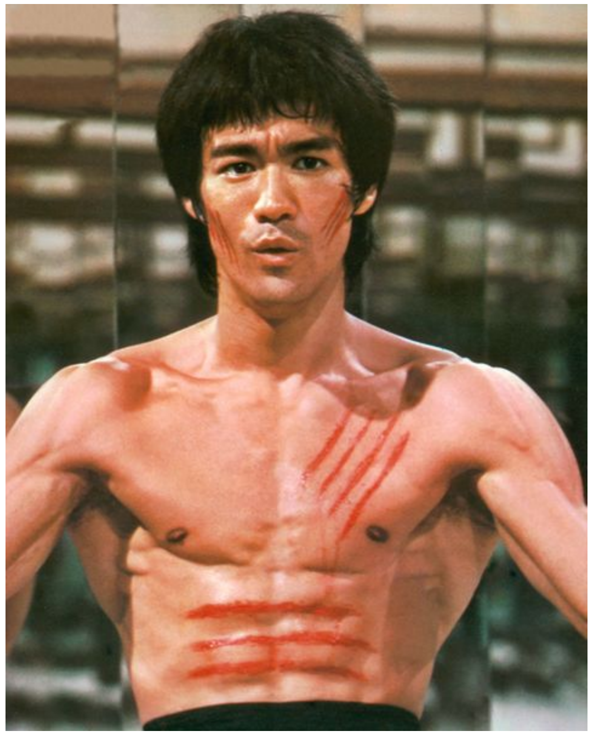



Side note: it's *rare* to find plots showing strong disagreement in LHC search papers. We tend to bury this more than we should.

# Enter the Fit

- Binned fits are the bread-and-butter of a large fraction of the LHC search program
- There are quite a few subtleties to them, so let's talk about them for a moment

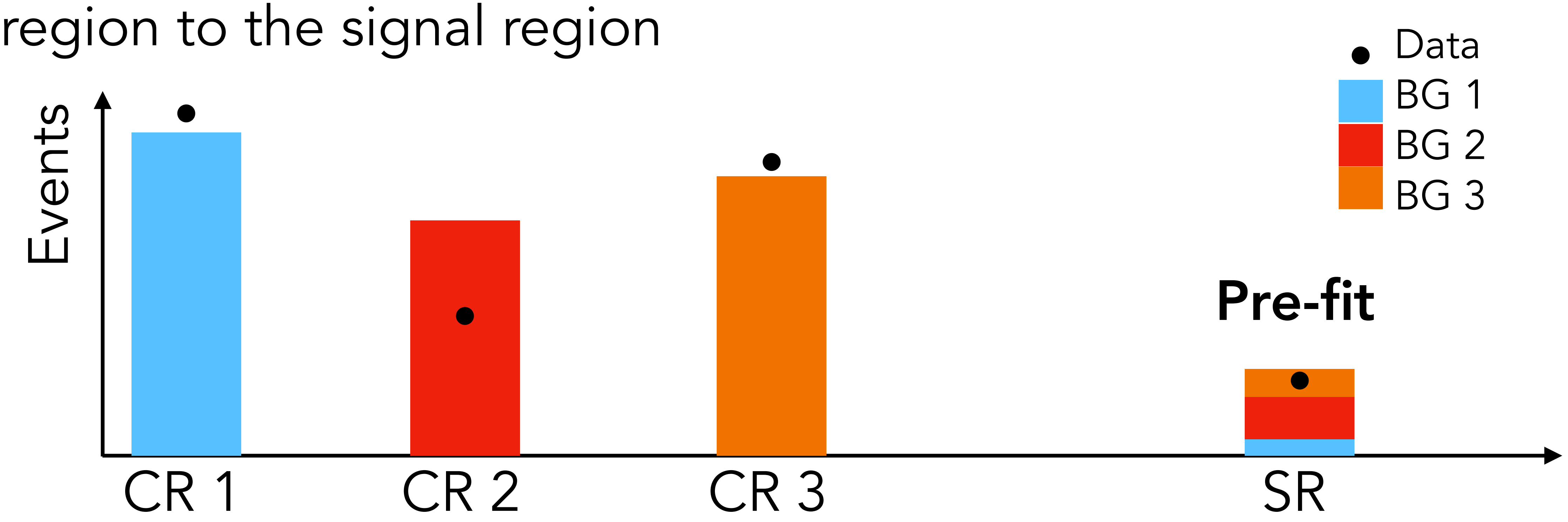


Slice your signal region up…

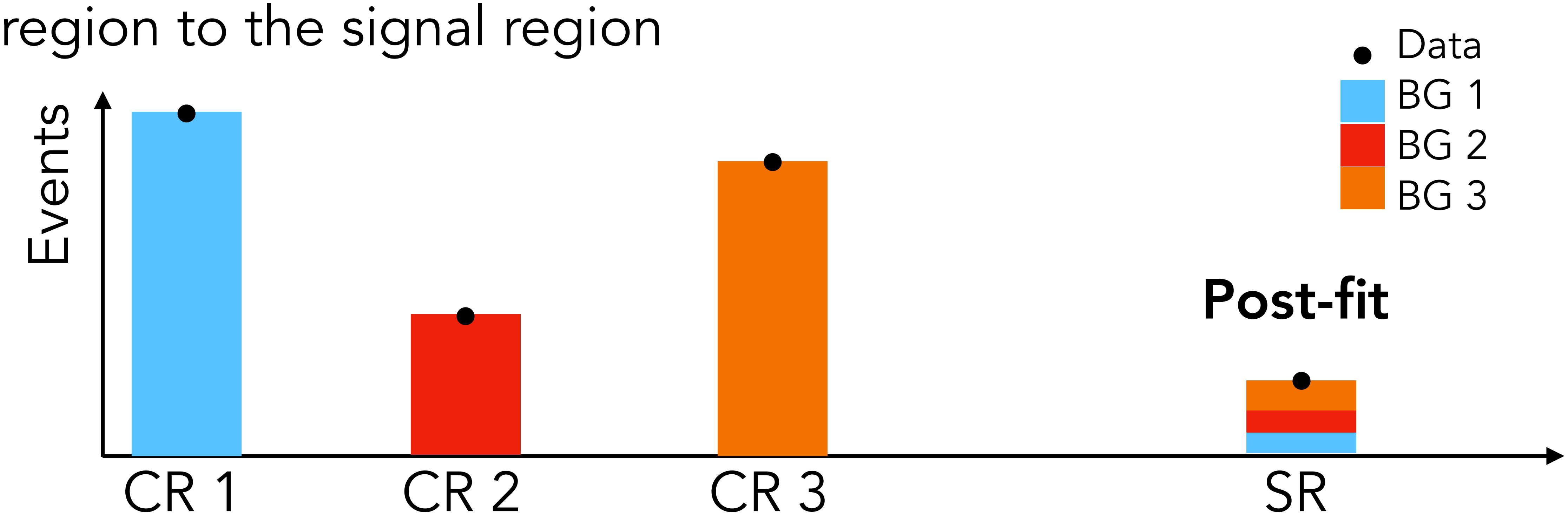
# Binning and Fitting: History

- In the dawn of time, some searches were done with "control regions" and "a signal region" (SR)
- Each "control region" (CR) was to **control** a background (often just the normalization)
  - These were made as pure in the background as possible
- Some Monte Carlo was used to extrapolate from the control region to the signal region



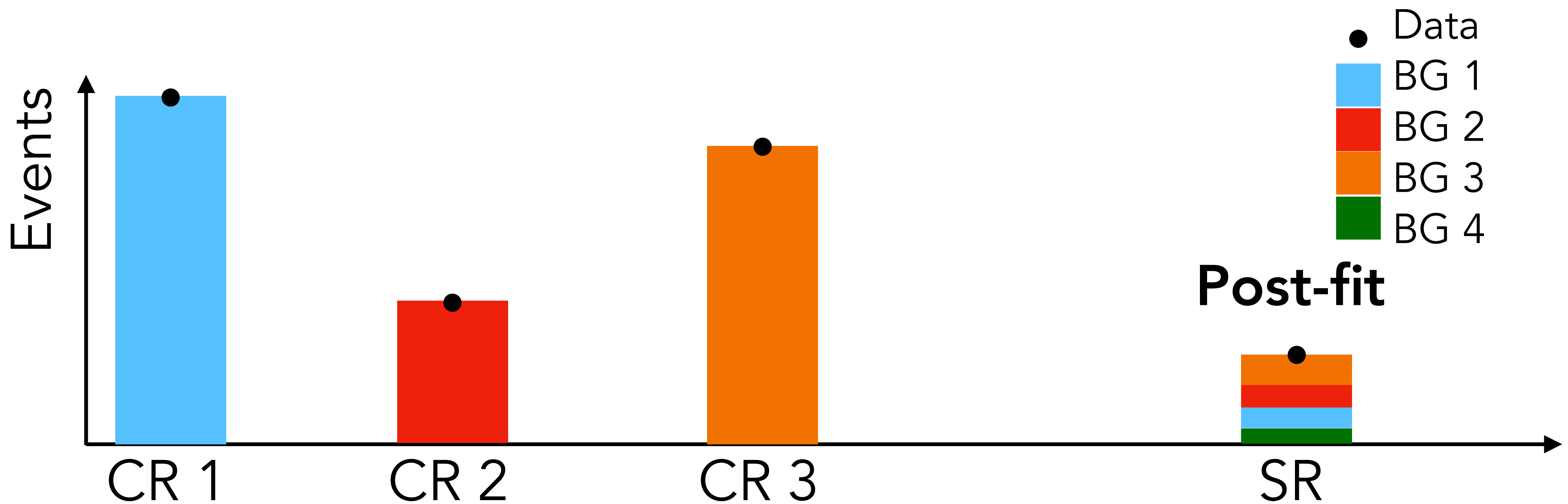
# Binning and Fitting: History

- In the dawn of time, some searches were done with "control regions" and "a signal region" (SR)
- Each "control region" (CR) was to **control** a background (often just the normalization)
  - These were made as pure in the background as possible
- Some Monte Carlo was used to extrapolate from the control region to the signal region



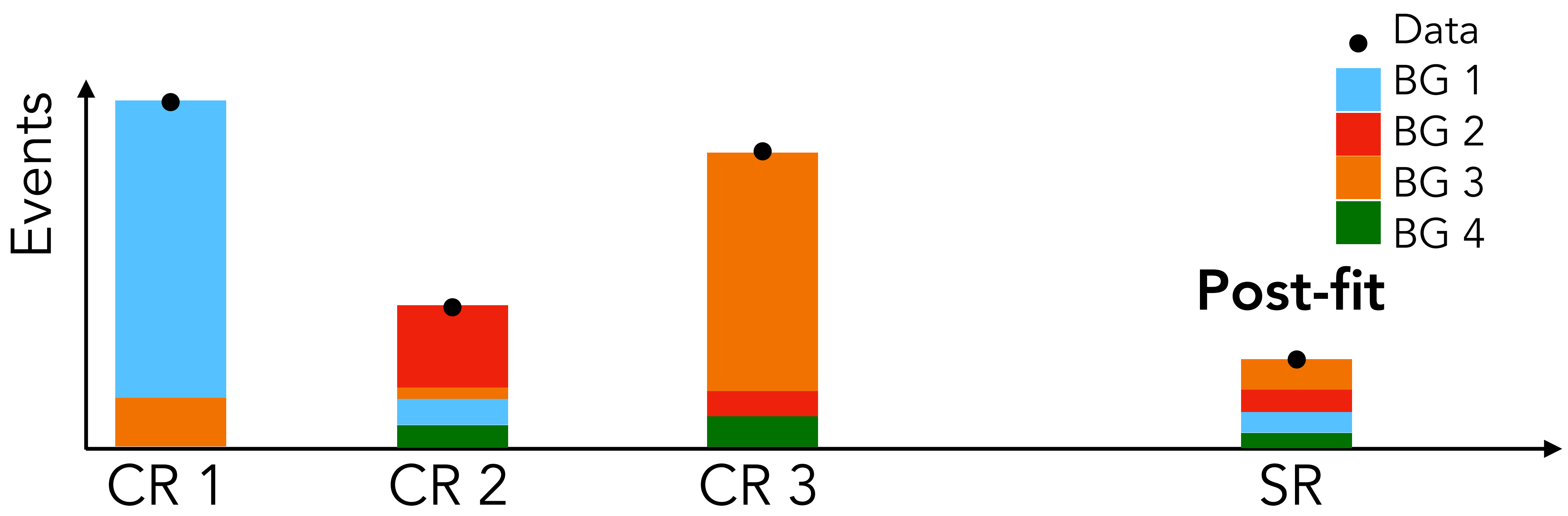
# Reality bites

- Of course, not all backgrounds can be “controlled” in that way, so we often have extras that have to be estimated purely with Monte Carlo simulation



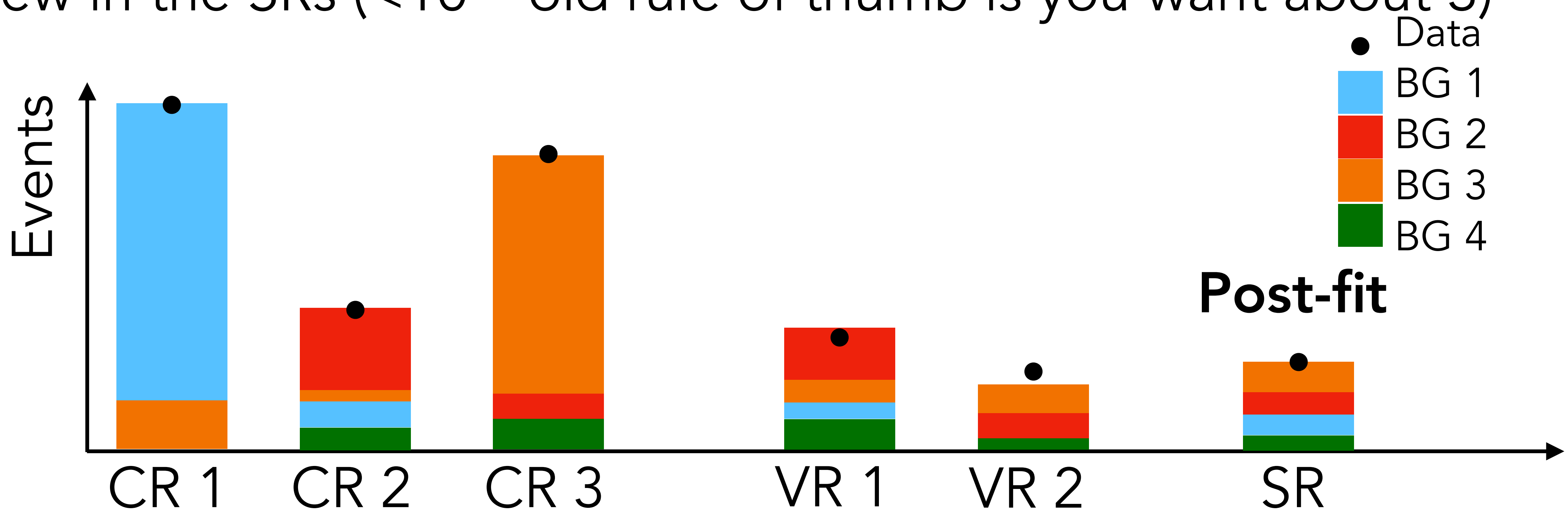
# Reality bites

- Of course, not all backgrounds can be "controlled" in that way, so we often have extras that have to be estimated purely with Monte Carlo simulation
- And, of course, not all CRs are actually pure in their background
  - We can no longer pretend that each CR "simply" normalizes a single background – we have to fit them all *together!*



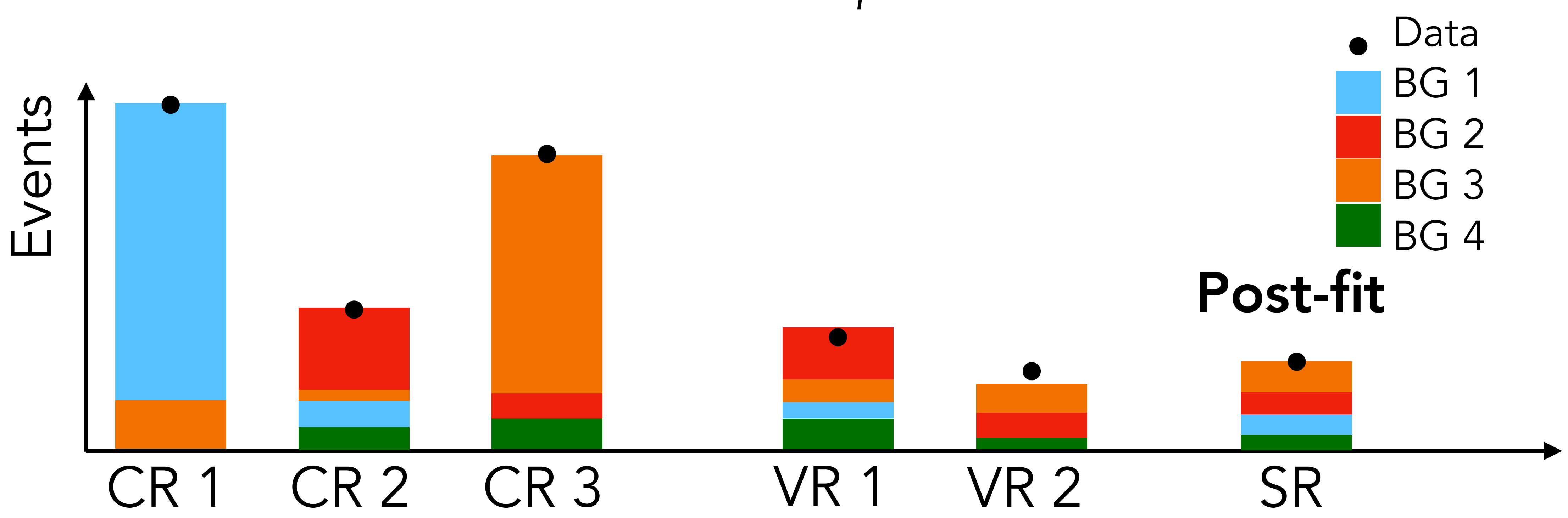
# Check my math!

- This is all complicated, so kinematically "between" the CRs and the SR, let's add some "validation regions" (VRs) to check that our background estimates did the right thing
- We fit in the CRs, check agreement in the VRs, and then look for new physics in the SRs
- Normally: many events in the CRs (>100s), fewer in the VRs, and few in the SRs (<10 – old rule of thumb is you want about 3)



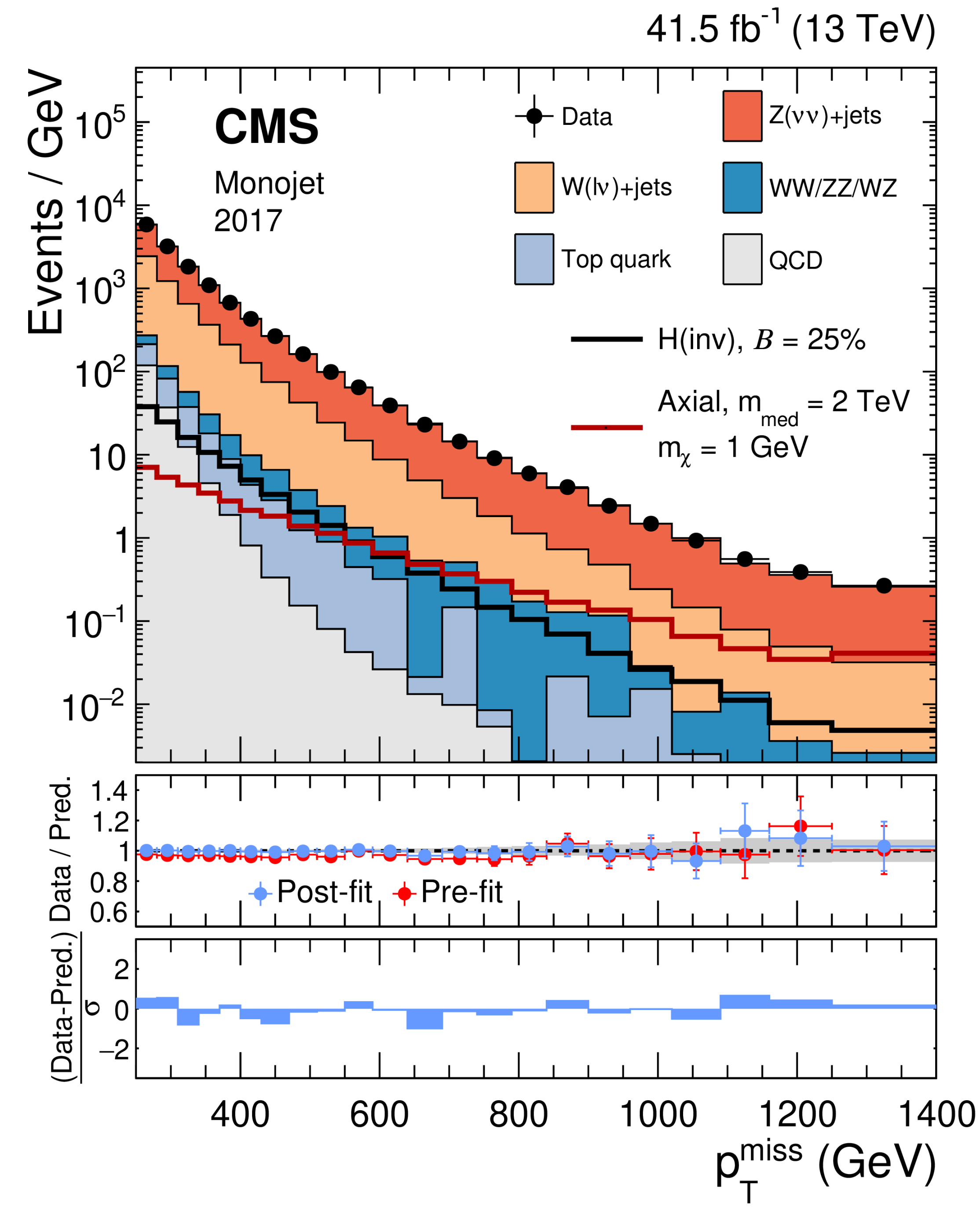
# Extrapolation

- There's a trade off in building all the CRs and SRs:
  - Make them larger (more events) – better "control" of the background, higher precision on the normalization factor, but very large uncertainties on the extrapolation from the CR to the SR
  - Move them closer to the signal region (fewer events) – larger normalization uncertainty, smaller extrapolation uncertainty
  - Which is better is *not obvious and depends on the search!*



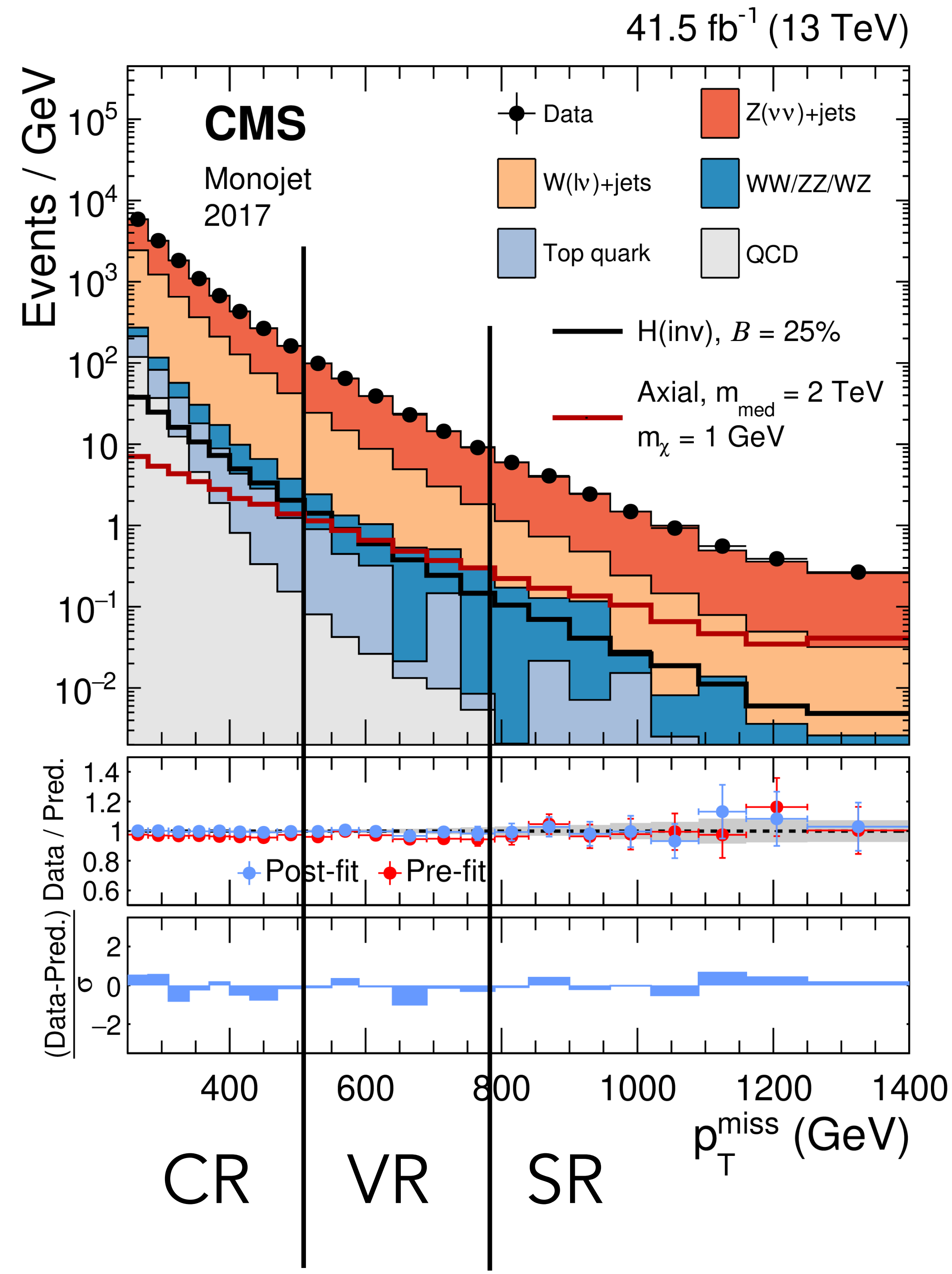
# What are CRs and SRs

- Let's take some long distribution in an observable like this one



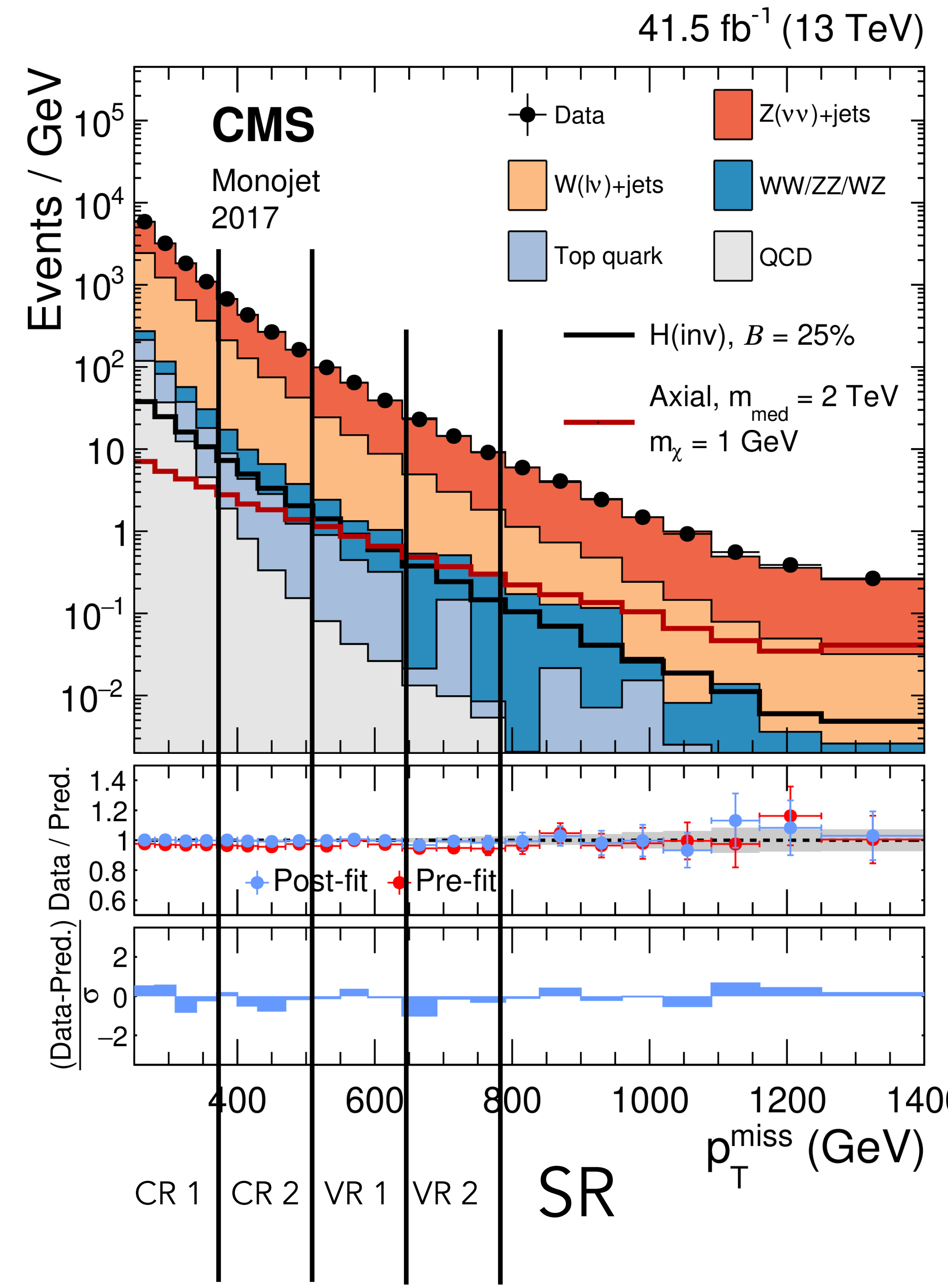
# What are CRs and SRs

- Let's take some long distribution in an observable like this one
- Simplest thing is to just chop up the distribution in an easy way
    - I can decide what a CR is based on some previous search (where I know there's no signal)



# What are CRs and SRs

- Let's take some long distribution in an observable like this one




- Simplest thing is to just chop up the distribution in an easy way
- I'm free to add more CRs and VRs
- This all looks great if my signal is mostly up in my SR

# TIME FOR AN ASIDE

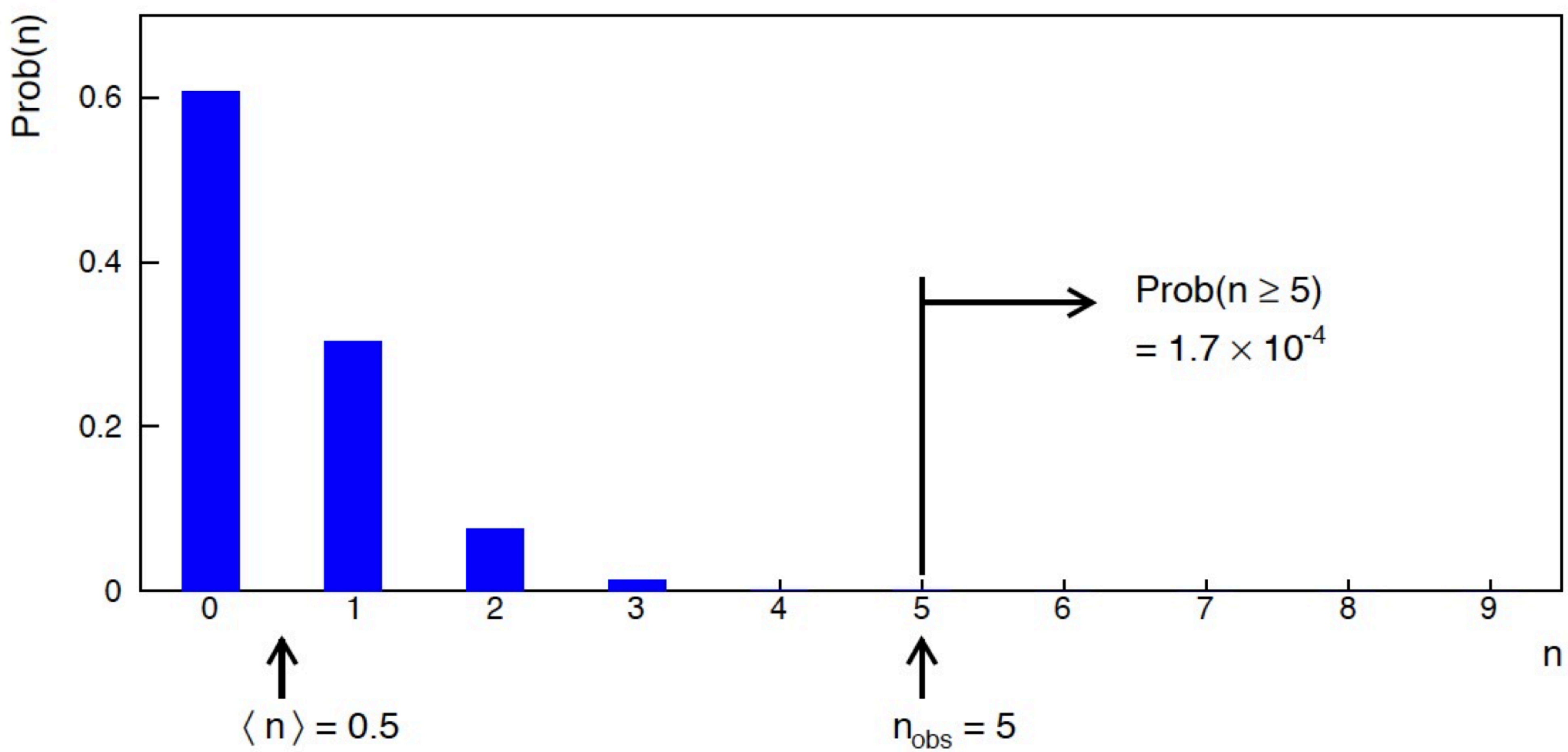
# Statistics are Magic

- Check out the academic training lectures by Glen Cowan for a really nice introduction to statistics for HEP physicists
- Here I'm just going to mention a few key statistical features
- And I'm going to jump around a little bit here (sorry)

Danger (and apologies): My statistics understanding is not rigorous and academic; rather it is gained from lots of experience and built on the resulting intuition. I expect the following explanations will work great for some of you and terribly for others; for those of you for whom it doesn't work, see Cowan above and his books…

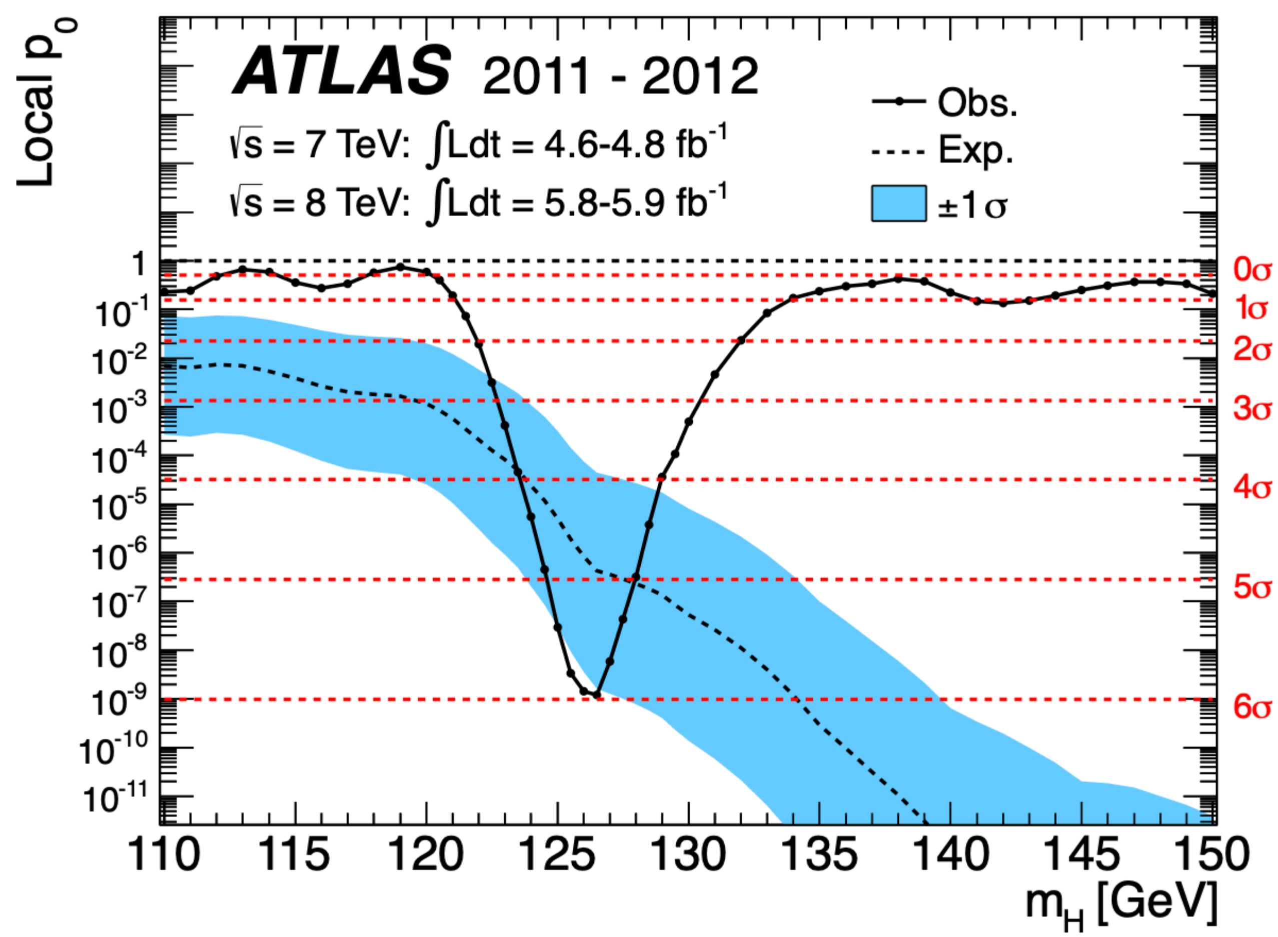
# Looking at distributions

- For simple counting experiments, the statistics are pretty easy
- If we expect on average 0.5 events and observe 5, that's unlikely
- A *small p-value* means something is unlikely
  - Very often we use a Gaussian approximation to turn that p-value into a "number of standard deviations"



# Looking at distributions

- For complicated tests, we play a similar statistical game
- Here $p_0$ is the probability of the background-only hypothesis
- A *small p-value* means something is unlikely
  - Very often we use a Gaussian approximation to turn that p-value into a "number of standard deviations"



# Testing for Signal

- Our searches come down to this question:

**Is my observation consistent with *my background estimation* or with *my background estimation plus a new physics signal?***

- First keen observation:

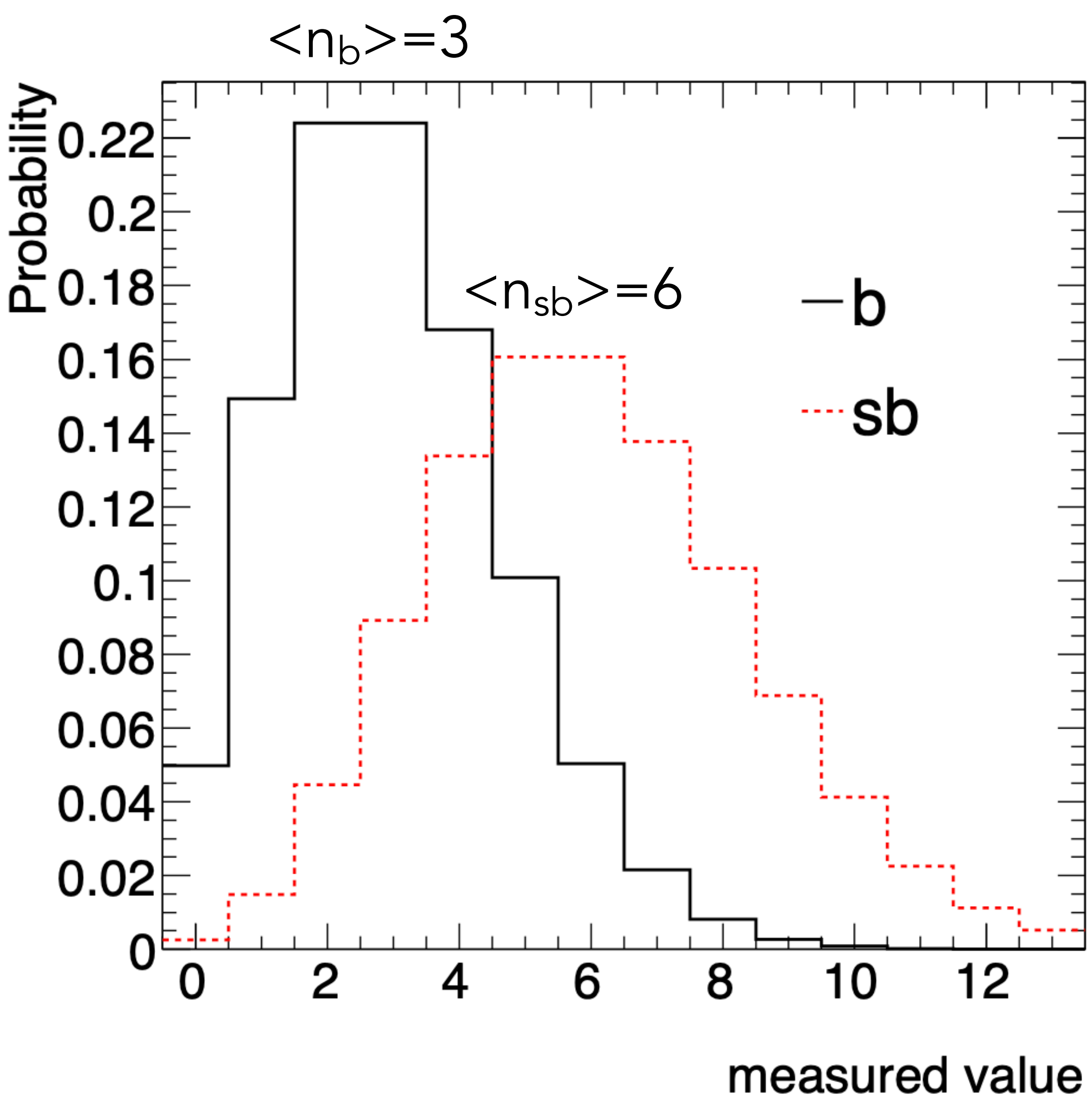
We **NEVER** test a hypothesis in a search

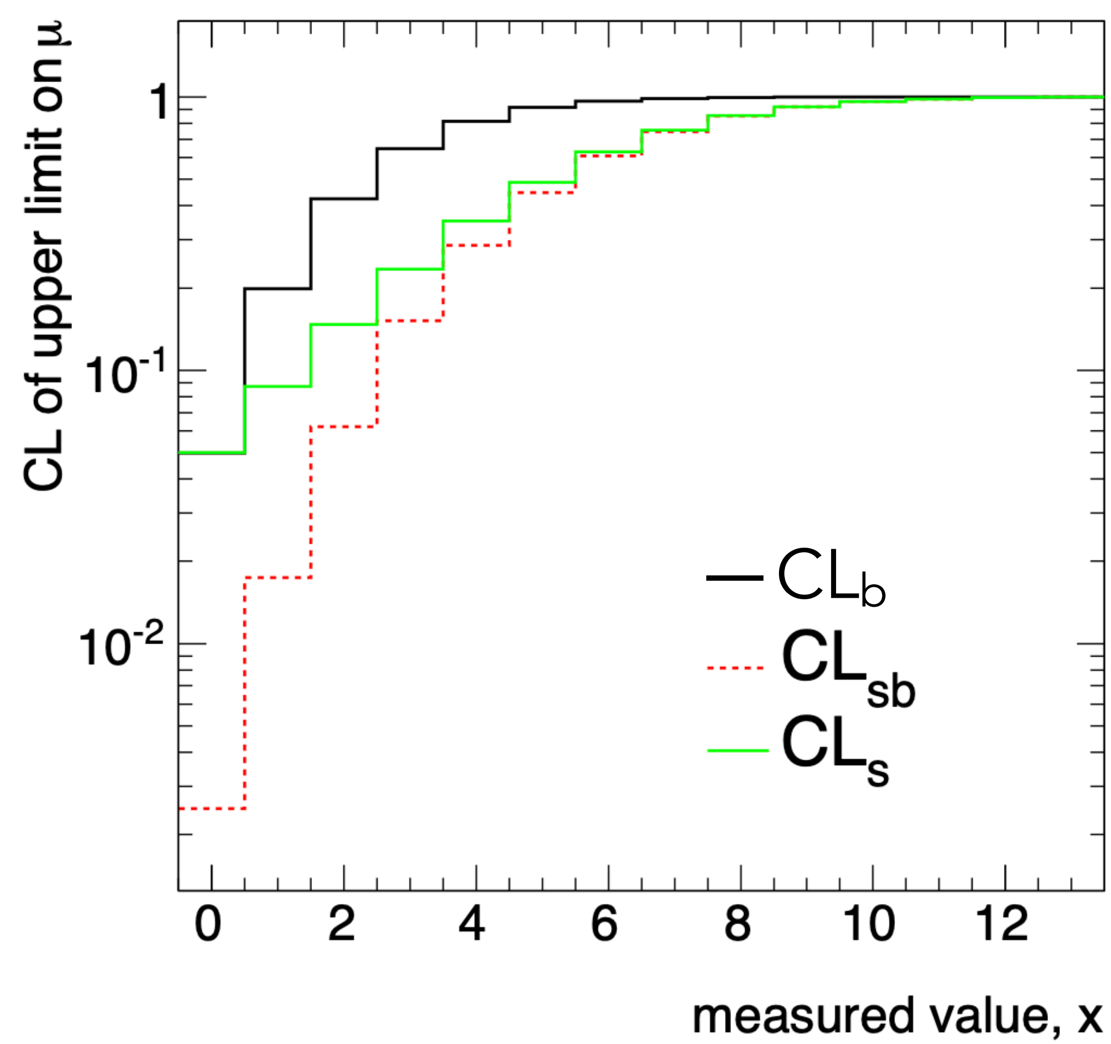
We **ONLY** *compare* two hypotheses

# Signal and Background

- In 2000, CLs was created with that simple observation: if our background estimate sucks, we shouldn't claim new physics has been discovered.

$$CL_s = \frac{CL_{sb}}{CL_b}$$




Figures from Bill Murray

# Signal and Background

- In 2000, CLs was created with that simple observation: if our background estimate sucks, we shouldn't claim new physics has been discovered.

$$CL_s = \frac{CL_{sb}}{CL_b}$$

- All LHC searches use CLs (agreed upon standard) to set 95% confidence level limits (ignoring the Bayesians here)
- In very rough terms, but so you've seen the lingo:
    - We build a likelihood function including all the information in the search
    - We make a profile likelihood ratio from that function
    - We make a test statistic using that profile likelihood ratio
    - We integrate that test statistic to get a p-value

# The important thing about Likelihoods

- For a simple 1-bin counting experiment, testing the mean of the distribution, a likelihood function is pretty simple:

Sample Size

Mean

$$\mathcal{L}(\mu, \sigma \mid x) = \left(2\pi\sigma^2\right)^{-n/2} \exp\left(-\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

Standard deviation

# The important thing about Likelihoods

- A likelihood for a multi-bin experiment (multi-bin fit!) is the *product of the likelihoods in each bin.*
    - These can get very complicated

$$\mathcal{L}=\underbrace{\overset{\substack{\text{Table}\\ \text{XXIa}}}{\prod_{i,b}} f\left(N_{ib}\,\middle|\,\mu\cdot S_{ib}\cdot\overset{\substack{\text{Syst in}\\ \text{Sec. V}}}{\prod_{r}}\nu_{br}(\theta_r)+\overset{\substack{\text{Table}\\ \text{I}}}{\sum_{k}}\beta_k\cdot B_{kib}\cdot\overset{\substack{\text{Syst in}\\ \text{Sec. VII C}}}{\prod_{s}}\nu_{bs}(\theta_s)\right)}_{\text{Poisson for SR with signal strength }\mu\text{; predictions }S,\ B}\cdot\underbrace{\overset{\substack{\text{Table}\\ \text{XXIb}\,\bullet}}{\prod_{l}} f\left(N_l\,\middle|\,\overset{\substack{\text{Table}\\ \text{I}}}{\sum_{k}}\beta_k\cdot B_{kl}\right)}_{\text{Poisson for profiled CRs}}\cdot\underbrace{\overset{\substack{\text{Syst in}\\ \{r,\ s\}}}{\prod_{t}} g(\vartheta_t|\theta_t)}_{\text{Gauss. for syst}}\cdot\underbrace{\overset{\substack{\text{Table}\\ \text{I}}}{\prod_{k}} f(\xi_k\,|\,\zeta_k\cdot\theta_k)}_{\text{Poiss. for MC stats}}\qquad(11)$$

$$L(\mu,\vec{\mu}_{\text{bkg}},\vec{\theta})=\prod_{\text{SR and CR bins}}\frac{E_i^{o_i}}{o_i!}e^{E_i}\prod N(\vec{\theta})$$

with

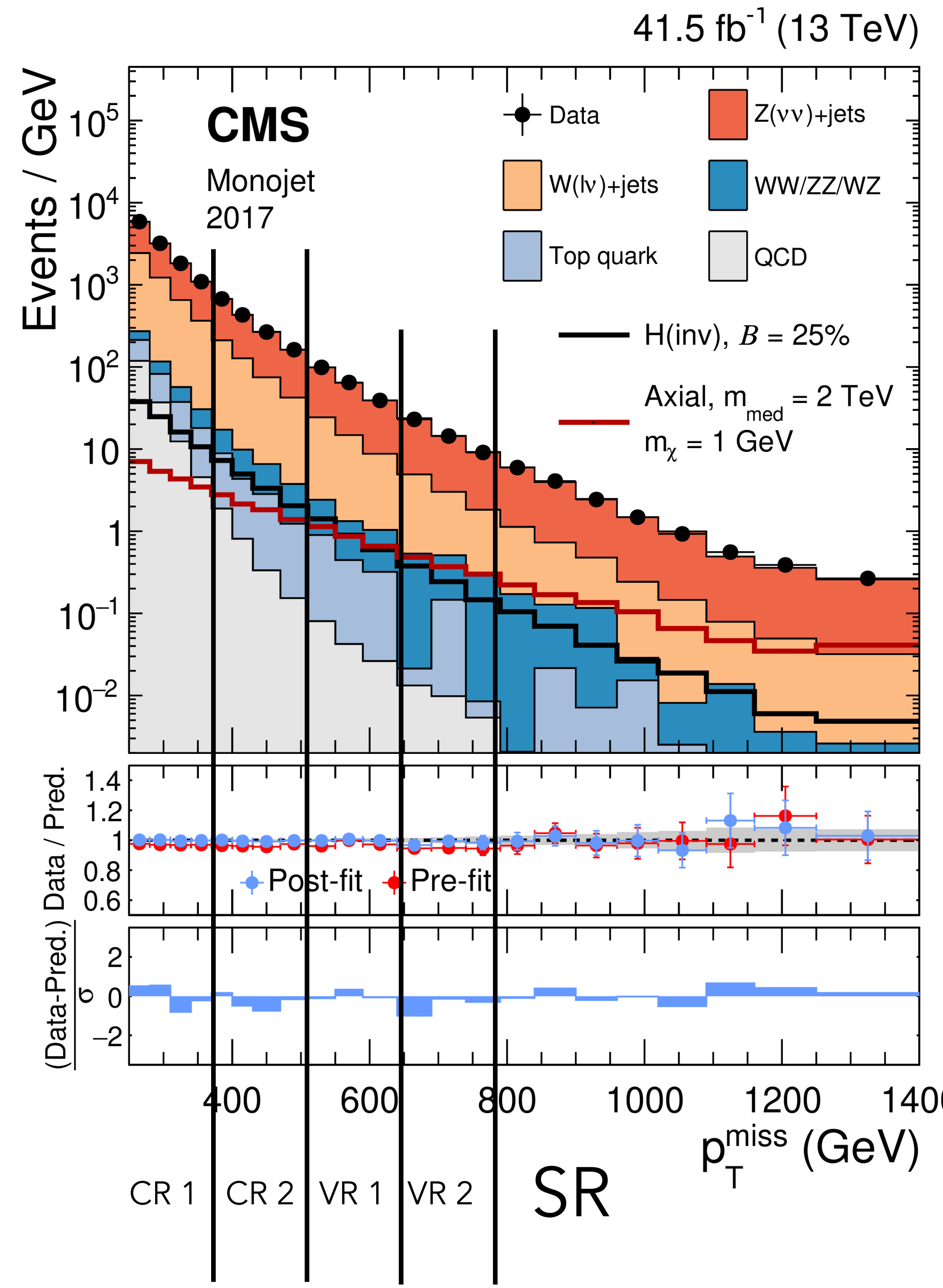
$$E_i=E_i(\mu,\vec{\mu}_{\text{bkg}},\vec{\theta})=\sum_{\text{process } j} b_{ij}(\mu^j_{\text{bkg}},\vec{\theta})+s_i(\mu,\vec{\theta})$$

- The likelihood ***does not know*** what you have called a control region and a signal region!

# AND WE'RE BACK

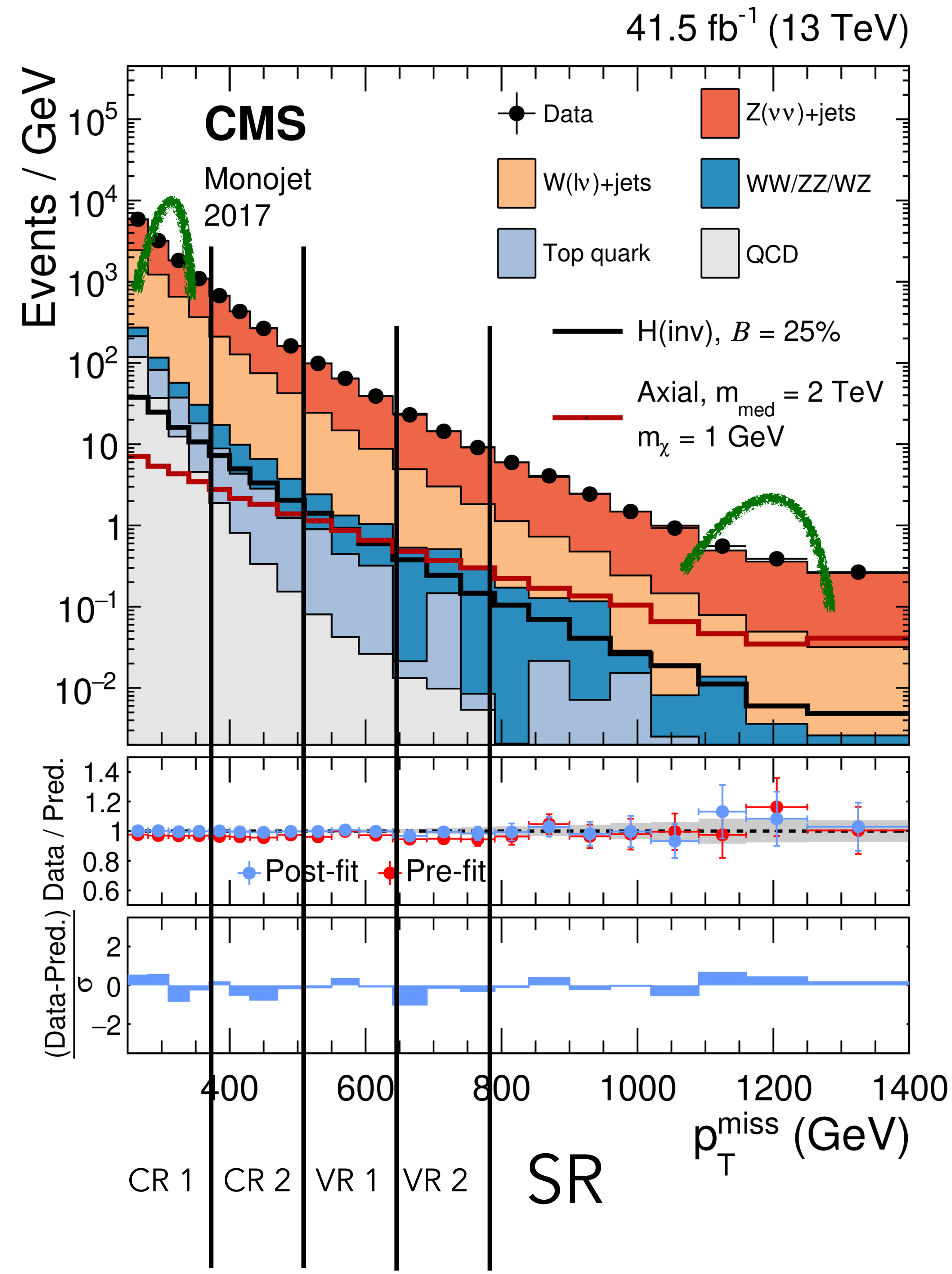
# What are CRs and SRs

- Building a likelihood to do the fit really means I have two choices for regions: **include them** in the fit, or **don't include them** in the fit
    - If I include a region in my fit, the fit will do its best to make my expectation match the data
- I can then dream up a few different fit configurations:
    - Fit the CRs only, look in the VRs and SRs (often done as a "discovery test")
    - Fit the CRs and SRs (often done as an "exclusion test")
    - Normally we don't include the VRs in the fit (~never)



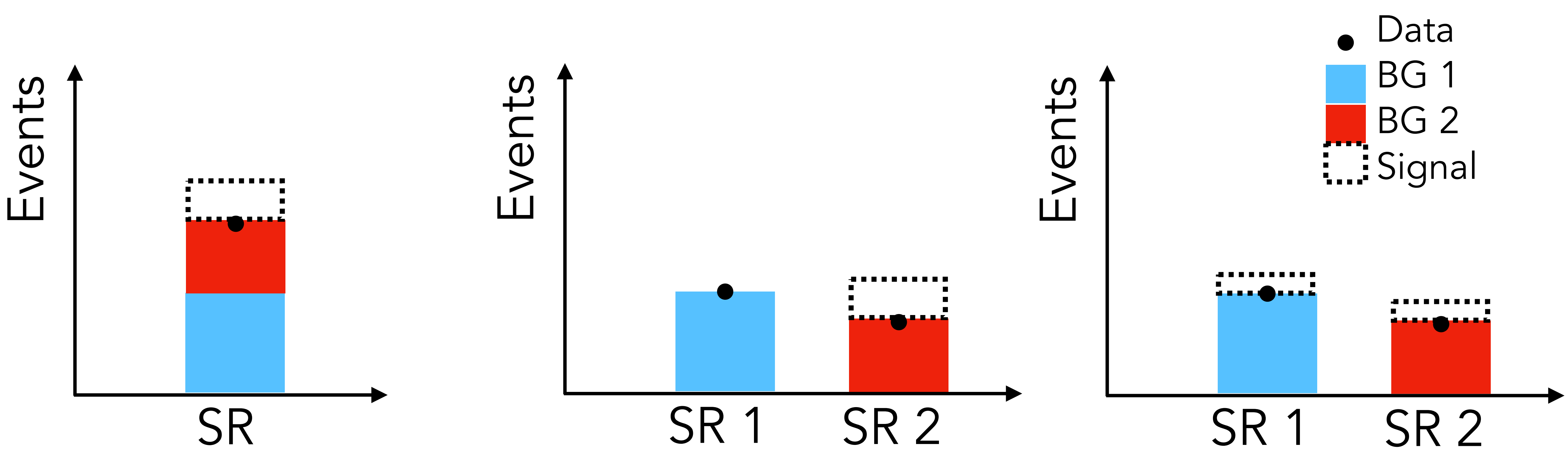
# The naive implication

- In a multi-bin fit, I *do not care* where the signal is and where the background is
- The fit will magically test the right bins for signal and use the others to constrain the background



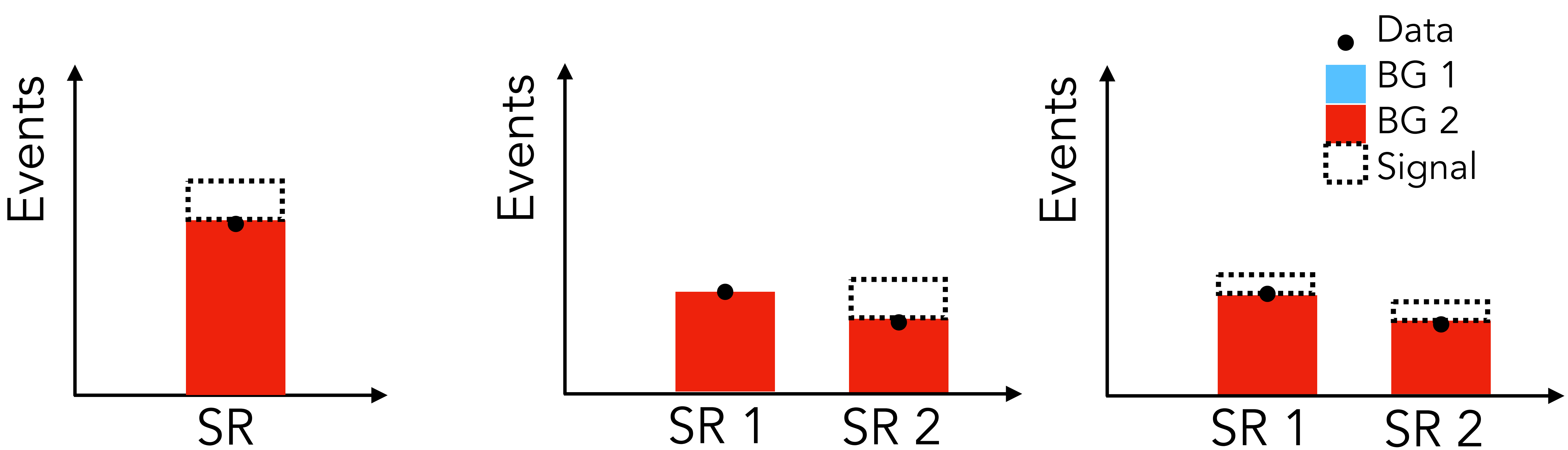
# A simple two-bin example

- Let's compare some tests!
- If I have a case where my signal regions are really different, and my signal is only in one, then I'm better off binning!
- If the signal is in both, I didn't really lose anything
    - This is key: I can't arbitrarily cut up the data and do better!



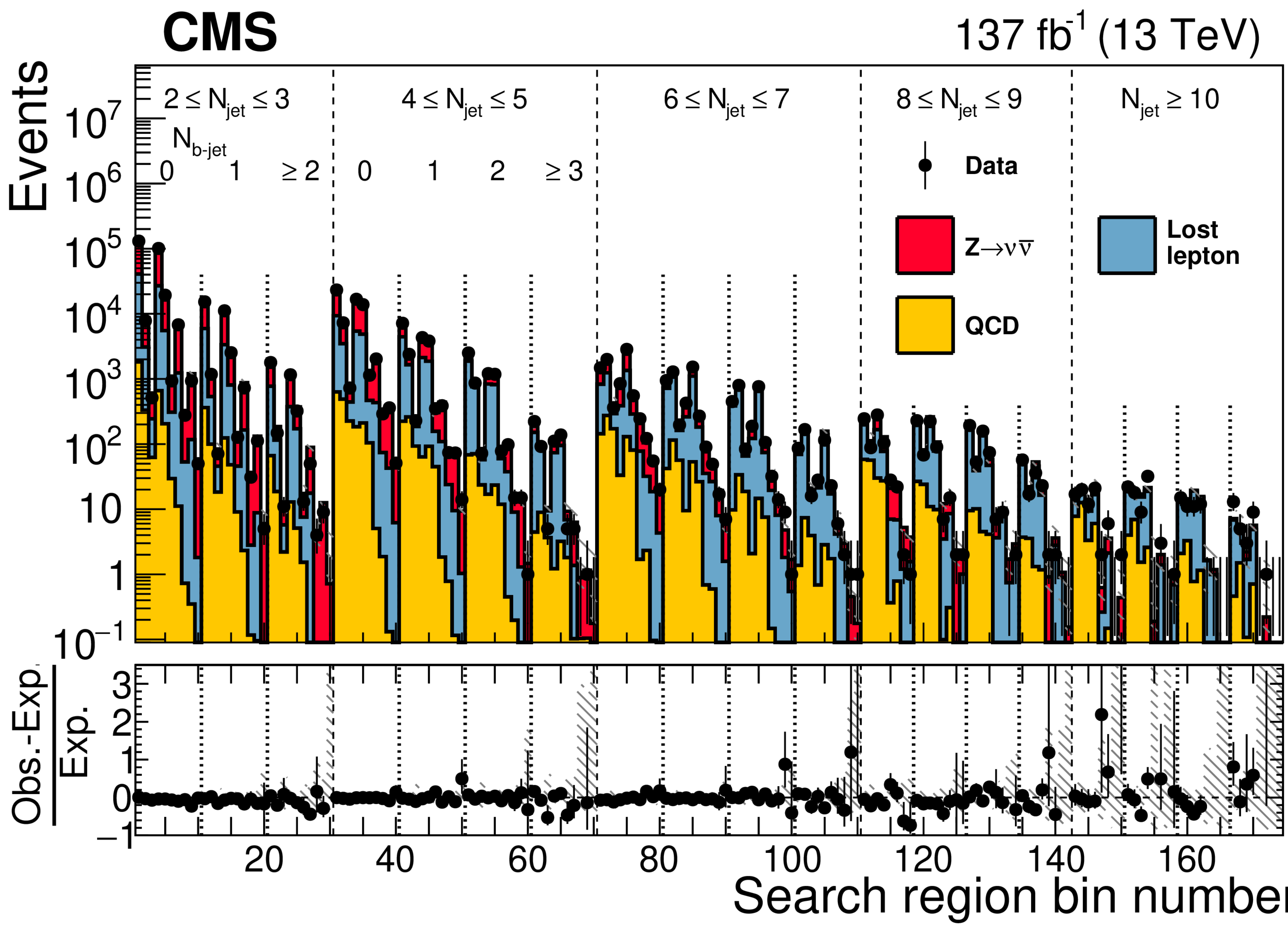
# A simple two-bin example

- Even better, if it’s the same background, one signal region effectively *acts like a control region* for the background!
- Multi-bin fits are *extremely powerful*, but we’ll talk more in the next bit about why they’re also *extremely hard* to do correctly



# Adding Complexity

- This setup is easy to scale, so it was not long before we saw searches that do things like this:



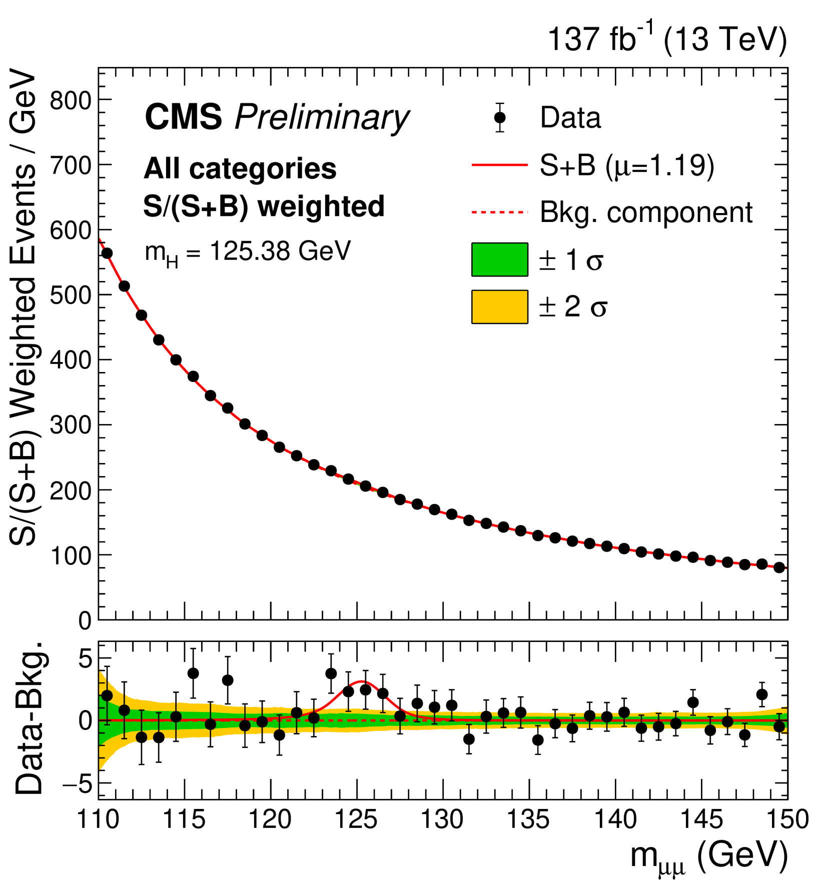
# Searching Step 2: Estimating Backgrounds

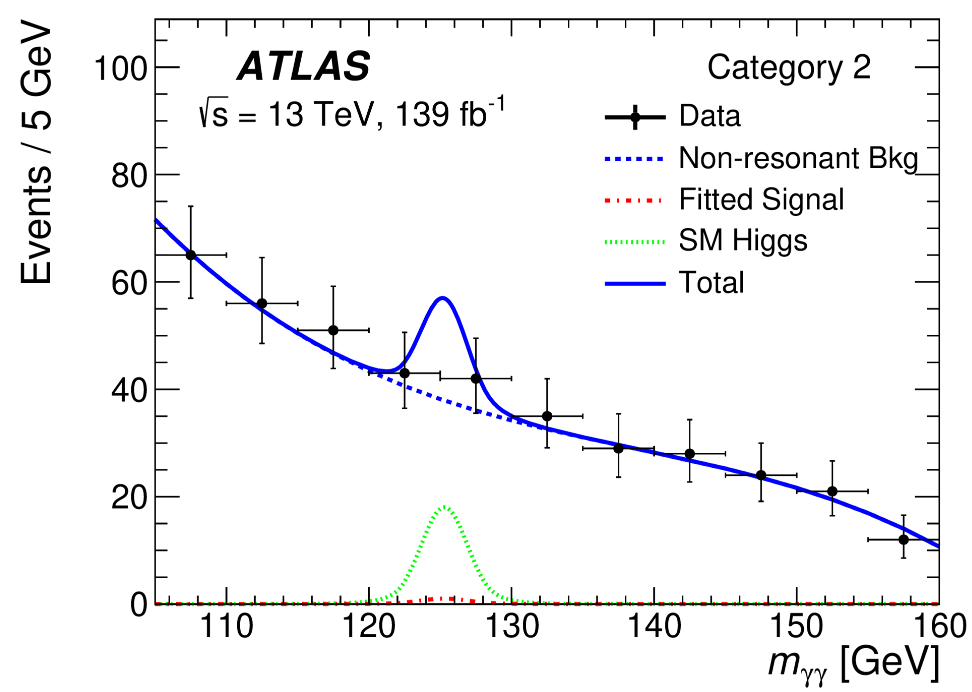
- We have decided on a *search strategy*. Now we need to estimate our backgrounds!

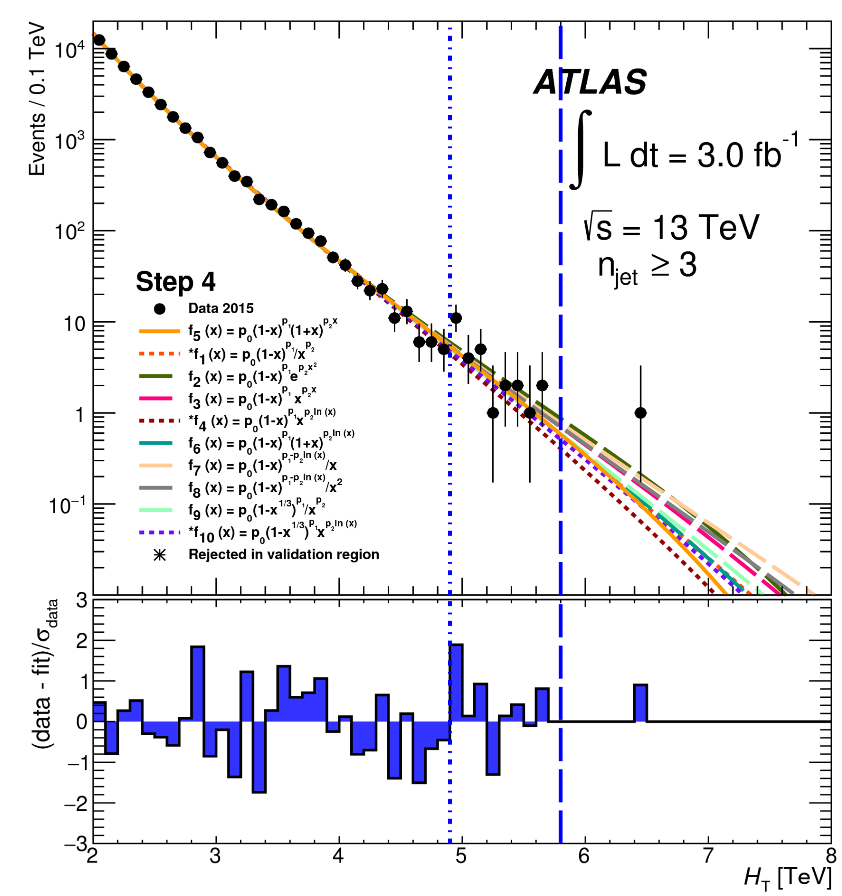
- Remember that the treatment of the *background* and the way you attack the *signal* need not be the same!
- You can do a multi-bin fit to estimate the background in a cut-and-count signal region, or shape fits to estimate the background in a multi-bin fit.

# Maybe easiest to do: Fit the BG!

- As mentioned earlier: fits are very nice.
- If you are fitting something, the function must be sufficiently general, but not too general
- You should be fitting *a background* with *a function*.
    - Don't try and mix things, you'll have a bad time.

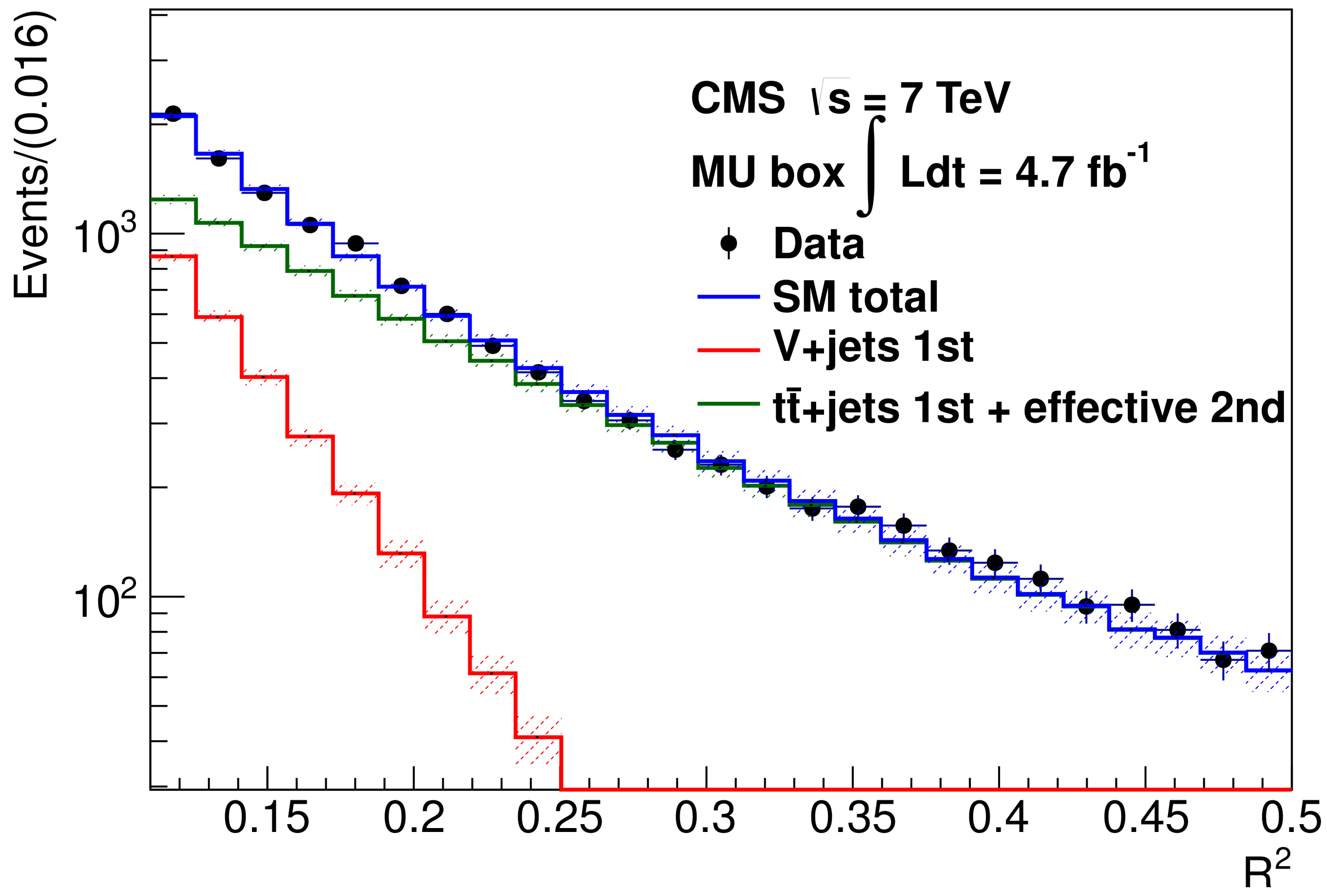


# Maybe easiest to do

- Motivating fit functions can be very tricky. Particularly if you are searching a *tail*, you need to be sure that the function captures the behavior in the *tail*.
- Some folks have attempted pretty complicated setups to do this.



# Easy to say, hard to do well

- “Just use Monte Carlo simulation”
- There is a lot to think about here
    - What accuracy do you need from the MC? Higher order? Multi-leg? This may depend on your signal region!
    - How do you establish uncertainties? There are **lots** of potential uncertainties one could consider, and they take a lot of effort to understand.
- There are as many opinions about doing this “correctly” as stars in the night sky. **FOLLOW YOUR GROUP’S RECOMMENDATIONS.**
    - **But remember:** Recommendations do not excuse you from using your brain.
    - If you don’t follow the recommendations *and* don’t have a good reason, you’re gonna have a really bad time during approval of the search.
    - If you see lots of searches doing the same thing, that’s not a coincidence.
- We will talk a bit about some of these things in a bit, but I’m going to resist the temptation to try to give you a recipe to follow.

# Pessimists Attack

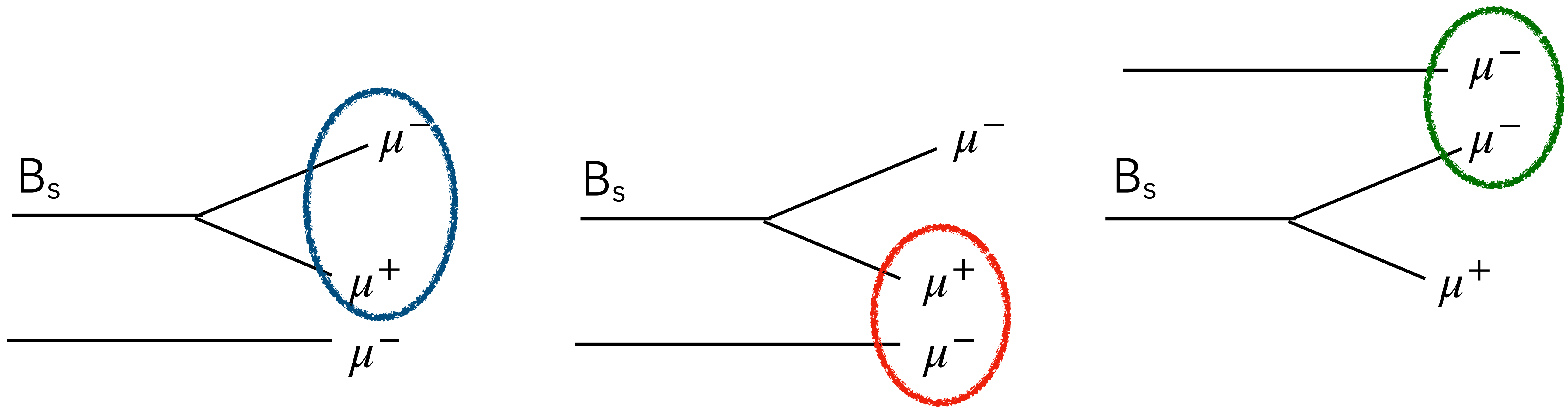
- You might have heard: "All Monte Carlo is junk; estimate your backgrounds from data!"
- That's waaaaaaaaaay too negative and oversimplified (but sometimes useful anyway)
- These people are pushing towards **data-driven backgrounds**

# Data-driven Backgrounds

- There are *lots* of flavors of data-driven backgrounds
  - Fits of all kinds (!) – we've spoken enough about these for the moment
  - Combinatoric estimates
  - Fake object estimates
  - Flavor, charge, or other symmetry-based estimates
  - ABCD
- Most of these require some care because of their *model dependence*.
- CMS do *much* more of this than ATLAS.

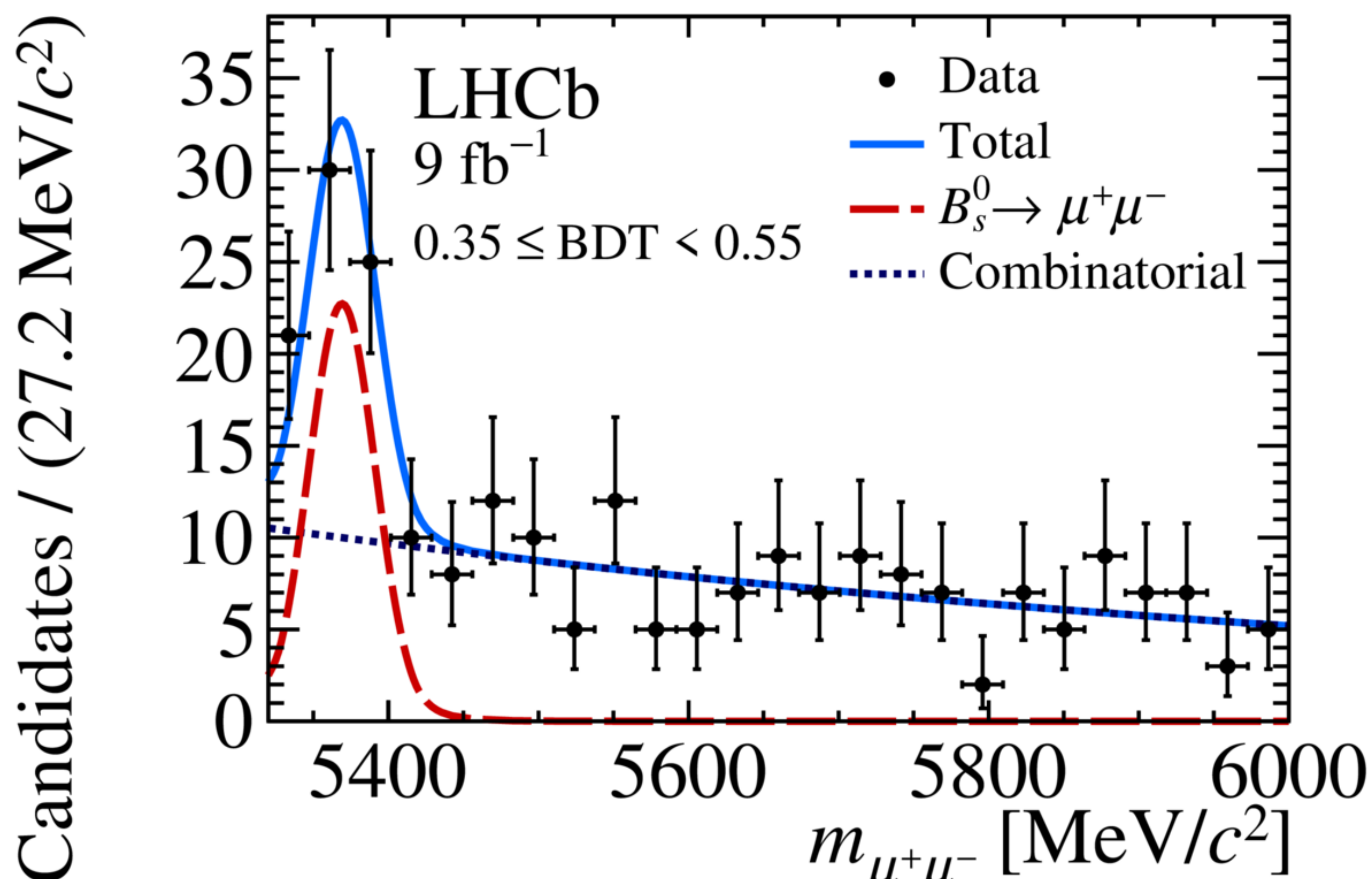
# Combinatorics

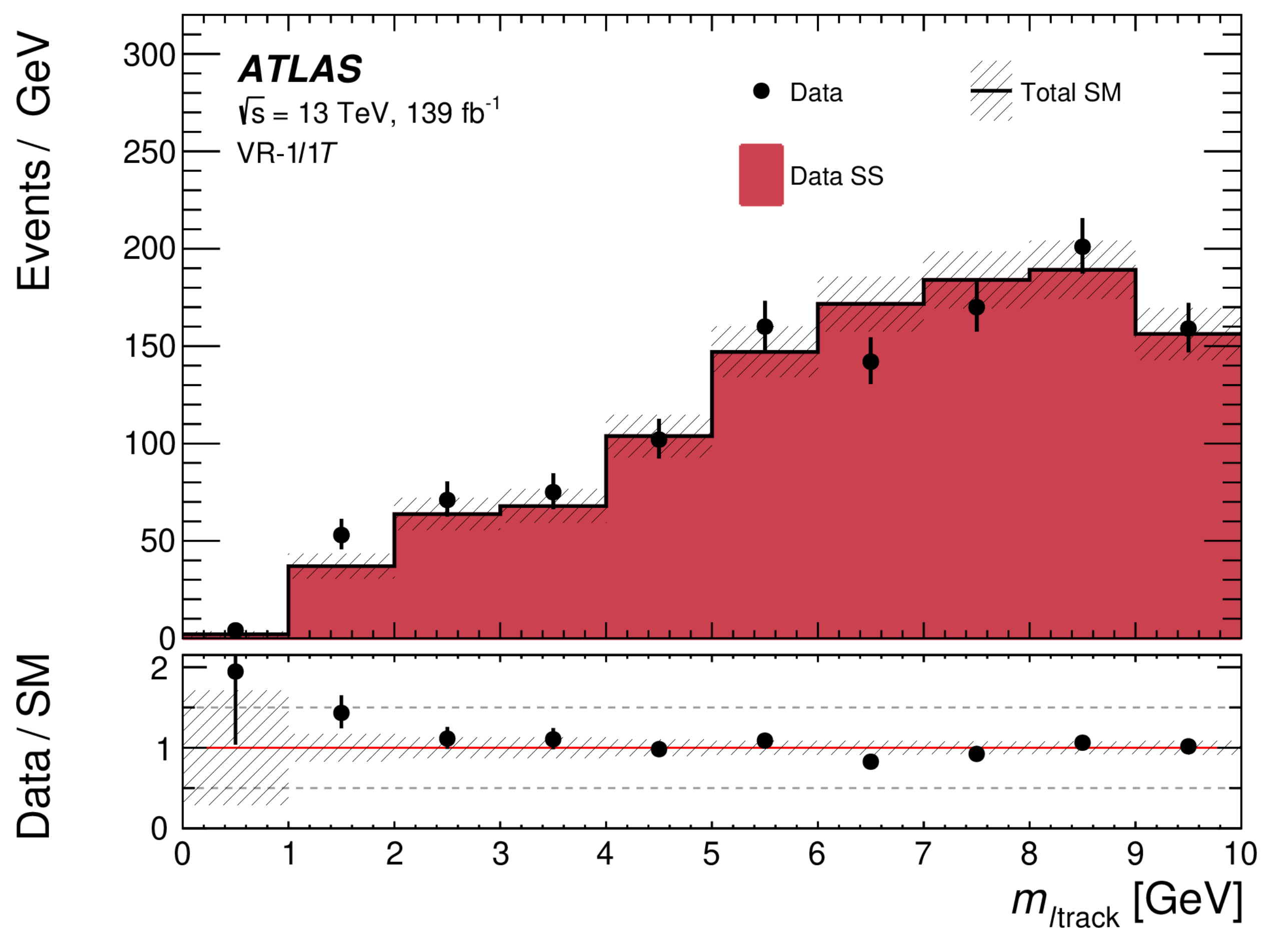
- This is a beautiful 'trick' that is common in B-physics measurements
- When your background is *combinatorial*, you can often e.g. use **same-sign** combinations to estimate **opposite-sign** combinations
  - If it applies, this is a *really* nice way to estimate your background



# Combinatorics

- This is a beautiful ‘trick’ that is common in B-physics measurements
- When your background is *combinatorial*, you can often e.g. use **same-sign** combinations to estimate **opposite-sign** combinations
  - If it applies, this is a *really* nice way to estimate your background



# Fakes (Fake leptons…)

- There are two common ways to estimate fake object backgrounds
  - The **matrix method** and the **fake factor method**
  - They are mathematically equivalent in some setups!
- Both boil down to very similar steps conceptually:
  - Create *loose* and *tight* object IDs (tight will be what you use in your SRs)
  - Estimate how likely a *true* object identified as loose is to be identified as tight (tag and probe; often as a function of kinematics like $p_T$)
  - Estimate how likely a *fake* object identified as loose is to be identified as tight (special ‘enhanced fake’ selections)
  - Measure the number of *loose* objects that would otherwise fall in your SRs or that would fall in some carefully selected region
  - Apply a factor or weight to those events with loose objects to estimate the number of *tight* objects in your SRs that are from *fake objects*
- There are lots of tricks and caveats, but that’s the basic deal

# Fakes

- Or in equations:

Real/fake efficiencies

Number of tight objects

Number of true objects

$$\begin{pmatrix} \langle n_T \rangle \\ \langle n_L \rangle \end{pmatrix} = \begin{pmatrix} \varepsilon_r & \varepsilon_f \\ 1-\varepsilon_r & 1-\varepsilon_f \end{pmatrix} \begin{pmatrix} n_R \\ n_F \end{pmatrix}$$

Number of loose objects

Number of fake objects

**Inverted:**

Number of true objects

Number of tight objects

$$\begin{pmatrix} n_R \\ n_F \end{pmatrix} = \frac{1}{\varepsilon_r - \varepsilon_f} \begin{pmatrix} \bar{\varepsilon}_f & -\varepsilon_f \\ -\bar{\varepsilon}_r & \varepsilon_r \end{pmatrix} \begin{pmatrix} \langle n_T \rangle \\ \langle n_L \rangle \end{pmatrix}$$

Number of fake objects

Real/fake efficiencies

Number of loose objects

# Fakes

- Or in equations:

Number of true objects

Number of tight objects

$$\begin{pmatrix} n_R \\ n_F \end{pmatrix} = \frac{1}{\varepsilon_r - \varepsilon_f} \begin{pmatrix} \bar{\varepsilon}_f & -\varepsilon_f \\ -\bar{\varepsilon}_r & \varepsilon_r \end{pmatrix} \begin{pmatrix} \langle n_T \rangle \\ \langle n_L \rangle \end{pmatrix}$$

Number of fake objects

Real/fake efficiencies

Number of loose objects

**Solved:**

$$\hat{n}_{T \cap F} = \frac{\varepsilon_f}{\varepsilon_r - \varepsilon_f} \left( \varepsilon_r (n_T + n_L) - n_T \right)$$

Number of tight fake objects

Real/fake efficiencies

Number of tight or loose objects

# Fakes

- It can get more complicated for more leptons…

$$\begin{pmatrix} \langle n_{tt} \rangle \\ \langle n_{tl} \rangle \\ \langle n_{lt} \rangle \\ \langle n_{ll} \rangle \end{pmatrix} = \begin{pmatrix} \varepsilon_{r1}\varepsilon_{r2} & \varepsilon_{r1}\varepsilon_{f2} & \varepsilon_{f1}\varepsilon_{r2} & \varepsilon_{f1}\varepsilon_{f2} \\ \varepsilon_{r1}\bar{\varepsilon}_{r2} & \varepsilon_{r1}\bar{\varepsilon}_{f2} & \varepsilon_{f1}\bar{\varepsilon}_{r2} & \varepsilon_{f1}\bar{\varepsilon}_{f2} \\ \bar{\varepsilon}_{r1}\varepsilon_{r2} & \bar{\varepsilon}_{r1}\varepsilon_{f2} & \bar{\varepsilon}_{f1}\varepsilon_{r2} & \bar{\varepsilon}_{f1}\varepsilon_{f2} \\ \bar{\varepsilon}_{r1}\bar{\varepsilon}_{r2} & \bar{\varepsilon}_{r1}\bar{\varepsilon}_{f2} & \bar{\varepsilon}_{f1}\bar{\varepsilon}_{r2} & \bar{\varepsilon}_{f1}\bar{\varepsilon}_{f2} \end{pmatrix} \begin{pmatrix} n_{rr} \\ n_{rf} \\ n_{fr} \\ n_{ff} \end{pmatrix}$$

- Usually people don't go beyond 2 or 3 fake leptons