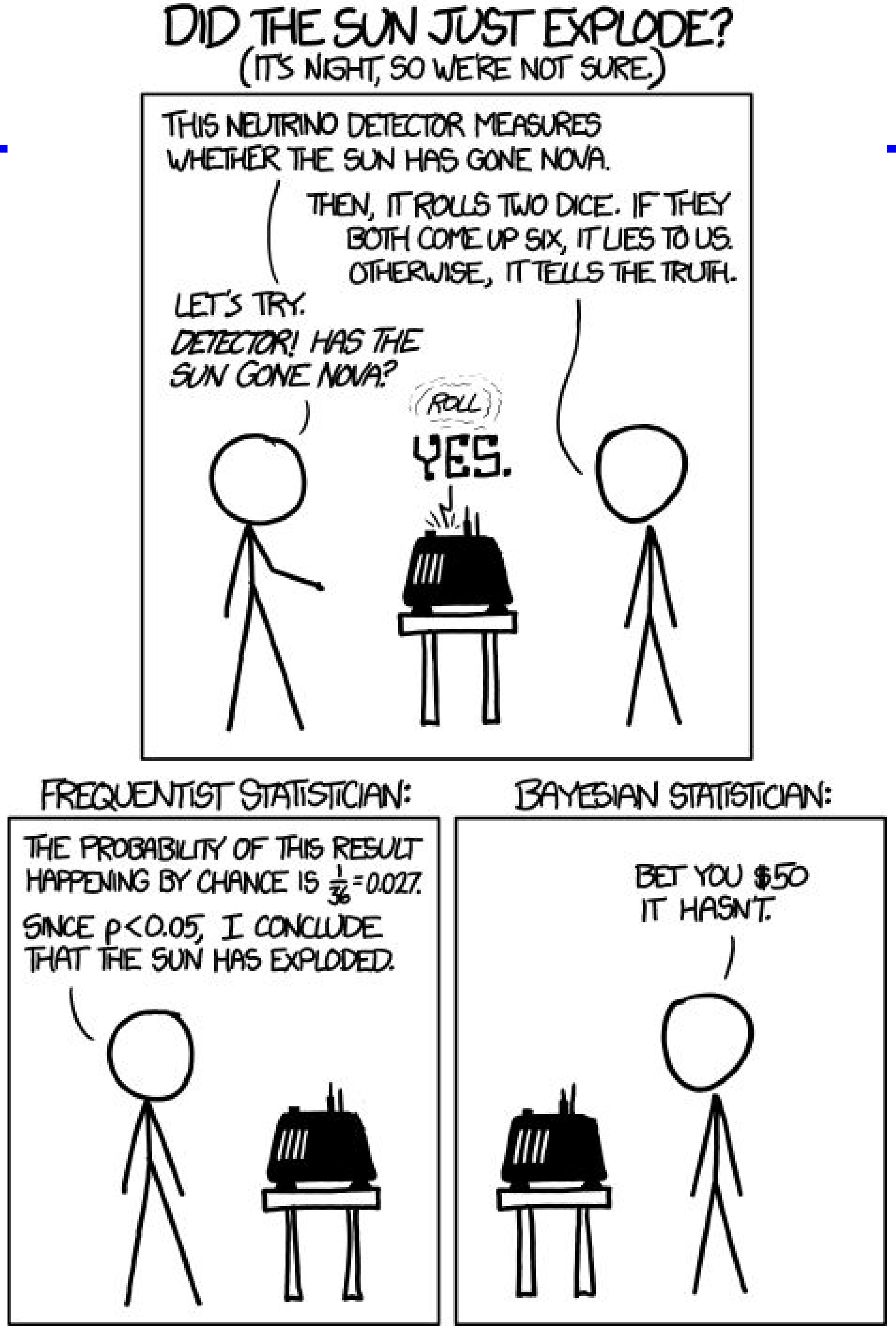
Hands-on session on statistical tools

Mario Pelliccioni

INFN School of Statistics 2017


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY.
DETECTOR! HAS THE SUN GONE NOVA?
(ROLL)
YES.
FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS 1/36 = 0.027. SINCE p<0.05, I CONCLUDE THAT THE SUN HAS EXPLODED.
BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

Welcome!

3 hours of classes

Covering a few most relevant use-cases for statistical analysis in HEP

Using RooFit and RooStats as main tools

You can use your laptops for the exercises (provided you installed ROOT with the --enable-roofit option)

CERN/other labs central clusters normally work too

(CMSSW/AliROOT work too)

Exercises will be in PyROOT, so python installation necessary

I will flash a few introductory slides for each topic

Twiki contains the exercises we will go through

https://twiki.cern.ch/twiki/bin/view/Main/INFNStatRooStats2017

RooFit and RooStats

RooFit: a ROOT library containing classes that allow to perform multi-dimensional (un)binned maximum likelihood/chi2 fits, toy-MC generation, plotting, etc

RooStats: a ROOT library that uses RooFit and provides classes to perform statistical interpretation of your results

Documentation

For most of what I do, I refer to the ROOT reference guide:

https://root.cern.ch/doc/master/classes.html

This includes RooFit and RooStats reference

RooFit manual (a bit outdated):

https://root.cern.ch/download/doc/RooFit_Users_Manual_2.91-33.pdf

RooStats documentation

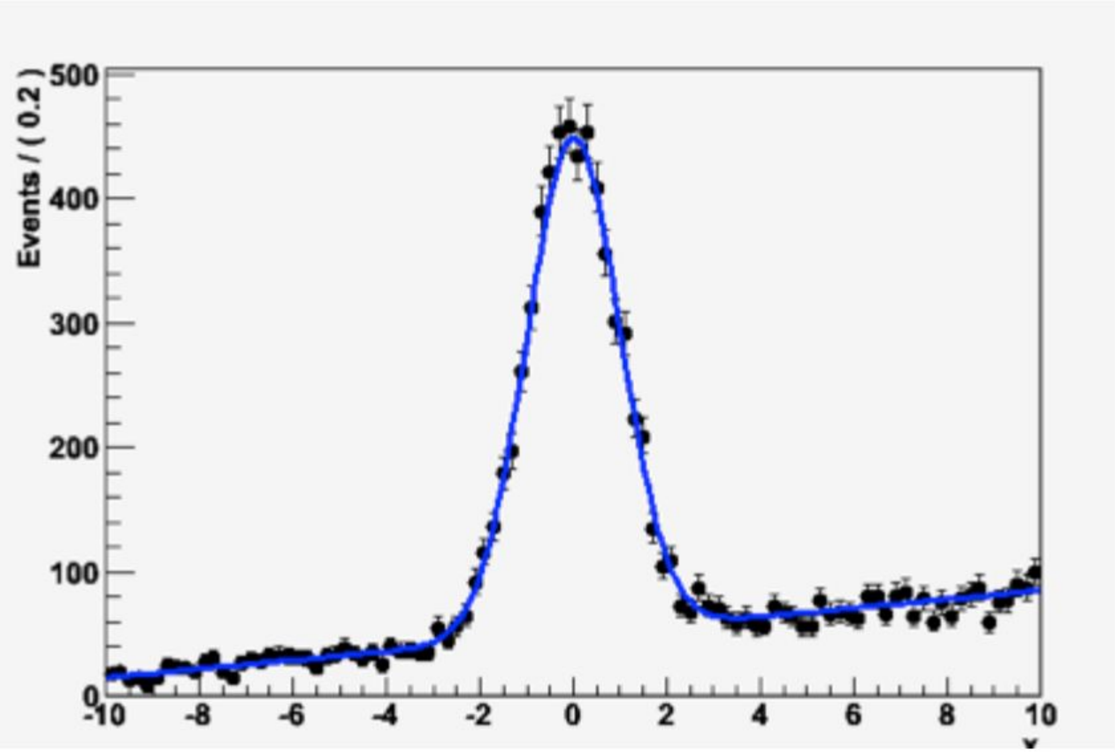
https://twiki.cern.ch/twiki/bin/view/RooStats/WebHome

More RooFit/RooStats examples (C++ based)

https://github.com/pellicci/UserCode/tree/master/RooFitStat_class

Why do we need RooFit?

- Focus on one practical aspect of many data analysis in HEP: How do you formulate your p.d.f. in ROOT
  - For ‘simple’ problems (gauss, polynomial) this is easy



  - But if you want to do unbinned ML fits, use non-trivial functions, or work with multidimensional functions you quickly find that you need some tools to help you

The origins

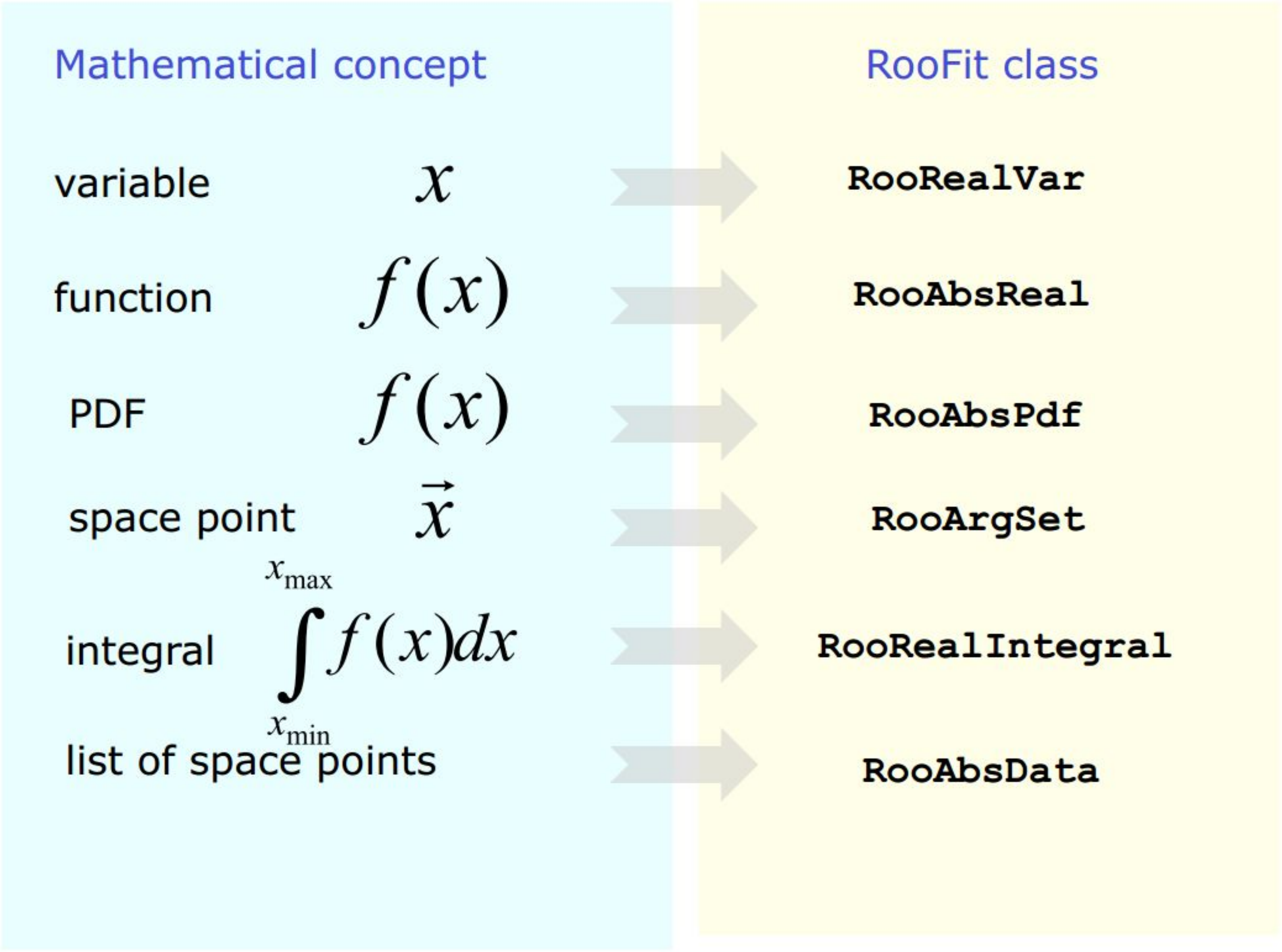
- BaBar experiment at SLAC: Extract sin(2β) from time_ dependent CP violation of B decay: e^+e^- → Υ(4s) → BB
  - Reconstruct both Bs, measure decay time difference
  - Physics of interest is in decay time dependent oscillation

$$
\begin{aligned}
&f_{sig} \cdot \left[\mathrm{SigSel}(m; \bar{p}_{sig}) \cdot \left(\mathrm{SigDecay}(t; q_{sig}, \sin(2\beta)) \otimes \mathrm{SigResol}(t \mid dt; r_{sig}) \right) \right] + \\
&(1 - f_{sig}) \left[\mathrm{BkgSel}(m; \bar{p}_{bkg}) \cdot \left(\mathrm{BkgDecay}(t; \vec{q}_{bkg}) \otimes \mathrm{BkgResol}(t \mid dt; \vec{r}_{bkg}) \right) \right]
\end{aligned}
$$

- Many issues arise
  - Standard ROOT function framework clearly insufficient to handle such complicated functions → must develop new framework
  - Normalization of p.d.f. not always trivial to calculate → may need numeric integration techniques
  - Unbinned fit, >2 dimensions, many events → computation performance important → must try optimize code for acceptable performance
  - Simultaneous fit to control samples to account for detector performance

“Dictionary”

- Mathematical objects are represented as C++ objects



Mathematical concept		RooFit class
variable	x	`RooRealVar`
function	$f(x)$	`RooAbsReal`
PDF	$f(x)$	`RooAbsPdf`
space point	$\vec{x}$	`RooArgSet`
integral	$\int_{x_{\min}}^{x_{\max}} f(x)dx$	`RooRealIntegral`
list of space points		`RooAbsData`

RooFit uses MINUIT for most of its work, it just provides an easy to use interface and optimizations

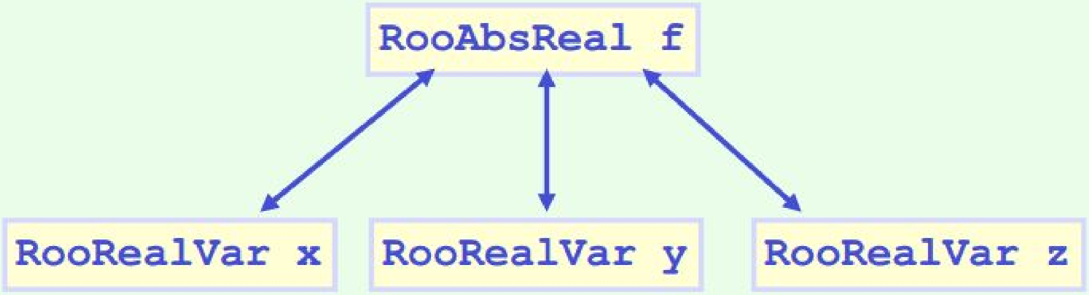
Design philosophy

- Represent relations between variables and functions as client/server links between objects

Math

$f(x,y,z)$

RooFit diagram



RooFit code

```
RooRealVar x(“x”,”x”,5) ;
RooRealVar y(“y”,”y”,5) ;
RooRealVar z(“z”,”z”,5) ;
RooBogusFunction f(“f”,”f”,x,y,z) ;
```

Variables

All variables (observables or parameters) are defined as **RooRealVar**

Several constructors available, depending on the needs:

```
var1 = ROOT.RooRealVar(“var1”,”My first var”,4.15)        //constant variable
var2 = ROOT.RooRealVar(“var2””My second var”,1.,10.); //valid range, no initial value
var3 = ROOT.RooRealVar(“var3””My third var”,3.,1.,10.); //valid range, initial value
```

You can also specify the unit (mostly for plotting purposes)

```
time = ROOT.RooRealVar(“time”,”Decay time”,0.,100.,”[ps]”);
```

You can change the properties of your RooRealVar later (setRange, setBins, etc.)

If you want to be 100% sure a variable will stay constant, use RooConstVar

Probability Density Functions

Each PDF in RooFit must inherit from RooAbsPdf

RooAbsPdf provides methods for numerical integration, events generation (hit & miss), fitting methods, etc.

RooFit provides a very extensive list of predefined functions (RooGaussian, RooPolynomial, RooCBShape, RooExponential, etc…)

If possible, always use a predefined function (if analytical integration or inversion method for generation are available, it will speed your computation)

You can always define a custom function using RooGenericPdf

Data Handling

Two basic classes to handle data in RooFit:

- **RooDataSet**: an unbinned dataset (think of it as a TTree). An ntuple of data

- **RooDataHist**: a binned dataset (think of it as a THXF)

Both types of data handlers can have multiple dimensions, contain discrete variables, weights, etc.

The perfect container

In order to “move” information among different RooFit/RooStats programs, one can use the RooWorkspace class

A **RooWorkspace** can contain:

- Variables
- PDFs
- DataSets

A RooWorkspace can be saved into a ROOT file

We’ll see how to use it.

The problem at hand

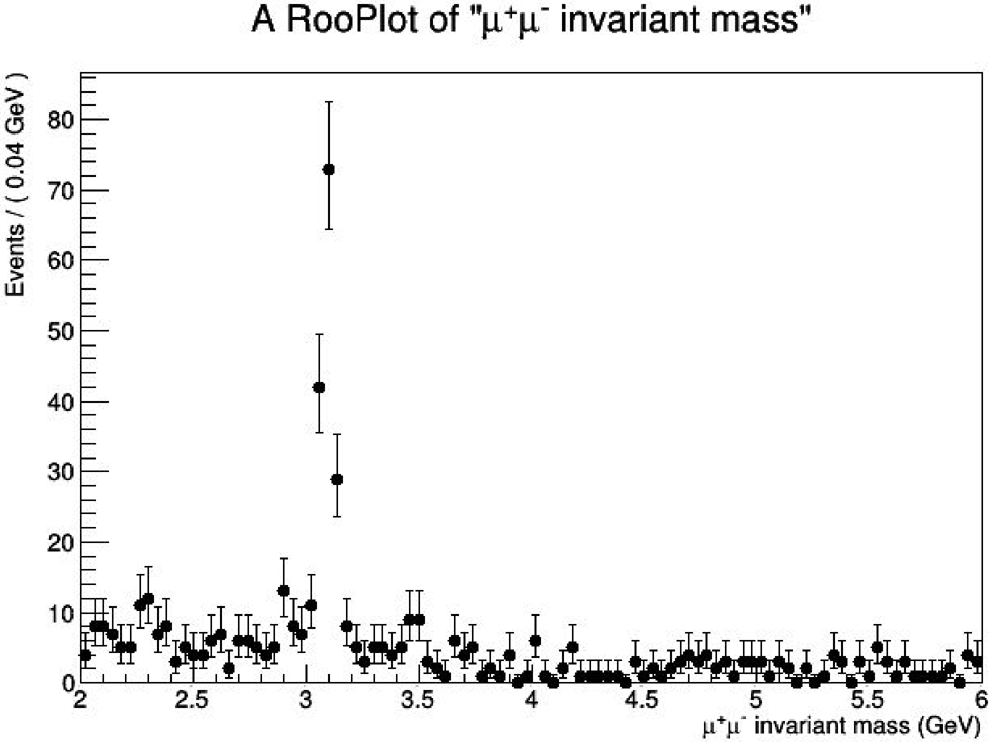
We will be analyzing a sample from the 2010 CMS data taking

All CMS data from 2010 is public → opendata.cern.ch

- I’ve isolated events with two opposite sign muons
- Calculated the invariant mass of the system
- Saved it into a RooDataSet (a 1D ntuple containing “mass” variable)

First, let’s look at the first three weeks of data taking (corresponds to about half a pb^{-1} of integrated lumi)

We’ll be studying this distribution



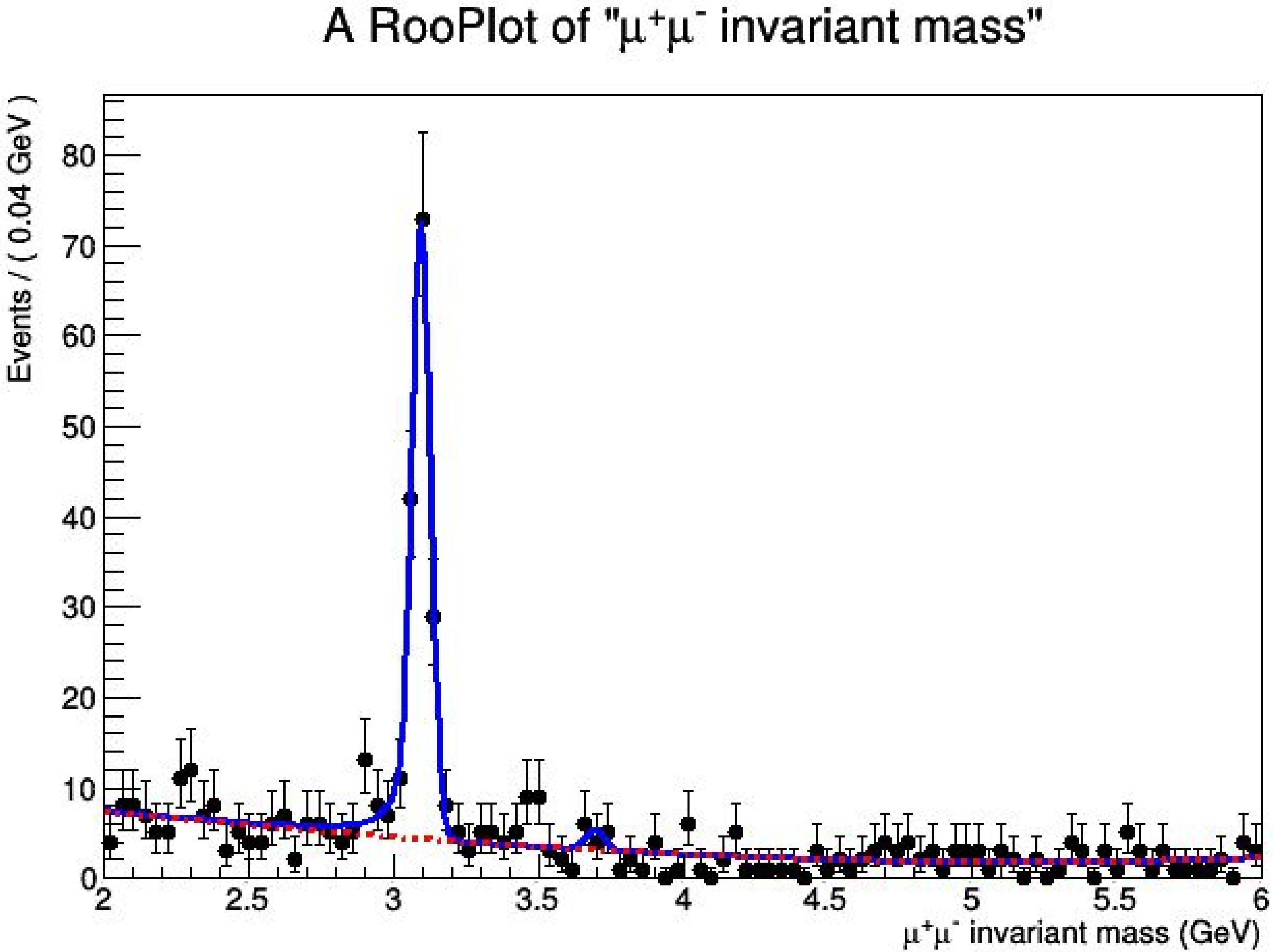
Exercise #0

The first exercise involves RooFit only

- Construct a J/ψ and ψ(2S) + background PDF
  - J/ψ with a Crystal Ball function
  - ψ(2S) with a Gaussian function
  - Background with a polynomial
- For now, the ψ(2S) will involve a very small amount of signal events
- Fit it, plot it, save it

We are going to use this program all the way through the exercises

Result of exercise #0

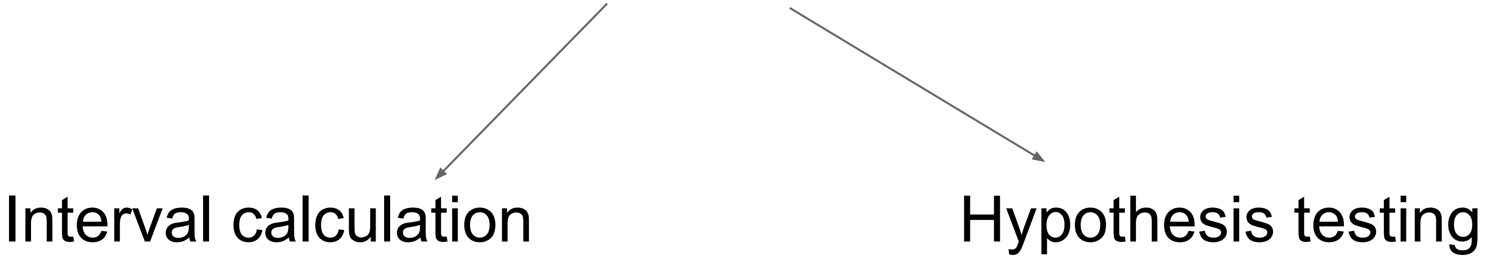


RooStats

Set of libraries for statistical interpretation of your results

→ communicates with RooFit via RooWorkspace

RooStats does essentially two things:



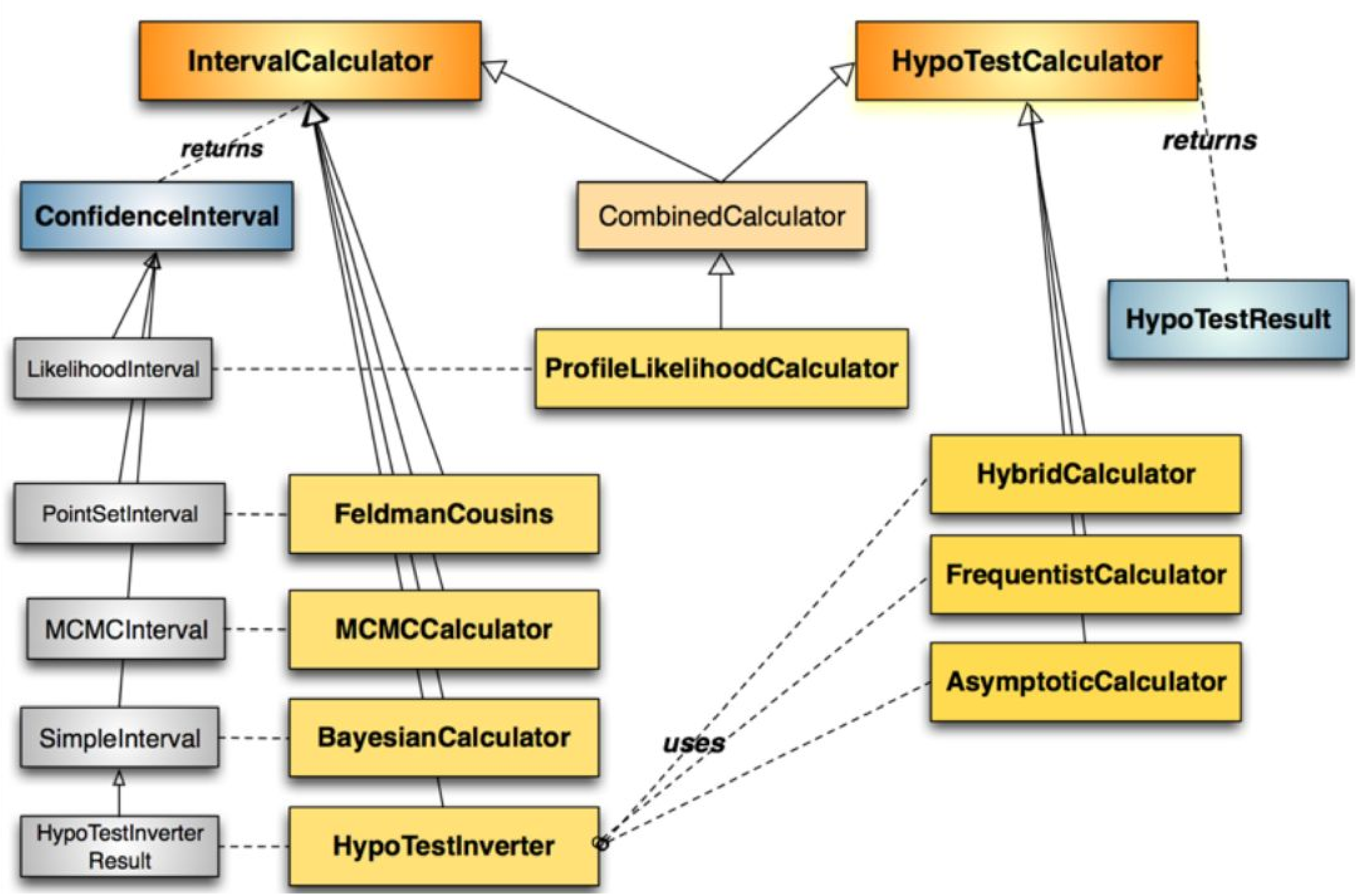
Interval calculation

Hypothesis testing

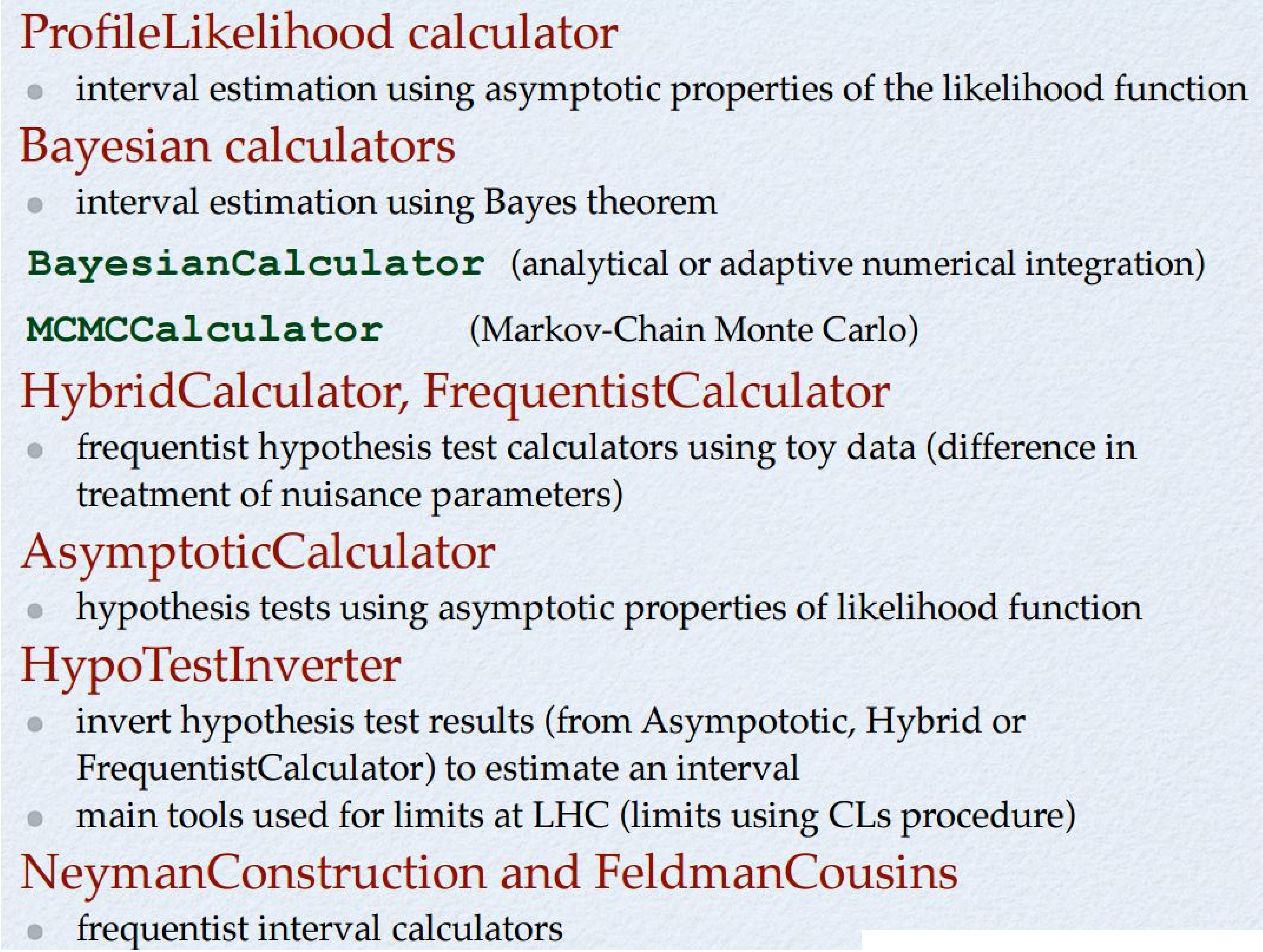
To do this, it uses “calculators”

RooStats design

C++ classes that reproduce statistical concepts



Main RooStats Calculators



ProfileLikelihood calculator

- interval estimation using asymptotic properties of the likelihood function

Bayesian calculators

- interval estimation using Bayes theorem

`BayesianCalculator` (analytical or adaptive numerical integration)

`MCMCCalculator` (Markov-Chain Monte Carlo)

HybridCalculator, FrequentistCalculator

- frequentist hypothesis test calculators using toy data (difference in treatment of nuisance parameters)

AsymptoticCalculator

- hypothesis tests using asymptotic properties of likelihood function

HypoTestInverter

- invert hypothesis test results (from Asympototic, Hybrid or FrequentistCalculator) to estimate an interval
- main tools used for limits at LHC (limits using CLs procedure)

NeymanConstruction and FeldmanCousins

- frequentist interval calculators

Exercise #1

Exercise #0 told us that there’s clearly no significant peak in the distribution

Is this actually clear? How do we quantify?

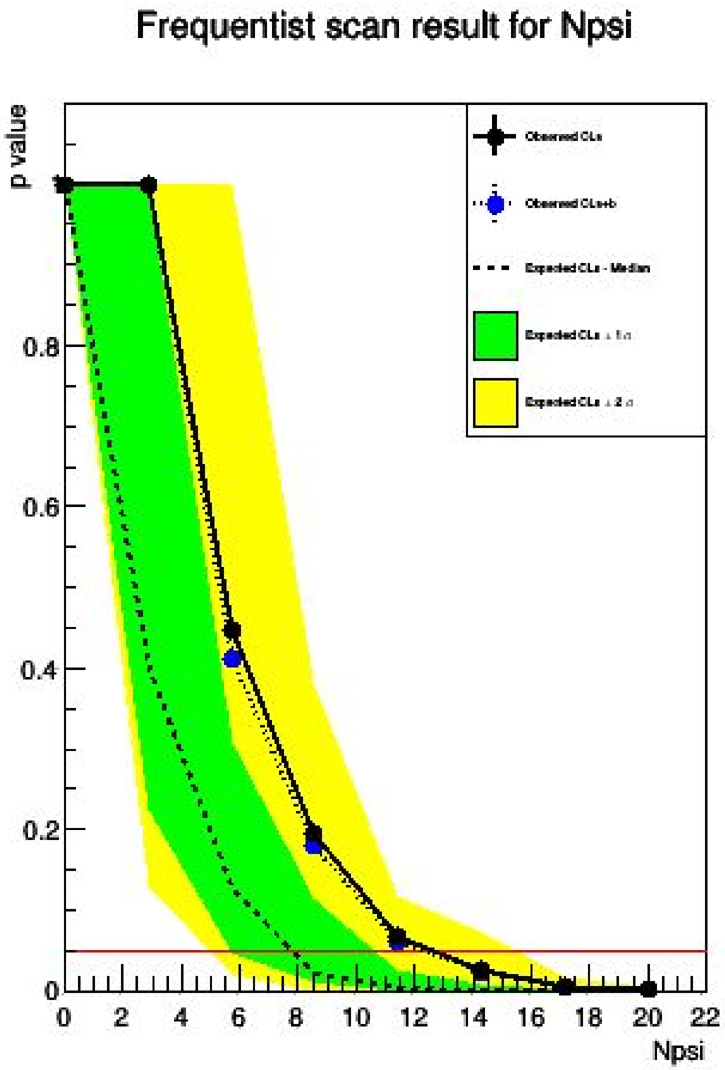
Exercise #2

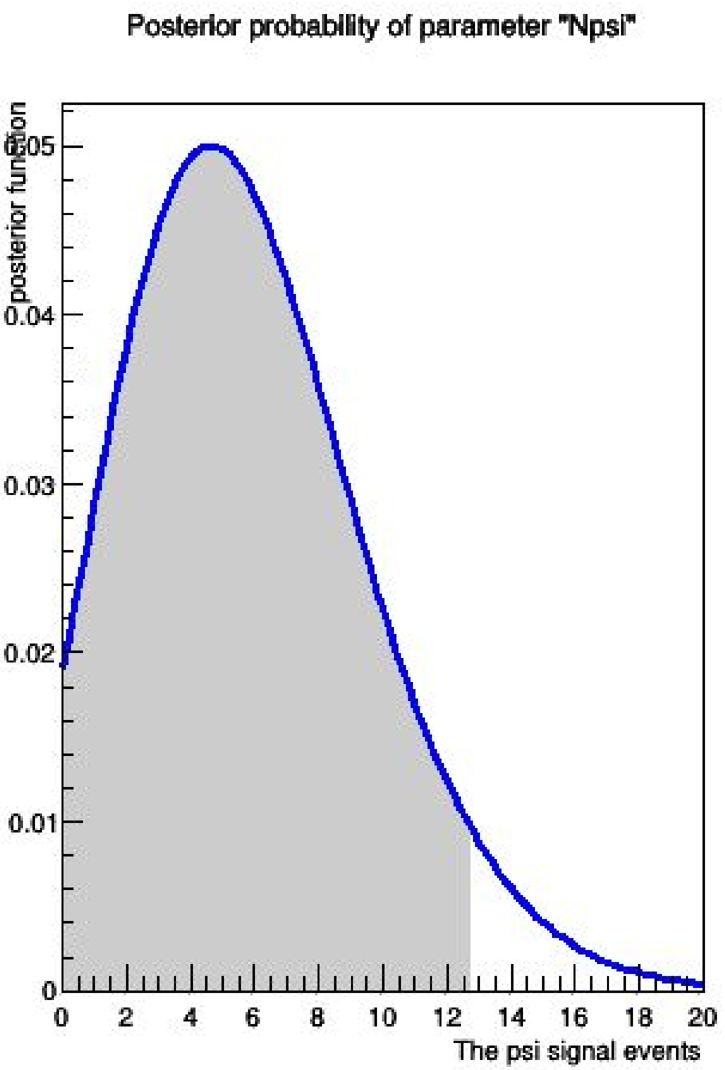
From exercise #1 we know that our excess is “not significant”.

The normal procedure here is to evaluate an upper limit on our parameter of interest. In this case, we will consider N_{sig} as our parameter of interest

In general, we are more interested in the cross section (so you need to reparametrize N_{sig} as cross*lumi → RooFormulaVar)

Result of exercise #2





Exercise #3

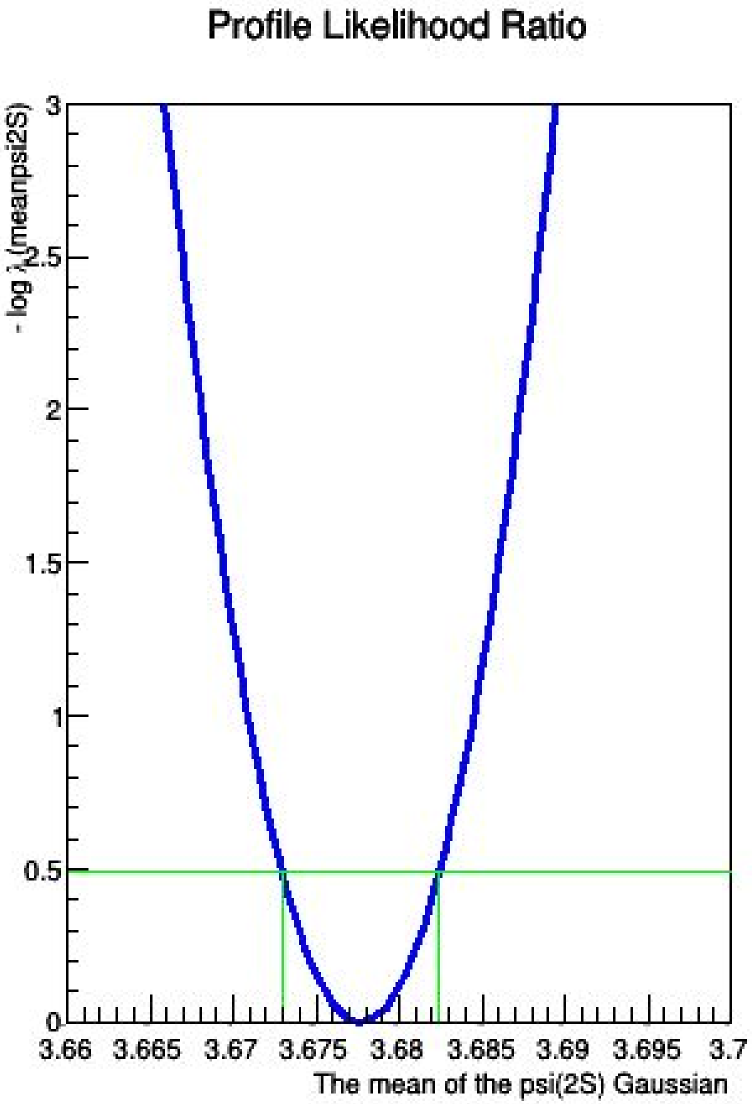
Let’s now go to a scenario where we have a significant excess

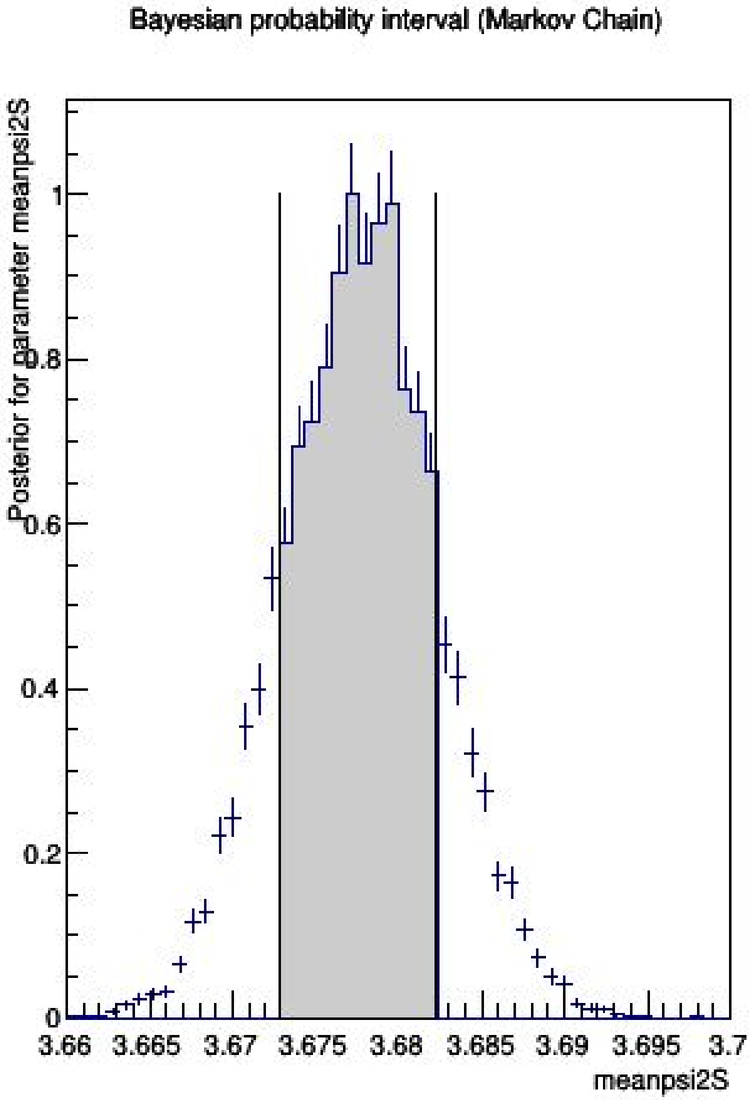
- Get the full 2010 statistics file
- Rerun exercise 0 and 1 to recreate the workspace and calculate the new significance

Now we can measure the properties of our discovery

For example the mass!

Result of exercise #3





Exercise #4

Up to now, all parameters but the parameter of interest could be fixed to a constant value

The basic assumption in this approach is that their uncertainty is negligible

This is sometimes acceptable, but in many cases your parameters have uncertainties that have to be taken into account

A possible approach: treat these uncertainties as a “penalty factor” in your likelihood function!

Dealing with systematics

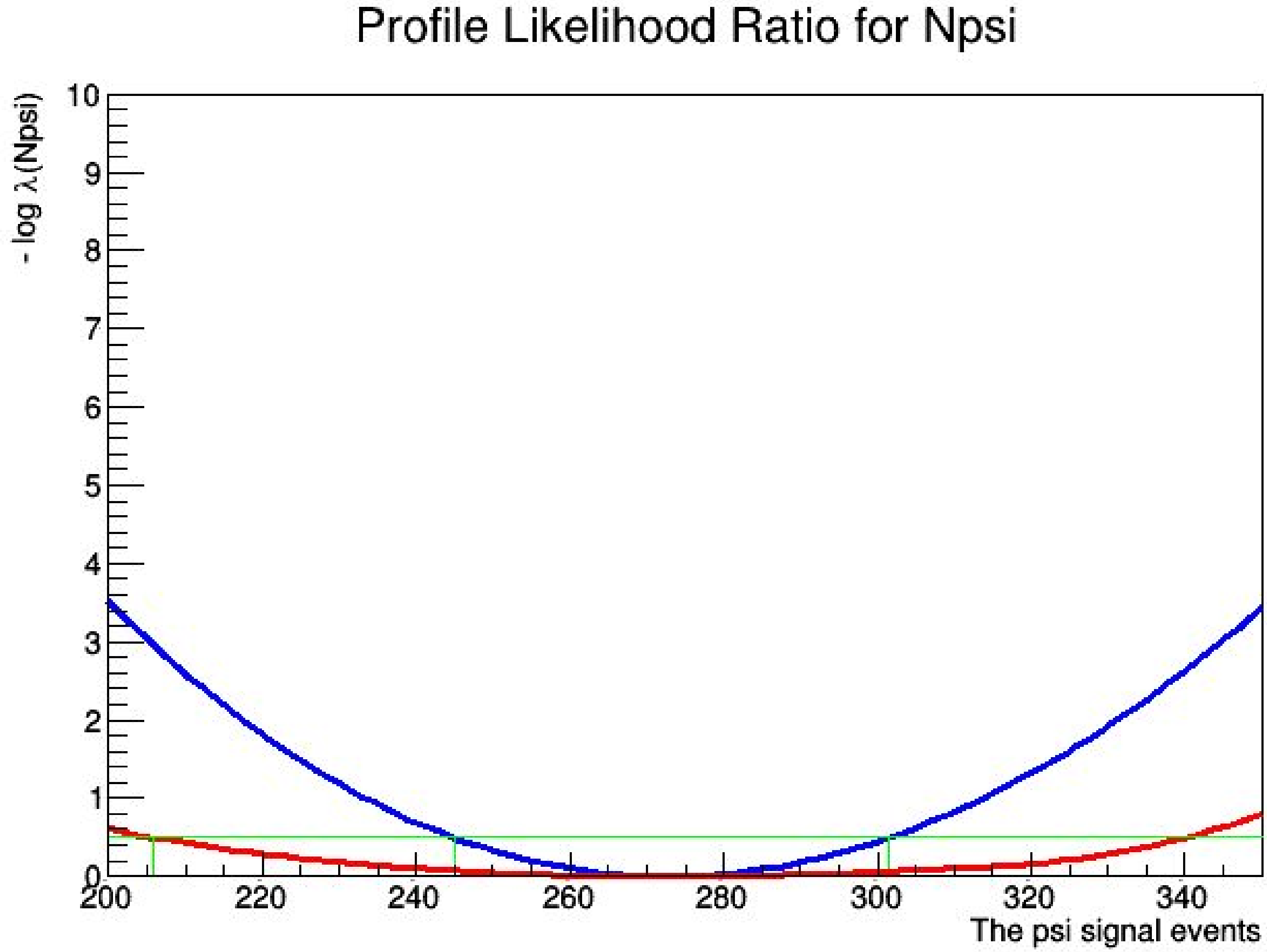
Imagine a 30% uncertainty on the efficiency (not realistic, but easier to see here)

- $N_{J/\psi,obs} = N_{J/\psi} * \alpha_{eff}$

- $\alpha_{eff} = k^{\beta eff}$, with β_{eff} is the new nuisance parameter, distribuited normally
- k = 1.3 is the 30% uncertainty on the efficiency

Then add to the likelihood function a normal Gaussian for β_{eff}

Result of exercise #4



That's all folks!