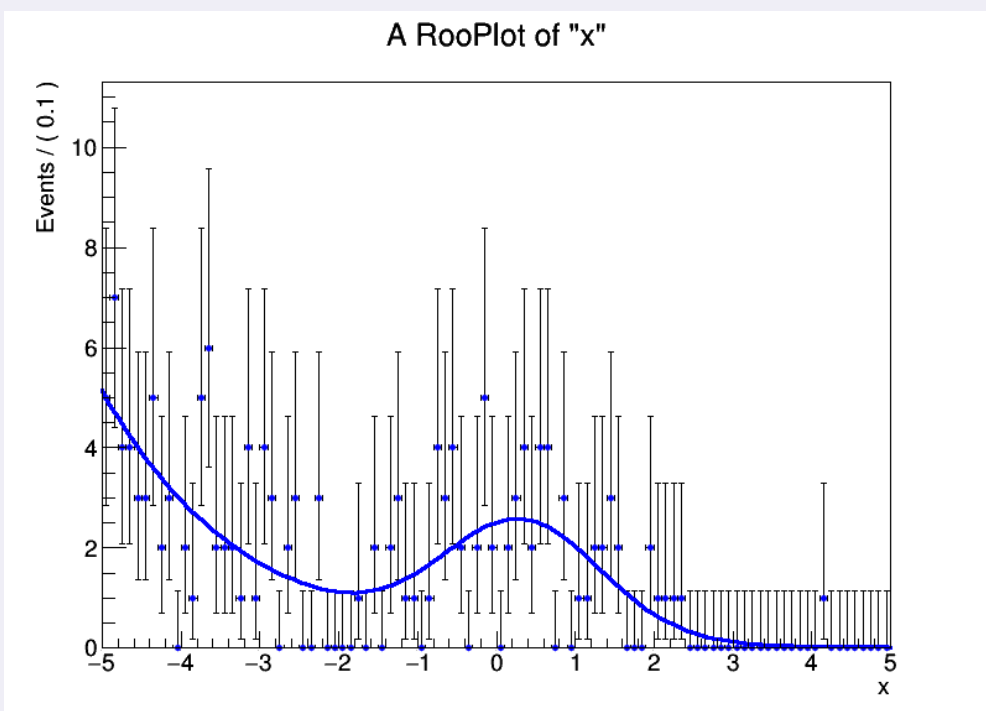


# RooFit, RooStats & HistFactory

a python data science *Cheat Sheet*

## A Simple Example

```
import ROOT
w = ROOT.RooWorkspace()
w.factory('Gaussian::g(x[-5,5],mu[-3,3],sigma[1])')
w.factory('Exponential::e(x,tau[-.5,-3,0])')
w.factory('SUM::model(s[50,0,100]*g,b[100,0,1000]*e)')
x = w.var('x')
pdf = w.pdf('model')
frame = x.frame()
data = pdf.generate(ROOT.RooArgSet(x))
data.plotOn(frame)
fitResult = pdf.fitTo(data,ROOT.RooFit.Save())
pdf.plotOn(frame)
frame.Draw()
```



## Models and Fitting

For a set of PDFs e.g. signalpdf and backgroundpdf and a fractional normalization fsig:

```
fsig = ROOT.RooRealVar("fsig","signal fraction",0.5,0.,1.)
model = ROOT.RooAddPdf("model","model",
    RooArgList(signalpdf,backgroundpdf),RooArgList(fsig))
```

We can create a negative log-likelihood (nll) function and from this and profile log-likelihood (pll) function. These relate estimators for the parameters in a model to some data.

```
nll = pdf.createNLL(data)
pll = nll.createProfile(paramOfInterest)
```

Various minimizers are available through the minuit package, these can act on all parameters or just those specified.

```
m = ROOT.RooMinimizer(nll)
m.migrad()           efficient and fast, struggles with nonlinearities
m.hesse()            calculates the full error matrix from second derivatives
m.minos(ParametersOfInterest) correct even if non-parabolic, but slow
```

## Using Data

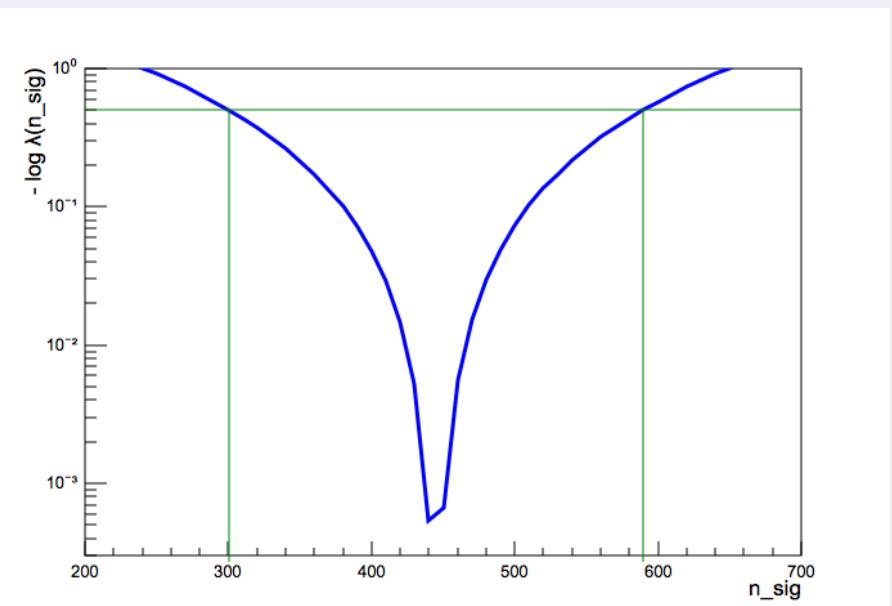
Data can be imported from histograms or generated from a model.

```
hh = ROOT.TH1F("hh","some histogram",21,-10,10)
x = ROOT.RooRealVar("x","x",-10,10)
real_data = ROOT.RooDataHist("data","dataset with x",x,hh)
pseudo_data = model.generate(ROOT.RooArgSet(x),10000)
```

## Statistical Tests with RooStats

### Profile Likelihood

```
pl = ROOT.RooStats.ProfileLikelihoodCalculator(data,ModelConfig)
pl.SetConfidenceLevel(0.683)           set desired interval
interval = pl.GetInterval()             perform calculation
firstPOI = mc.GetParametersOfInterest().first() only one POI
lowerLimit = interval.LowerLimit(firstPOI) extract limits
upperLimit = interval.UpperLimit(firstPOI) this is a number
plot = ROOT.RooStats.LikelihoodIntervalPlot(interval)
plot.Draw() Custom plot object but regular frame->plotOn also works.
```

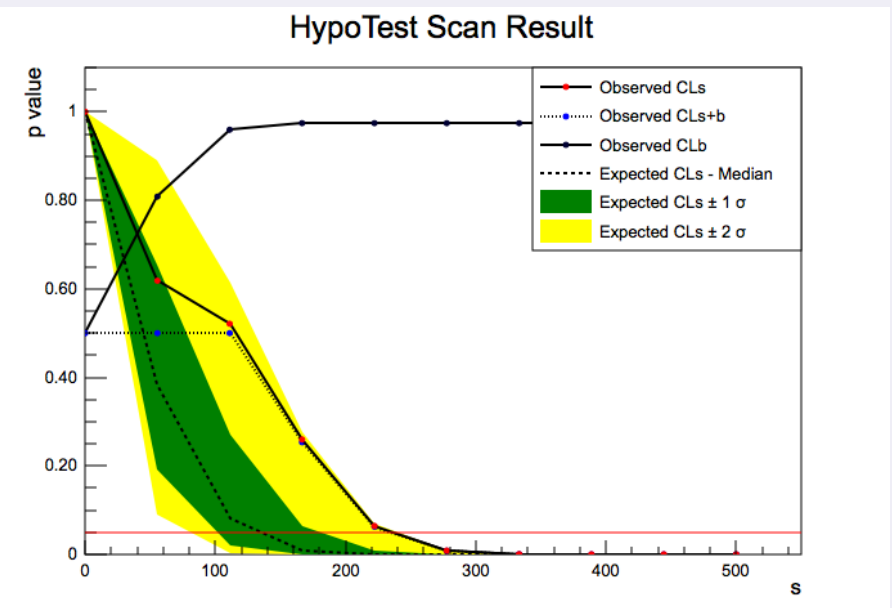


### Hypothesis Test (asymptotic formula)

```
ac = ROOT.RooStats.AsymptoticCalculator(data, sbModel, bModel)
asResult = ac.GetHypoTest() where sbModel is signal+background
asResult.Print() bModel = background only. Gives p-value & Z0
```

### Limits using Hypothesis Test Inversion (CLs Limits)

```
ac = ROOT.RooStats.AsymptoticCalculator(data, sbModel, bModel)
calc = ROOT.RooStats.HypoTestInverter(ac) as above
calc.SetConfidenceLevel(0.95) desired limit
calc.UseCLs(True) scan CLs+b or CLs values
toymcs = calc.GetHypoTestCalculator().GetTestStatSampler()
prof1l = ROOT.RooStats.ProfileLikelihoodTestStat(sbModel.GetPdf())
toymcs.SetTestStatistic(prof1l) Profile likelihood test statistics
calc.SetFixedScan(npoints,poi.getMin(),poi.getMax()) set range
r = calc.GetInterval() or scan until reaching the desired precision
upperLimit = r.UpperLimit() the result for this confidence level.
plot = ROOT.RooStats.HypoTestInverterPlot("HTI_Result_Plot",
    "HypoTest Scan Result",r)
plot.Draw("CLb 2CL")
```



## Workspaces and Histfactory

```
w = ROOT.RooWorkspace("w") Make a workspace
getattr(w,'import')(pdf1) Move a pdf into the workspace
w.factory('SUM::model(n_sig[5,0,10]*pdf1,n_bkg[10,0,100]*pdf2)')
w.var("n_sig").setVal(2) Factory can be used to create models
model = w.pdf("model") objects (e.g PDFs) are extracted by name
data = model.generate(ROOT.RooArgSet(x)) data to be used
getattr(w,'import')(data) most WS's need a PDF and data
mc = ROOT.RooStats.ModelConfig("ModelConfig",w) configure model
mc.SetPdf(model) target subset of variables
mc.SetParametersOfInterest(ROOT.RooArgSet(w.var("n_sig"))) POI
mc.SetSnapshot(ROOT.RooArgSet(w.var("n_sig"))) preserve values
mc.SetObservables(ROOT.RooArgSet(w.var("x"))) target range
w.defineSet("nuisParams","n_bkg") there can be hundreds of NPs
nuis = getattr(w,'set')("nuisParams") added to WS as normal
mc.SetNuisanceParameters(nuis) NPs distinguished from POIs in model
getattr(w,'import')(mc) import ModelConfig to WS
w.writeToFile("outputdir/name.root",True) save the workspace to file
```

```
meas = ROOT.RooStats.HistFactory.Measurement("meas", "meas")
meas.SetPOI( "SigXsecOverSM" )
chan = ROOT.RooStats.HistFactory.Channel( "SignalRegion" )
chan.SetStatErrorConfig(0.05, "Poisson")
chan.SetData( data_hist )
signal = ROOT.RooStats.HistFactory.Sample( "signal" )
signal.SetHisto( signal_hist )
signal.AddNormFactor( "SigXsecOverSM", 1, 0, 3)
signal.AddOverallSys( "flat_uncertainty", low_val, high_val )
signal_shape = ROOT.RooStats.HistFactory.HistoSys("shape_uncert")
signal_shape.SetHistoHigh( sig_up_hist )
signal_shape.SetHistoLow( sig_down_hist )
signal.AddHistoSys( signal_shape )
chan.AddSample( signal )
hist2workspace =
    ROOT.RooStats.HistFactory.HistoToWorkspaceFactoryFast(meas)
workspace = hist2workspace.MakeSingleChannelModel(meas, chan)
```

## RooVariables, RooPdfs, and Data

### Variables and P.D.Fs

```
observable = ROOT.RooRealVar("x","x",-10,10)
mean = ROOT.RooRealVar("mean","Mean",-10,10)
sigma = ROOT.RooRealVar("sigma","Width",3,-10,10)
gauss = ROOT.RooGaussian("gauss","pdf title",x,mean,sigma)
```

### Common P.D.Fs

```
ROOT.RooBifurGauss("name", "title", x, μ, σL, σR)
ROOT.RooExponential("name", "title", x, c)
ROOT.RooPolynomial("name", "title", x, RooArgList(c1,c2)
ROOT.RooPoisson("name", "title", x, η)
```

Bifurcated Gaussian  $f(x; \mu, \sigma) = \frac{1}{N} \cdot \exp(-(x - \mu)^2 / (2\sigma(x - \mu)^2))$   
Exponential  $f(x; c) = \frac{1}{N} \exp(cx)$   
Polynomial  $f(x; c_0, \dots, c_n) = \frac{1}{N} \cdot (1 + \sum_{k=1}^n c_k x^k)$   
Poisson  $f(x; \eta) = \frac{1}{x!} \cdot \eta^x \exp(-\eta)$