



Hands-on Session

RooStats Exercises

(Part 2)

Lorenzo Moneta

CERN

Grégory Schott

Karlsruhe Institute of Technology (KIT)

Hybrid Calculator



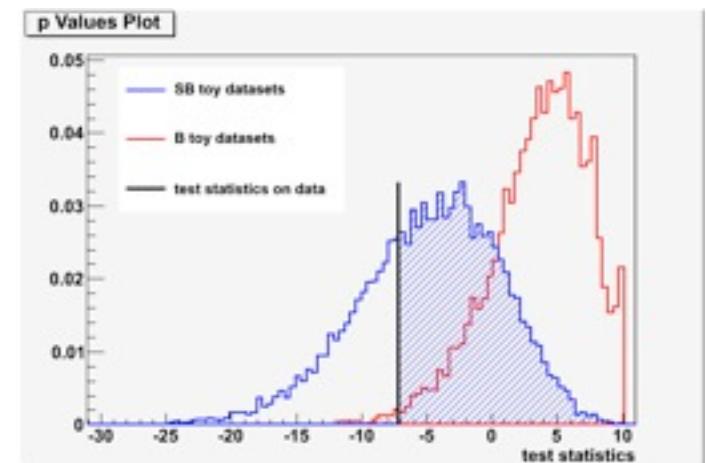
- ❖ Hypothesis test calculator (method used at LEP)
- ❖ uses by default the likelihood ratio as test statistics Q

$$Q(m_H) = \frac{L(s+b)}{L(b)}$$

- ❖ optionally one can use the number of events or the profile likelihood ratio

- ❖ generate toy MC experiments to obtained the distribution of Q
- ❖ From the empirical distribution obtained estimate the p-values:
 - ❖ CL_{sb} and CL_b
 - ❖ CL_s is obtained as the ratio CL_{sb} / CL_b

- ❖ Frequentist method
- ❖ Cousins-Highland integration over the nuisance parameters (Bayesian)
 - ❖ toys are generated with nuisance parameters smeared according to their distribution



Exercise 3: HybridCalculator



- ❖ Run HybridCalculator to perform an hypothesis test
 - ❖ test B only hypothesis vs S+B
 - ❖ set first value of all parameters S, B, sigSigma (e.g. initial or best fit values)
- ❖ For the HybridCalculator we need the background pdf, the pdf of the nuisance parameters (see “priorNuisance” in slide 8) and the nuisance parameters set
 - ❖ see http://root.cern.ch/root/htmldoc/RooStats_HybridCalculator.html

```
RooAbsPdf * nuisPdf = myWS.pdf("priorNuisance");
RooAbsPdf * modelBkg = myWS.pdf("modelBkg");
HybridCalculator * hc=new HybridCalculator(*data,*model,*modelBkg, &nuisPar, nuisPdf);
hc->SetNumberOfToys(5000);
// default is 1 likelihood ratio, 0 is number of events, 2 profile likelihood ratio
hc->SetTestStatistic(1);
```

- ❖ do the hypothesis test running the toys

```
HybridResult* hcresult=hc->GetHypoTest();
double clsb_data = hcresult->CLsplusb();
double clb_data = hcresult->CLb();
double cls_data = hcresult->CLs();
double data_significance = hcresult->Significance();
```

- ❖ plot the result (test statistic distributions

```
// plot the result: distribution of test statistic for null and alternate hypothesis
HybridPlot* hcPlot=hcresult->GetPlot("hcPlot","p Values Plot",100);
hcPlot->Draw();
```

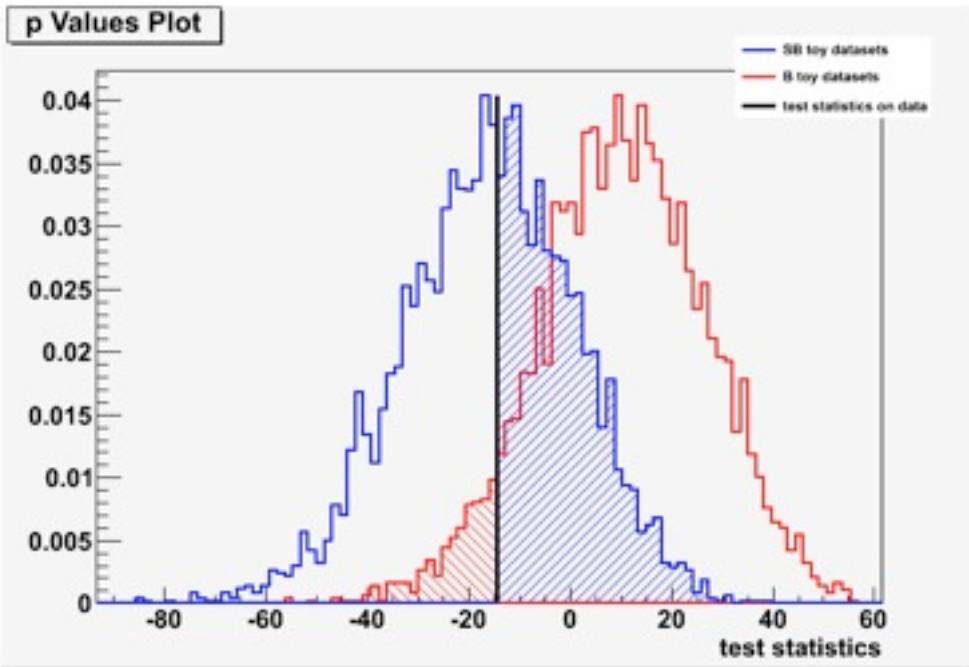
Exercise 3: Hybrid Result



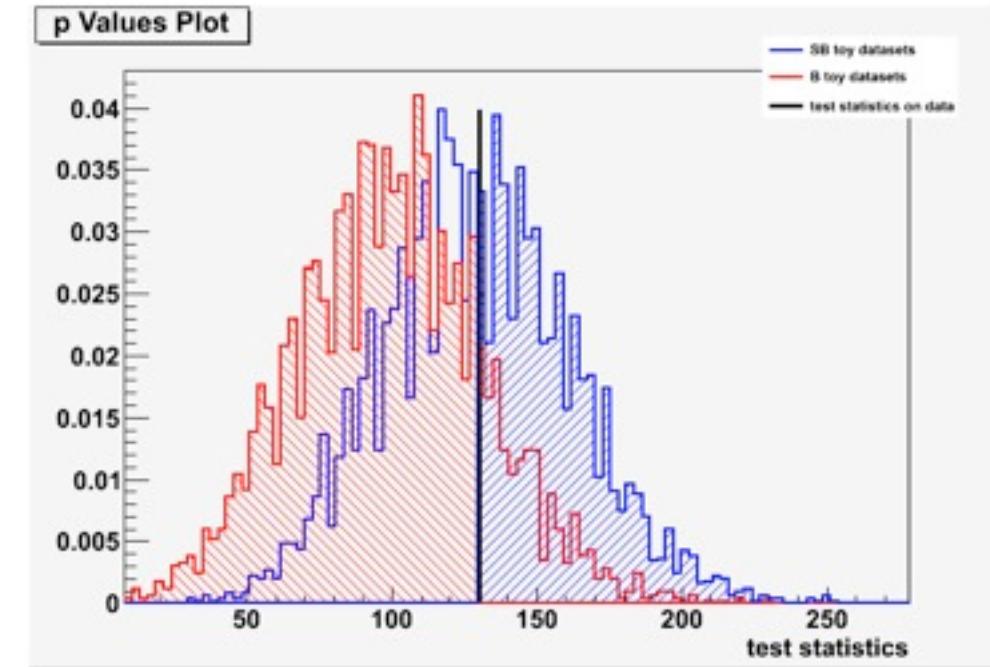
- Result obtained (remember to set initial values correctly)

5000 toys, initial parameter values, test statistic:

likelihood ratio



number of events



HYBRID CALCULATOR result:

CL_b: 0.9322
 CL_s: 0.505471
 CL_sb: 0.4712
 significance: 1.49238

HYBRID CALCULATOR result:

CL_b: 0.836
 CL_s: 0.614593
 CL_sb: 0.5138
 significance: 0.97815

Exercise 4: BayesianCalculator



- ◆ Run BayesianCalculator for interval in parameter of interest
 - ◆ class works only for 1D, can use MCMCCalculator for multi-dimensional problems
- ◆ Build from data, model and prior for p.o.i.
 - ◆ see http://root.cern.ch/root/html/doc/RooStats_BayesianCalculator.html
- ◆ Need first to define a prior for p.o.i.
 - ◆ Example in case of Poisson model define uniform prior

```
myWS.factory("Uniform::priorPOI(S)");
BayesianCalculator bcalc(*data,*model,poi,*priorPOI,&nuisPar);
```

- ◆ normally would get interval by doing

```
bcalc.SetConfidenceLevel(0.68);
SimpleInterval * bInterval = bcalc.GetInterval();
lowerLimit = bInterval->LowerLimit(); upperLimit = bInterval->UpperLimit();
```

- ◆ but there is a bug in 5.25.04 (sorry!). Workaround, use posterior pdf and ROOT class TF1 (16% and 84% quantiles are the interval limits)

```
RooAbsPdf * posteriorPdf = bcalc->GetPosteriorPdf(); // get the posterior PDF
TF1 * f1 = posteriorPdf->asTF(poi);
double q[2]; double pval[2];
pval[0] = 0.16; pval[1] = 0.84; // define the quantiles for a 68% interval (16% and 84%)
f1->GetQuantiles(2,q,pval);
f1->DrawClone(); // for drawing the posterior pdf
std::cout << "68% CL interval: [ " << q[0] << " - " << q[1] << " ] \n";
```

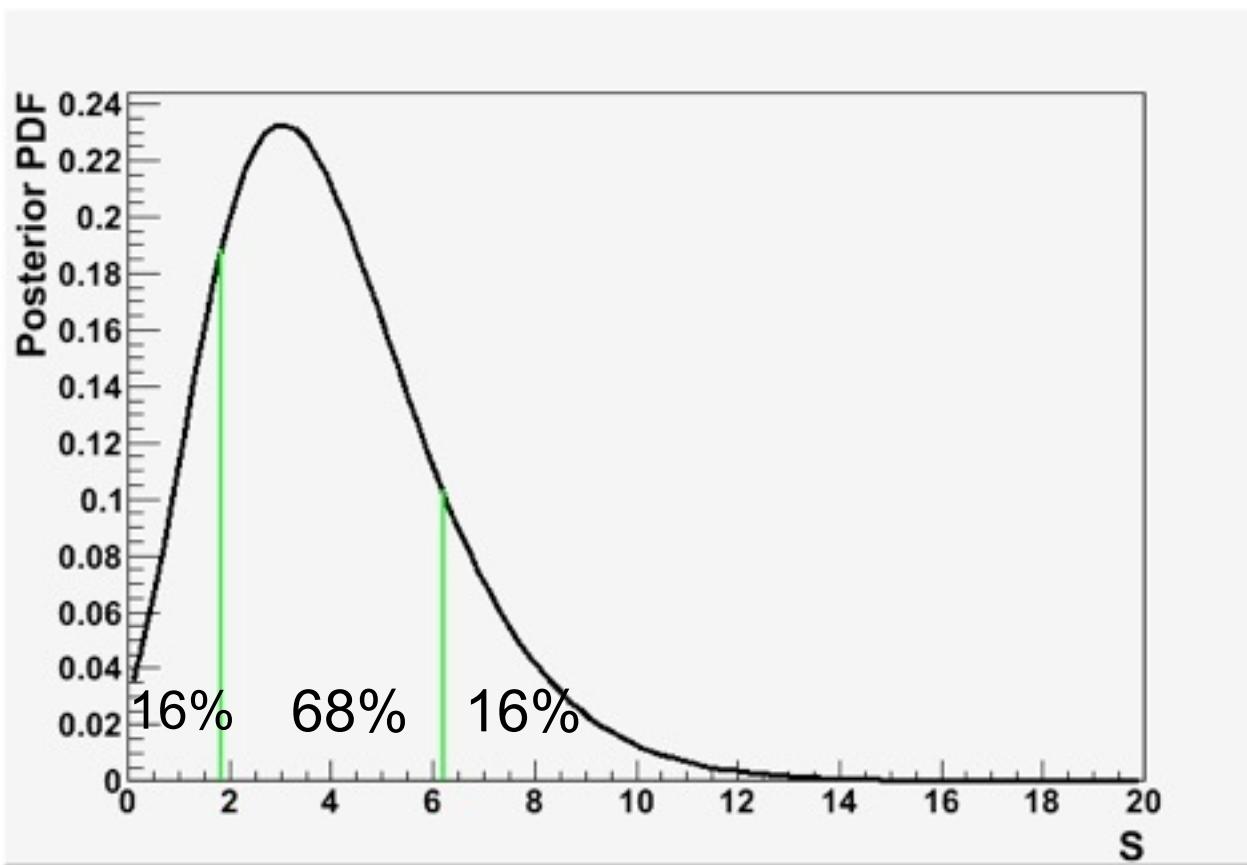
Exercise 4: Result



- ★ Can draw the posterior and its interval (when bug will be fixed)

```
RooPlot * bplot = bcalc.GetPosteriorPlot();
bplot->Draw();
```

- ★ Draw for the moment the TF1 with interval limit



68% CL interval: [1.82826 - 6.19458]

Exercise 4b: MCMCCalculator



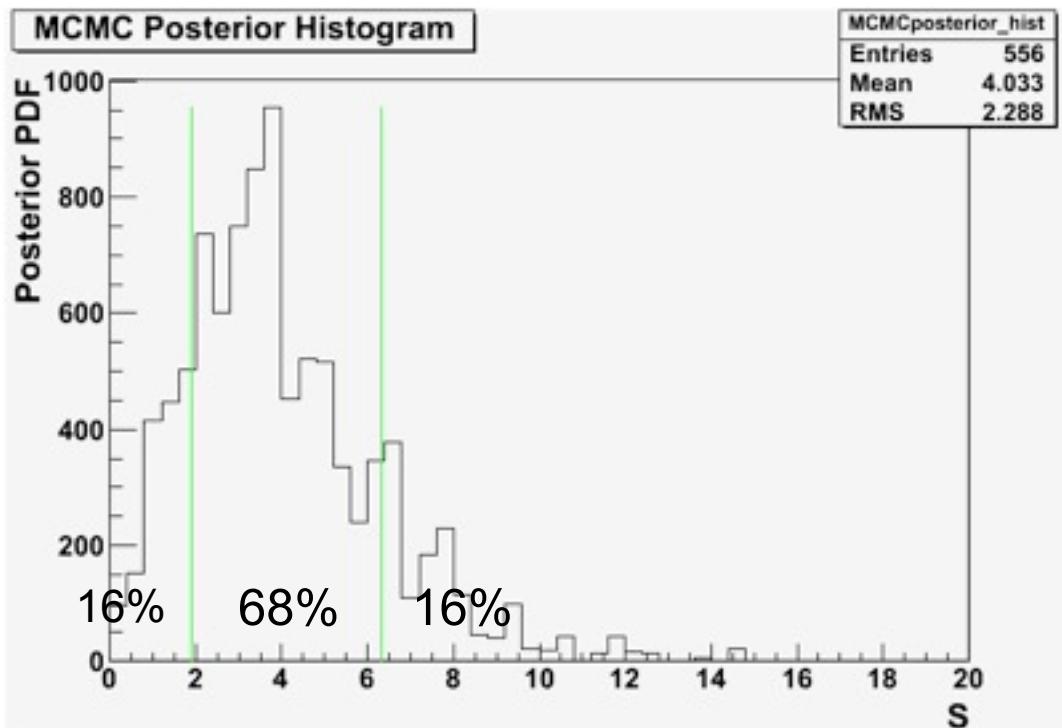
- Run same example as before but using the MCMC calculator

- see http://root.cern.ch/root/html/doc/RooStats_MCMCCalculator.html

```
MCMCCalculator mccalc(*data,*model,poi,*priorPOI);
mccalc.SetConfidenceLevel(0.68);
MCMCInterval * mcInterval = mcCalc.GetInterval();
```

- Also for a bug in LowerLimit/UpperLimit use posterior histogram as before

```
TH1* posterior = mcInterval->GetPosteriorHist();
posterior->GetQuantiles(2,q,p);
```



68% CL interval:
[1.90416 - 6.30061]

with default MC parameters

```
mccalc.SetNumIter(100000);
mccalc.SetNumBins(50);
```

Solutions



- ❖ Download the tar file :
 - ❖ http://www.cern.ch/moneta/temp/exercises_2.tar.gz
 - ❖ contains also macro for next exercises
- ❖ Poisson model:
 - ❖ likelihood analysis + bayesian calculators:
 - ❖ PoissonModel_full.C
- ❖ Gaussian over flat background model:
 - ❖ likelihood analysis and hybrid calculator
 - ❖ GaussianModel_full.C
- ❖ Next exercises:
 - ❖ PoissonModel_FC.C (FeldmanCousins)
 - ❖ rs801_HypoTestInverter.C
 - ❖ GaussianModel_VarMass.C (3 methods with Gaussian model and floating mass)

Exercise 5: Feldman-Cousins



◆ Run Feldman-Cousins calculator

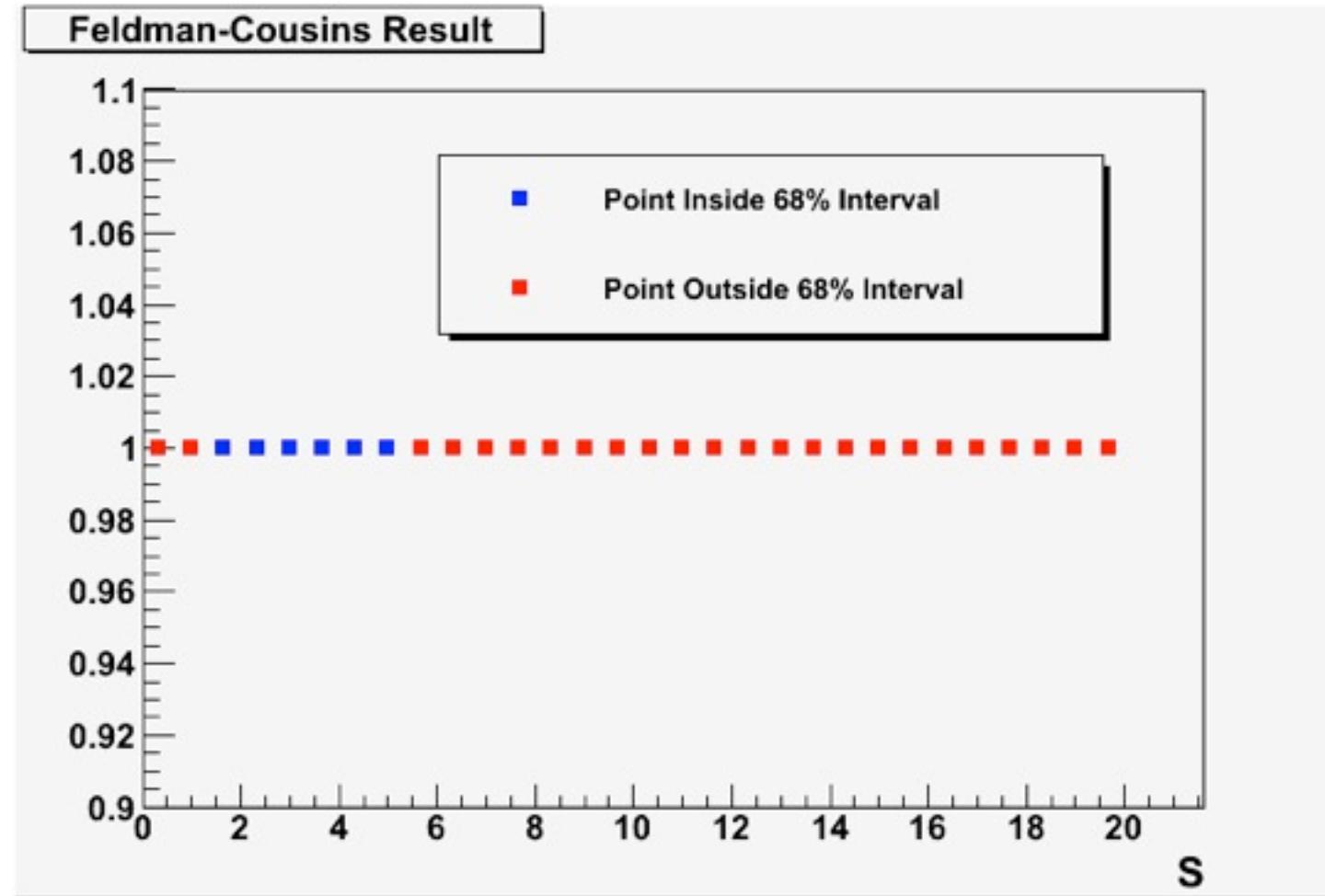
- ◆ see http://root.cern.ch/root/html/doc/RooStats_FeldmanCousins.html

```
FeldmanCousins fc;
fc.SetData(*data);
fc.SetPdf(*model);
fc.SetParameters(poi);
// set the distribution creator, which encodes the test statistic
fc.SetConfidenceLevel(.68); // set size of test
fc.UseAdaptiveSampling(true);
// number counting analysis: dataset always has 1 entry with N events observed
fc.FluctuateNumDataEntries(false);
fc.SetNBins(30); // number of points to test per parameter
```

```
// Result is a ConfInterval class
ConfInterval * fcInterval = fc.GetInterval();
```

```
// can only scan point to test if inside or outside:
RooAbsData * parameterScan = fc.GetPointsToScan();
for(Int_t i=0; i<parameterScan->numEntries(); ++i){
    const RooArgSet * point = parameterScan->get(i);
    double svalue = point->getRealValue("S");
    cout << "S value " << svalue;
    if (fcInterval->IsInInterval(*point) )
        cout << " is inside \n";
    else
        cout << " is outside \n";
}
```

Feldman-Cousins Result



- see code in macro PoissonModel_FC.C to produce the plot
- can verify (if you wish) with class TFeldmanCousins

```
TFeldmanCousins fc(0.68);
ll = fc.CalculateLowerLimit(nobs, nexp); ul = fc.CalculateUpperLimit(nobs, nexp);
```



◆ Run HypoTest Inverter calculator

- ◆ see http://root.cern.ch/root/html/doc/RooStats_HypoTestInverter.html

```
// create first an hybrid calculator (no systematics)
HybridCalculator hc(*data,*model,*modelBkg);

// after having configure the hybrid calculator can create hypotest inverter
HypoTestInverter myInverter("myInverter",&hc,S);
// optionally use CLS
// myInverter.UseCLs(true);
```

◆ run the scan

```
// scan fixed (in a grid or automatically)
// myInverter.RunFixedScan(5,1,6);
double target = 0.05; // for a 95% CL upper limit
myInverter.RunAutoScan(1,6,target,0.005); //
//myInverter.RunOnePoint(3.9);

// get result and plot
HypoTestInverterResult* results = myInverter.GetInterval();
std::cout << "The computed upper limit is: " << results->UpperLimit() << std::endl;

HypoTestInverterPlot myInverterPlot("myInverterPlot","",results);
TGraphErrors* gr1 = myInverterPlot.MakePlot();
gr1->Draw("ALP");
```

◆ see macro PoissonModel_HypoTestInv.C

Note on Poisson Model generation



- ⊕ For running the Hybrid calculator and the HypoTestInverter with a Poisson model you need to generate the events in a slightly different way in RooFit
 - ★ create a dummy observable x (with a uniform pdf)
 - ★ generate (and fit) events using Extended()
 - ◆ i.e. with Poisson fluctuation on the total number

```
// define observable x
myWS.factory("x[0,0,1]" );
myWS.factory("Uniform::sigPdf(x)" );
myWS.factory("Uniform::bkgPdf(x)" );
RooAbsPdf * model = (RooAbsPdf*) myWS.factory("SUM::model(S[2,0,20]*sigPdf,B[1,0,10]*bkgPdf)" );
// Background only pdf
RooAbsPdf * modelBkg = (RooAbsPdf*) myWS.factory("ExtendPdf::modelBkg(bkgPdf,B)" );

// Background distribution (if systematics) for background
RooAbsPdf * priorBkg = (RooAbsPdf *) myWS.factory("Gaussian::priorBkg(B,1,0.5)");
RooArgSet nuisPar(*myWS.var("B"));

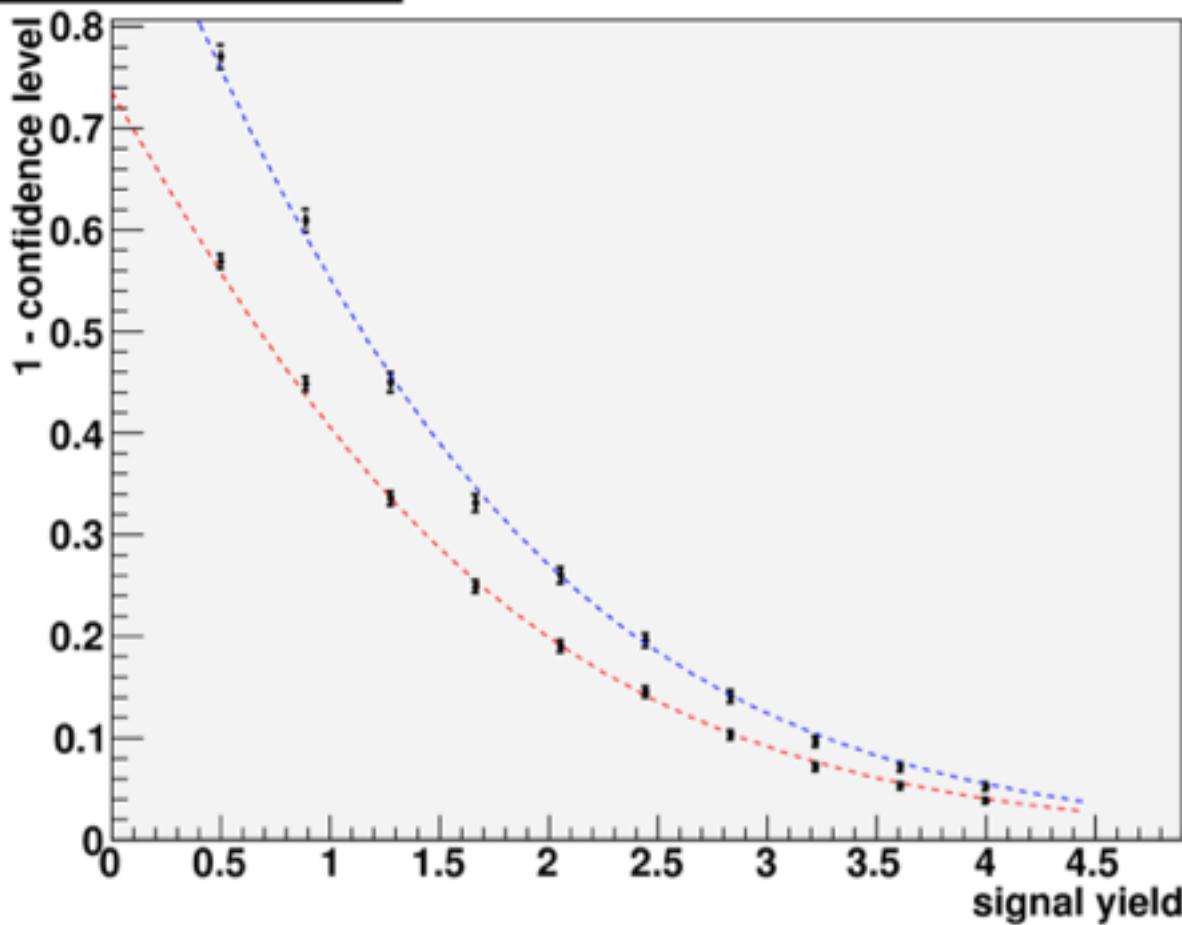
// to generate always different event use seed = 0 (default)
(RooRandom::randomGenerator())->SetSeed(seed);

// Generate data (one bin but has Poisson fluctuation)
RooAbsData* data = myWS.pdf("model")->generate(*myWS.var("x"),Extended(),Name("data"));
int nobs = data->numEntries();
std::cout << "number of observed events " << nobs << std::endl;
```

- ★ see macro PoissonModel_HypoTestInv.C
- ★ run HybridCalculator and HypoTestInverter

HypoTestInvertor::RunFixedScan

HypoTestInvertor example



1 measured event, 1 expected event

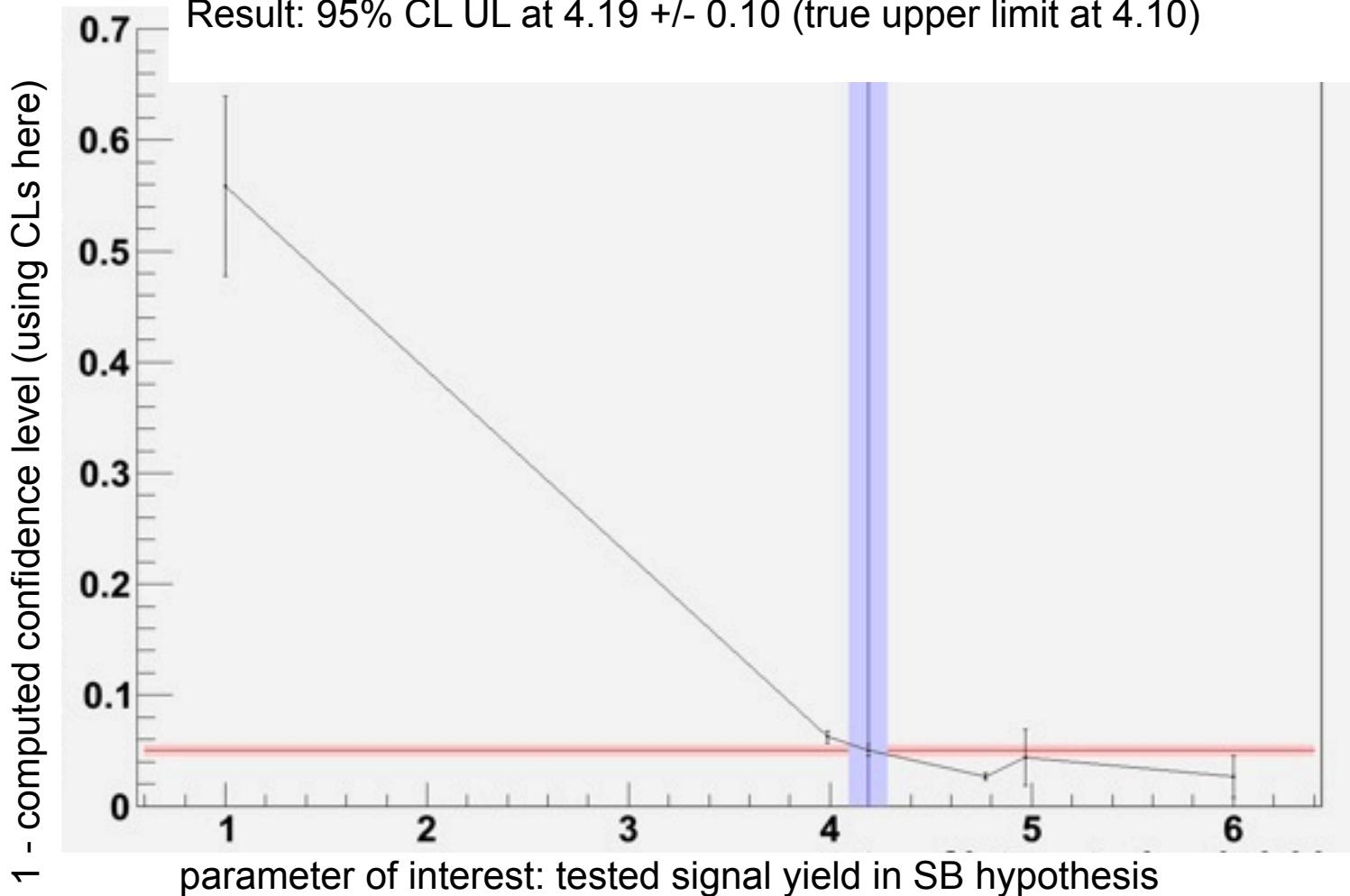
Ran with UseCLs() – blue and UseCLs(false) – red

Toy hypothesis tests and analytically computed p-values compared

95% CL UL on signal yield:
- 3.74 with CLsb
- 4.10 with CLs
reproduced by the class

HypoTestInvertor::RunAutoScan

Performs 3 iterations with low number of toys to quickly locate the upper limit
The increase the number of toys to decrease the error bars
3 more iterations to locate precisely the upper limit
Propagate errors on CLs to the upper limit (using slope of the curve there)
Result: 95% CL UL at 4.19 ± 0.10 (true upper limit at 4.10)



Exercise 7: Floating Mass



- ◆ Similar Gaussian over flat background model but with floating mass
- ◆ modify GaussianModel.C as follows:
 - ◆ add a new parameter of interest, trueMass

```
// Observable  
myWS.factory("mass[0,500]" );  
  
// Pdf of signal  
myWS.factory("Gaussian::sigPdf(mass,trueMass[200,150,300],sigSigma[0,100])" );
```

- ◆ add also prior for the new parameter (for Bayesian analysis)

```
// Prior for signal  
myWS.factory("Uniform::priorS(S)" );  
myWS.factory("Uniform::priorMass(trueMass)" );  
myWS.factory("PROD::priorPOI(priorS,priorMass)" );
```

- ◆ create new POI set

```
// POI set  
RooArgSet poi(*myWS.var("S"), myWS.var("trueMass")) ;
```

- ◆ Can run as before likelihood method

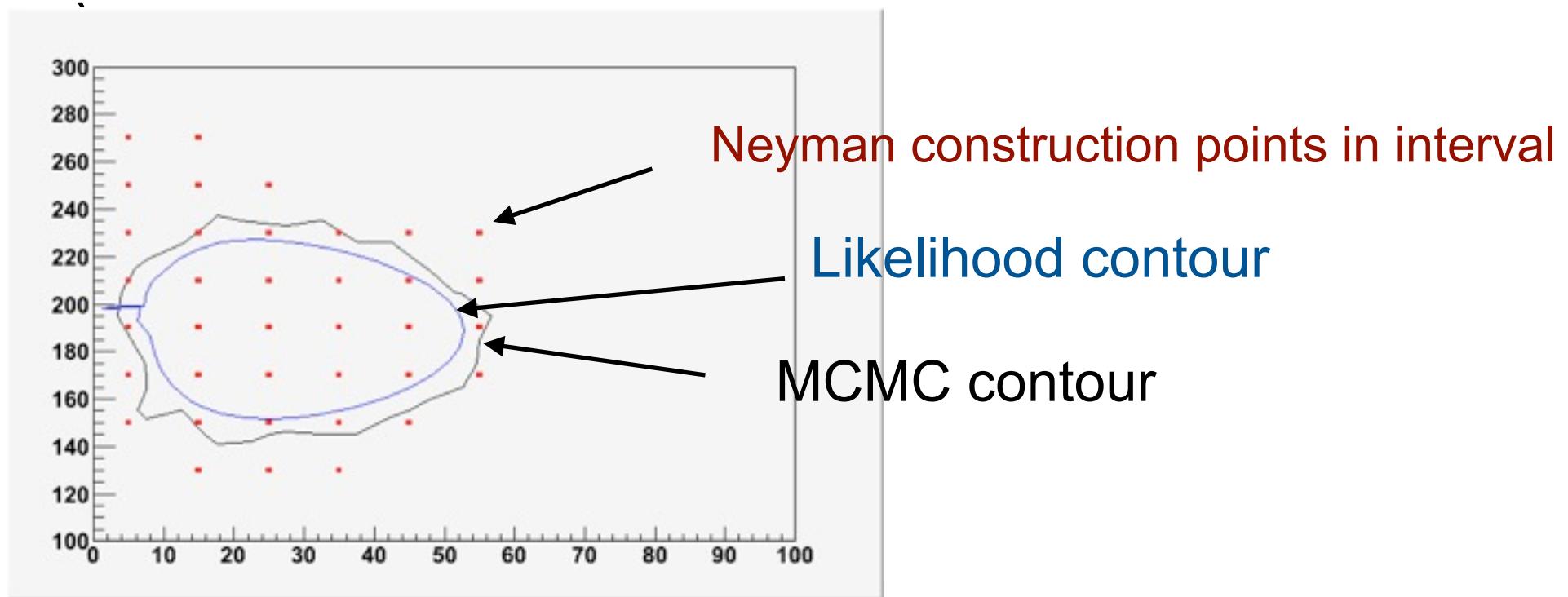
```
// create Profile Likelihood Calculator (with 68% CL, can be set also later)  
ProfileLikelihoodCalculator plc(*data,*model,poi,0.68);
```

- ◆ lplot.Draw() will produce now a contour plot

Exercice 6: Floating Mass , 3 methods



- ❖ Run on the floating mass example:
 - ❖ profile likelihood
 - ❖ MCMCCalculator
 - ❖ FeldmanCousins
- ❖ Should get a plot similar to this (90% CL):



see macro GaussianModel_VarMass.C for making the plot

RooFit interface to BAT package

Interface <http://cern.ch/schott/public/BCRooInterface/>
to the Bayesian Analysis Toolkit <http://www.mppmu.mpg.de/bat/>

Here is how to install BAT (on Ixplus using the tcsh shell):

```
wget http://www.mppmu.mpg.de/bat/source/BAT-0.3.1.tar.gz  
tar xvfz BAT-0.3.1.tar.gz  
cd BAT-0.3.1  
.configure --prefix=${PWD}  
make  
make install  
setenv BATINSTALLDIR ${PWD}  
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:  
    ${BATINSTALLDIR}/lib  
setenv CPATH ${BATINSTALLDIR}/include  
cd models/RooInterface  
make
```

To run the same study but with BAT, copy then
the input ROOT file (containing the workspace, ie.
`WS_GaussOverFlat_withSystematics_floatingMass.root`) in the `RooInterface`
directory and run with:

```
./runRooInterface.exe WS_GaussModel.root myWS  
gv bat_plots.ps
```

