ROOT and statistics tutorial

Solutions to exercises

Attilio Andreazza

Università di Milano and INFN

Caterina Doglioni

Université de Genève

Higgs->γγ

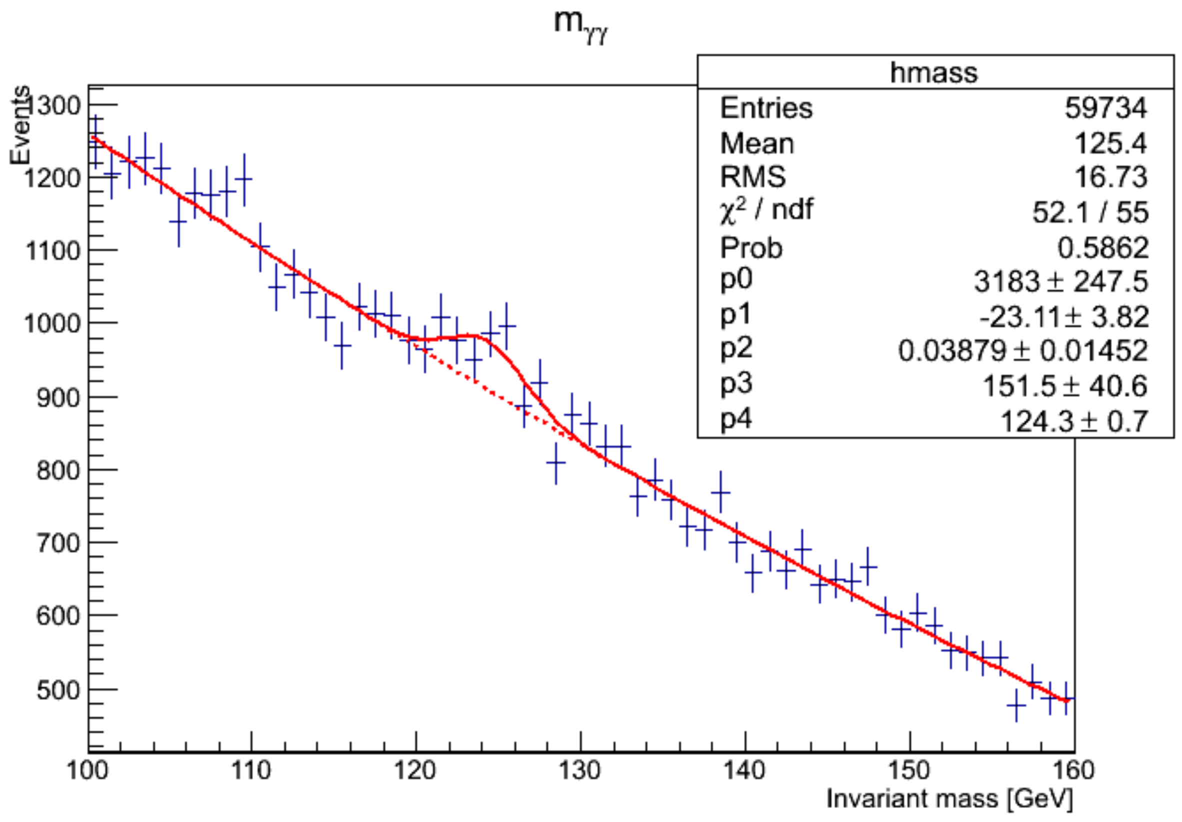
- The mass resolution is dominated by the sampling term and can be approximated by

$$\sigma_m / m = 0.205 / \sqrt{m[\mathrm{GeV}]}$$

  - Notice you can try a **`pol1`** fit, this will give a sensible approximation, because any differentiable function can be approximated by a straigth line, but the parametrization $\sigma_m = p_0 + mp_1$ is different from the one suggested in the exercise $\sigma_m = \sqrt{(Am)^2 + B^2 m + C^2}$. The p_0, p_1 have no defined physical meaning, instead A, B and C are related to calorimeter performance.
- The background has been generated using a 2-degree polynimial.
  - Using a 3-degree one does not worsen the fit, and is a viable solution, but it does not improve its quality either. Usually one prefers to use the smaller set of parameters, if the additional ones do not provide a good gain.
  - The macro **`readTree_background.C`** provides a solution without using a **`TFitResultPtr`**.
- Macros to perform the combined final fit and the signal strength plot are, respectively:
  - **`readTree_combined.C`**
  - **`readTree_strenght.C`**
  - ...see next pages for more details

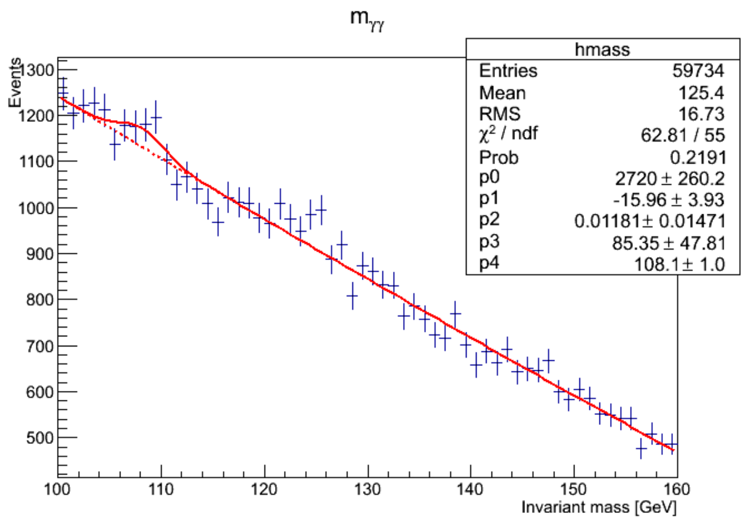
readTree_combined.C

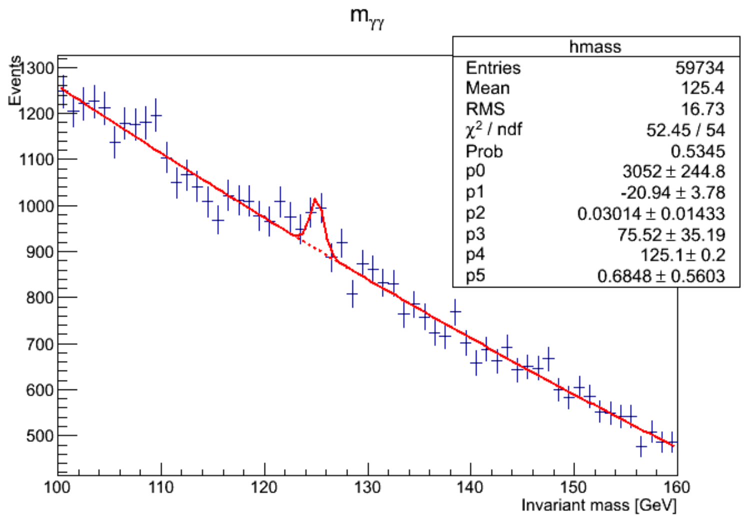
- The function to be used for the fit must have a form like:
  `"[0]+[1]*x+[2]*x**2+([3]/(0.205*sqrt([4])))*exp(-0.5*(x-[4])**2/(0.205*sqrt([4]))**2)"`
  - The approach in the macro is:
    1. to do an initial fit with a background only shape (resulting in a p-value of 0.19)
    2. To use these parameters as initial value for the combi-ned fit.
  - If setting an initial value of p_4 near 120, one gets the displayed fit, with a 3.7σ significance.
  - If setting an initial value neat 110, significance.



readTree_combined.C

- If setting an initial value neat 110, one gets a peak at m_H=108 GeV but with only 1.8σ significance. (top plot)
- If the Gaussian width is a free parameter (bottom plot).
  - Since data are used to determine one more parameter, that results in a higher statistical uncertainty on the fit: **the significance drops to a 2.1σ**.
  - The fitted width is unphysical: 700 MeV is much narrower than detector resolution: basically the fit tends to follow too closely the statistical fluctutations.
  - The rejection power against statistical fluctuations is reduces: the 180 GeV peak has low significance with the default approach because it has a "wrong" width.
  - **The Higgs is a small signal over a large background:** ***for observation it is critical all our knowledge is included in the fit.***

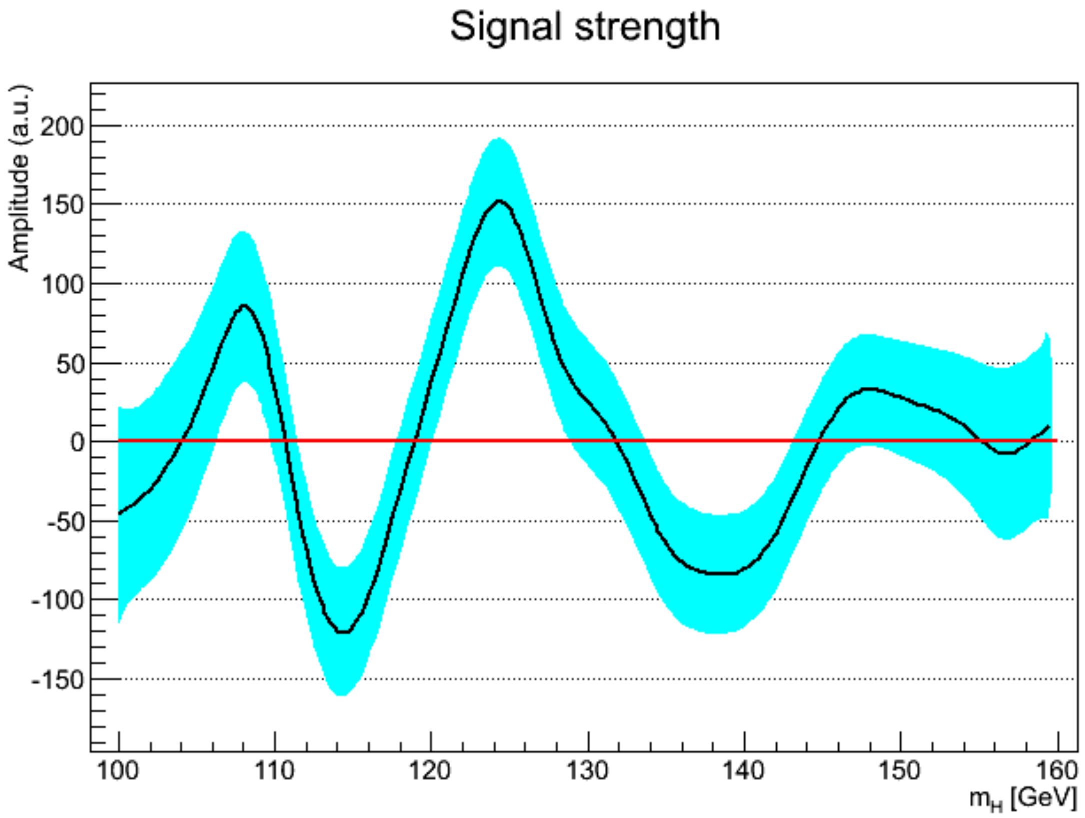




readTree_strength.C

- This macro simply repeat several times the fit, using the same function as **`readTree_combined.C`**, but fixing the mass value.

- The only free parameter of the Higgs peak is not the amplitude p_3.
- The amplitudes for different mass hypothesis are save into a **`TGraphErrors`** object.
- The amplitude may be either positive (excess of events), or negative (lack of events).
- Significances should be at the same order as from the previous fits.



- Look at the macro as an example for filling and drawing **`TGraphError`** objects.

Higgs->4ℓ

- An example macro performing all comuputations is **Solution.C**.
- This macro actually use an auxiliary library **CLCalculator.C** (but of course there are many ways)
- It contains two main functions:
  - **ProbabilitySampling**, to compute the distribution of $P\left(N \mid \langle N\rangle, \sigma_{\langle N\rangle}\right)$
  - **CL**, to compute the confidence level for rejecting a signal hypothesis (argument **exclusion** set to to **true**) or the bacground-only hypothesis (**exclusion** set to to **false**)
  - This splitting is useful for the optional part of the exercise.
- The solution is implemented in RooFit, in you want to use directly a TRandom object, you can replace ProbabilitySampling with the following function:

```
TH1* ProbabilitySampling(Double_t mu, Double_t smu=0) {
  TRandom  generator(0);
  Int_t nmax = 3*(mu+3*smu)+6; Int_t nEvents = 100000;
  TH1F* myh = new TH1F("myh","N distribution",nmax,-0.5,nmax-.5);
  for (Int_t i=0; i<nEvents; i++) {
    Double_t nu1 = generator.Gaus(mu,smu);
    myh->Fill(generator.Poisson(nu1));
  }
  return myh;
}
```

Confidence Levels for specific hypotheses

- When computing the CL of excluding the background only hypothesis the results are:
  - `2011 data 2 bkg events expected, 4 observed: CL=0.85689`
  - `2012 data 3 bkg events expected, 9 observed: CL=0.99605`
  - `Total data 5.1 bkg events expected, 13 observed: CL=0.99764`
- CL of rejection of Higgs hypothesis, in case the experiment would have seen no excess:
  - `2011 data 2 bkg+2 Higgs events expected, assuming 2 observed: CL=0.76264`
  - `2012 data 3 bkg+3 Higgs events expected, assuming 3 observed: CL=0.84785`
  - `Total data 5.1 bkg+5.3 Higgs events expected, assuming 5 observed: CL=0.94661`
- Adding the uncertainty on the background smears the pure Poisson distributions used before and results overall in a reduction of the CL.
- When computing the again the CL of excluding the background only hypothesis the results are:
  - `2011 data 2 bkg events expected, 4 observed: CL=0.8528`
  - `2012 data 3 bkg events expected, 9 observed: CL=0.99541`
  - `Total data 5.1 bkg events expected, 13 observed: CL=0.99574`
- CL of rejection of Higgs hypothesis (with systematics):
  - `2011 data 2 bkg+2 Higgs events expected, assuming 2 observed: CL=0.75011`
  - `2012 data 3 bkg+3 Higgs events expected, assuming 3 observed: CL=0.83075`
  - `Total data 5.1 bkg+5.3 Higgs events expected, assuming 5 observed: CL=0.92541`

Exclusion limits

- As mentioned in the initial text of the exercise this part is optional since it is much more complex in coding:
  - While before one could do an ad-hoc simulation for each value, now one needs to solve the equation

    $$P\left(N \leq N_{\mathrm{obs}} \mid N_{\mathrm{B}} + N_{\mathrm{S}}\right) = 1 - \alpha$$

    as a function of the variable N_S.
- The solving can be done with any of the methods you know. In the proposed solution we embed the function into a global **`TF1`** object named **`ConfidenceLevelCalculator`** which has as parameters N_{obs}, and the uncertainty on N_B, and as variable N_B+N_S.
  - Embedding into a **`TF1`** allows using the **`GetX()`** method:
    `myTF1.GetX(y,xmin,xmax,precision)`
    which solves the equation myTF1(x)=y in the interval [x_{min},x_{max}], within the required precision.
- Besides that there is some additional complication for the expected limits, since they requires to compute what N would correspond to the various fraction of the pdf integral.
- A possible implementation of the computation is in function **`ExpectedExclusionLimit`**
  ...hope that usage of standard vectors in C++ is known to everybody,

Exclusion limits

- The observed exclusion limits and the expected ones are given in the following printouts.
  For the expected limits the first number is the expected limit (center of the dashed line), while the first pair is the ±1σ expectation (green band) and the second pair is the ±2σ expectation (yellow band).

```
Using 2011 data
Excluded value at 95% CL: Ns>9.2
Expected limits. NS > 4.4  +5.8-2.8  +8.5-1.1
Using 2012 data
Excluded value at 95% CL: Ns>15.7
Expected limits. NS > 4.8  +7.6-1.8  +10.2-0.0
Using all data
Excluded value at 95% CL: Ns>20.0
Expected limits. NS > 5.6  +8.3-2.8  +12.0-0.0
```

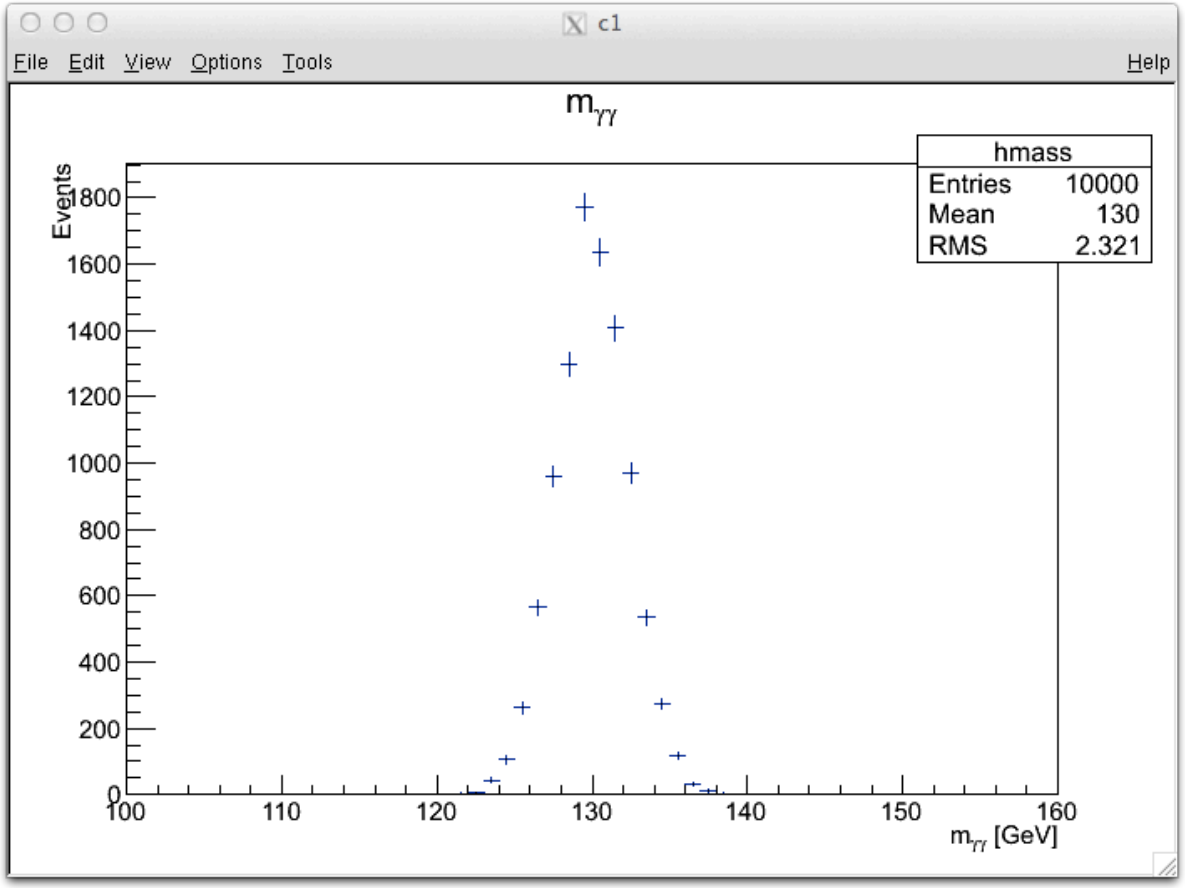
- To compare with the band point, you have to divide the resulting values by the expected signal. Results are qualitatively similar, but not exactly the same (especially for the ±2σ bands) because the real likelihood used by ATLAS is more complex than our simple example

More complex operations

- To do any more serious manipulation on the TTree content, one needs to use at least a macro.
- Among the dounloaded files threse is a small macro example: **readTree_basic.C**

- Execute this macro:
  root –l –x readTree_basic.C
  it will display the distribution of γγ invariant mass.

- **Let's see what's inside!**



The basic macro: part 1

```
void readTree_basic() {
  Char_t *filename = "Higgs130.root";
  // Retrieve the TTree
  TFile* myFile = TFile::Open(filename);
  TTree* tree = (TTree*)(myFile->Get("tree"));
  Double_t Et1, eta1, phi1, Et2, eta2, phi2;
  tree->SetBranchAddress("Et1" ,&Et1 );
  tree->SetBranchAddress("eta1",&eta1);
  tree->SetBranchAddress("phi1",&phi1);
  tree->SetBranchAddress("Et2" ,&Et2 );
  tree->SetBranchAddress("eta2",&eta2);
  tree->SetBranchAddress("phi2",&phi2);
  // Book histograms
  TH1F* hmass = new TH1F("hmass","m_{#gamma#gamma}",60,100.,160.);
  hmass->GetXaxis()->SetTitle("m_{#gamma#gamma} [GeV]");
  hmass->GetYaxis()->SetTitle("Events");
```

To executed automatically same name as file

File to read (it changes during the exercise)

In a macro need to retrieve the object properly (still using their names

This is where information read from file is stored

Book the histogram

Label the axes! (otherwise old professors complain)

The basic macro: part 2

```
// Loop over the events
  Long64_t events = tree->GetEntries();
  for (int i=0; i<events; i++) {
    tree->GetEntry(i);
    TLorentzVector g1,g2;
    g1.SetPtEtaPhiM(Et1,eta1,phi1,0.);
    g2.SetPtEtaPhiM(Et2,eta2,phi2,0.);
    TLorentzVector gg=g1+g2;
    hmass->Fill( gg.M() );
  }
  hmass->Draw("e");
}
```

Get the number of events in the TTree

Load the event i in memory

`TLorentzVector`
is a powerful class, implementing the whole Lorentz vector alegebras and many, many useful methods.

See all of them at:
`http://root.cern.ch/root/html/TLorentzVector.html`

Fitting is easy when using some predefined functions, like a Gaussian.
Add before:
`hmass->Fit(“gaus”);`
Get the resolution of $\gamma\gamma$ mass reconstrution.

Step 1: which is the resolution on $m_{\gamma\gamma}$?

- Compute the resolution on the mass peak for the following samples (the file name indicates the simulated Higgs boson mass):
  - `Higgs110.root`
  - `Higgs120.root`
  - `Higgs130.root`
  - `Higgs140.root`
- After the fit the resulting parameters can be retrived by the instructions:

```
hmass->GetFunction("gaus")->GetParameter(2);
hmass->GetFunction("gaus")->GetParError(2);
```

Function name

- **Provide a reasonable parameterization of the mass resolution as a function of the m_H.**

Hint

In this mass region the Higgs boson is a narrow resonance, so its observed mass is dominated by the experimental resolution.

$$m_H = \sqrt{2E_{\gamma 1}E_{\gamma 2}\left(1-\cos\theta_{\gamma 1,\gamma 2}\right)}$$

The main contribution is the uncertainty in energy reconstruction, which is usually parameterized as:

$$\frac{\sigma_E}{E} = A \oplus \frac{B}{\sqrt{E}} \oplus \frac{C}{E}$$

Calibration Sampling Noise

In our simulation one of these three terms will dominate.

Background model

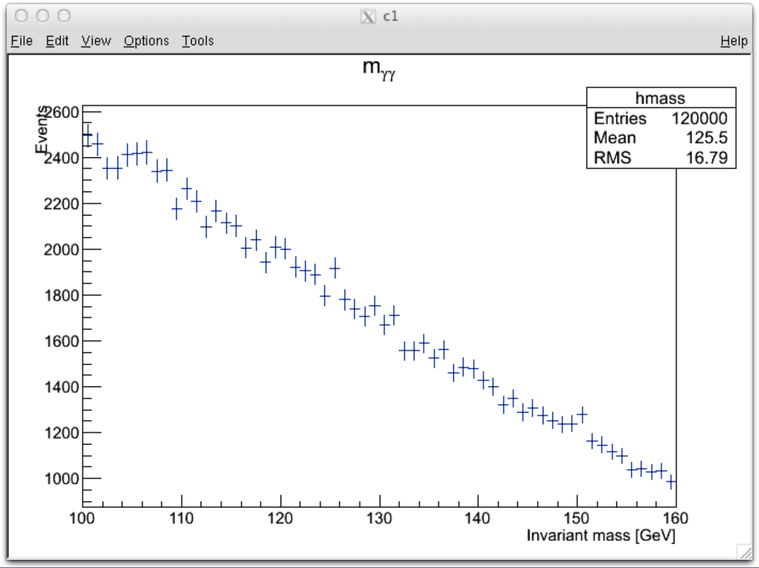
- Use the file: **Background.root**
- Content of this file are di-photons from background events: our aim is to define a model for their $m_{\gamma\gamma}$ distribution.
- Let's fit the resulting histograms with different functions:
  - Exponential $p_0*\exp(-m_{\gamma\gamma}/p_1)$
  - 1st, 2nd and 3rd degree polinomials
- Use a more general approach:

```
TF1* myF = new TF1("myF","[0]*exp(-x/[1])",100.,160.);
myF->SetParameter(0,1000.);
myF->SetParameter(1,30.);
hmass->Fit(myF);
```

Parameters

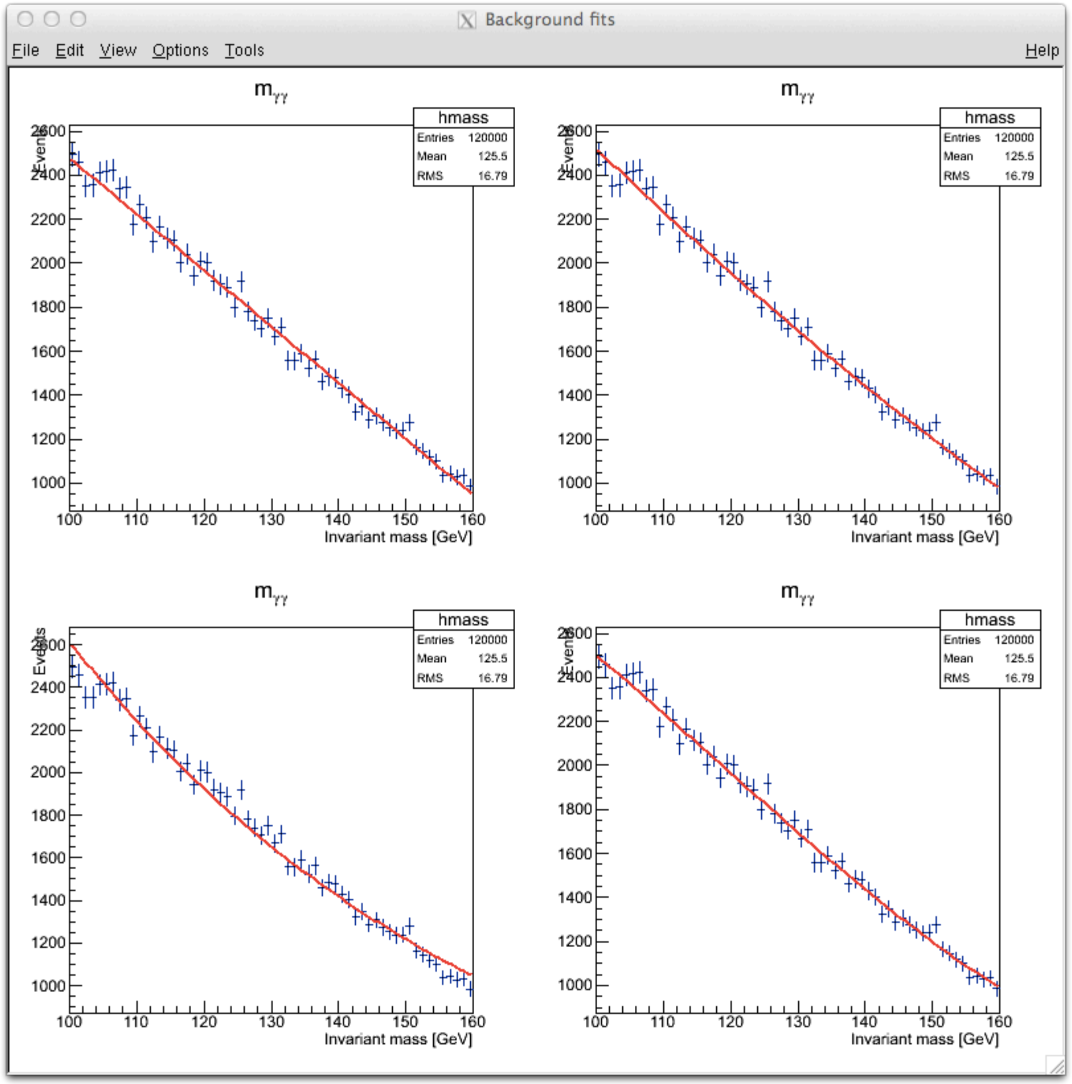
Need to set initial values for all parameters

Fit using a pointer to the TF1 fit function



```
hmass
Entries   120000
Mean       125.5
RMS        16.79
```

How to discriminate these models?



Hint

If wanting to have multiple plots of the same histogram, use the method `DrawClone` instead of `Draw`.

You need a TCanvas for each clone:

```
TCanvas *c1 = new TCanvas();
hmass->Fit(myF1);
hmass->DrawClone("e");
TCanvas *c2 = new TCanvas();
hmass->Fit(myF2);
hmass->DrawClone("e");
```

Or to divide a Tcanvas in zones:

```
TCanvas *c1 = new TCanvas();
c1->Divide(2,2);
c1->cd(1)
hmass->Fit(myF1);
hmass->DrawClone("e");
c1->cd(2);
hmass->Fit(myF2);
hmass->DrawClone("e");
```

Step 2: choose the background model.

- When invoked with option `"S"`, the `Fit` method gives back a pointer to a `TFitResult` object:
  `TFitResultPtr fit1 = hmass->Fit(myF1,"S");`

- This object has many accessors to the result and quality of the fit:
  - **`Double_t par = fit1->Parameter(0);`**
  - **`Double_t spar= fit1->ParError(0);`**
  - **`Double_t chi2= fit1->Chi2();`**
  - **`Int_t NDF = fit1->Ndf();`**

Retrieve the value of fitted parameter

Get its uncertainty

χ^2 and number of degrees of freedom of the fit

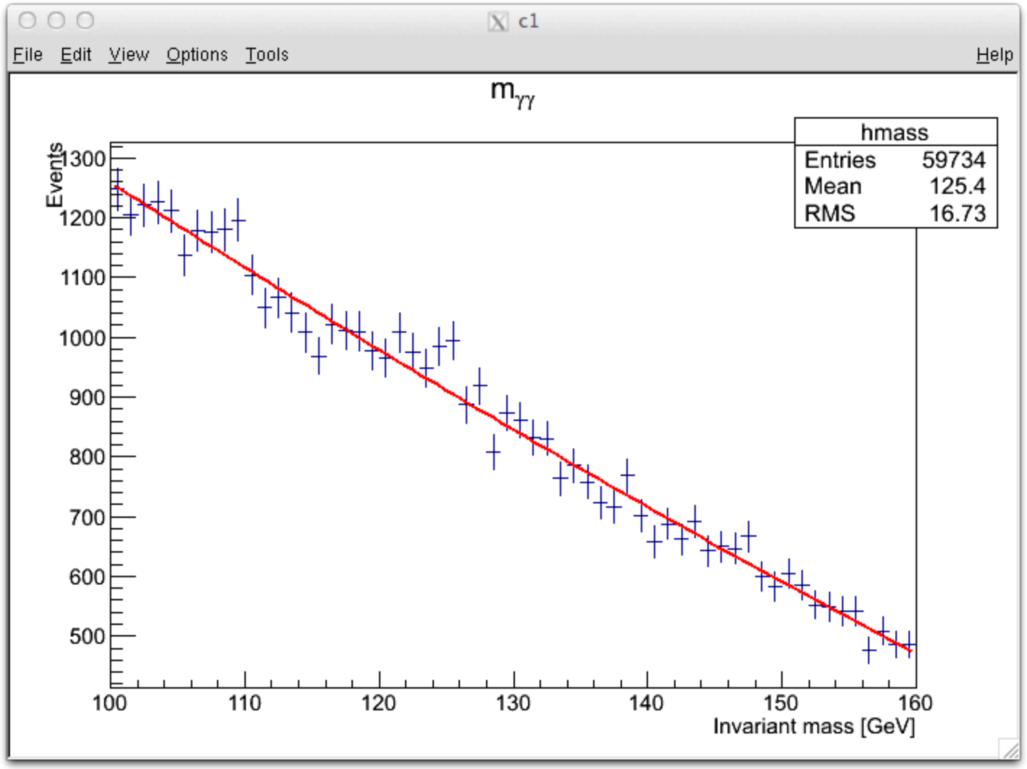
- The function
  `TMath::Prob(chi2,NDF)`
  provides the p-value for the χ^2 fit (probability that, if the disribution follows the given parameterization, the χ^2 will be worse than the observed one).
- **Which function will you choose?**

Rules of thumb

1. The p-value of the χ^2 fit must be reasonable (it cannot be perfect if we don't know the real functional form of the background.)
2. If there are parameters compatible with 0, it indicates they are not useful in improving quality of the data descripton.

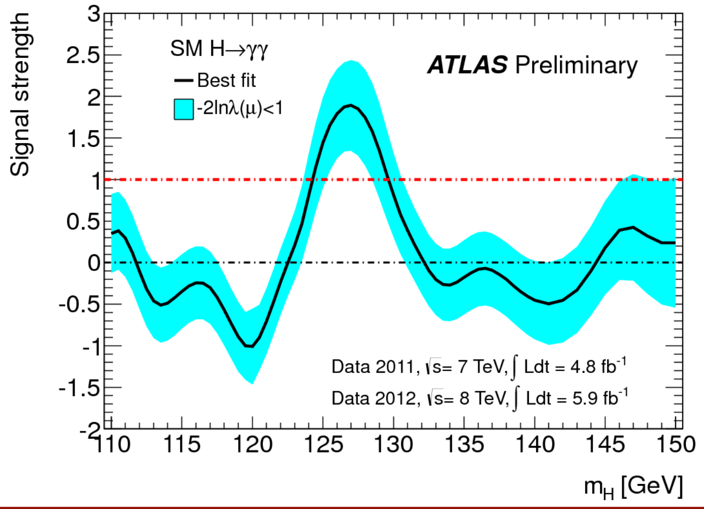
Opening the black box

- Use the file: **`Data.root`**
- Content of this file are di-photons with both signal and background:
  - But you do not know the mass, nor the amount of the signal (it may be zero).
- First, fit it just with the background model found in the previous step:
  - What's the value of the χ^2 and its p-value?
- Fit adding to your background a Gaussian whose width depends on m_H (from your initial study)
  - Be careful in setting initial values of the parameters!
  - To be more specific:
    - How much is the mass of the excess about 110 GeV?
    - How much is its significance?
      (as significance take the amplitude of the fitted Gaussian divided by its uncertainty)



The signal strength plot

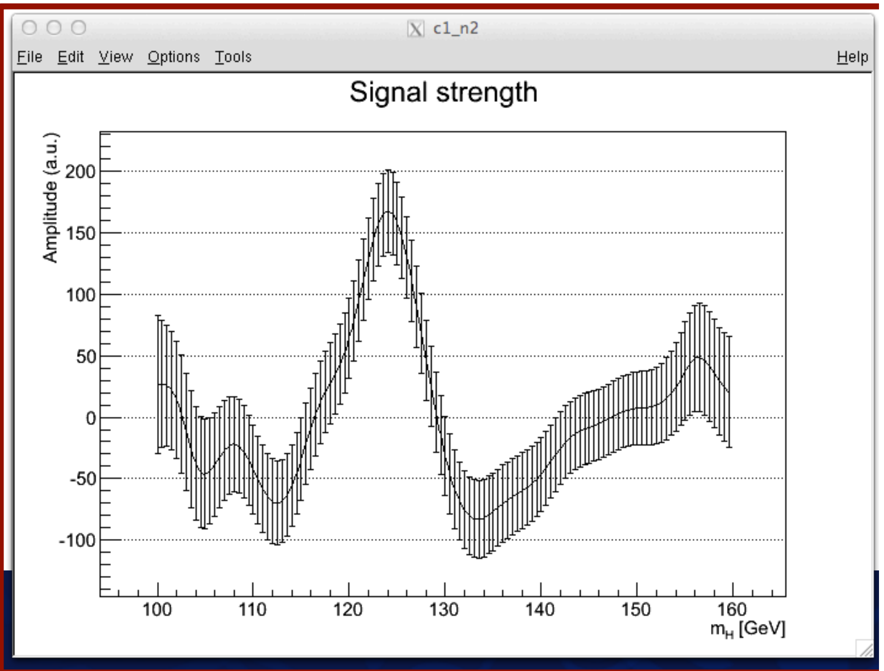
- A more systematic way to proceed is the so-called signal strength:
  - Fix a mass hypothesis
  - Fit the shape for that hypothesis
  - Look at the signal amplitude vs. mass hypothesis
  - *Normally divided for expected signal amplitude* (so that our model as strenght=1): we do not do it here.



```
TGraphErrors* myGraph = new TGraphErrors();
for (Int_t i=0; i<120; i++) {
  …
  myHiggs->FixParameter(4,mH);
  TFitResultPtr result = hmass->Fit(myHiggs,"SQ");
  myGraph->SetPoint(i,mH,result->Parameter(3));
  myGraph->SetPointError(i,0.,result->ParError(3));
  …
}
myGraph->Draw("ACP");
```

Fix mass parameter

Retrieve amplitude and its error



If using RooFit

- You can use the same approach, but before fitting you need to translate the **`TH1`** histogram with the mass distribution in a **`RooDataHist`** object.

- To fix a **`RooRealVar`** (you may need that in a loop):

```
Variable = value;
Variable.setConstant(true);
```