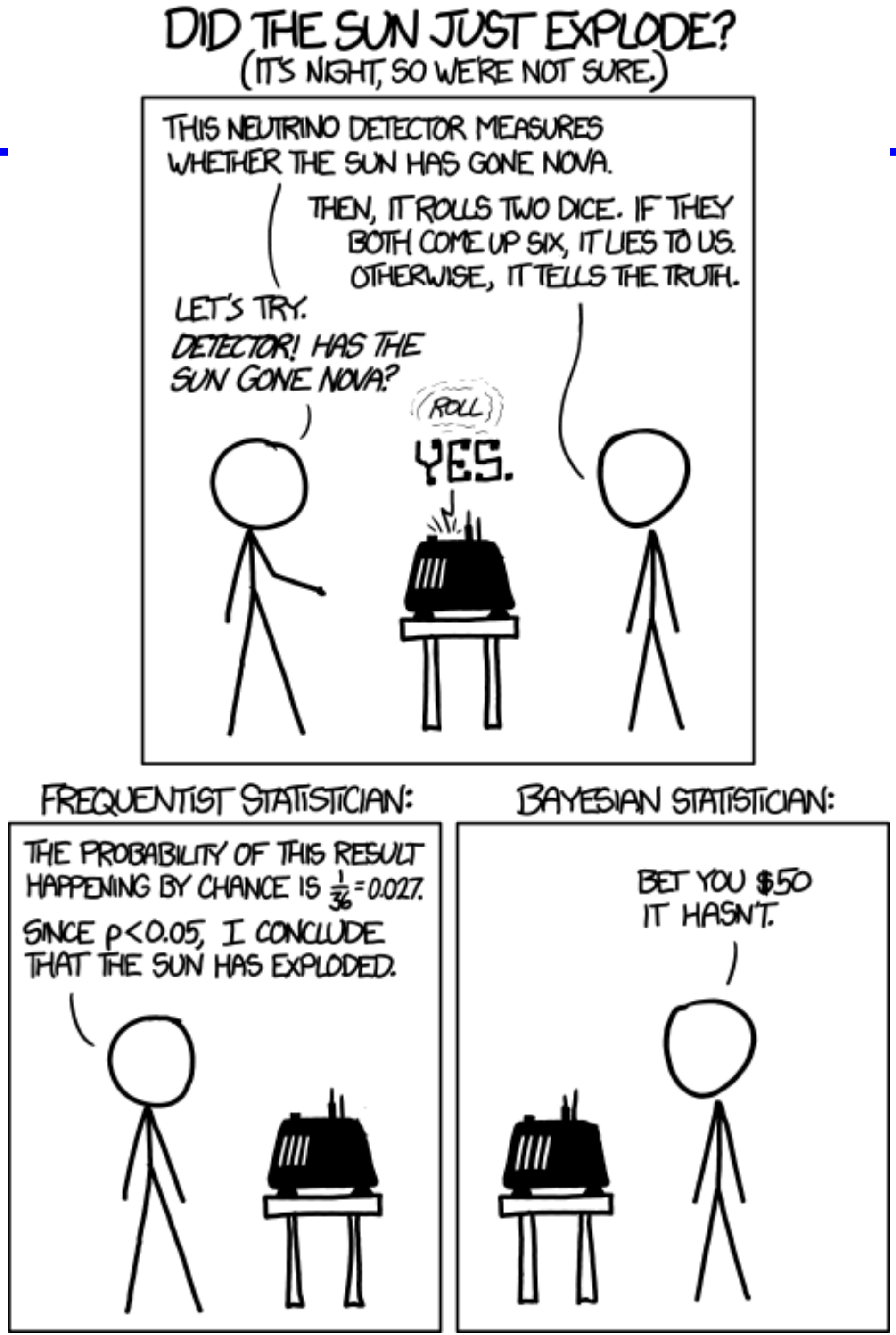
Hands-on session on statistical tools

Mario Pellicccioni

INFN School of Statistics 2015


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 $\frac{1}{36}$=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)

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/INFNStatRooStats

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/root/html534/ClassIndex.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RooFit manual (a bit outdated):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

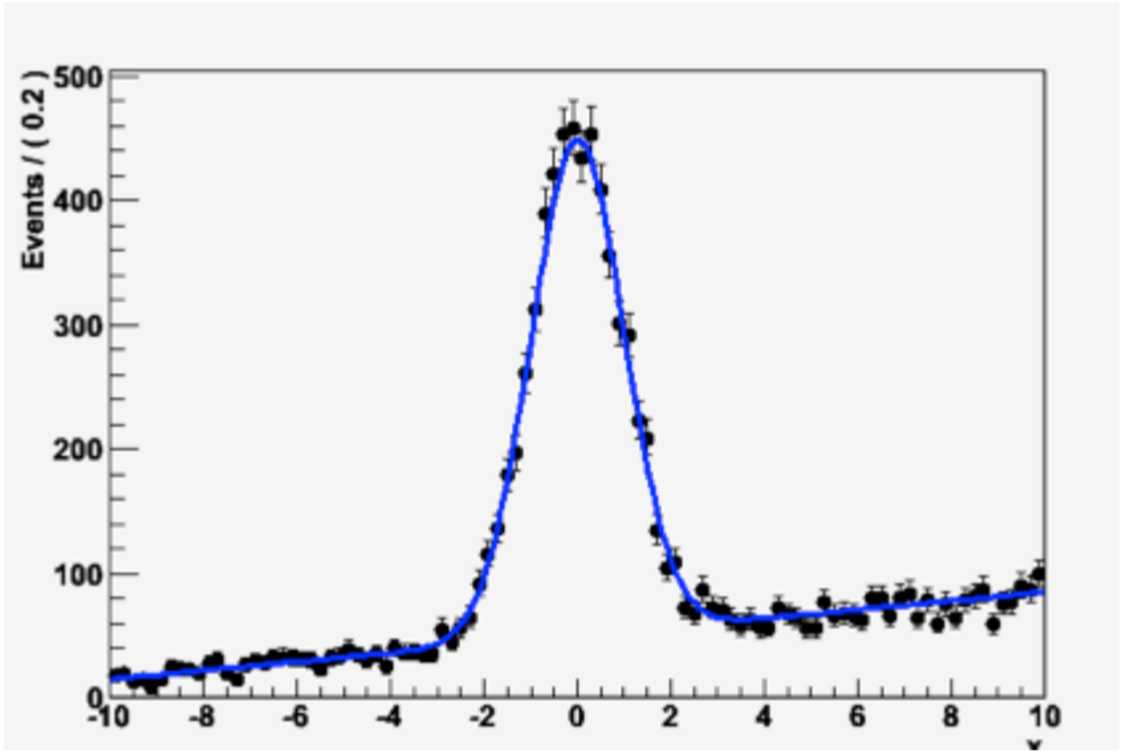
https://twiki.cern.ch/twiki/bin/view/RooStats/WebHome

More RooFit/RooStats examples

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: $e^+e^- \rightarrow \Upsilon(4s) \rightarrow BB$
  - Reconstruct both Bs, measure decay time difference
  - Physics of interest is in decay time dependent oscillation

$$f_{sig} \times \left[\mathrm{SigSel}(m;\vec{p}_{sig}) \times \left(\mathrm{SigDecay}(t;\vec{q}_{sig},\sin(2\beta)) \otimes \mathrm{SigResol}(t\,|\,dt;\vec{r}_{sig})\right)\right] +$$
$$(1-f_{sig})\left[\mathrm{BkgSel}(m;\vec{p}_{bkg}) \times \left(\mathrm{BkgDecay}(t;\vec{q}_{bkg}) \otimes \mathrm{BkgResol}(t\,|\,dt;\vec{r}_{bkg})\right)\right]$$

- 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

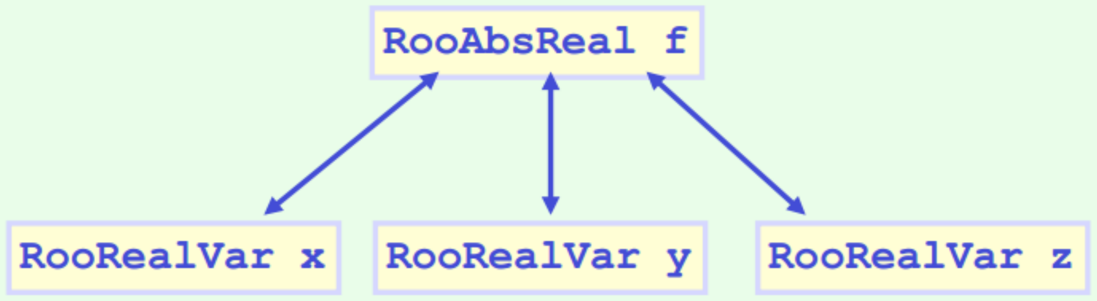
Design philosophy

- 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:

RooRealVar var1(“var1”,”My first var”,4.15);   //constant variable

RooRealVar var2(“var2””My second var”,1.,10.); //valid range, no initial value

RooRealVar var3(“var3””My third var”,3.,1.,10.); //valid range, initial value

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

RooRealVar time(“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.

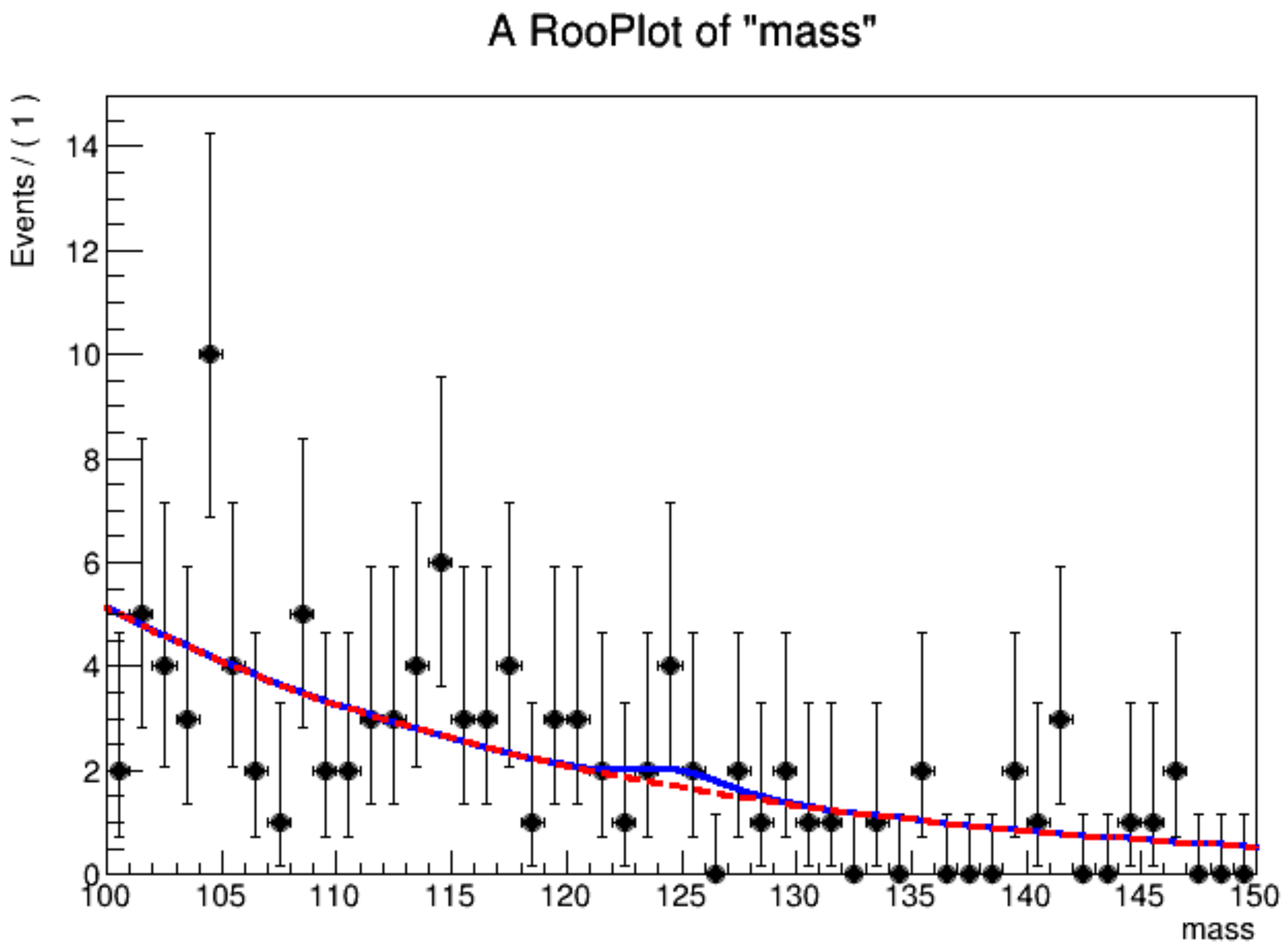
Exercise #0

The first exercise involves RooFit only

- Construct a signal + background PDF
  - Signal is a Gaussian function
  - Background is an exponential
- For now, it will involve a very small amount of signal events
- Generate a sample using this PDF
- Fit it, plot it, save it

We are going to use this macro all the way through the exercises, changing the S/B to use different statistical methods

Result of exercise #0



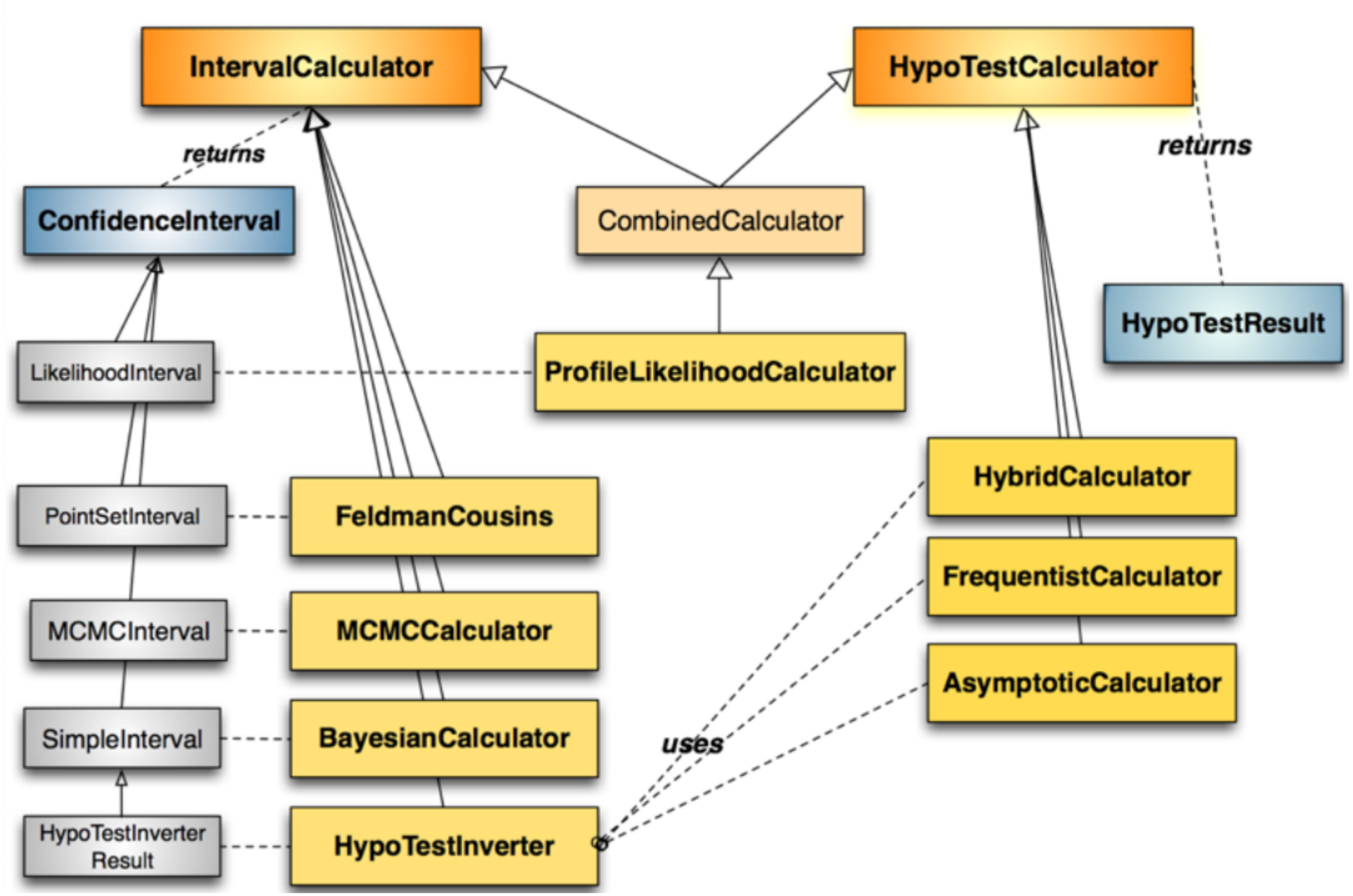
RooStats

Set of libraries for statistical interpretation of your results

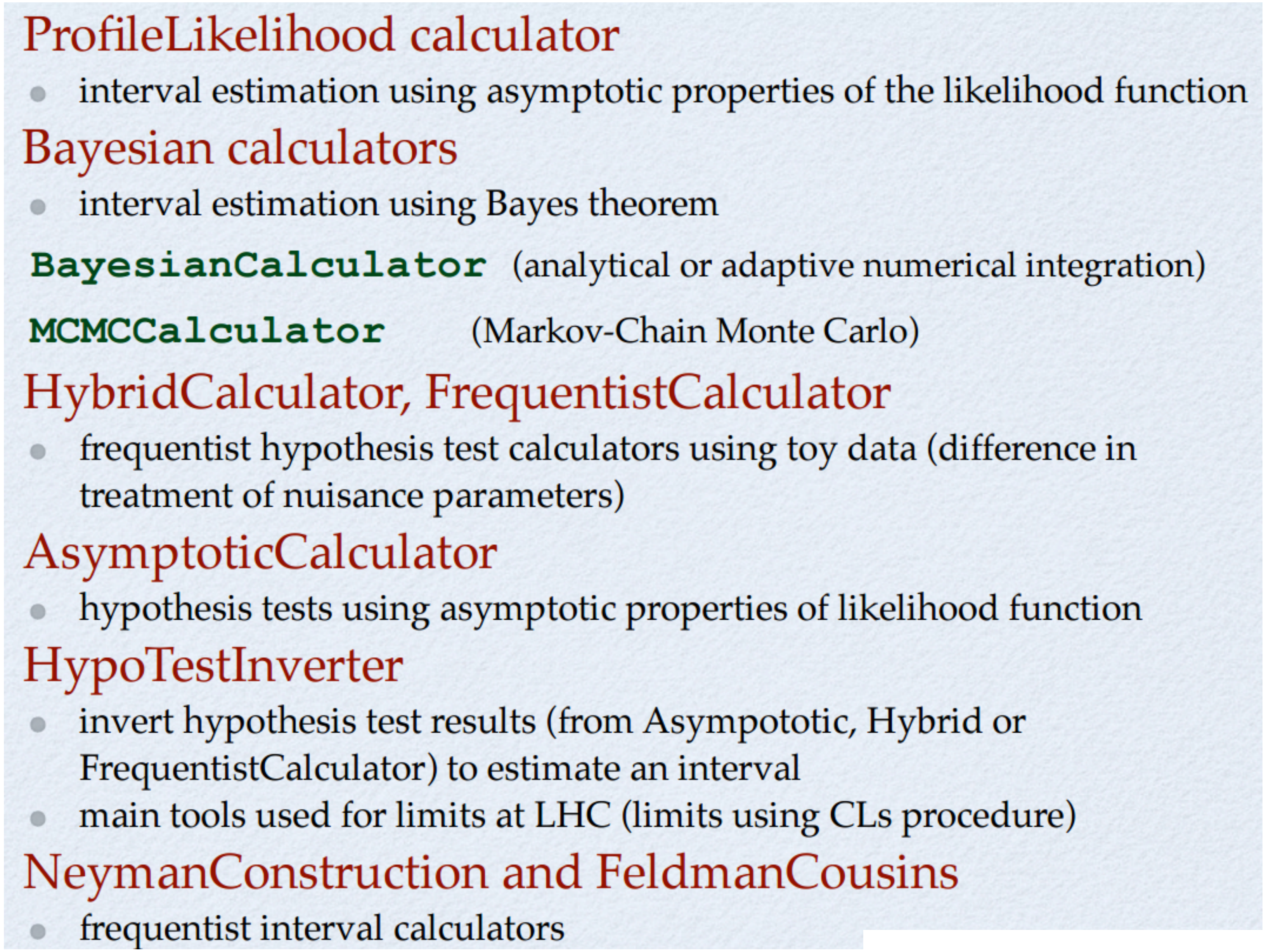
- “communicates” with RooFit via RooWorkspace
- Provides:
  - Significance estimates
  - Upper limit calculations
  - Confidence intervals
  - Probability intervals
  - Hypothesis testing

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 gave us a distribution where there’s clearly no significant signal present.

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)

Exercise #3

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

Rerun exercise_0.C with 50 signal events and 450 background events

Rerun exercise_1.C and observe the change in significance

Now we can measure the properties of our discovery

For example the mass!

Exercise #4

Up to now, all parameters but the parameter of interest were 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!

Adding a scale factor

Let's imagine your original likelihood was:

$$\mathcal{L} = \mathcal{L}(\theta_1, ..., \theta_i, ..., \theta_n)$$

modify your likelihood in this way

$$\mathcal{L} = \mathcal{L}(\theta_1, ..., \kappa_i \cdot \theta_i, ..., \theta_n)$$

and add a term to account for the uncertainty on the scale factor

$$\mathcal{L} = \mathcal{L}(\theta_1, ..., \kappa_i \cdot \theta_i, ..., \theta_n) \cdot Gauss(\kappa_i, b, \sigma_{\kappa_i,syst})$$

in this exercise, we consider b=1 (so no bias on the scale factor)