PREFIT-School
PRecision Effective Field Theory School
2-13 March 2020 at DESY, Hamburg

# Fitting: Session 2

*Andrea C. Marini, Nick Wardle*

**Assistants:**

*Lydia Brenner*

*Sebastian Wuchterl*

*Adinda de Wit*

## Statistical introduction

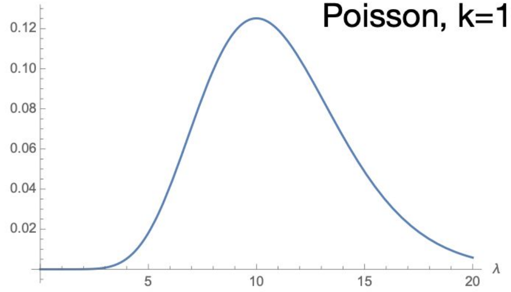
# Likelihood functions

- and Maximum Likelihood Estimators (MLE)

$$\mathcal{P}(k|\lambda) = \frac{e^{-\lambda}\lambda^k}{k!} \longleftrightarrow \mathcal{L}(\lambda|k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

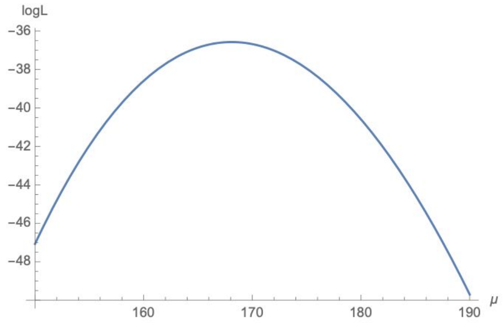
- Probability is normalized, Likelihood is not

Poisson, k=10



Poisson,

k=168, 176, 164, 179, 165, 161, 159, 157, 175, 177



$$\mathcal{L}(\lambda|k_i) = \prod_i \frac{e^{-\lambda}\lambda_i^k}{k_i!}$$

## Morphing

- How do we interpolate two shapes (f1,f2)?

**Vertical Morphing:**

$$f(x|\alpha) = \alpha f_1(x) + (1-\alpha) f_2(x)$$

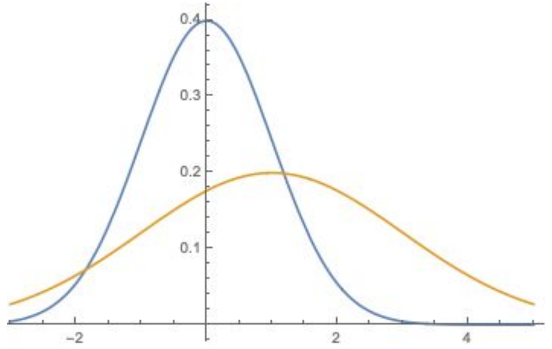
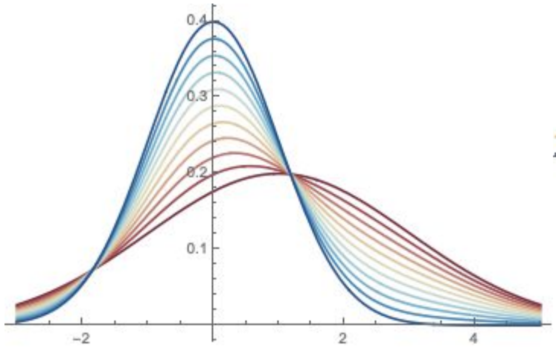
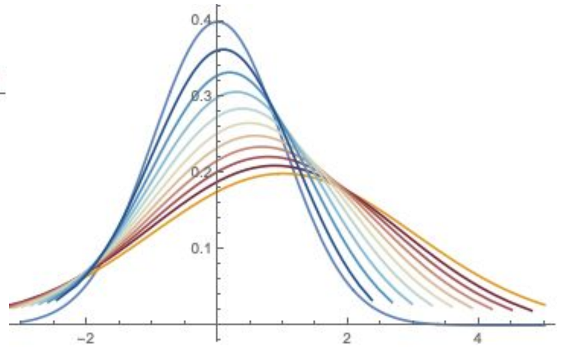
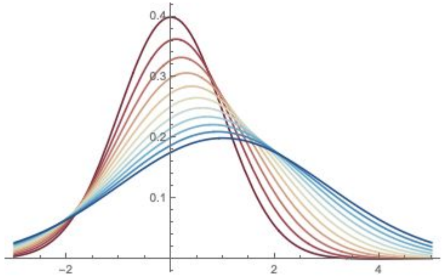
**Horizontal morphing:**

$$F^{-1}(y|\alpha) = \alpha F_1^{-1}(x) + (1-\alpha) F_2^{-1}(x)$$

**Parameters interpolation:**

Fit and interpolate parameters

$$f(x|\alpha) = f(x, (1-\alpha)\vec{\theta}_1 + \alpha\vec{\theta}_2)$$

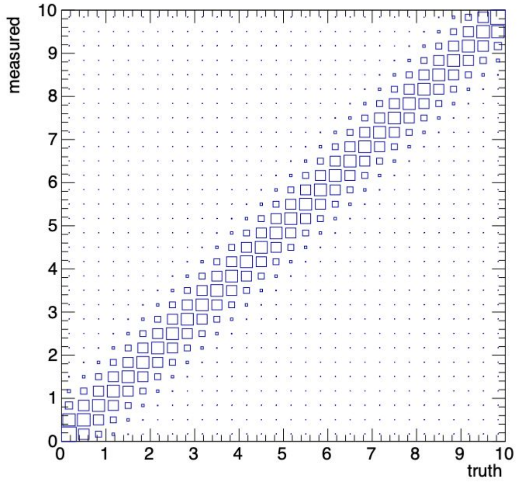





A.C. Marini, N. Wardle

# Unfolding: 2 words

- Undo detector resolution effects
- Used to measure differential distributions

$$\vec{y} = \tilde{\mathbf{R}} \cdot \vec{x}_{\mathrm{T}} + \vec{b},$$

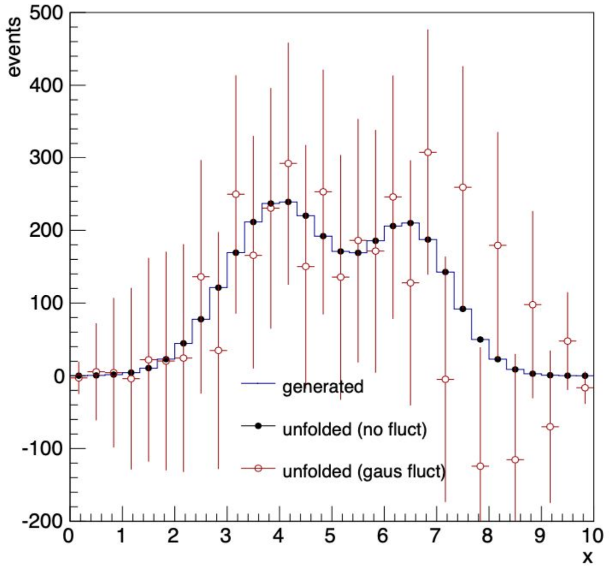


- Response matrix will give migrations probabilities across the bins

Can be done also w/ RooFit

# ML Solution & Regularization

- The ML solution (inversion) suffers from high variance and negative correlation across the bins
- These large spikes are often un-physical and distributions can be smooth out due to the fact that the truth distribution is continuous



**Regularized Unfolding:**

- **Bin-by-bin:** neglect migrations, and rescale each component of the histograms
- **early stop iterative** (bayesian):
  - use the Bayes theorem to measure the truth given the observation and a prior (MC)
  - re-iterate, the early stop of the iterations will give a regularized result
- **Tickonov regularization:**
  - modify the ML solution in order to add a penalization term on the curvature of the distributions (or residuals)

## Covariance matrix

- Covariance

$$\mathbb{V}[x,y] \overset{\text{def}}{=} \mathbb{E}[(x-\mu_x)\cdot(y-\mu_y)]$$

- Correlation

$$\rho = \mathbb{V}[x,y]/\sqrt{\mathbb{V}[x]\mathbb{V}[y]}$$

- Covariance Matrix

$$V = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$$

# Error propagation

- Error propagation (normally distributed)

$$\vec{y} = f(\vec{x}) \qquad y(\vec{x}) \sim y(\mu) + \mathbf{A}(x - \mu)$$

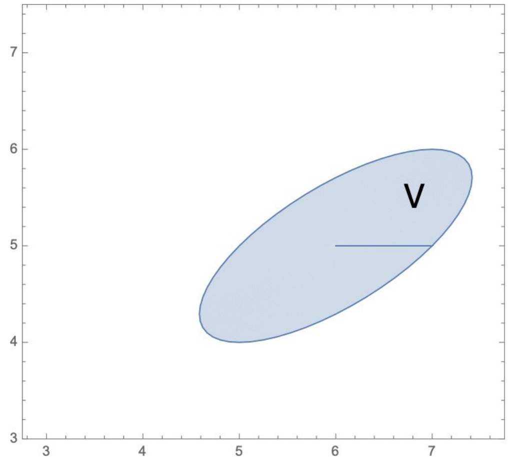
$$\mathbf{A}_{ki} = \left.\frac{\partial y_k}{\partial x_i}\right|_{x=\mu}$$

**Taylor serie**

$$\mathbf{U} = \mathbf{A} V \mathbf{A}^{\mathrm{T}}$$

**Linear Algebra**

- U is the covariance matrix of y



# Example of LSE

- LSE for two correlated measurements
  - x1,x2 with a common error sigmaC

$$V = \begin{pmatrix} \sigma_1^2 + x_1^2\sigma_c^2 & x_1x_2\sigma_c \\ x_1x_2\sigma_c & \sigma_2^2 + x_2^2\sigma_c^2 \end{pmatrix}$$

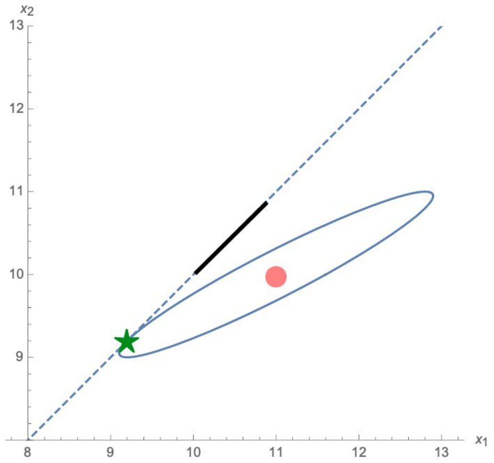
prove it!

$$\hat{x} = \frac{x_1^2\sigma_2^2 + x_2^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2 + (x_1 - x_2)^2\sigma_c^2}$$

- x1= 10+/-.5
- x2=11+/-.5
- sC = 20%

$$\hat{x} = 8.90 \pm 2.92$$

- and this is outside x1,x2!!!



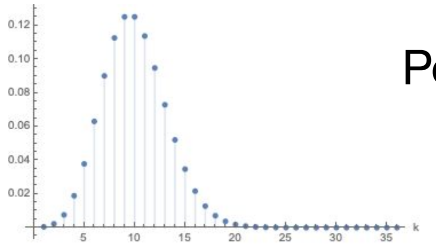
## Hands On

# Objective of Hands On

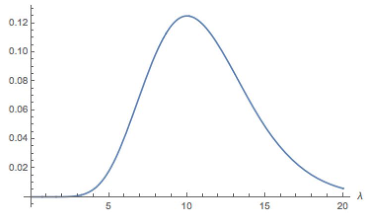
What you will see today:

Material: https://github.com/amarini/Prefit2020

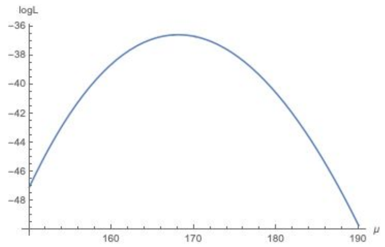
- Template likelihood
- Simultaneous fits
- Inclusion of simple systematics effects



Poisson PDF



Poisson likelihood



Product of Poisson likelihood

## Part 1

# Part 1

## Part 1: Template analysis

Sometimes we do not have a parametric form to describe our fit but rather rely on simulated events.

The likelihood takes the form of:

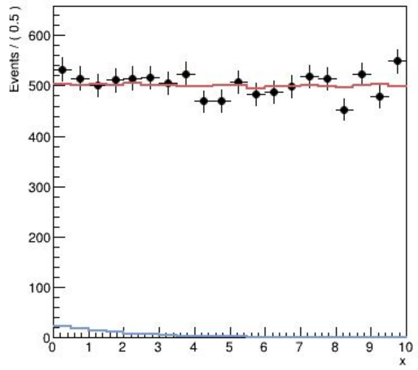
$$\mathcal{L} = \prod_i \mathcal{P}(x_{\text{obs}}|x_{\text{exp}})$$

or equivalently (using the multinomial distribution):

$$\mathcal{L} = \mathcal{P}(N_{\text{obs}}|N_{\text{exp}})\mathcal{M}(x_{\text{obs}}|p_{\text{exp}})$$

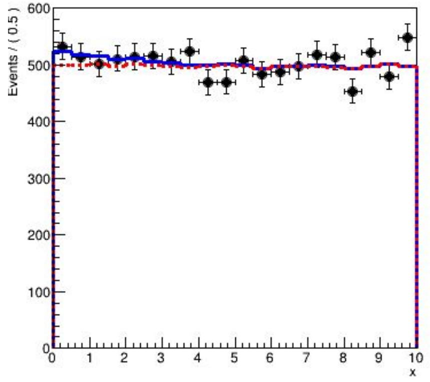
We start with a simple fit where we have a signal shape(h_sig), a background shape h_bkg, and the data distribution, h_data.

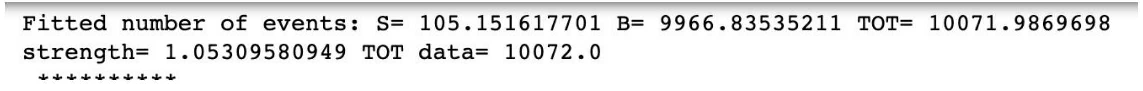
```
## Read the inputs and plot the histograms
fIn=ROOT.TFile.Open("input.root")

#fIn.ls()
h_data=fIn.Get("h_data__x")
h_sig=fIn.Get("h_sig__x")
h_bkg=fIn.Get("h_bkg__x")
```



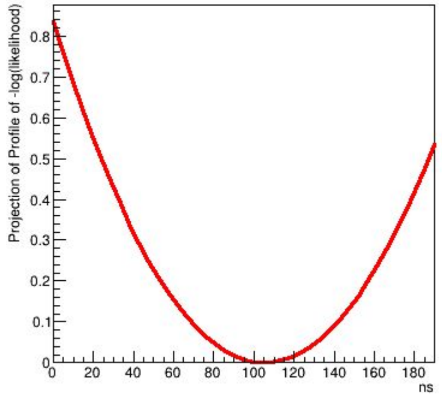
# Part 1: Assignment

1. Create a binned likelihood and minimize it.
2. How many signal events do you fit?
3. Plot the postfit distributions (sig and bkg)
4. Plot the profile likelihood of the n. of signal events





```
Fitted number of events: S= 105.151617701 B= 9966.83535211 TOT= 10071.9869698
strength= 1.05309580949 TOT data= 10072.0
 ***********
```



# Solution and explanation

```
# create the binned dataset to fit
x=ROOT.RooRealVar("x","x",1,0,10)
hist=[h_data,h_sig,h_bkg]
name=["data","sig","bkg"]
dh=[ROOT.RooDataHist("dh_%s"%n,"%s distribution"%n,ROOT.RooArgList(x),h) for n,h in zip(name,hist)]
## s and b pdf
pdf=[ROOT.RooHistPdf("hp_%s"%n, "%s",ROOT.RooArgSet(x), d) for n,d in zip(name[1:],dh[1:])]

rv_sig,rv_bkg=ROOT.RooRealVar("ns","ns",h_sig.Integral(),0,5*h_sig.Integral()),ROOT.RooRealVar("nb","nb",h_bkg.Integral
model_s=ROOT.RooAddPdf("model_s","model_s",ROOT.RooArgList(pdf[0],pdf[1]), ROOT.RooArgList(rv_sig,rv_bkg) )


#fr=model_s.fitTo(dh[0],ROOT.RooFit.Extended())

nll=model_s.createNLL(dh[0],ROOT.RooFit.Extended())
minuit=ROOT.RooMinuit(nll)
minuit.migrad()
minuit.improve()
minuit.hesse()
minuit.minos()

print "Fitted number of events: S=",rv_sig.getVal(),"B=",rv_bkg.getVal(),"TOT=",rv_sig.getVal()+rv_bkg.getVal()
print "strength=",rv_sig.getVal()/h_sig.Integral(), "TOT data=",h_data.Integral()
```

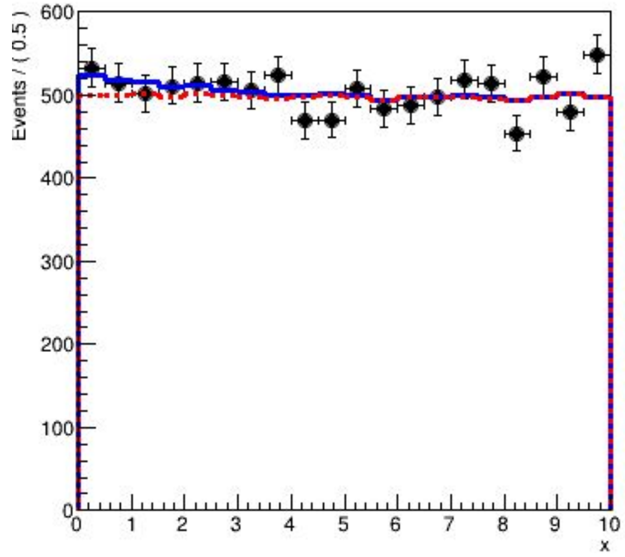
```
Fitted number of events: S= 105.151617701 B= 9966.83535211 TOT= 10071.9869698
strength= 1.05309580949 TOT data= 10072.0
 ***********
```

**== (Barlow)**

A.C. Marini, N. Wardle

## Solutions: plot best-fit

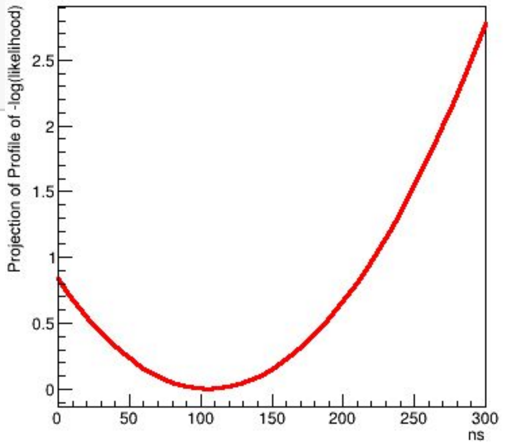
```
# plot fit
frame = x.frame()
dh[0].plotOn(frame)
model_s.plotOn(frame)
model_s.plotOn(frame, ROOT.RooFit.Components("hp_bkg"),
                ROOT.RooFit.LineStyle(ROOT.kDashed),
                ROOT.RooFit.LineColor(ROOT.kRed)
               )
c=ROOT.TCanvas("c_fit","c_fit",400,400)
frame.Draw() ;
c.Draw()
```



# Solutions: plot profile likelihood

Construct the profile likelihood and plot it

```
frame = rv_sig.frame(ROOT.RooFit.Bins(20), ROOT.RooFit.Range(0,300))

#Plot the profile likelihood in nsig
pll_frac = nll.createProfile(ROOT.RooArgSet(rv_sig))
pll_frac.plotOn(frame, ROOT.RooFit.LineColor(ROOT.kRed))

c=ROOT.TCanvas("c_pl","c_pl",400,400)
frame.Draw()
c.Draw()
```



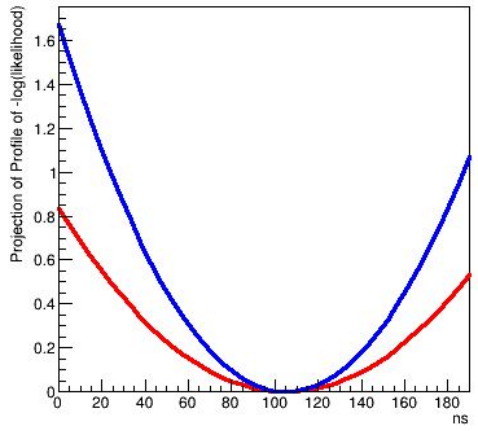
## Part 2

# Part 2: Assignment

## Simultaneous Fit

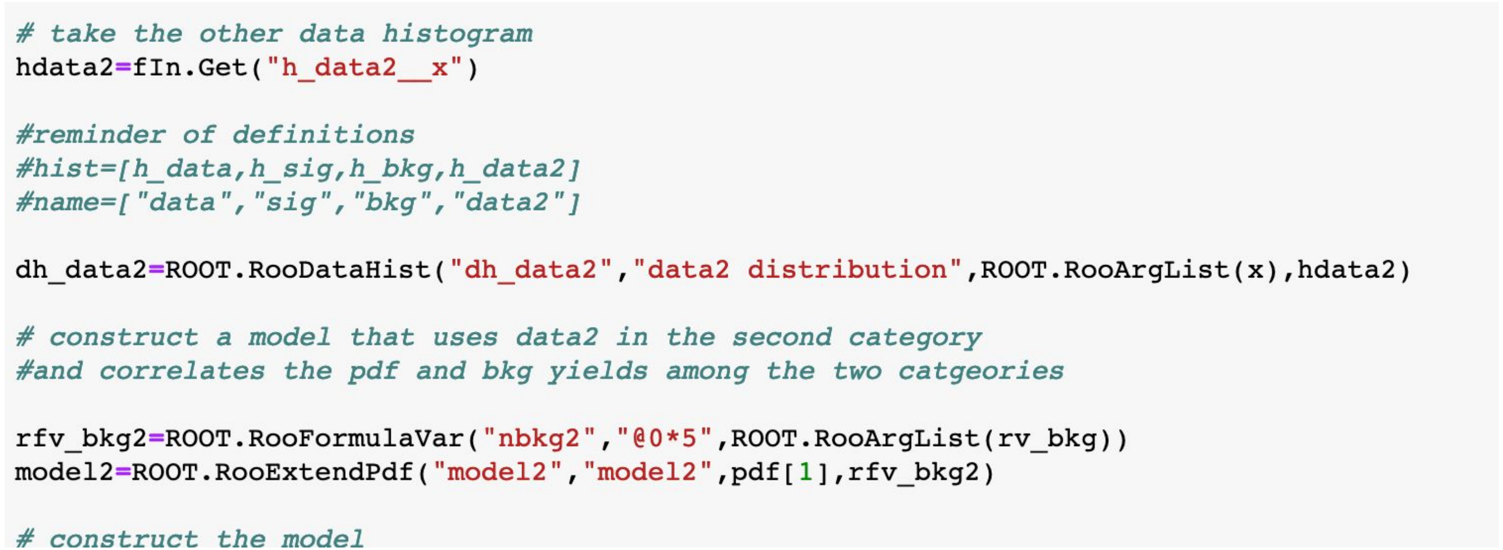
the bkg region has 5 times the expected yields for the bkg in the sig region. Correlate the two categories with this transfer factor.

1. Construct the model for the second category.
   Correlate the n of bkg events seen in each of the two with the right factors
2. Construct the simultaneous dataset and pdf (hint: look up `RooSimultaneous`)
3. Minimize the nll and plot the profile likelihood for n sig
4. Plot the two post fit datasets
5. How does it compare to the previous version?
6. Construct the sum of the nll for the two model
7. How does it compare to the previous method?



# Solutions

Take the additional hist and build the normalization parameter

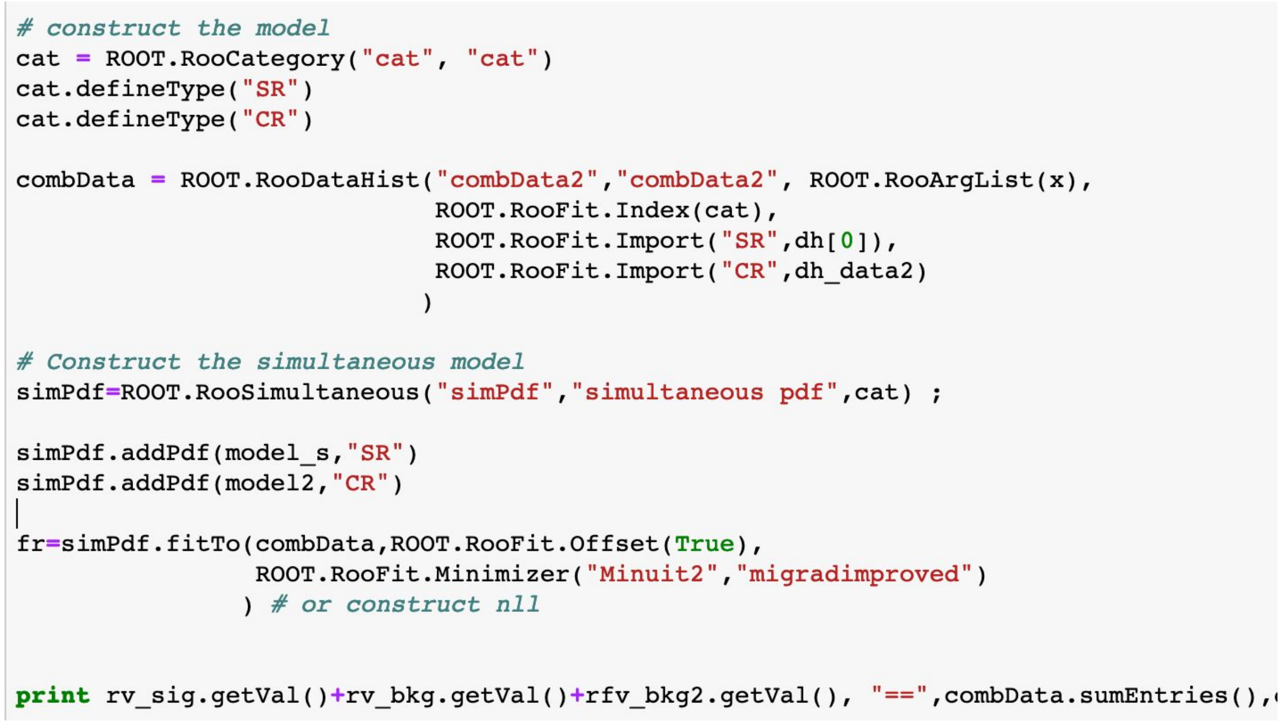


```
# take the other data histogram
hdata2=fIn.Get("h_data2__x")

#reminder of definitions
#hist=[h_data,h_sig,h_bkg,h_data2]
#name=["data","sig","bkg","data2"]

dh_data2=ROOT.RooDataHist("dh_data2","data2 distribution",ROOT.RooArgList(x),hdata2)

# construct a model that uses data2 in the second category
#and correlates the pdf and bkg yields among the two catgeories

rfv_bkg2=ROOT.RooFormulaVar("nbkg2","@0*5",ROOT.RooArgList(rv_bkg))
model2=ROOT.RooExtendPdf("model2","model2",pdf[1],rfv_bkg2)

# construct the model
```

A.C. Marini, N. Wardle

# Solutions

## Build the Simultaneous dataset/pdf



```
# 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")

fr=simPdf.fitTo(combData,ROOT.RooFit.Offset(True),
                ROOT.RooFit.Minimizer("Minuit2","migradimproved")
               ) # or construct nll


print rv_sig.getVal()+rv_bkg.getVal()+rfv_bkg2.getVal(), "==",combData.sumEntries(),
```

```
59898.0003081 == 59898.0
```

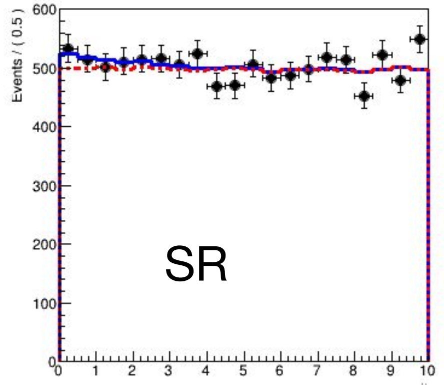
**(Barlow)**

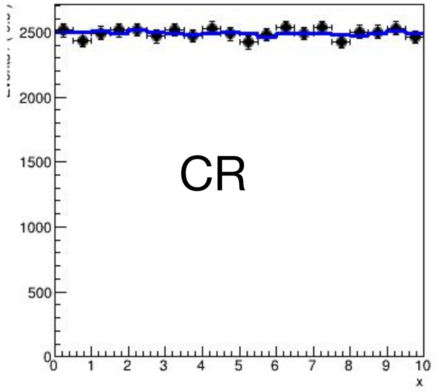
# Solution: plot slices

Careful with normalizations. In each region it is not true anymore
That each pdf is normalized to the integral in that region.

You can also use the projection of the comb dataset

```
## Plot slices

frame1 = x.frame(ROOT.RooFit.Bins(20),ROOT.RooFit.Title("Signal Region")) ;

dh[0].plotOn(frame1)
model_s.plotOn(frame1,ROOT.RooFit.Normalization(rv_sig.getVal()+rv_bkg.getVal(),ROOT.RooAbsReal.NumEvent) )
model_s.plotOn(frame1, ROOT.RooFit.Normalization(rv_sig.getVal()+rv_bkg.getVal(),ROOT.RooAbsReal.NumEvent),
               ROOT.RooFit.Components("hp_bkg"),
               ROOT.RooFit.LineStyle(ROOT.kDashed),
               ROOT.RooFit.LineColor(ROOT.kRed)
              )


frame2 = x.frame(ROOT.RooFit.Bins(20),ROOT.RooFit.Title("Control Region")) ;

dh_data2.plotOn(frame2)
model2.plotOn(frame2,ROOT.RooFit.Normalization(rfv_bkg2.getVal(),ROOT.RooAbsReal.NumEvent))

c=ROOT.TCanvas("c_sim","c_sim",800,400)

c.Divide(2)
c.cd(1)
frame1.Draw()
c.cd(2) ;
frame2.Draw()
c.Draw()
```



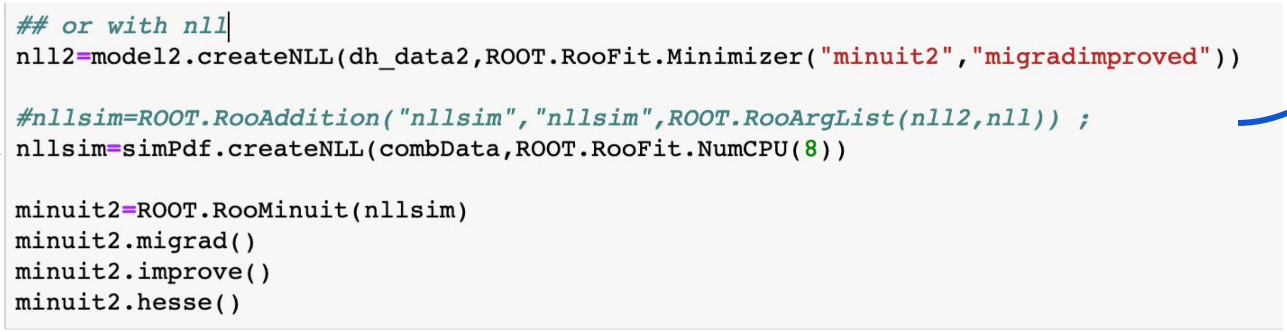


## Solutions: joint likelihood

Ask RooFit ….

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

… or sum the two log-likelihood !



```
## or with nll
nll2=model2.createNLL(dh_data2,ROOT.RooFit.Minimizer("minuit2","migradimproved"))

#nllsim=ROOT.RooAddition("nllsim","nllsim",ROOT.RooArgList(nll2,nll)) ;
nllsim=simPdf.createNLL(combData,ROOT.RooFit.NumCPU(8))

minuit2=ROOT.RooMinuit(nllsim)
minuit2.migrad()
minuit2.improve()
minuit2.hesse()
```

# Consideration on templates

$$\mathcal{L} = \prod_i \mathcal{L}_i \qquad \mathcal{L}_i = \prod_k \mathcal{P}(x_{\mathrm{obs},k} | x_{\mathrm{exp},k})$$

Each bin in a template is an independent Poisson (the bin-width is lost).

The stat part of the likelihood function runs over all the bins in all categories (order independent).

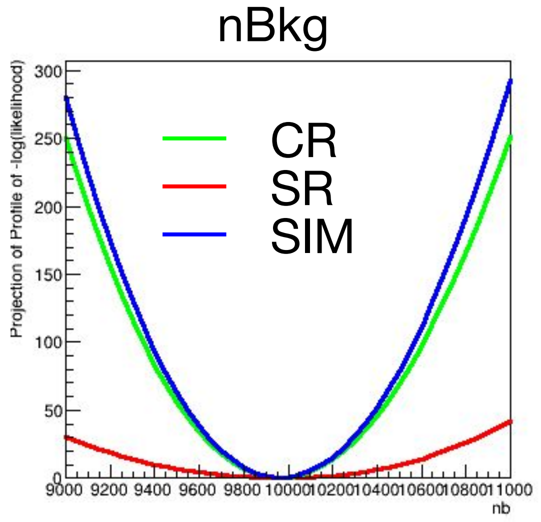
We can have a larger histogram with all the bins unrolled in.

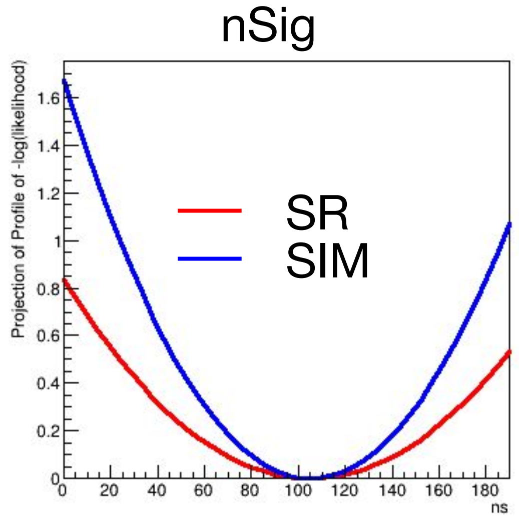
The writing in categories is to help us in **understanding** the fit and **easy** the write down of the systematics

# Adding a CR for bkg estimation

By adding a control region we constrain the background much more.

This constraint is seen in the profiling of the signal in the joint fit.





A.C. Marini, N. Wardle

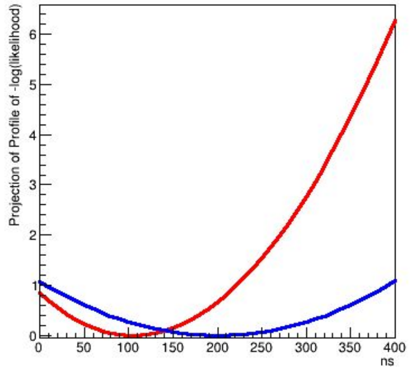
# Part 3

## Nuisances: Todo

Let's go back to the only SR fit.
Add a polynomial modification to the shape,

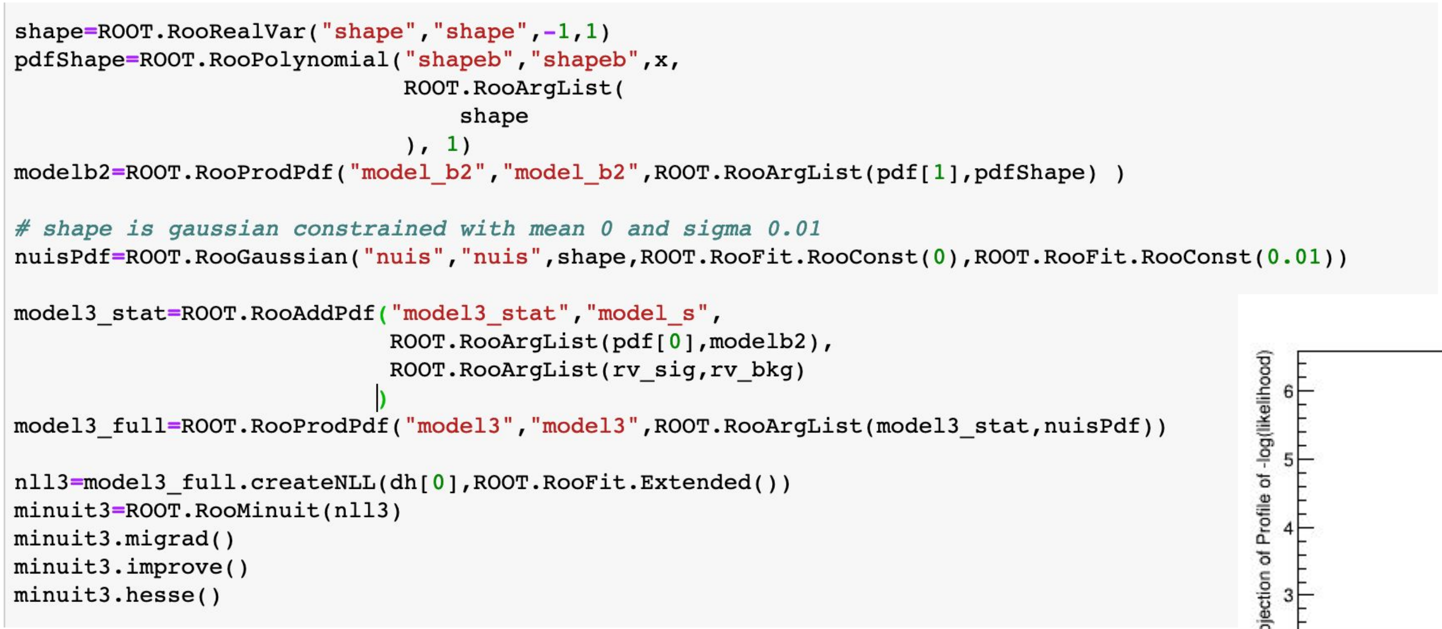
such as we get a linear modification of the background shape of 10% gaussian constraint.

1. Construct the shape modifier.
2. Construct the new background model.
3. Construct the nuisance pdf
4. Incorporate the nuisance pdf in the likelihood function and minimize it
5. Compare with previous results

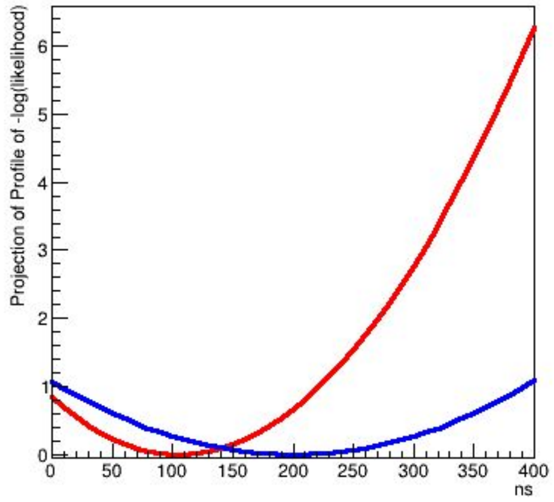


A.C. Marini, N. Wardle

# Solutions



```
shape=ROOT.RooRealVar("shape","shape",-1,1)
pdfShape=ROOT.RooPolynomial("shapeb","shapeb",x,
                            ROOT.RooArgList(
                                shape
                            ), 1)
modelb2=ROOT.RooProdPdf("model_b2","model_b2",ROOT.RooArgList(pdf[1],pdfShape) )

# shape is gaussian constrained with mean 0 and sigma 0.01
nuisPdf=ROOT.RooGaussian("nuis","nuis",shape,ROOT.RooFit.RooConst(0),ROOT.RooFit.RooConst(0.01))

model3_stat=ROOT.RooAddPdf("model3_stat","model_s",
                           ROOT.RooArgList(pdf[0],modelb2),
                           ROOT.RooArgList(rv_sig,rv_bkg)
                          )
model3_full=ROOT.RooProdPdf("model3","model3",ROOT.RooArgList(model3_stat,nuisPdf))

nll3=model3_full.createNLL(dh[0],ROOT.RooFit.Extended())
minuit3=ROOT.RooMinuit(nll3)
minuit3.migrad()
minuit3.improve()
minuit3.hesse()
```



The nuisance will allow the background to move, and therefore the fitted signal.

The latter, will have a broader likelihood function.

# Thanks!

# Tools … python

Python, scipy and numpy can be valuable options for the future.

Here the solution to Part 1 in numpy:

```
import ROOT
import numpy as np
from scipy.optimize import minimize
from scipy.stats import poisson
```

```
fIn=ROOT.TFile.Open("input.root")
```

```
## read h_data, h_sig and h_bkg and import them in numpy array
names=["data","sig","bkg"]
th1 = [fIn.Get("h_"+n+"__x") for n in names ]
arr = [np.array([h.GetBinContent(i+1) for i in range(0,h.GetNbinsX())]) for h in th1] ## arr[0] ->

data= np.array([ th1[0].GetBinContent(i+1) for i in range(0,th1[0].GetNbinsX()) ] )
exp= np.array( [ [h.GetBinContent(i+1) for i in range(0,h.GetNbinsX()) ] for h in th1[1:] ]  )
```

```
def nll(r):
    global exp,data
    return np.sum(- poisson.logpmf(data, np.dot(r,exp) ) )
```

```
def nll_fix_s(rsig):
    def nll_bkg(rbkg):
        return nll(np.append(rsig,rbkg))
    b0=np.array([1.])
    bf=minimize(nll_bkg,b0,method='SLSQP',jac=False)
    return bf.fun
```

```
#optional
def jac(r):
    global exp,data
    #sum _i
    return np.sum( (np.ones(data.shape)-np.divide(data,np.dot(r,exp)))*exp,axis=1)
```

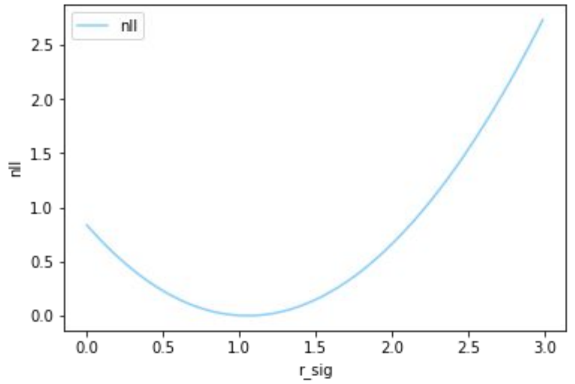
# Results with scipy

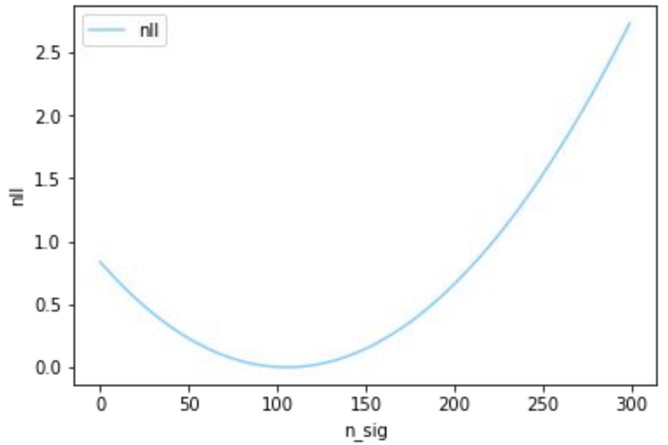
```
## minimize the likelihood function
r0=np.array([1.,1.])
rbf=minimize(nll,r0,
             method='SLSQP',
             jac=False);
rbf.x ## strength modifiers.
```

```
array([1.05243109, 0.99609794])
```

```
##compute profile likelihood vs nsig
nsig=np.arange(.9,1.3,0.01)# start, stop, step
l = np.array([nll_fix_s(s) for s in nsig]).flatten()
```

```
import matplotlib.pyplot as plt
fig = plt.figure()
plt.xlabel('r_sig')
plt.ylabel('-nll')
ax = plt.axes()
plt.plot (nsig, l-rbf.fun,color='lightskyblue',label="nll")
plt.legend()
plt.show()
```


nll
nll
r_sig


nll
nll
n_sig