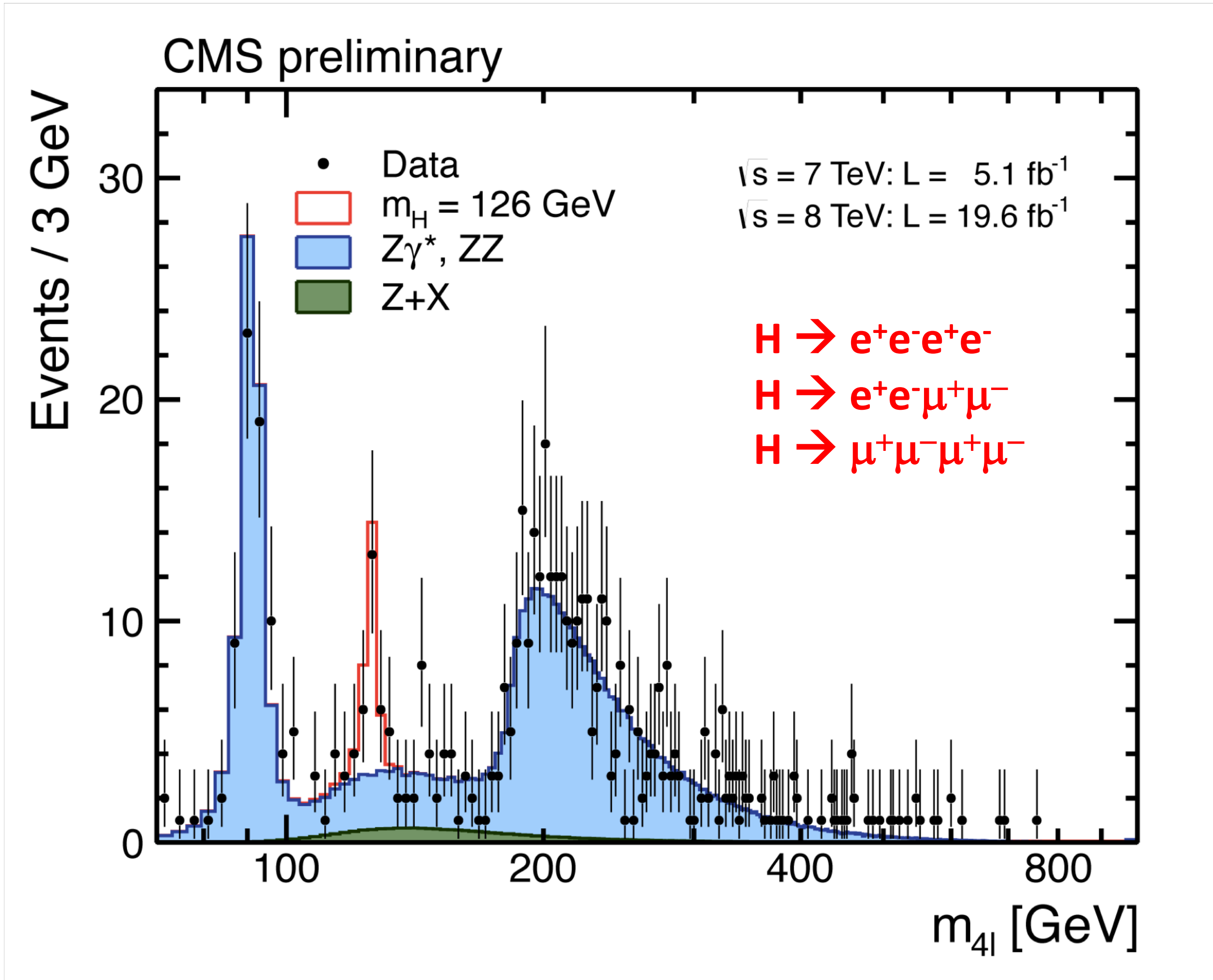
A common problem in physics

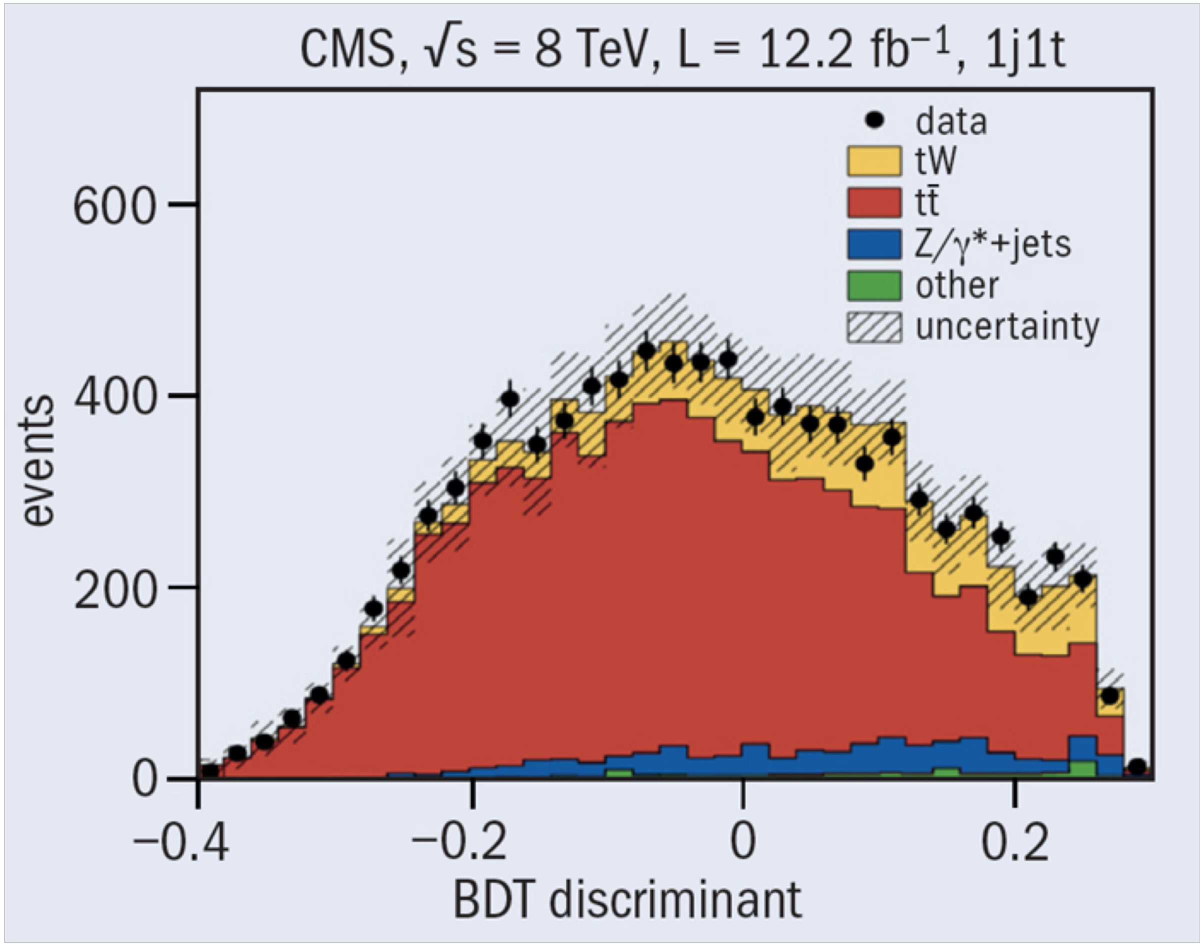
- Dig out “signal” for some interesting but rare process out of a sea of “background”
- Toy example here from high energy physics
- First step: select “events” with patterns of final state particles/kinematics according to what your signal should look like to distinguish it from background
- Most likely this selection will give you both signal and background events
- You then have to
  - Decide whether you have a signal or not
  - If so, estimate how “strong” your signal is
    - because you are trying to “measure” something physical, eg, a cross-section

Sometimes it is fairly obvious



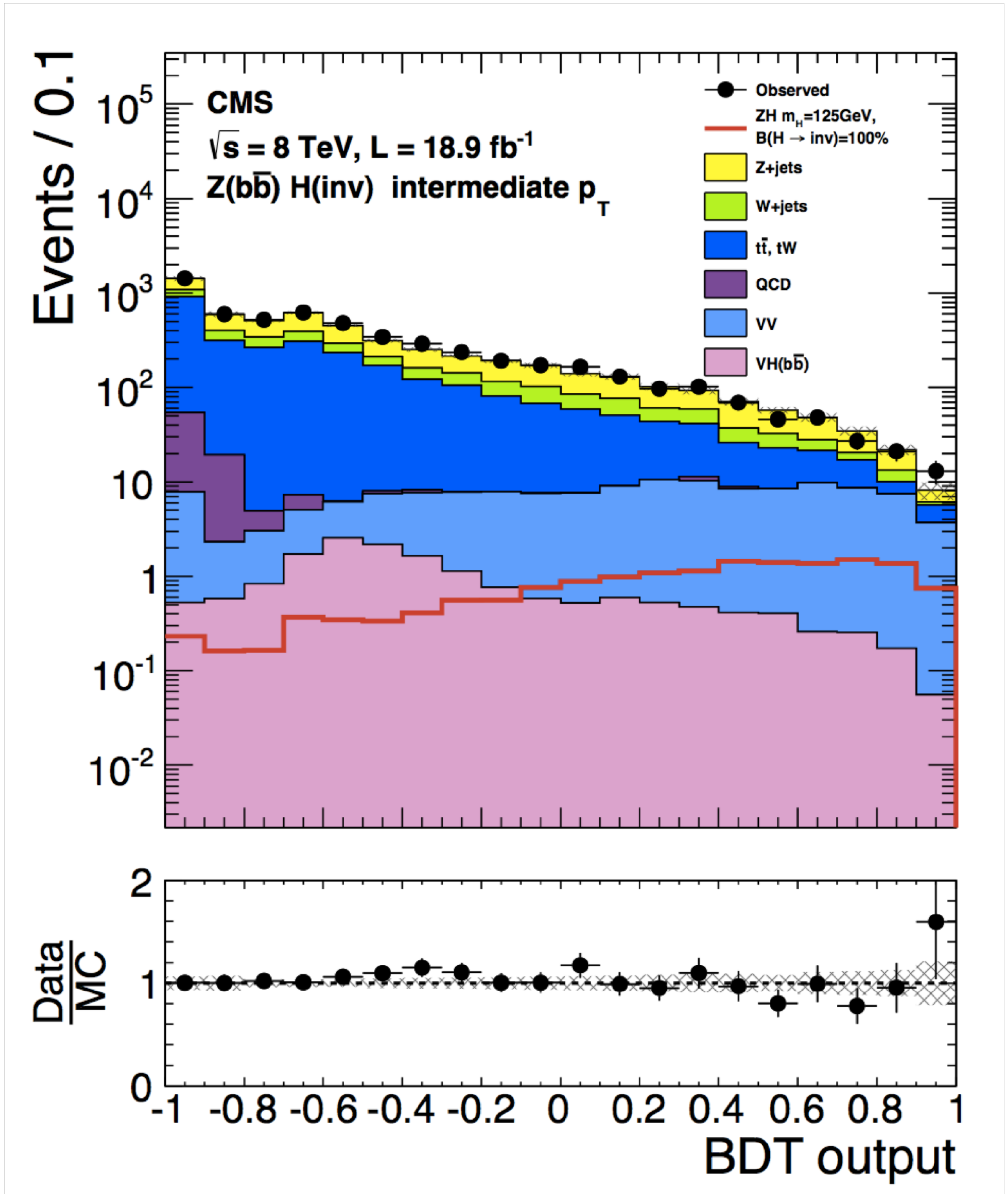
Obviously there is a signal. Precisely estimating the strength of the signal is a different story

Sometimes less so



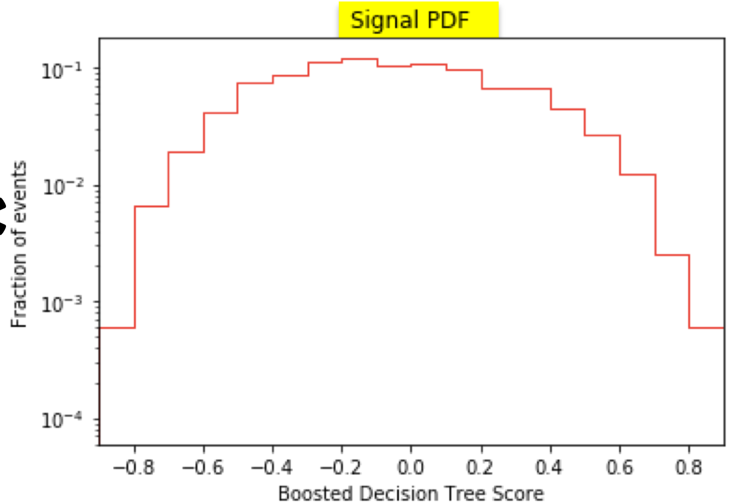
Here the signal is "pp→tW". (t=top quark…W=W boson)
What is plotted is a "machine learning" "boosted decision tree" discriminant which combines a lot of information into one single variable that looks different for signal and background

Sometimes it's kind of hopeless



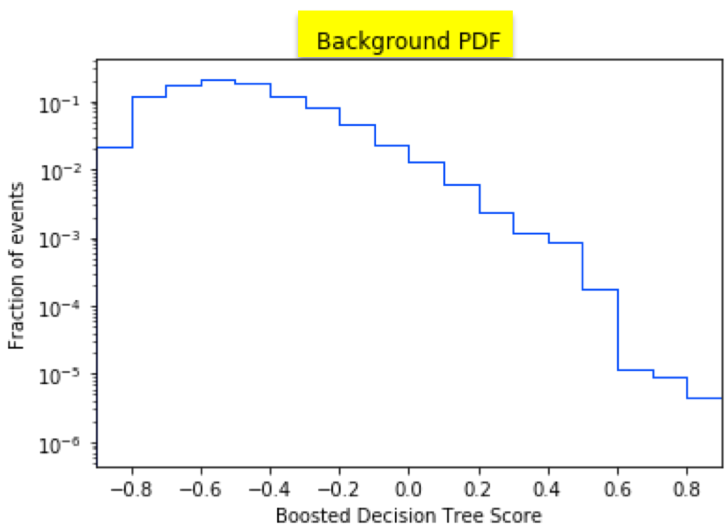
- In general have to fit the data distribution to sum of signal (S) + background (B) distributions
- This could mean fitting
  - A single 1D histogram
  - A 2D (or 3D, or..) histogram
  - Several different histograms simultaneously
  - Or even single events in an unbinned fashion
- Obviously you need to select and treat your data intelligently, and understand the pdfs for S and B
  - Including uncertainties
- We now look at a semi-realistic example

S*

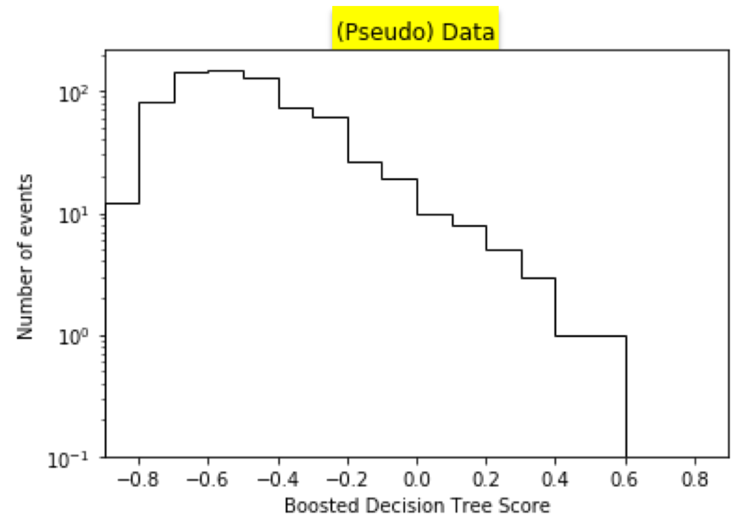

Signal PDF
Fraction of events
Boosted Decision Tree Score
10⁻¹
10⁻²
10⁻³
10⁻⁴
−0.8 −0.6 −0.4 −0.2 0.0 0.2 0.4 0.6 0.8

+

B*


Background PDF
Fraction of events
Boosted Decision Tree Score
10⁻¹
10⁻²
10⁻³
10⁻⁴
10⁻⁵
10⁻⁶
−0.8 −0.6 −0.4 −0.2 0.0 0.2 0.4 0.6 0.8

=


(Pseudo) Data
Number of events
Boosted Decision Tree Score
10²
10¹
10⁰
10⁻¹
−0.8 −0.6 −0.4 −0.2 0.0 0.2 0.4 0.6 0.8

- Want to estimate number of signal (S) and background events (B)
  - **actually, we usually mostly only care about S**
  - See "Appendix" for what we really mean by this
- Use the histograms as PDFs
- Because many of the bins are low stats, need Poisson Uncertainties → use NLL
- $\{s_i\}$ and $\{b_i\}$ are the binned pdfs for S and B
  - Normalized to 1, ie, $\Sigma s_i = 1$ and $\Sigma b_i = 1$
- $\{d_i\}$ are contents of i-th bin in the (pseudo) data
- The model for the $\{d_i\}$ is

$$\mu_i = S*s_i + B*b_i$$

$$\mathcal{L} = \prod p(d_i) = \prod \frac{e^{-\mu_i}\mu_i^{d_i}}{d_i!}$$

$$-\log\mathcal{L} = \sum \mu_i - d_i \log \mu_i$$

$$-\log\mathcal{L} = \sum S\cdot s_i + B\cdot b_i - d_i \log(S\cdot s_i + B\cdot b_i)$$

$$-\log\mathcal{L} = S + B - \sum d_i \log(S\cdot s_i + B\cdot b_i)$$

This is actually called "extended log likelihood"
(I dropped an additive constant from the NLL)

Note: it does not look at all like a χ^2.

But as was discusses previously in the large d_i limit it is the same as χ^2 within a factor of ½

We will fit with Minuit

```
# iminuit tutorial at
# https://nbviewer.jupyter.org/github/iminuit/iminuit/blob/master/tutorial/basic_tutorial.ipynb
import numpy as np
import matplotlib.pyplot as plt
from iminuit import Minuit
```

Define the negative log likelihood function

```
# d, s_pdf, and b_bdf are np.arrays
# d       = contents of data histogram
# s_pdf   = contents of signal pdf histogram
# b_pdf   = contents of background pdf histogram
# S       = number of signal events
# B       = number of background events
# Assume that pdf's are normalized, eg s_pdf.sum()=1
def NLL(S,B):
    temp = d * np.log(S*s_pdf + B*b_pdf)
    return S + B - temp.sum()
```

/home/pi/physrpi/campagnari/python/maxLikFit.py

```python
# d, s_pdf, and b_bdf are np.arrays
# d      = contents of data histogram
# s_pdf  = contents of signal pdf histogram
# b_pdf  = contents of background pdf histogram
# S      = number of signal events
# B      = number of background events
# Assume that pdf's are normalized, eg s_pdf.sum()=1
def NLL(S,B):
    temp = d * np.log(S*s_pdf + B*b_pdf)
    return S + B - temp.sum()

# Setup the fitter.  S and B are the initial guesses
# print_level=0 --> suppress print of intermediate information
# errordef = 0.5   because for NLL 1 sigma errors are  from
#                  NLL-NLL(at minimum) = 0.5
# error S and error B are initial steps to look for minimum
m = Minuit(NLL, S=10., B=500., print_level=1,
           errordef=0.5, error_S=1.0, error_B=1.0)

# migrad is the basic minimizatio
m.migrad()
```

Note: I could have told Minuit to impose the constraint S>0 and/or B>0

This is best to be avoided for technical reasons.

However: with S and/or B < 0 some of the calls to NLL could result in taking log of negative number. I should probably have protected against it, but luckily it did not happen ☺

FCN = -2515.3836311930554	TOTAL NCALL = 46	NCALLS = 46
EDM = 2.6719919762844715e-06	GOAL EDM = 5e-06	UP = 0.5

Valid	Valid Param	Accurate Covar	PosDef	Made PosDef
True	True	True	True	False
Hesse Fail	HasCov	Above EDM		Reach calllim
False	True	False		False

+	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	S	23.5165	10.2353					No
1	B	701.541	27.9716					No

I run the code through a "jupyter notebook"
This gives a nicely formatted output.
It does not look as nice when run normally.
But you get the same info.

Green means "it worked"

The answer is S = 23.5 ± 10.2

More sophisticated analysis: “minos”

```
# minos does not assume that the NLL is
# parabolic and calculates asymmetric errors
m.minos(var="S")
m.print_param()
```

Minos status for S: VALID

Error	-9.516333676056222	10.933566962130033
Valid	True	True
At Limit	False	False
Max FCN	False	False
New Min	False	False

+	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	S	23.5165	10.2353	-9.51633	10.9336			No
1	B	701.541	27.9716					No

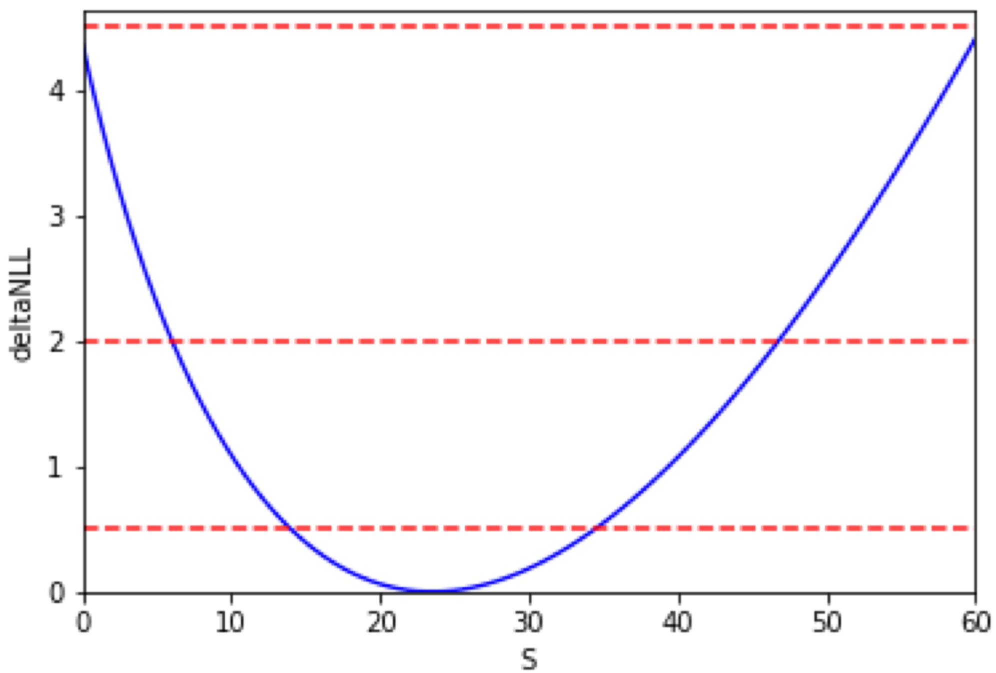
$$S = 23.5^{+10.9}_{-9.5}$$

PS: the dataset had 700 background events and 25 signal events

```
# Profile scan of the fitted function (NLL).
# At each FIXED value of S, fit again for B,
# extract the NLL at the minimu, subtract
# the NLL at the GLOBAL minimum, and plot it
xxx, yyy, _ = m.mnprofile('S', subtract_min=True, bins=100, bound=(0,60))
# m.mnprofile does all the work...
# Now we just plot the results
# deltaNLL = 0.5 (2, 4.5 ) corresponds to 1 (2, 3) sigma
fig3, ax3 = plt.subplots()
ax3.plot(xxx,yyy,linestyle='solid', color='b')
ax3.set_xlim(min(xxx), max(xxx))
ax3.set_ylim(0.)
ax3.set_xlabel('S')
ax3.set_ylabel('deltaNLL')
ax3.plot([min(xxx), max(xxx)], [0.5, 0.5], linestyle='dashed', color='red')
ax3.plot([min(xxx), max(xxx)], [2.0, 2.0], linestyle='dashed', color='red')
ax3.plot([min(xxx), max(xxx)], [4.5, 4.5], linestyle='dashed', color='red')
```

One line of code to do the scan

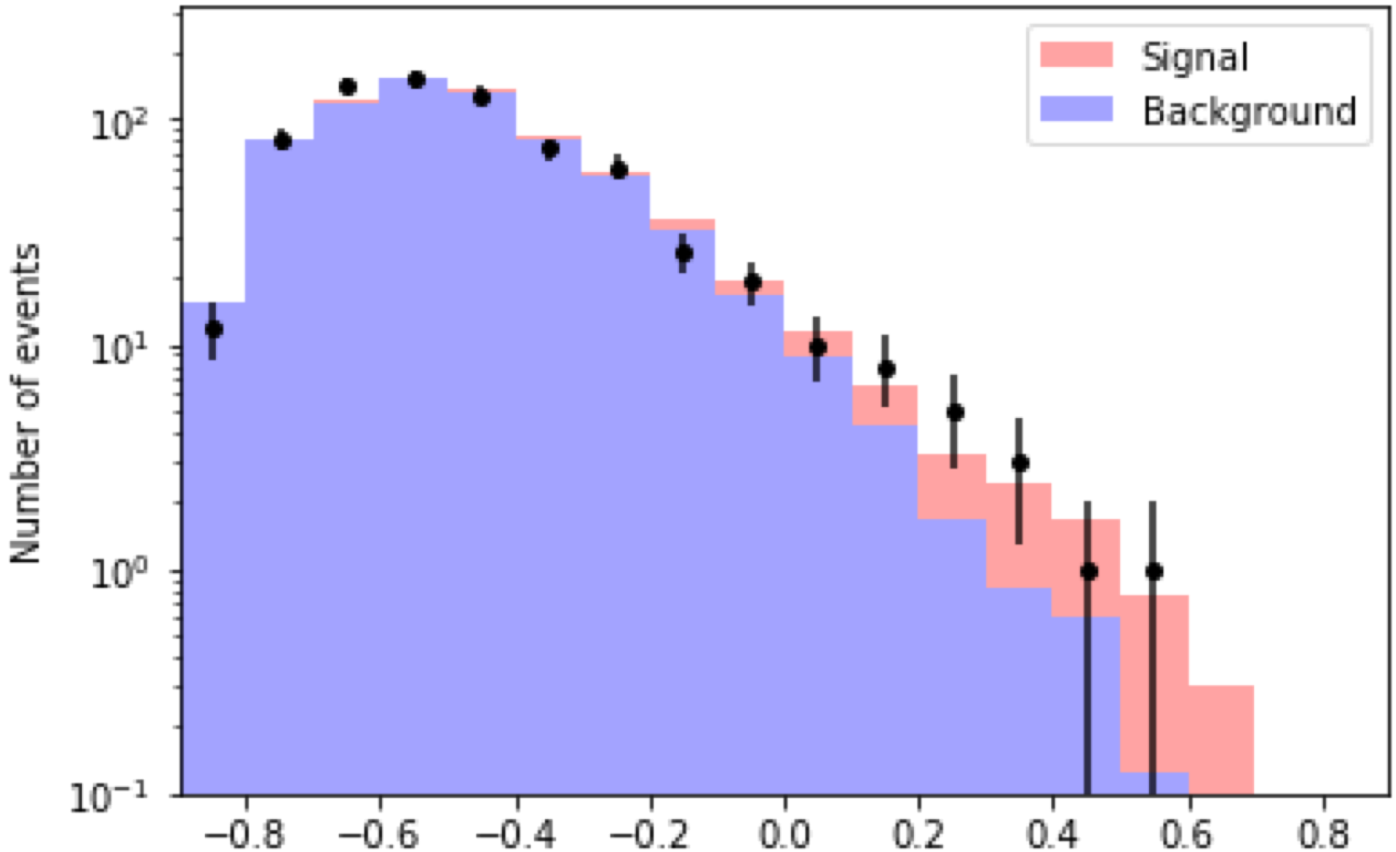
Just plotting stuff



Can see by eye that it is not exactly parabolic.

The asymmetry around the minimum is what gives asym. minos errors

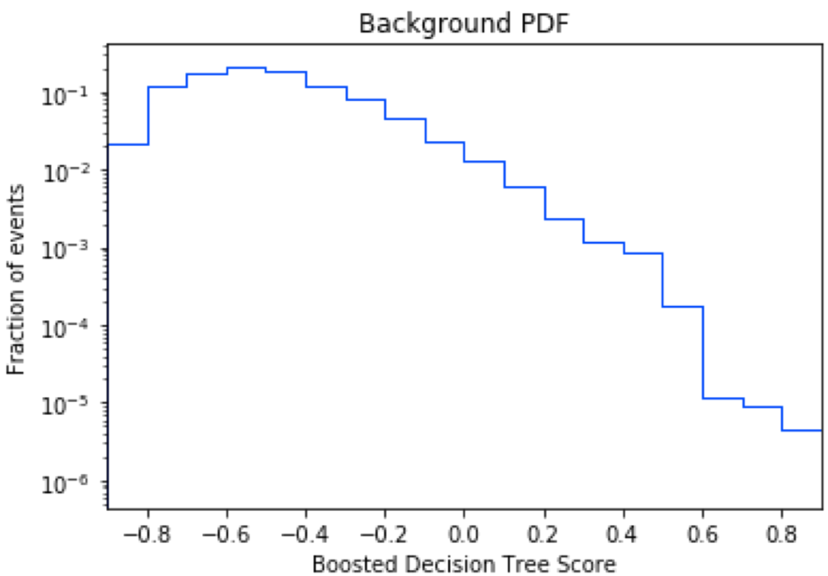
This is almost 3σ away from 0, but not quite.

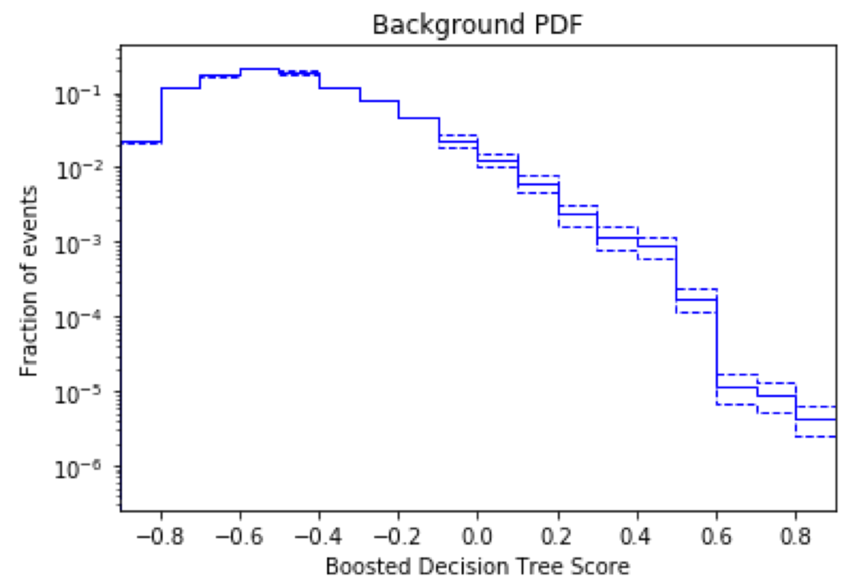


Data here is plotted with sqrt(N) uncertainties.
Not quite right, but conventional

Systematics (simple example)

Imagine we do not know the background pdf perfectly



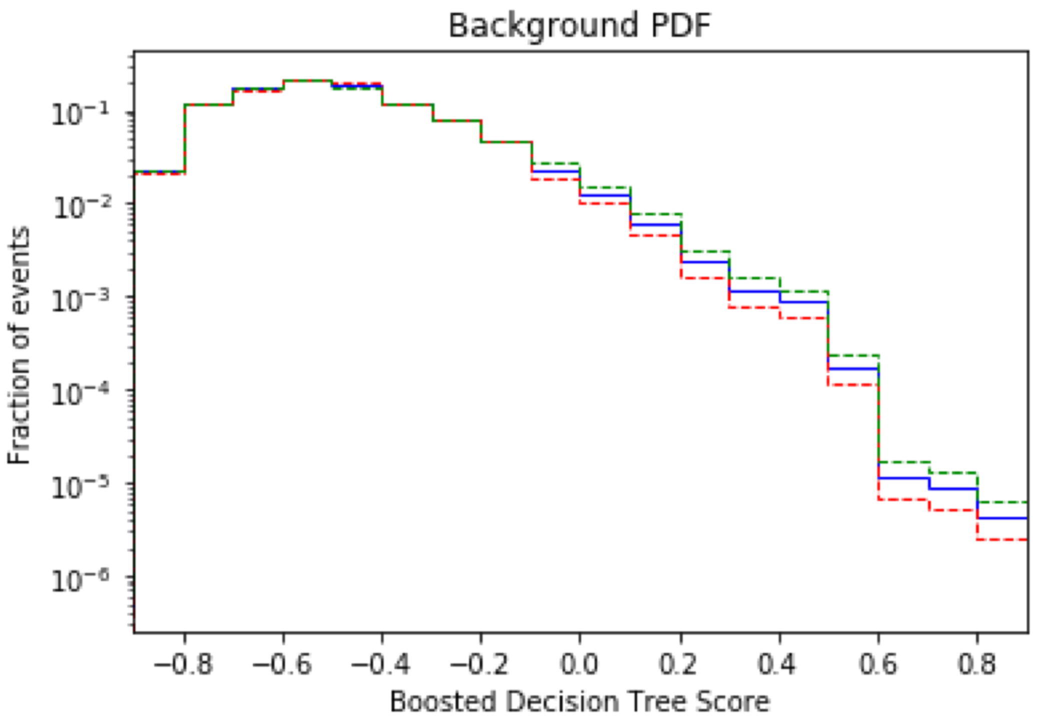


This is our best guess for the bg pdf (what we have used so far) . Comes from some separate study we may have done

The dashed lines “bracket” at the 1 sigma level our lack of knowledge of the bg pdf. This “information” also would come from some study

<u>How to take this “shape uncertainty” into account?</u>

Introduce a new parameter α that smoothly interpolates the options for the background pdf:



b_pdf: the best guess
b1_pdf: one alternative
b2_pdf: the other alternative

$\alpha>0$:
pdf = b_pdf + α (b1_pdf-b_pdf)
$\alpha<0$:
pdf = b_pdf - α (b2_pdf-b_pdf)

This insures that for
$\alpha=0$: pdf = b_pdf
$\alpha=1$: pdf = b1_pdf
$\alpha=-1$: pdf = b2_pdf
(and all pdf's in-between or even more "like" b1_pdf or b2_pdf)

Our best guess for α is $\alpha=0$, with $\sigma=1$.
Let the data itself figure out what α should be
Modify the likelihood to include information on α
α **becomes a 3rd parameter to be fit for**
Include "prior" information on α in likelihood
"Nuisance parameter"

$$\mathcal{L} \rightarrow \mathcal{L}\, e^{-\frac{\alpha^2}{2}}$$

$$-\log \mathcal{L} \rightarrow -\log \mathcal{L} + \frac{\alpha^2}{2}$$

Old code, no shape uncertainty

```
# d, s_pdf, and b_bdf are np.arrays
# d      = contents of data histogram
# s_pdf  = contents of signal pdf histogram
# b_pdf  = contents of background pdf histogram
# S      = number of signal events
# B      = number of background events
# Assume that pdf's are normalized, eg s_pdf.sum()=1
def NLL(S,B):
    temp = d * np.log(S*s_pdf + B*b_pdf)
    return S + B - temp.sum()
```

```
# Setup the fitter.  S and B are the initial guesses
# print_level=0 --> suppress print of intermediate information
# errordef = 0.5   because for NLL 1 sigma errors are  from
#                  NLL-NLL(at minimum) = 0.5
# error_S and error_B are initial steps to look for minimum
m = Minuit(NLL, S=10., B=500., print_level=1,
         errordef=0.5, error_S=1.0, error_B=1.0)
```

New code, with shape uncertainty

```
# d, s_pdf, b_bdf, b1_pdf, b2_pdf are np.arrays
# d      = contents of data histogram
# s_pdf  = contents of signal pdf histogram
# b_pdf  = contents of default background pdf histogram
# b1_pdf = contents of alternative 1 to b_pdf
# b2_pdf = contents of alternative 2 too b_pdf
# S      = number of signal events
# B      = number of background events
# alpha  = parameter to interpolate between pdfs
# Assume that pdf's are normalized, eg s_pdf.sum()=1
def new_NLL(S,B,alpha):
    # code below insures that
    # alpha=0  ---> use b_pdf
    # alpha=1  ---> use b1_pdf
    # alpha=-1 ---> use b2_pdf
    # (and smoothly interpolates vs. alpha)
    if alpha>0:
        new_b_pdf = b_pdf + alpha*(b1_pdf-b_pdf)
    else:
        new_b_pdf = b_pdf - alpha*(b2_pdf-b_pdf)

    # should be already normalized, but make sure
    new_b_pdf = new_b_pdf / new_b_pdf.sum()
    temp = d * np.log(S*s_pdf + B*new_b_pdf)
    return S + B - temp.sum() + alpha*alpha/2.
```

```
# Setup the fitter.  S, B, alpha are the initial guesses
# print_level=0 --> suppress print of intermediate information
# errordef = 0.5    because for NLL 1 sigma errors are  from
#                   NLL-NLL(at minimum) = 0.5
# error_S, error_B, alpha: are initial steps to look for minimum
new_m = Minuit(new_NLL, S=10., B=500., alpha=0., print_level=1,
       errordef=0.5, error_S=1.0, error_B=1.0, error_alpha=0.1)
```

Old code, no shape uncertainty

```python
# d, s_pdf, and b_bdf are np.arrays
# d      = contents of data histogram
# s_pdf  = contents of signal pdf histogram
# b_pdf  = contents of background pdf histogram
# S      = number of signal events
# B      = number of background events
# Assume that pdf's are normalized, eg s_pdf.sum()=1
def NLL(S,B):
    temp = d * np.log(S*s_pdf + B*b_pdf)
    return S + B - temp.sum()
```

Shows the flexibility of Minuit.
The function to minimize is a NLL with a bunch of Poisson terms from data counts, but also with an additional term that has nothing to do with data counts and is most certainly not Poissonian

New code, with shape uncertainty

```python
# d, s_pdf, b_bdf, b1_pdf, b2_pdf are np.arrays
# d      = contents of data histogram
# s_pdf  = contents of signal pdf histogram
# b_pdf  = contents of default background pdf histogram
# b1_pdf = contents of alternative 1 to b_pdf
# b2_pdf = contents of alternative 2 too b_pdf
# S      = number of signal events
# B      = number of background events
# alpha  = parameter to interpolate between pdfs
# Assume that pdf's are normalized, eg s_pdf.sum()=1
def new_NLL(S,B,alpha):
    # code below insures that
    # alpha=0  ---> use b_pdf
    # alpha=1  ---> use b1_pdf
    # alpha=-1 ---> use b2_pdf
    # (and smoothly interpolates vs. alpha)
    if alpha>0:
        new_b_pdf = b_pdf + alpha*(b1_pdf-b_pdf)
    else:
        new_b_pdf = b_pdf - alpha*(b2_pdf-b_pdf)

    # should be already normalized, but make sure
    new_b_pdf = new_b_pdf / new_b_pdf.sum()
    temp = d * np.log(S*s_pdf + B*new_b_pdf)
    return S + B - temp.sum() + alpha*alpha/2.
```

```python
# Setup the fitter.  S and B are the initial guesses
# print_level=0 --> suppress print of intermediate information
# errordef = 0.5   because for NLL 1 sigma errors are  from
#                  NLL-NLL(at minimum) = 0.5
# error_S and error_B are initial steps to look for minimum
m = Minuit(NLL, S=10., B=500., print_level=1,
         errordef=0.5, error_S=1.0, error_B=1.0)
```

```python
# Setup the fitter.  S, B, alpha are the initial guesses
# print_level=0 --> suppress print of intermediate information
# errordef = 0.5   because for NLL 1 sigma errors are  from
#                  NLL-NLL(at minimum) = 0.5
# error_S, error_B, alpha: are initial steps to look for minimum
new_m = Minuit(new_NLL, S=10., B=500., alpha=0., print_level=1,
         errordef=0.5, error_S=1.0, error_B=1.0, error_alpha=0.1)
```

Results with shape unc.

Minos status for S: VALID

Error	-11.025370084171879	12.60857601321337
Valid	True	True
At Limit	False	False
Max FCN	False	False
New Min	False	False

±	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	S	19.9292	11.8446	-11.0254	12.6086			No
1	B	705.074	28.7197					No
2	alpha	-0.412914	0.746692					No

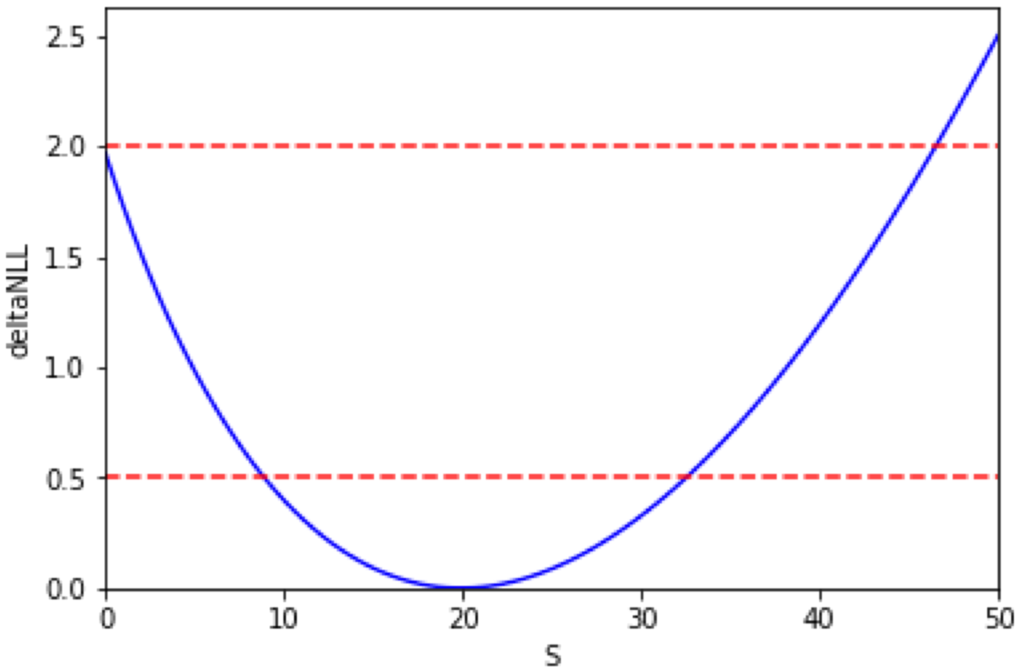
$$S = 19.9^{+12.6}_{-11.0}$$

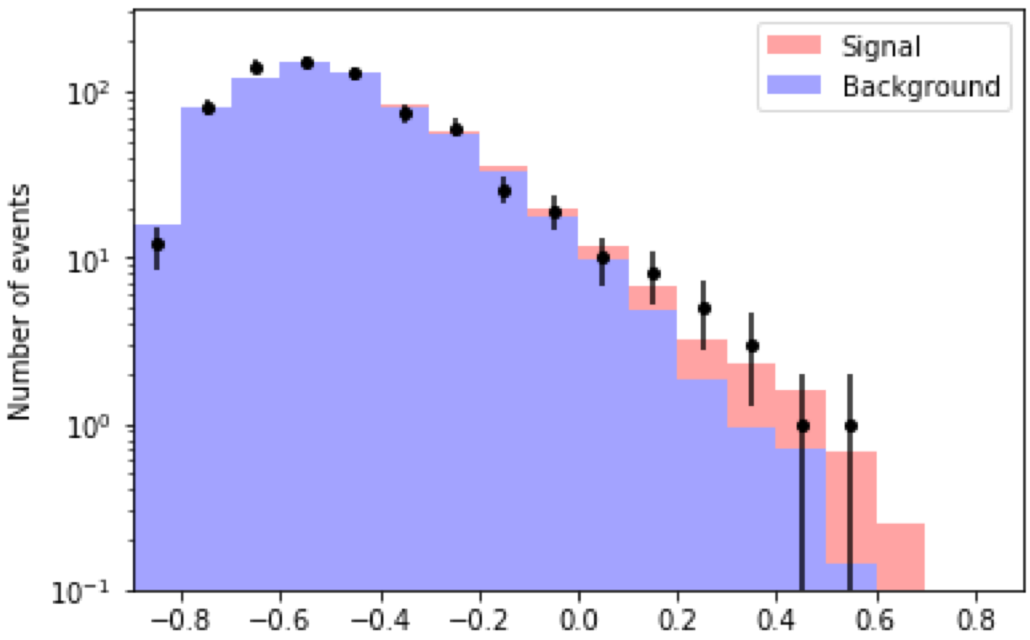
used to be

$$S = 23.5^{+10.9}_{-9.5}$$

Uncertainty increased because we told the fitter that we are not entirely sure about the shape of the background (makes sense that we lose accuracy)

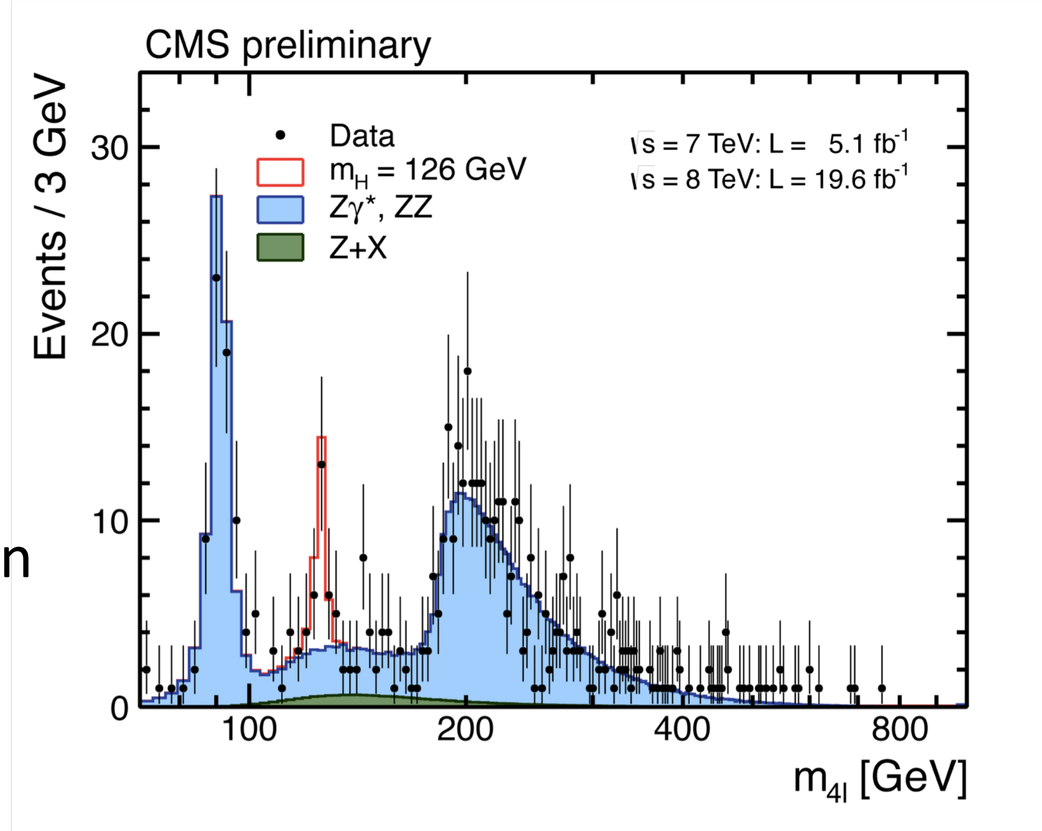
Results with shape unc.





Just about 2σ from zero

If the signal (or the background!) varies very fast, it makes sense to make the bins small enough to capture the important features of the distribution



In this example the signal (in red) is sharply peaked.
The binsize is *small enough*.

A binsize of 50 GeV would have completely lost all the information on the signal.

Could make binsize even smaller

In the limit that we let the binsize go to zero, we can perform an **unbinned** ML fit

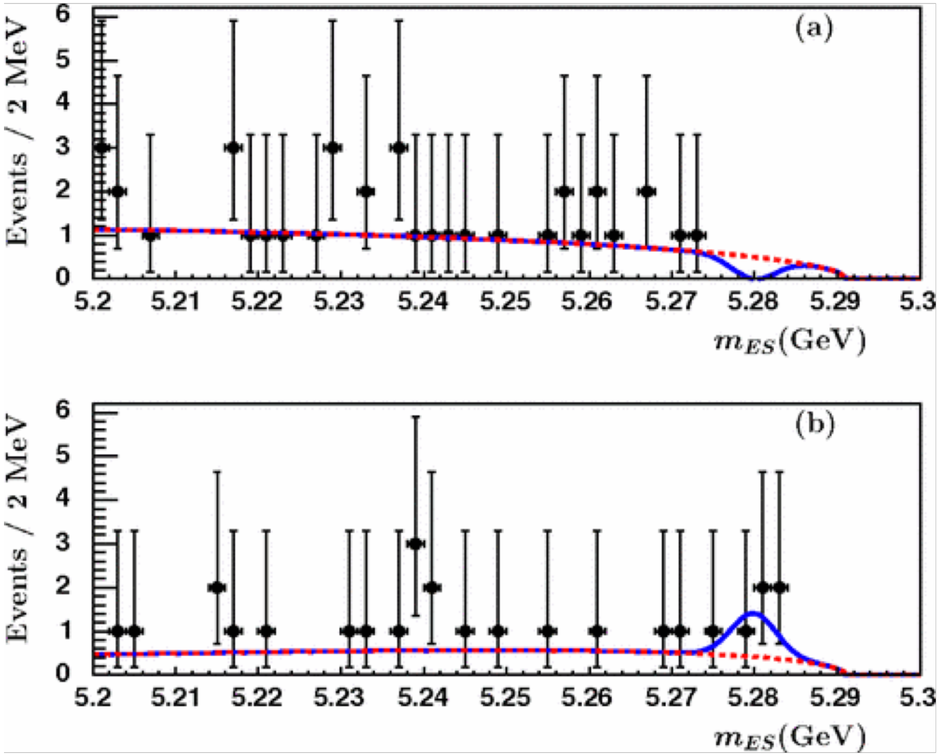
Example of unbinned fit results

The S+B fit is blue
The B component is red

This is an interesting case because the fitted S in the top panel is negative!

$$S = -1.6^{+0.7}_{-0.0}$$

Note: the fit is unbinned but the data are plotted binned (how else could we plot it?)



In order to perform an unbinned fit it is best if the the pdf's for S and B are continuous functions (instead of histograms)
But how does the procedure that we outlined change?

Binned fit:

$$-\log \mathcal{L} \quad = \quad S + B - \sum d_i \log(S \cdot s_i + B \cdot b_i)$$

d_i = number of data counts in the *i-th* bin

$S\ s_i$ = number of expected counts in the *i-th* bin from signal

$B\ b_i$ = number of expected counts in the *i-th* bin from background

The sum is over <u>bins</u>

Unbinned fit:

$$-\log \mathcal{L} = S + B - \sum \log(S \cdot s(x_i) + B \cdot b(x_i))$$

x_i = measured value of discriminating variable for *i-th* event

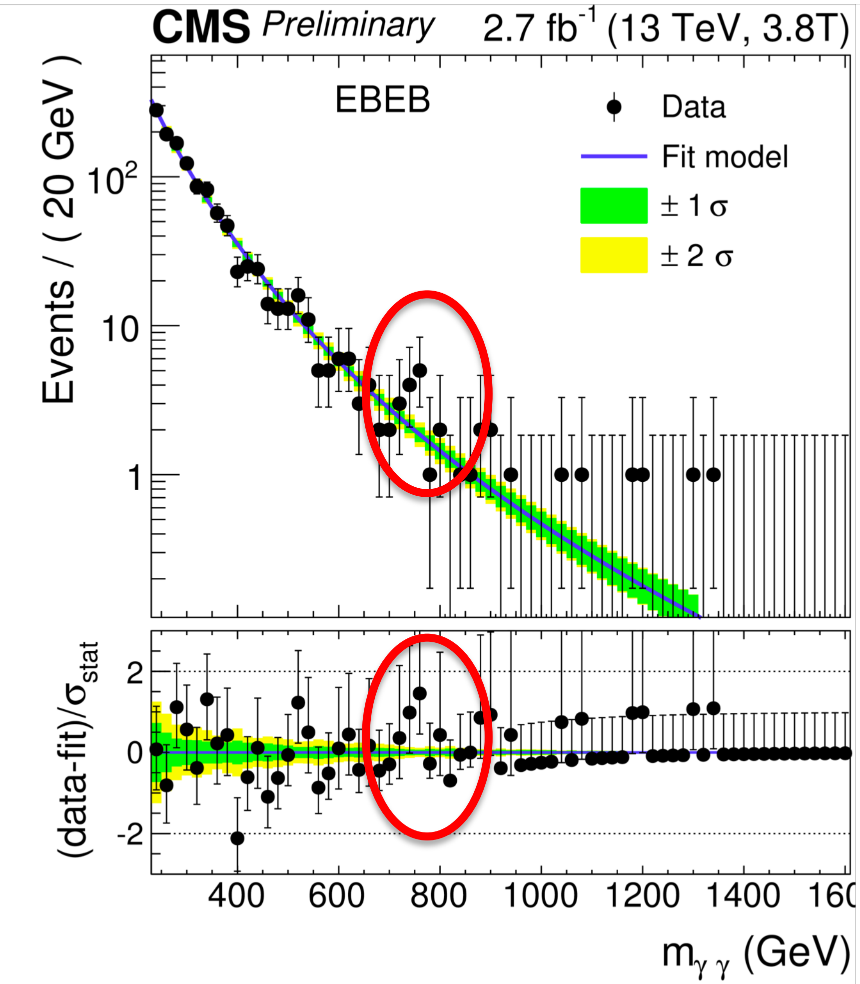
s(x) = pdf for signal

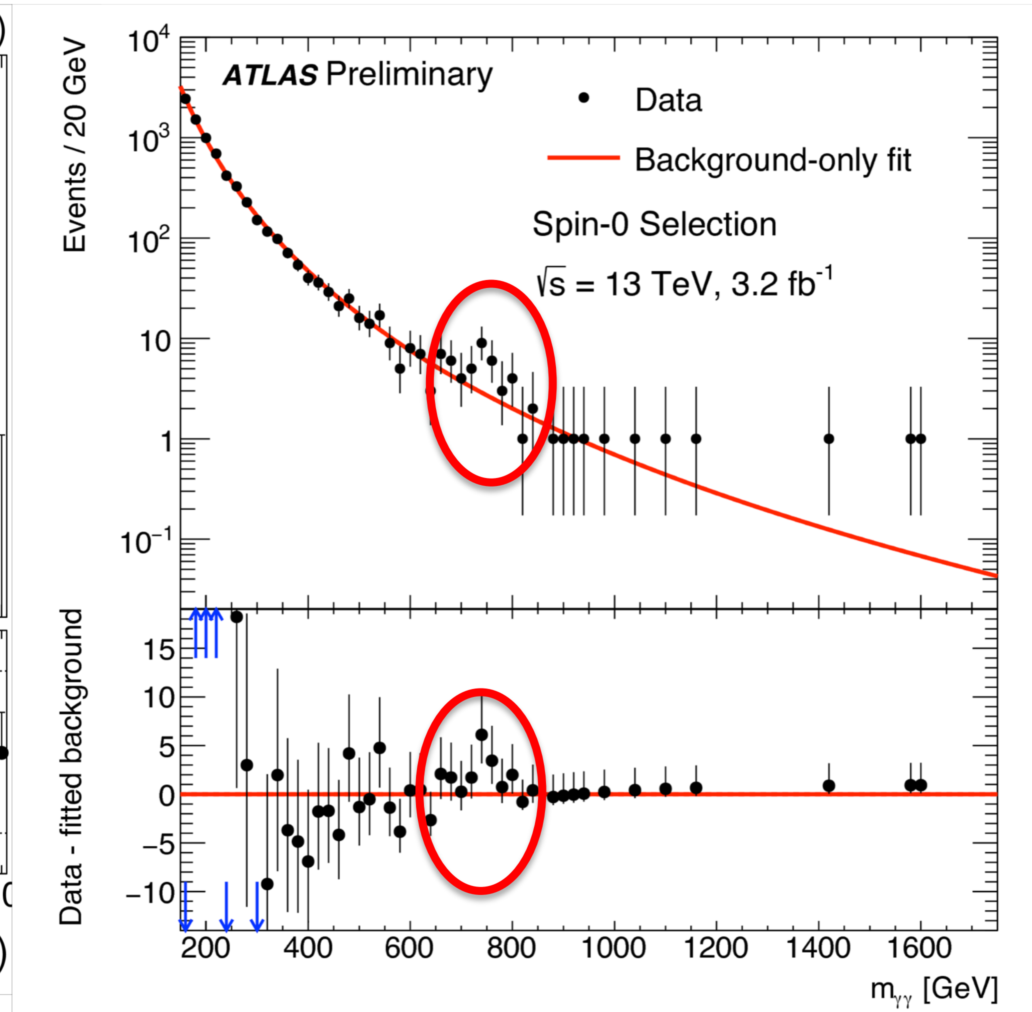
b(x) = pdf for background

The sum is over <u>events</u>

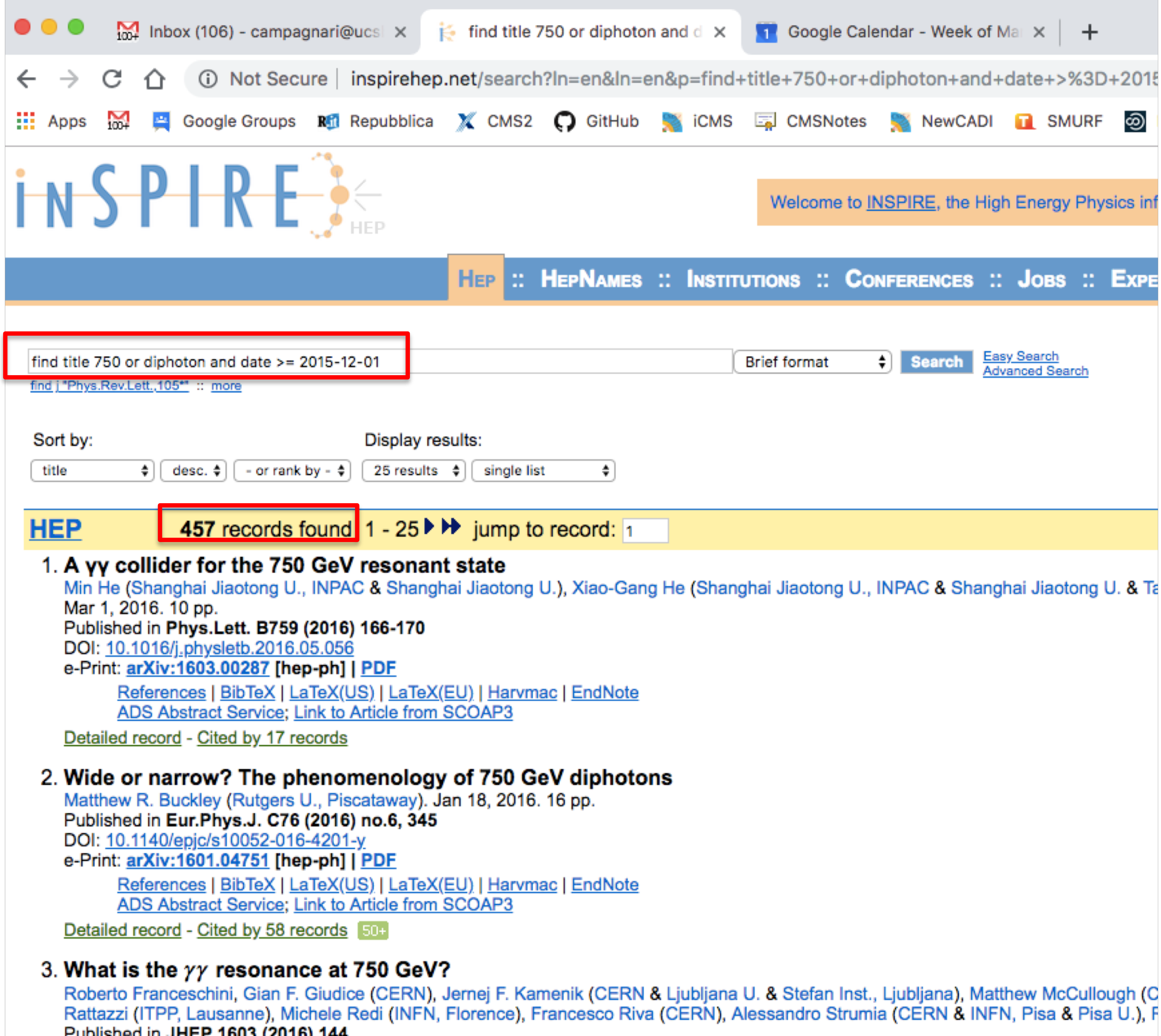
You will doing an unbinned fit in a future homework

A surprise: diphoton mass bump near 750 GeV?





Data from 1st higher energy LHC run (May-Nov 2015) presented in mid-December 2015


Inbox (106) - campagnari@ucs | find title 750 or diphoton and d | Google Calendar - Week of Ma

Not Secure | inspirehep.net/search?ln=en&ln=en&p=find+title+750+or+diphoton+and+date+>%3D+2015

Apps | Google Groups | Repubblica | CMS2 | GitHub | iCMS | CMSNotes | NewCADI | SMURF

INSPIRE HEP

Welcome to INSPIRE, the High Energy Physics inf

HEP :: HEPNAMES :: INSTITUTIONS :: CONFERENCES :: JOBS :: EXPE

find title 750 or diphoton and date >= 2015-12-01

Brief format | Search | Easy Search | Advanced Search

find j "Phys.Rev.Lett.,105*" :: more

Sort by: title | desc. | - or rank by -

Display results: 25 results | single list

HEP **457 records found** 1 - 25 jump to record: 1

1. **A γγ collider for the 750 GeV resonant state**
Min He (Shanghai Jiaotong U., INPAC & Shanghai Jiaotong U.), Xiao-Gang He (Shanghai Jiaotong U., INPAC & Shanghai Jiaotong U. & Ta
Mar 1, 2016. 10 pp.
Published in **Phys.Lett. B759 (2016) 166-170**
DOI: 10.1016/j.physletb.2016.05.056
e-Print: **arXiv:1603.00287 [hep-ph]** | **PDF**
References | BibTeX | LaTeX(US) | LaTeX(EU) | Harvmac | EndNote
ADS Abstract Service; Link to Article from SCOAP3
Detailed record - Cited by 17 records

2. **Wide or narrow? The phenomenology of 750 GeV diphotons**
Matthew R. Buckley (Rutgers U., Piscataway). Jan 18, 2016. 16 pp.
Published in **Eur.Phys.J. C76 (2016) no.6, 345**
DOI: 10.1140/epjc/s10052-016-4201-y
e-Print: **arXiv:1601.04751 [hep-ph]** | **PDF**
References | BibTeX | LaTeX(US) | LaTeX(EU) | Harvmac | EndNote
ADS Abstract Service; Link to Article from SCOAP3
Detailed record - Cited by 58 records 50+

3. **What is the $\gamma\gamma$ resonance at 750 GeV?**
Roberto Franceschini, Gian F. Giudice (CERN), Jernej F. Kamenik (CERN & Ljubljana U. & Stefan Inst., Ljubljana), Matthew McCullough (C
Rattazzi (ITPP, Lausanne), Michele Redi (INFN, Florence), Francesco Riva (CERN), Alessandro Strumia (CERN & INFN, Pisa & Pisa U.), F
Published in JHEP 1603 (2016) 144

How the $\gamma\gamma$ Resonance Stole Christmas

Nathaniel Craig[♠], Patrick Draper[♠], Can Kilic[♦], and Scott Thomas[♣]

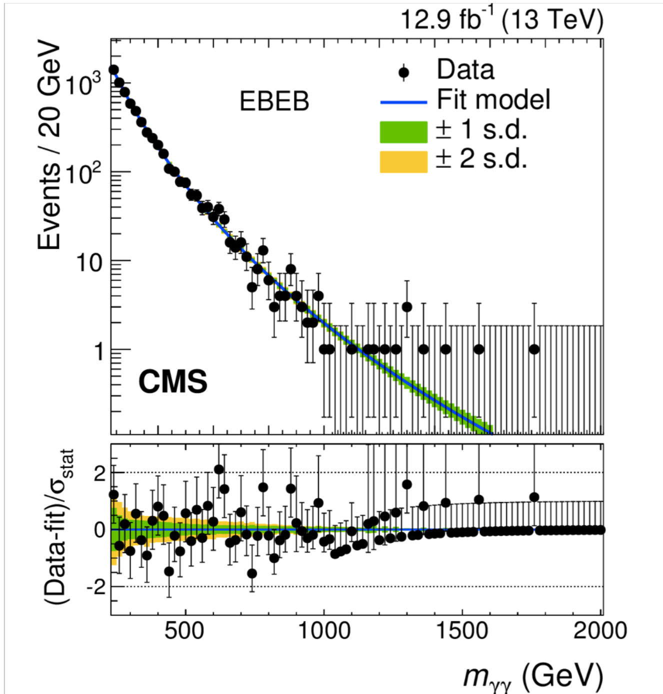
[♠] *Department of Physics, University of California, Santa Barbara, CA 93106, USA*

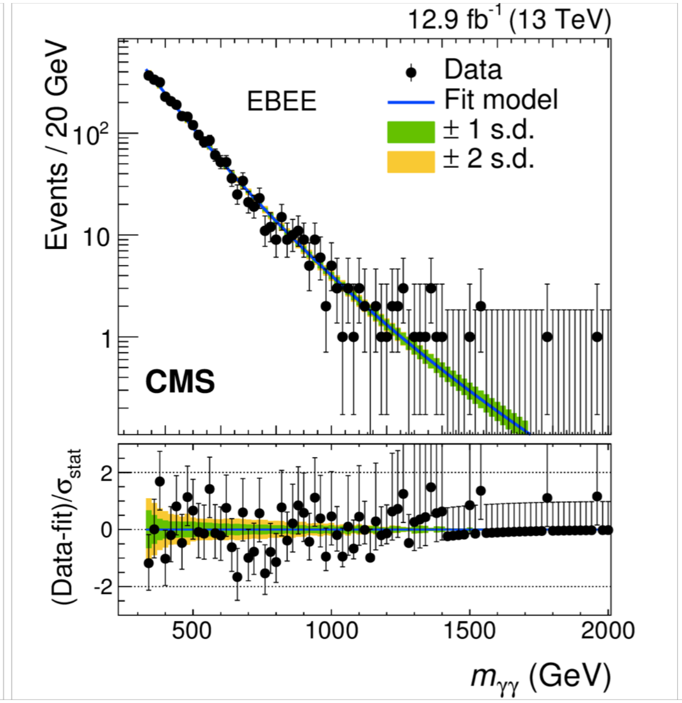
[♦] *Weinberg Theory Group, Department of Physics and Texas Cosmology Center, The University of Texas at Austin, Austin, TX 78712, USA*

[♣] *New High Energy Theory Center, Rutgers University, Piscataway, NJ 08854, USA*

Abstract

The experimental and theoretical implications of heavy di-gauge boson resonances that couple

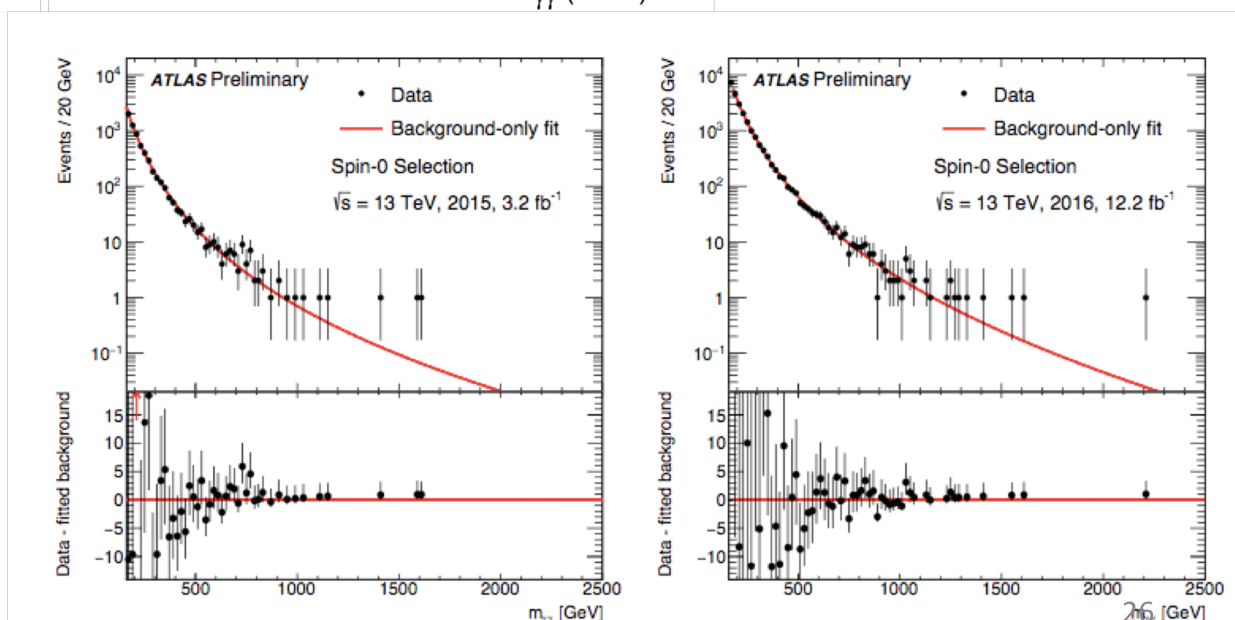




Updated results from August 2016.

~ 4 times more data

The peak is gone



APPENDIX

Technicality

- The uncertainty on S returned by the extended log likelihood fit is **no**t the uncertainty on the number of signal events contained in the particular dataset
- This fact can be understood with a simple example
  - Suppose B=0.
  - Only 1 bin
  - Let N = the number of events
  - Since B=0, S=N
  - There is no uncertainty whatsoever on the number of signal events contained in this dataset. I have N events, there is no background, so all of them are signal. Full stop.
- Let's see what the formalism actually gives us, see next page....

$$
\begin{aligned}
-\log\mathcal{L}(S,B) &= S + B - \sum d_i \log(S\cdot s_i + B\cdot b_i) \\
-\log\mathcal{L}(S,B=0) &= S - d_1\log(S\cdot s_1) \\
-\log\mathcal{L}(S) &= S - N\log S
\end{aligned}
$$

This is minimized for *S=N* as expected.

The change in NLL moving away from the minimum *S=N* by δN is

$$
\begin{aligned}
\Delta\mathrm{NLL} &= -\log\mathcal{L}(N+\delta N) + \log\mathcal{L}(N) \\
\Delta\mathrm{NLL} &= \delta N - N\log(1+\frac{\delta N}{N})
\end{aligned}
$$

Expanding the log for small δN/N and large N):

$$
\Delta\mathrm{NLL} \approx \delta N - N(\frac{\delta N}{N} - \frac{1}{2}(\frac{\delta N}{N})^2) = \frac{\delta N^2}{2N}
$$

Thus, the “1 sigma” uncertainty δN that one obtains
by setting ΔNLL= ½ is: $\delta N = \sigma = \sqrt{N}$

This is the usual counting uncertainty in the Gaussian regime that you can interpret in a frequentist or baysean sense, as you like.