RooFit in Data Fitting

Jie Xiao

Overview

- ❑ **Maximum likelihood estimates**
- ❑ **Some Basic RooFit**
- ❑ **Simultaneous fit**
- ❑ **Physics parameter determination**

- Based on Fitting lessons in Prefit2020: https://github.com/amarini/Prefit2020
- Course from Cowan: http://www.pp.rhul.ac.uk/~cowan/stat_course.html

Maximum likelihood estimates

Parameter estimation

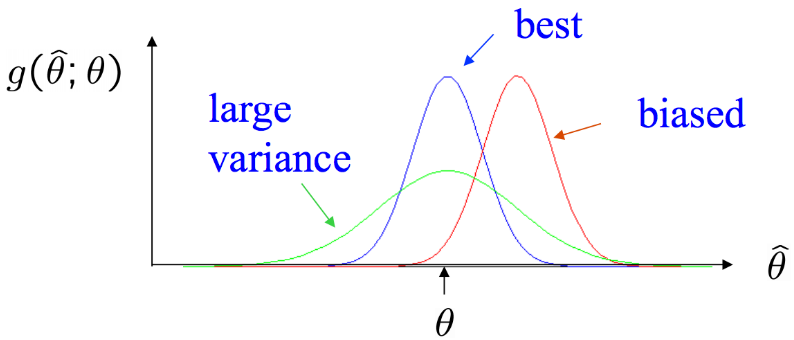
- The parameters of a pdf* are constants that characterize its shape, e.g.

$$f(x;\theta) = \frac{1}{\theta} e^{-x/\theta}, (\ x \ is\ random\ variable, \theta\ is\ parameter)$$

- If we have a sample of Observed values: $\vec{x} = (x_1, \ldots, x_n)$
- We want to find some function of the data to estimate the parameter(s)

  $$\hat{\theta}(\vec{x})$$

  estimator written with a hat.
- If we were to repeat the entire measurement, the estimates from each would follow a pdf



- We want small (or zero) bias (systematic error)
- And we want a small variance (statistical error)

* probability density function

Maximum likelihood estimators

- Suppose the entire result of an experiment (set of measurements) is a collection of numbers **x**, and suppose the joint pdf for the data **x** is a function that depends on a set of parameters **θ**:

$$f(\vec{x};\vec{\theta})$$

- Evaluate this function with the data obtained and regard it as a function of the parameter(s). This is the likelihood function:

$$L(\vec{\theta}) = f(\vec{x};\vec{\theta}) \quad (\boldsymbol{x} \text{ constant })$$

For independent and identically distributed data, the joint pdf for the whole data sample is:

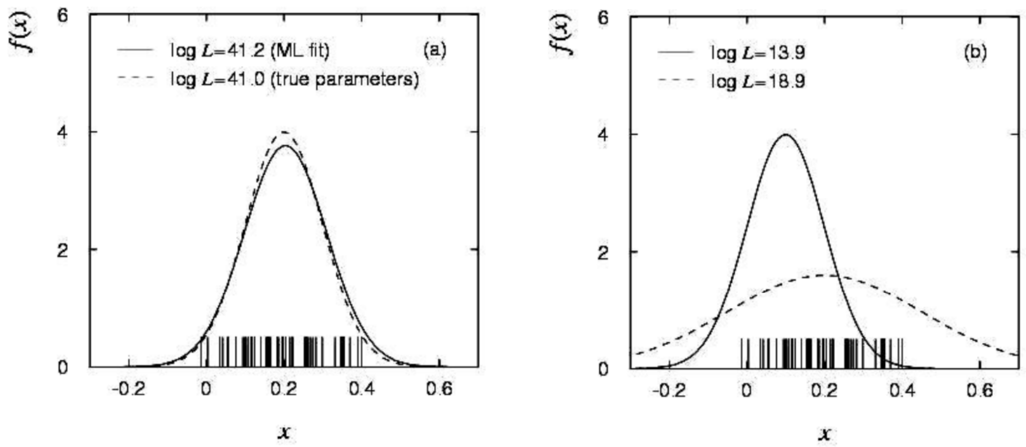
$$f(x_1, \ldots, x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)$$

In this case the likelihood function is

$$L(\vec{\theta}) = \prod_{i=1}^{n} f\left(x_i; \vec{\theta}\right) \; (x_i \text{ constant })$$

Maximum likelihood estimators

- If the hypothesized **θ** is close to the true value, then we expect a high probability to get data like that which we actually found. So we define the maximum likelihood (ML) estimator(s) to be the parameter value(s) for which the likelihood is maximum.



- Often we want to separate parameters which are physics parameters of interest $(\mathrm{POI} = \vec{\mu}\,)$ vs uninteresting parameters $(\,\mathrm{NP} = \vec{\theta}\,)$
- Typically (though certainly not always!) the nuisance parameters are constrained by some external measurements (e.g. Jet energy scales) - we introduce **constraint terms**

$$\pi\left(\vec{\theta}_0|\vec{\theta}\right) \sim p\left(\vec{\theta}|\vec{\theta}_0\right)$$

π is the probability to observe that outcome some value of the NPs

- So then we have

$$L(\vec{\mu},\vec{\theta}) \sim f(\vec{x}|\vec{\mu},\vec{\theta}) \cdot \pi\left(\vec{\theta}_0|\vec{\theta}\right)$$

Marginalization/Profiling

- In general, we would also have many more than 1 nuisance parameter (usually, there is one per source of systematic uncertainty). In these cases, reporting the N-dim likelihood is not feasible and not interesting.

- Instead, we tend to remove the nuisance parameters from the likelihood by one of two methods

Marginalisation or Profiling

- The two are often synonymous with **Bayesian** vs **Frequentist** methods
  - We won't go into the long debate about which is better/worse/right/wrong etc., but rather just review the techniques we use in combine which are one or the other....

Marginalization

- Recall Bayes' theorem

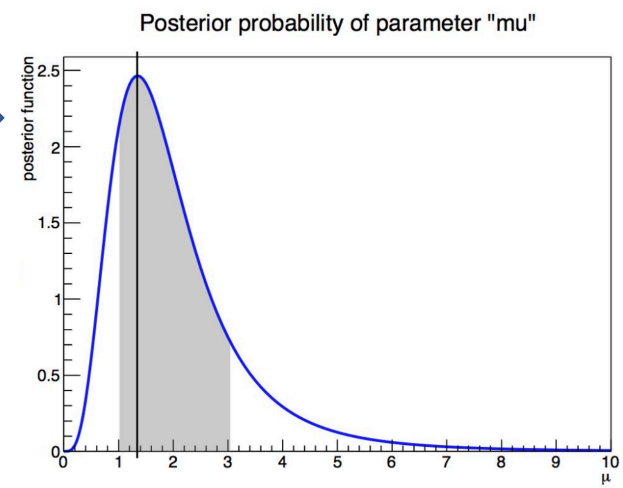
$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

- For our purposes, we can write this as

$$p(\mu|\vec{x}) = \int \frac{\boxed{p(\vec{x}|\mu,\theta)p(\theta)}p(\mu)}{p(\vec{x})} d\theta$$

$p(\mu)$ is known as "prior", $p(\mu|\vec{x})$ is posterior probability

- For cut-and-count analysis, the posterior might look like



- The value of mu which is the most common is 1.333

- We could also ask which region contains 68 % of the posterior distribution. This is known as a 68% **credible interval**

Profiling

- The other common method to remove nuisance parameters from the likelihood function is to find the value θ for which **maximizes** the likelihood at each value of μ. This is known as **profiling** over the nuisance parameters.

$$\mathcal{L}(\mu, \theta) \to \mathcal{L}(\mu, \hat{\theta}(\mu)) := \max_{\theta} \mathcal{L}(\mu, \theta)$$

  Or dropping, implicit dependencies,

$$\mathcal{L}(\mu, \theta) \to \mathcal{L}(\mu)$$

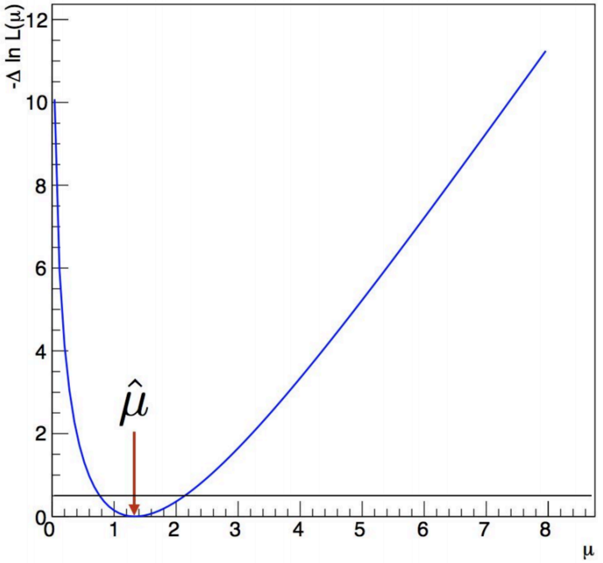
- Very often, to avoid dealing with small or large values of likelihoods, we take negative logs of the likelihood -> maximum likelihood = **minimum Negative Log Likelihood**

- We often subtracting the value at this minimum:

$$-\ln \mathcal{L}(\mu) \to -\ln \mathcal{L}(\mu) - (-\mathcal{L}(\hat{\mu})) = -\Delta \ln \mathcal{L}(\mu)$$

- **Wilkes' theorem** tells us that we can obtain.a 68 % **confidence interval** from the region for which:

$$-\Delta \ln \mathcal{L}(\mu) < 0.5$$



Extended Maximum Likelihood

- Sometimes regard **n** not as fixed, but as a **Poisson** r.v., mean **ν**.
  The (extended) likelihood function is:

$$L(\nu,\vec{\theta}) = \frac{\nu^n}{n!}e^{-\nu} \prod_{i=1}^{n} f(x_i;\vec{\theta})$$

- Here is an example: Consider two types of events (e.g., signal and background) each of which predict a given pdf for the variable x: $f_s(x)$ and $f_b(x)$.
  Let $\mu_\mathrm{S} = \theta\nu, \mu_\mathrm{b} = (1-\theta)\nu$, goal is to estimate μ_s, μ_b.

$$f(x;\mu_\mathrm{s},\mu_\mathrm{b}) = \frac{\mu_\mathrm{s}}{\mu_\mathrm{s}+\mu_\mathrm{b}} f_\mathrm{s}(x) + \frac{\mu_\mathrm{b}}{\mu_\mathrm{s}+\mu_\mathrm{b}} f_\mathrm{b}(x)$$

$$P(n;\mu_\mathrm{s},\mu_\mathrm{b}) = \frac{(\mu_\mathrm{s}+\mu_\mathrm{b})^n}{n!} e^{-(\mu_\mathrm{s}+\mu_\mathrm{b})}$$

$$\rightarrow \ln L(\mu_\mathrm{s},\mu_\mathrm{b}) = -(\mu_\mathrm{s}+\mu_\mathrm{b}) + \sum_{i=1}^{n} \ln\left[(\mu_\mathrm{s}+\mu_\mathrm{b}) f(x_i;\mu_\mathrm{s},\mu_\mathrm{b})\right]$$

Some Basic RooFit

RooFit introduction

- **RooFit** is an OO (Object-Oriented) analysis environment built on ROOT.
- In Roofit, any variable, data point, function, PDF (etc.) is represented by a c++ object

Model the distribution of observables **x** in terms of

- Physical parameters of interest **p**
- Other parameters **q** to describe detector effects (resolution,efficiency,...)

↓ **RooFit**

Probability density function **F(x;p,q)**

- normalized over allowed range of the observables **x** w.r.t the parameters **p** and **q**

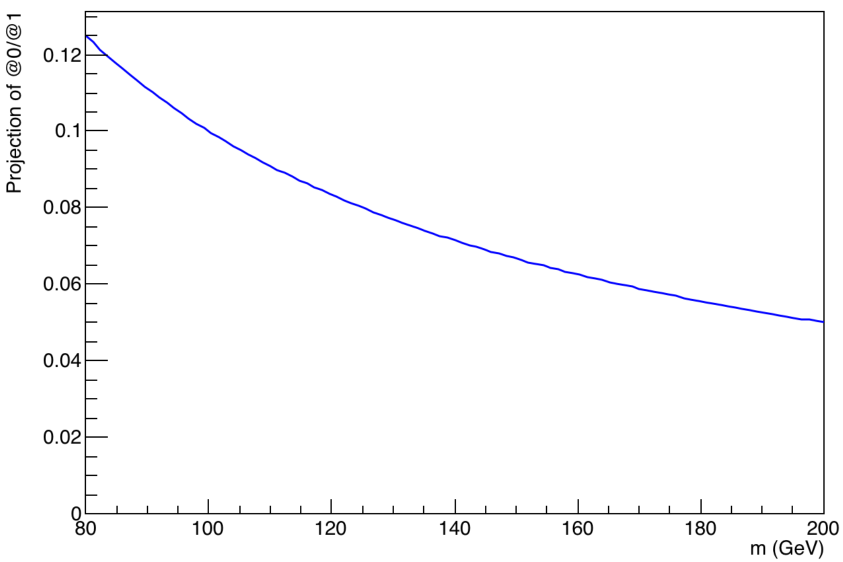
Class in RooFit

- **RooRealVar**: To define most basic variables

```
//Reconstruct the decay products of the H–boson
RooRealVar mass("m","m (GeV)",100,80,200);
//Assume the resolution of the invariant mass is 10 GeV
RooRealVar sigma("resolution","#sigma",10,0,20);
```

- **RooFormulaVar**: Construct more exotic variables out of **RooRealVar**s

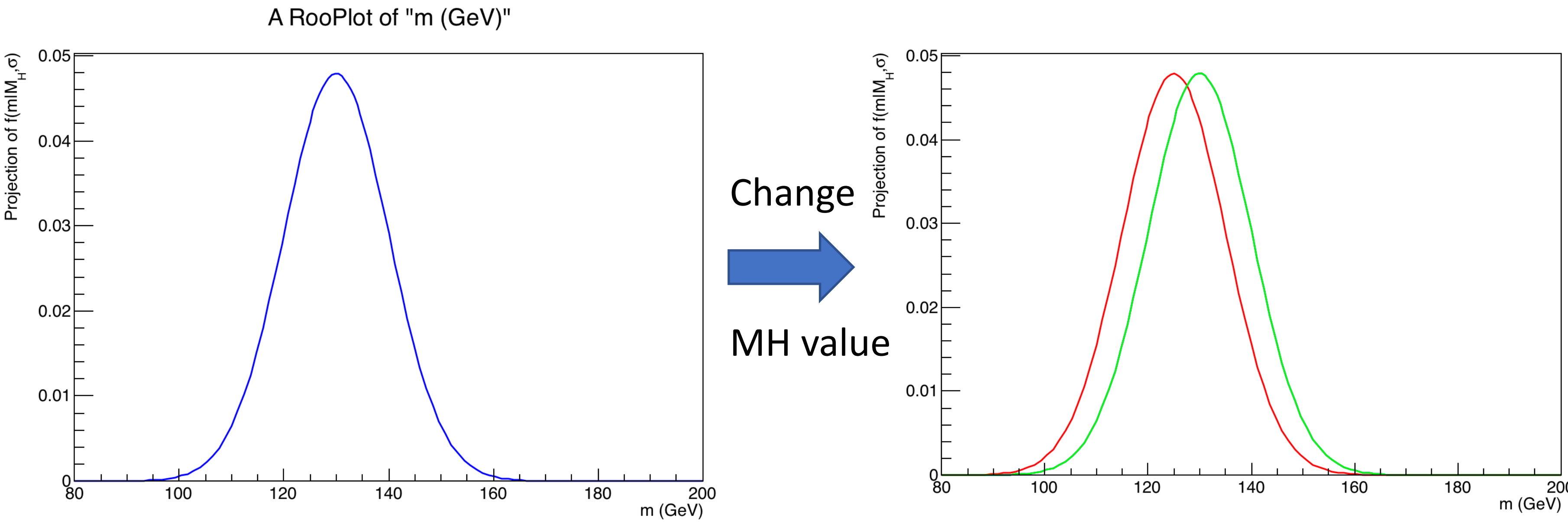
```
//Make a function which represented the relative resolution
RooFormulaVar func("R","@0/@1",RooArgList(sigma,mass));
```



Class in RooFit

- The main objects we are interested in using from RooFit are **"probability denisty functions"** or (PDFs)*.

```
RooRealVar MH("MH","mass of the Hypothetical Boson (H-boson) in GeV",125,120,130);
//A simple Gaussian shape for example in RooFit language
RooGaussian gauss("gauss","f(m|M_{H},#sigma)",mass,MH,sigma);
```

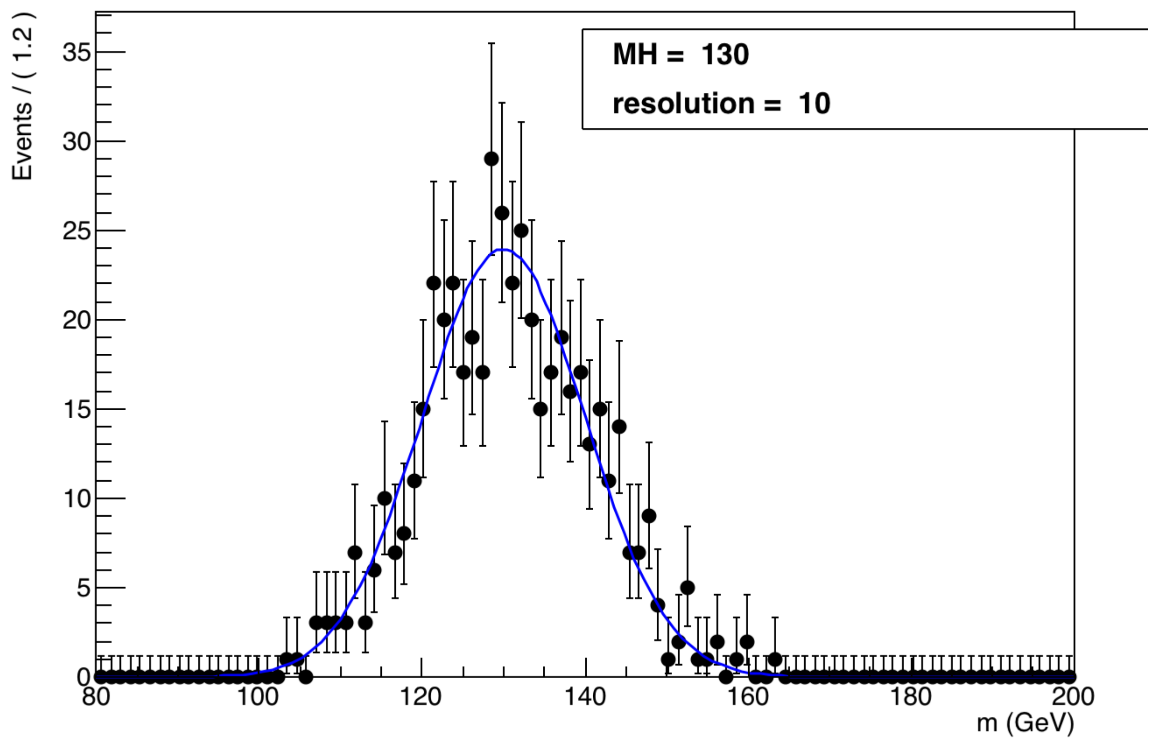


*There is also support for a generic pdf in the form of a RooGenericPdf:
https://root.cern.ch/doc/v608/classRooGenericPdf.html

Class in RooFit

- Toy-generation: PDFs can be used to generate Monte Carlo data

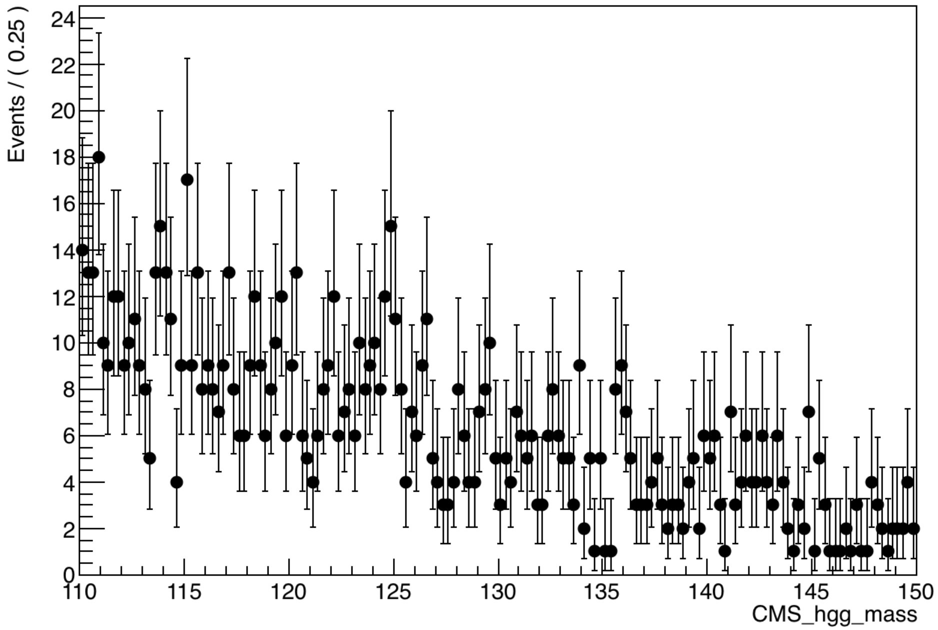
```
//The arguments are the set of observables, follwed by the number of events
RooDataSet *data = (RooDataSet*) gauss.generate(RooArgSet(mass),500);
```



Class in RooFit

- **RooWorkspace*:** keeping a persistent link between the objects for a model
  A useful way to share data and PDFs/functions etc among CMS collaborators

```
TFile *file = TFile::Open("tutorial.root");
RooWorkspace *wspace = (RooWorkspace*) file->Get("workspace");
// This workspace contains one RooDataSet and one RooRealVar
RooDataSet *hgg_data = (RooDataSet*) wspace->data("dataset");
RooRealVar *hgg_mass = (RooRealVar*) wspace->var("CMS_hgg_mass");
```

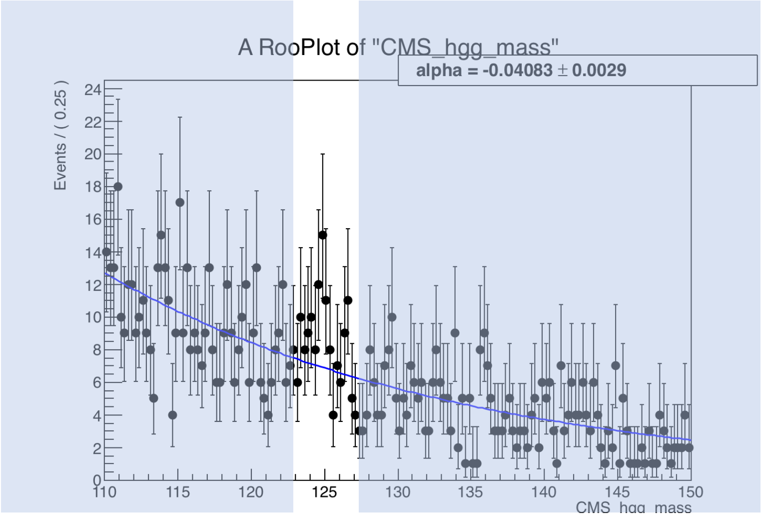


*The root file is here: https://github.com/amarini/Prefit2020/

Likelihoods and Fitting to data

- **Maximum likelihood estimator** is common in HEP and is known to give unbiased estimates for things like distribution means etc.
- It's also common to multiply this by -1 and minimize the resulting **N**egative **L**og **L**ikelihood

```
RooRealVar alpha("alpha","#alpha",-0.05,-0.2,0.01);
RooExponential expo("exp","exponential function",*hgg_mass,alpha);
//RooFit can construct the NLL for us.
RooNLLVar *nll = (RooNLLVar*) expo.createNLL(*hgg_data);
//To minimize nll
RooMinimizer minim(*nll);
minim.minimize("Minuit2","migrad");
```



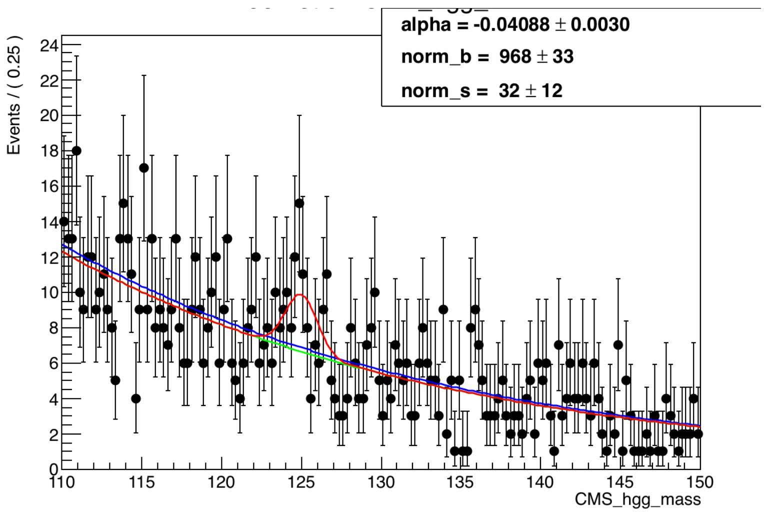
- There is a small region near 125 GeV for which our fit doesn't quite go through the points.

Likelihoods and Fitting to data

- Create Signal+Background model

$$\ln L(\mu_{\rm s}, \mu_{\rm b}) = -(\mu_{\rm s}+\mu_{\rm b}) + \sum_{i=1}^{n} \ln\left[(\mu_{\rm s}+\mu_{\rm b}) f(x_i; \mu_{\rm s}, \mu_{\rm b})\right]$$

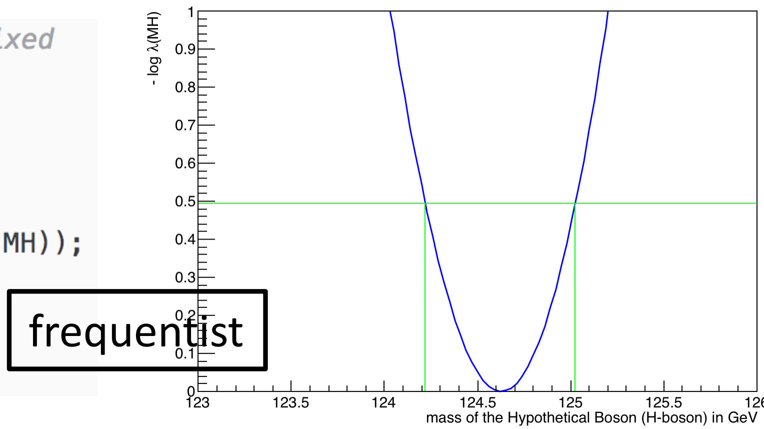
```
// Signal pdf
RooGaussian hgg_signal("signal","Gaussian PDF",*hgg_mass,MH,sigma);
//fraction of yields for the signal and background
//or absolutely where each PDF has its own normalisation
RooRealVar norm_s("norm_s","N_{s}",10,100);
RooRealVar norm_b("norm_b","N_{b}",0,1000);
const RooArgList components(hgg_signal,expo);
const RooArgList coeffs(norm_s,norm_b);
//"Signal+Background model" by creating a RooAddPdf
RooAddPdf model("model","f_{s+b}",components,coeffs);
//fit to the overall number of observed events: Extended()
model.fitTo(*hgg_data,RooFit::Extended());
```



Likelihoods and Fitting to data

- **Nuisance Parameters**
  - In HEP, there are some parameters POI, while others are known as nuisance parameters.
- There are two schools of thought for removing nuisance parameters
  - Frequentists use profiling
  - Bayesians use marginalization

```
//For simplicity, we will assume that the values of α and N_b are fixed
alpha.setConstant(true);
norm_b.setConstant(true);
MH.setRange(123,126);
//Class for profile likelihoods: ProfileLikelihoodCalculator
RooStats::ProfileLikelihoodCalculator plc(*hgg_data,model,RooArgSet(MH));
//According Wilks' theorem, find 1σ interval
plc.SetConfidenceLevel(0.68);
RooStats::LikelihoodInterval *interval = plc.GetInterval();
```



- Two methods agree very well in this case

  68% (frequentist) interval = (124.22,125.026)

  68% (Bayes shortest) interval = (124.176,125.031)

Simultaneous fit

Simultaneous fit

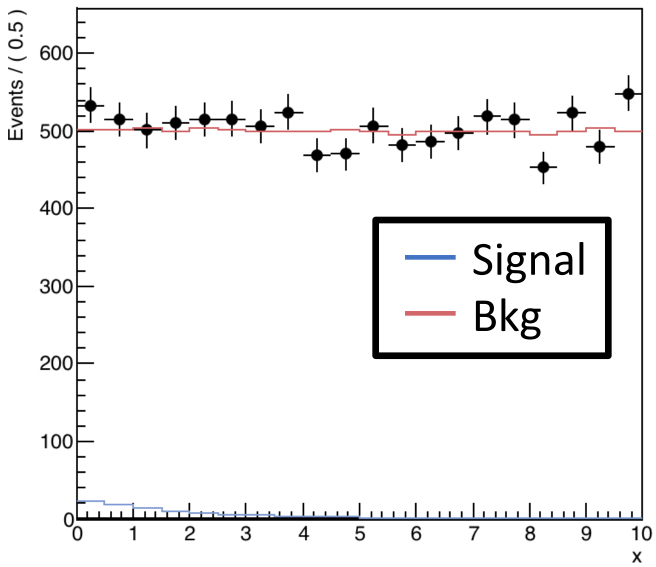
❑ Pseudo data preparation

- Generate background and signal pdf

```
# Generate toys, bkg is uniform, signal is exponential
pdf_true = [ROOT.RooUniform("b","b",ROOT.RooArgSet(x)),ROOT.RooExponential("s","s",x,t)]
n_true = [10000.,100.]
ds_obs = [pdf.generate(ROOT.RooArgSet(x),n) for pdf,n in zip(pdf_true,n_obs)]
```

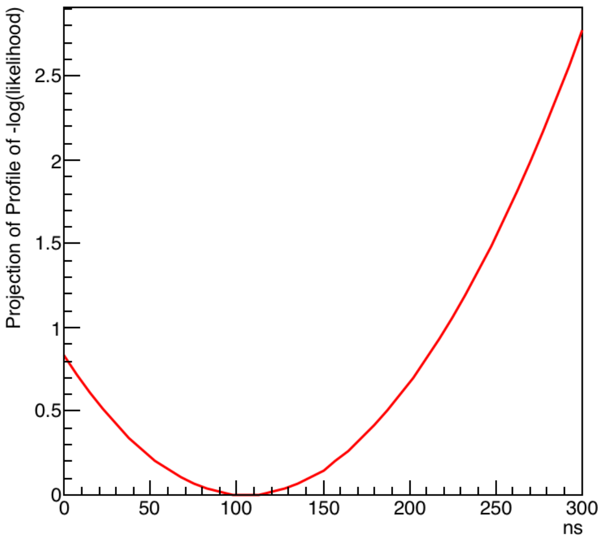
- Also generate signal region and control region

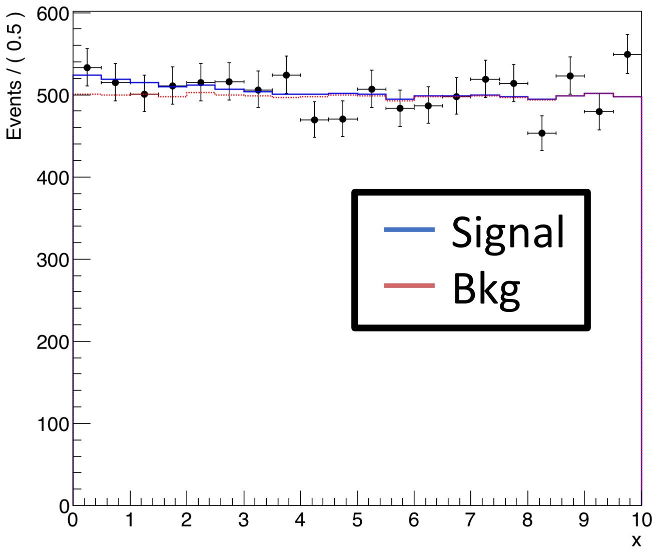
```
# signal region
sr=ROOT.RooAddPdf("data_true","data_true",\
    ROOT.RooArgList(pdf_true[0],pdf_true[1]), ROOT.RooArgList(f))
n_sr=n_true[0]+n_true[1]*1.2
# control region
cr=pdf_true[0]
n_cr=n_true[0]*5
```



Simultaneous fit

- **Fit to signal region to extract parameters**
- Fit to the number of signal events





- Best fit: ns= 105.0028

Simultaneous fit

- **Simultaneous pdfs**
- Include as many channel as we want by making the product of the likelihood functions

$$\mathscr{L} = \prod_i \mathscr{L}_i$$

```
#signal region model: bkg pdf: pdf[0]; sig pdf: pdf[1]
model_s=ROOT.RooAddPdf("model_s","model_s",\
    ROOT.RooArgList(pdf[0],pdf[1]), ROOT.RooArgList(rv_sig,rv_bkg))
```

Simultaneous fit

❑ Simultaneous pdfs

- Include as many channel as we want by making the product of the likelihood functions

$$\mathscr{L} = \prod_i \mathscr{L}_i$$

```
#signal region model: bkg pdf: pdf[0]; sig pdf: pdf[1]
model_s=ROOT.RooAddPdf("model_s","model_s",\
    ROOT.RooArgList(pdf[0],pdf[1]), ROOT.RooArgList(rv_sig,rv_bkg))
# construct a model that uses data2 in the second category
# and correlates the pdf and bkg yields among the two catgeories
dh_data2=ROOT.RooDataHist("dh_data2","data2 distribution",ROOT.RooArgList(x),hdata2)
rfv_bkg2=ROOT.RooFormulaVar("nbkg2","@0*5",ROOT.RooArgList(rv_bkg))
model2=ROOT.RooExtendPdf("model2","model2",pdf[1],rfv_bkg2)
```

Simultaneous fit

- **Simultaneous pdfs**
- Include as many channel as we want by making the product of the likelihood functions

$$\mathscr{L} = \prod_i \mathscr{L}_i$$

```
#signal region model: bkg pdf: pdf[0]; sig pdf: pdf[1]
model_s=ROOT.RooAddPdf("model_s","model_s",\
    ROOT.RooArgList(pdf[0],pdf[1]), ROOT.RooArgList(rv_sig,rv_bkg))
# construct a model that uses data2 in the second category
# and correlates the pdf and bkg yields among the two catgeories
dh_data2=ROOT.RooDataHist("dh_data2","data2 distribution",ROOT.RooArgList(x),hdata2)
rfv_bkg2=ROOT.RooFormulaVar("nbkg2","@0*5",ROOT.RooArgList(rv_bkg))
model2=ROOT.RooExtendPdf("model2","model2",pdf[1],rfv_bkg2)
# construct the model
cat = ROOT.RooCategory("cat", "cat")
cat.defineType("SR")
cat.defineType("CR")
combData = ROOT.RooDataHist("combData2","combData2", ROOT.RooArgList(x),
                            ROOT.RooFit.Index(cat),
                            ROOT.RooFit.Import("SR",dh[0]),
                            ROOT.RooFit.Import("CR",dh_data2)
)
```

Simultaneous fit

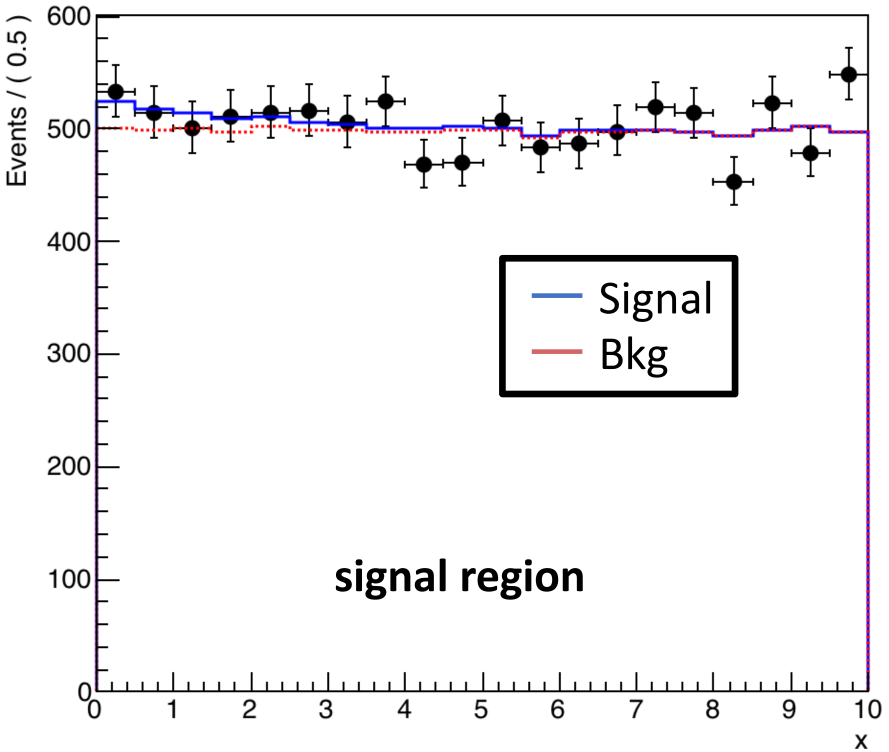
- **Simultaneous pdfs**
  - Include as many channel as we want by making the product of the likelihood functions

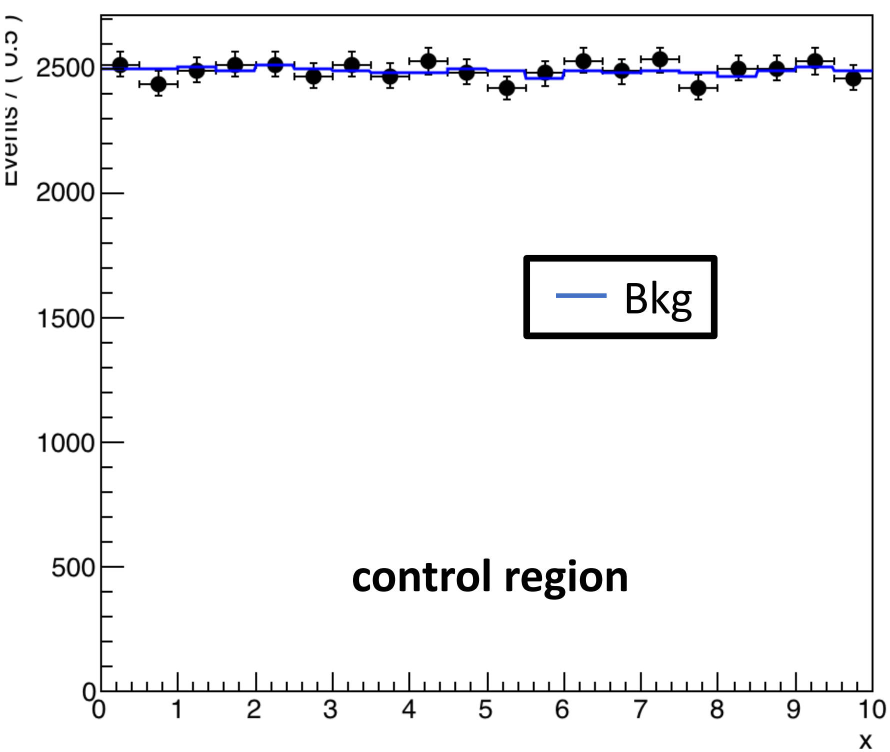
$$\mathscr{L} = \prod_i \mathscr{L}_i$$

```
#signal region model: bkg pdf: pdf[0]; sig pdf: pdf[1]
model_s=ROOT.RooAddPdf("model_s","model_s",\
    ROOT.RooArgList(pdf[0],pdf[1]), ROOT.RooArgList(rv_sig,rv_bkg))
# construct a model that uses data2 in the second category
# and correlates the pdf and bkg yields among the two catgeories
dh_data2=ROOT.RooDataHist("dh_data2","data2 distribution",ROOT.RooArgList(x),hdata2)
rfv_bkg2=ROOT.RooFormulaVar("nbkg2","@0*5",ROOT.RooArgList(rv_bkg))
model2=ROOT.RooExtendPdf("model2","model2",pdf[1],rfv_bkg2)
# construct the model
cat = ROOT.RooCategory("cat", "cat")
cat.defineType("SR")
cat.defineType("CR")
combData = ROOT.RooDataHist("combData2","combData2", ROOT.RooArgList(x),
                            ROOT.RooFit.Index(cat),
                            ROOT.RooFit.Import("SR",dh[0]),
                            ROOT.RooFit.Import("CR",dh_data2)
)
# Construct the simultaneous model
simPdf=ROOT.RooSimultaneous("simPdf","simultaneous pdf",cat) ;
simPdf.addPdf(model_s,"SR")
simPdf.addPdf(model2,"CR")
# Fit
fr=simPdf.fitTo(combData,ROOT.RooFit.Offset(True),
                ROOT.RooFit.Minimizer("Minuit2","migradimproved")
) # or construct nll
```

Simultaneous fit

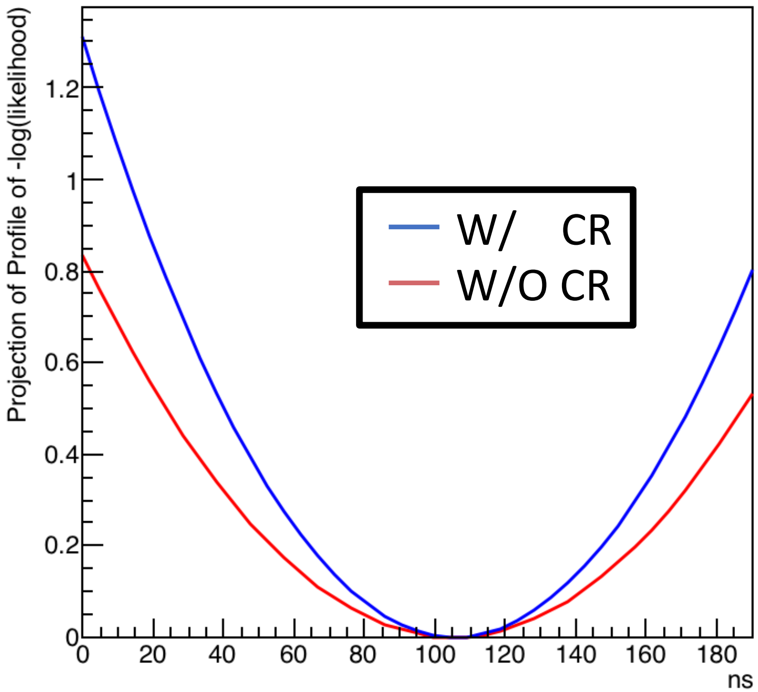
- Fit results for signal region and control region



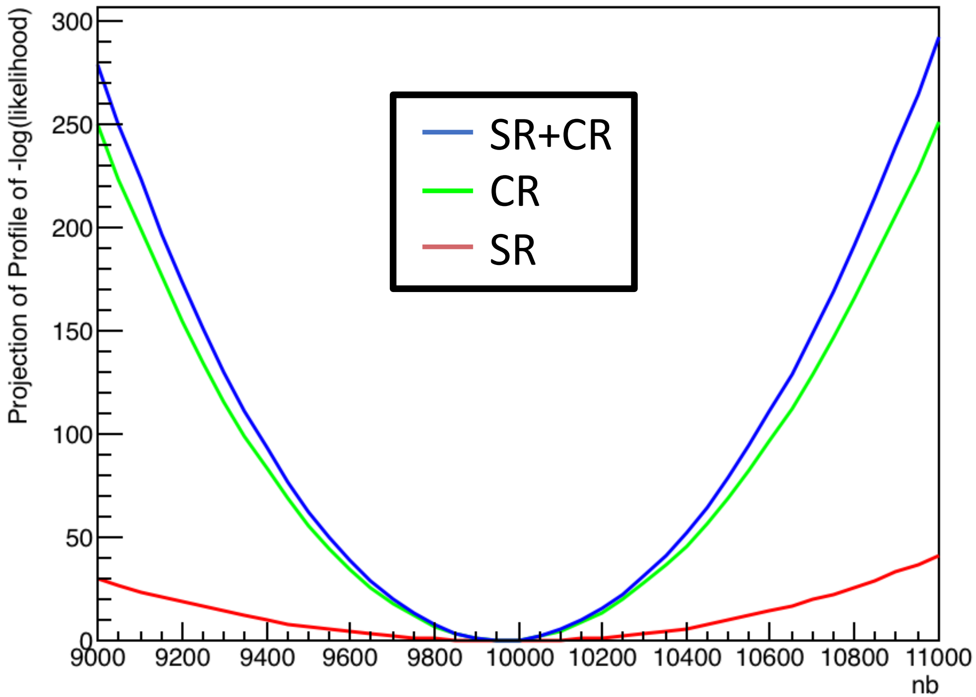


Simultaneous fit

- ❑ **Profile of –log(likelihood)**



Number of signal



Number of bkg

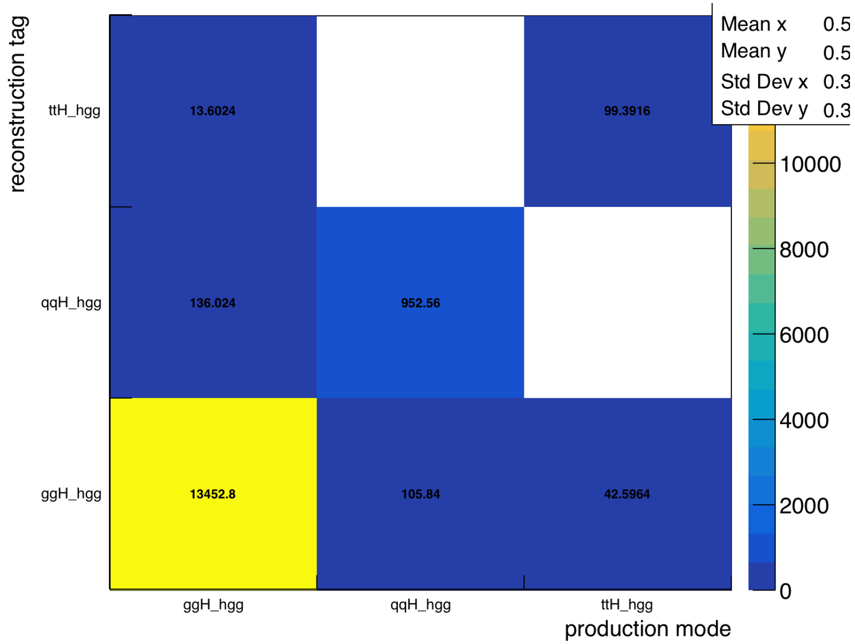
Physics parameter determination

Physics parameter determination

- ❑ Three categories* of data.

```
TFile**         inputs_session3.root
 TFile*         inputs_session3.root
  KEY: TH1F     data_ggH_hgg__mgg;1     Histogram of data_ggH_hgg__mgg
  KEY: TH1F     data_qqH_hgg__mgg;1     Histogram of data_qqH_hgg__mgg
  KEY: TH1F     data_ttH_hgg__mgg;1     Histogram of data_ttH_hgg__mgg
  KEY: TH2D     Response;1      sig
```

- ❑ The 'Response' shows migration between each category at recostruction level.



*The root file is here: https://github.com/amarini/Prefit2020/tree/master/Session%203

Physics parameter determination

- ❑ Background model is Exponential pdf
- ❑ Signal model is Gaussian pdf

```
##############################################
#            BACKGROUND MODEL               #
##############################################
par_b = [ROOT.RooRealVar("exp_1_%s"%t,"exp_parameter_%s"%t,-0.2,-1, -0.01) for t in tags ]
models_b = [ ROOT.RooExponential("b_%s"%t,"b_%s"%t,x,b) for t,b in zip(tags,par_b) ]
n_bkg =[ROOT.RooRealVar("nbkg_%s"%t,"nbkg_%s"%t,h.Integral(),\
    .5*h.Integral(),2.0*h.Integral()) for t,h in zip(tags,th1)]
##############################################
#            SIGNAL MODEL                   #
##############################################
## for simplicity the mean and sigma of the gaussians are all identical
mean=ROOT.RooRealVar("mean","mean",125)
sigma=ROOT.RooRealVar("sigma","sigma",1.5)
# can be different for each entry in the response matrix.
smodel=ROOT.RooGaussian("s_model_generic","smodel",x,mean,sigma)
```

Physics parameter determination

- Construct strength modifiers for the truth components

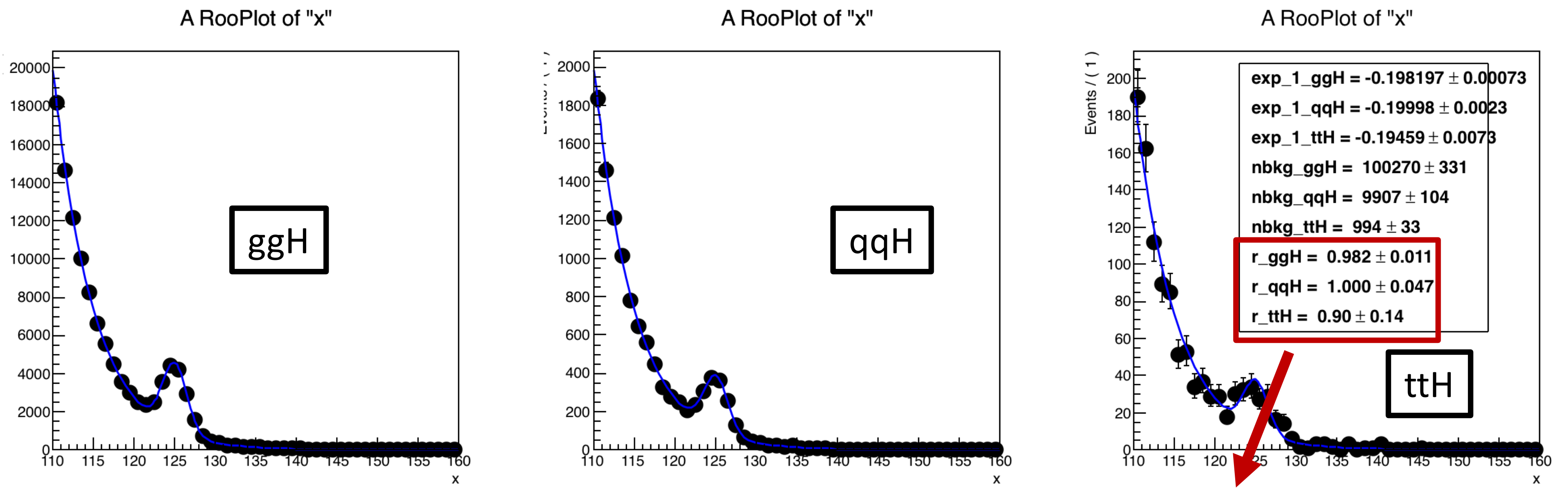
```
# construct strength modifiers for the truth components
r =[ROOT.RooRealVar("r_%s"%t,"r_%s"%t,1, 0.,5.0) for t in tags]
# contents from response matrix.
e =[ [ ROOT.RooRealVar("e_%s_cat%s"%(truth,tag),"",th2_resp.GetBinContent(ix+1,iy+1) ) \
    for ix,truth in enumerate(tags) ] for iy,tag in enumerate(tags) ]
# contruct scaling
scaling = [ [ ROOT.RooFormulaVar("scaling_%s_cat%s"%(truth,tag),"@0*@1",\
    ROOT.RooArgList(e[iy][ix],r[ix])) \
    for ix,truth in enumerate(tags) ] for iy,tag in enumerate(tags)  ]
```

- Each category will have contributions from other production mode

```
################################################
#             PER CAT MODEL                    #
################################################
model_s = [
    ROOT.RooAddPdf("model_s_%s"%t,"",\
        ROOT.RooArgList(smodel,smodel,smodel,models_b[iy]),\
        ROOT.RooArgList(scaling[iy][0],scaling[iy][1],scaling[iy][2],n_bkg[iy]))
    for iy,t in enumerate(tags)
]
```

Physics parameter determination

- Simultaneous fit to three categories



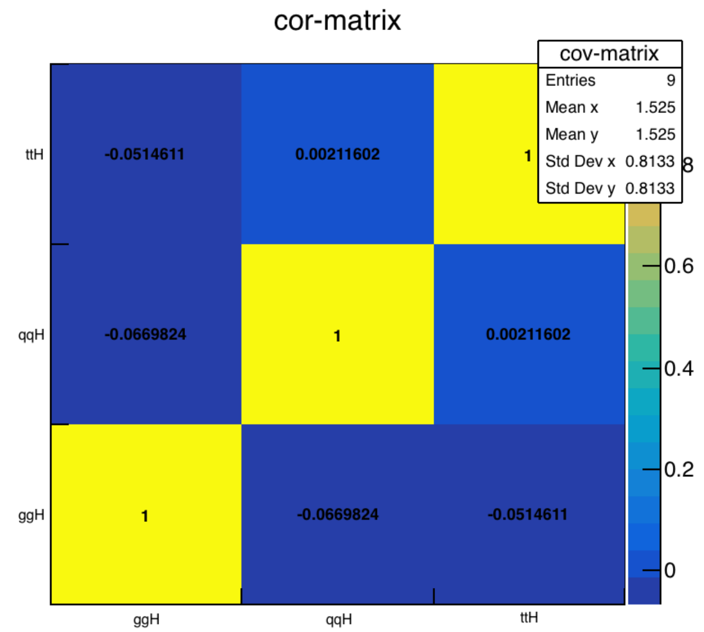
Signal strength

Physics parameter determination

- Convert these strengths into cross-sections if we know the reference cross-sections

```
reference_cross_sections=[ 48.58, 3.78, 0.5071 ]
strengths:
r_ggH: 0.981747 +/- (-0.011194,0.011270) -> 47.693288
r_qqH: 1.000471 +/- (-0.046289,0.046872) -> 3.781781
r_ttH: 0.902884 +/- (-0.141690,0.148022) -> 0.457853
```

- We can also take a look at the correlation between these strengths.



Physics parameter determination

- ❑ Extract the values of the EFT coefficients which best match our data
  - Use the SILH basis
  - And will be interested in 2 parameters c'_G and the linear combination $c_{HW} - c_B$.
- ❑ Now we know the scaling should be

$$r_{ggH} = 1+8.73c'_G + 19.5(c'_G)^2$$

$$r_{qqH} = 1-0.6985(c_{HW} - c_B) + 25.53(c_{HW} - c_B)^2$$

$$r_{ttH} = 1. + (0.115 + 0.0297c'_G)c'_G + 1.797(c_{HW} - c_B)^2$$

```
# if eft =0 --> SM. ggH, qqH, ttH
eft=[ROOT.RooRealVar("cG1","cG1",0.,-20,20),ROOT.RooRealVar("cHWmB","cHWmB",0.,-20,20) ]
eft_scaling_function = [ ROOT.RooFormulaVar("ggH_scaling","1. + cG1*(8.73+19.5*cG1)",ROOT.RooArgList(eft[0]) ),\
    ROOT.RooFormulaVar("qqH_scaling","1. + 25.53 * cHWmB *cHWmB +cHWmB*(-0.6985)", ROOT.RooArgList(eft[0],eft[1]) ),\
    ROOT.RooFormulaVar("ttH_scaling","1. + (0.115+0.0297*cG1)*cG1+1.797*cHWmB*cHWmB",ROOT.RooArgList(eft[0],eft[1]) ),
    ]
# for ix,truth in enumerate(tags) ] for iy,tag in enumerate(tags) ]
eft_scaling = [ [
    ROOT.RooFormulaVar("y_%s_cat%s"%(truth,tag),"@0*@1",ROOT.RooArgList(e[iy][ix],eft_scaling_function[ix]))
    for ix,truth in enumerate(tags) ] for iy,tag in enumerate(tags)
]
```

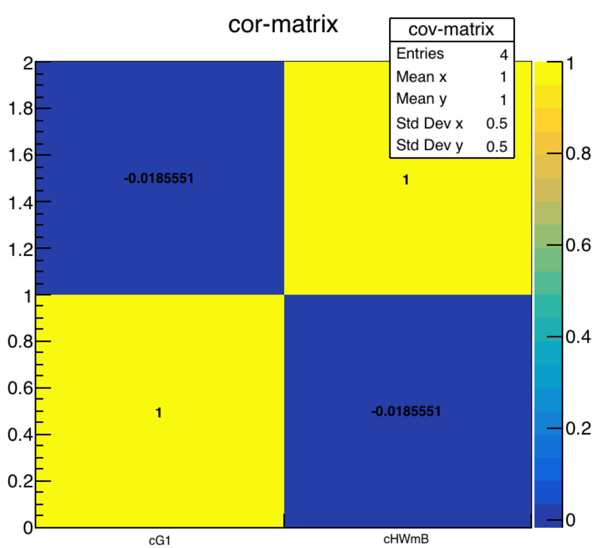
- ❑ Fit to data

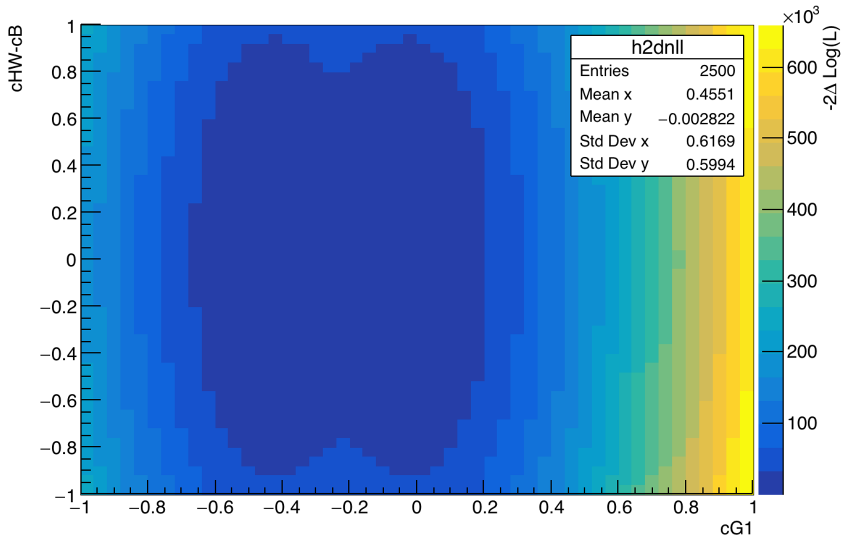
```
strengths:
cG1: -0.002140 +/- (0.000000,0.000000)
cHW-cB: 0.004434 +/- (-0.031411,0.058284)
```

Physics parameter determination

- Check the correlation matrix of c'_G and $c_{HW} - c_B$.
- Draw 2D likelihood scan

```
nll = eftPdf.createNLL(combData)
h2dnll = ROOT.TH2F("h2dnll",";cG1;cHW-cB;-2#Delta Log(L)",50,-1,1,50,-1,1)
for ix in range(h2dnll.GetNbinsX()):
    eft[0].setVal(h2dnll.GetXaxis().GetBinCenter(ix+1)) # TH1/TH2 bins start at 1
    for iy in range(h2dnll.GetNbinsY()):
        eft[1].setVal(h2dnll.GetYaxis().GetBinCenter(iy+1))
        h2dnll.SetBinContent(ix+1,iy+1,2*nll.getVal()-nll2minimum)
c2d = ROOT.TCanvas()
h2dnll.Draw("COLZ")
c2d.Draw()
```





Summary

❑ Basic RooFit

- Define and use objects in RooFit, constrcute and minimize likelihood

❑ Simultaneous likelihoods

- Add additional constraints on parameters from including multiple categories

❑ Physics parameter determination

- Multiple signal processes can contribute to multiple categories
- Fit to experimental data to determine differences from expected contributions can be used to constraint EFT coefficients