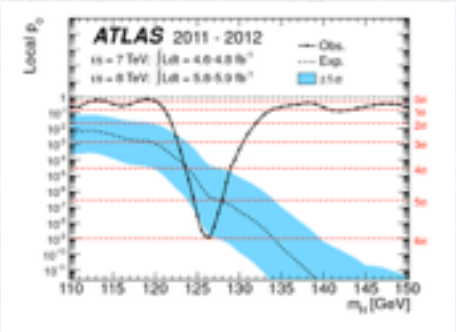
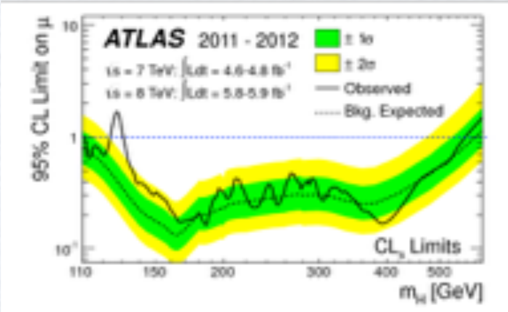
RooStats

Lecture and Tutorials





Outline

- Introduction to Fitting in ROOT
- Introduction to RooFit
  - Basic functionality and model building using the workspace
  - Composite models
- Exercises on RooFit: building and fitting models
- RooStats:
  - Introduction
  - Interval estimation tools (Likelihood/Bayesian)
  - Exercises on interval/limit estimation
- Hypothesis Test
- Frequentist interval/limit calculator (CLs)
  - Exercises on frequentist interval/limit estimation and discovery significance (hypothesis test)
- Building models with the HistFactory tool

RooStats Goal

- Common framework for statistical calculations
  - work on arbitrary models and datasets
    - factorize modeling from statistical calculations
  - implement most accepted techniques
    - frequentists, Bayesian and likelihood based tools
  - possible to easy compare different statistical methods
  - provide utility for combinations of results
  - using same tools across experiments facilitates the combinations of results

Statistical Applications

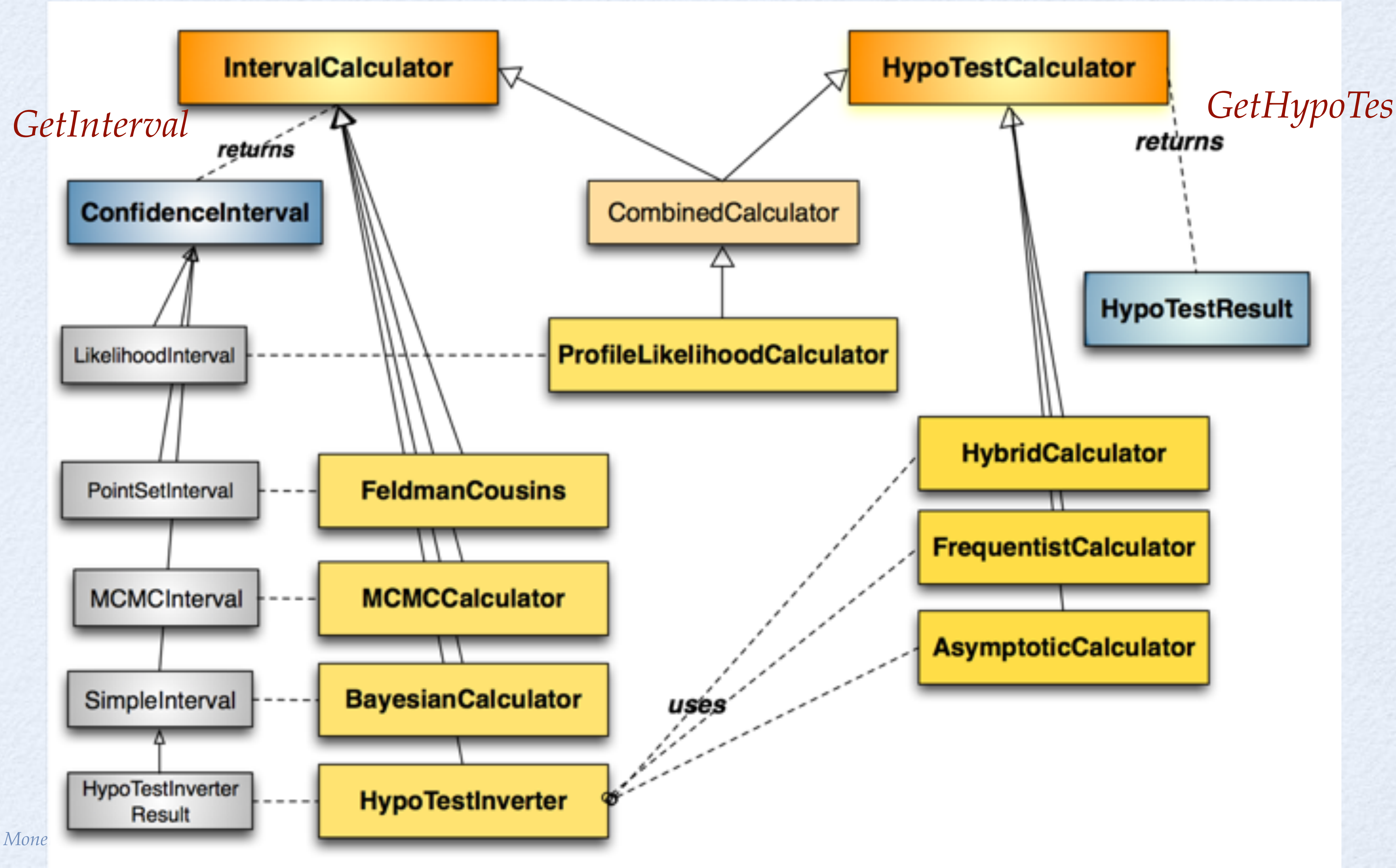
- Statistical problems:
  - point estimation (covered by RooFit)
  - estimation of confidence (credible) intervals
  - hypothesis tests
  - goodness of fit (not addressed)

RooStats Technology

- Built on top of RooFit
  - generic and convenient description of models (probability density function or likelihood functions)
  - provides *workspace* (RooWorkspace)
    - container for model and data and can be written to disk
    - inputs to all RooStats statistical tools
    - convenient for sharing models (e.g digital publishing of results)
  - easily generation of models (workspace factory and HistFactory tool)
  - tools for combinations of model (e.g. simultaneous pdf)
- Use of ROOT core libraries:
  - minimization (e.g. Minuit), numerical integration, etc...
  - additional tools provided when needed (e.g. Markov-Chain MC)

RooStats Design

- C++ interfaces and classes mapping to real statistical concepts



RooStats Interfaces

- **IntervalCalculator**
  - built from a model (workspace + ModelConfig) and data set
  - has the function:
    - `ConfInterval * GetInterval();`
- **ConfInterval**
  - built from a given confidence level
  - `bool IsInInterval(const RooArgSet * point)` can tell if a point is inside or outside the interval

RooStats Interfaces (2)

- **HypoTestCalculator**
  - built from a data set and null and alternate models (e.g. background and signal plus background)
    - model can be common and defined only by different parameter values (S = 0 and S = Standard Model)
  - has the function:
    - `HypoTestResult * GetHypoTest();`
- **HypoTestResult**
  - `double NullPValue();` `double Significance();`
  - `double AlternatePValue();`
  - `SamplingDistribution * GetAlt/NullDistribution();`
    SamplingDistribution is the sampled test statistic distribution

RooStats Calculator classes

Interval Calculators

- ProfileLikelihoodCalculator
  - interval estimation using asymptotic properties of the likelihood function
    - also an hypothesis test calculator (same as AsymptoticCalculator)
- BayesianCalculator
  - interval estimation based on Bayes theorem using adaptive numerical integration
- MCMCCalculator
  - Bayesian calculator using Markov-Chain Monte Carlo
- HypoTestInverter
  - invert hypothesis test results to estimate an interval
    - CLs limits, FC interval
- NeymanConstruction and FeldmanCousins
  - frequentist interval calculators

HypoTest Calculators

- HybridCalculator, FrequentistCalculator
  - frequentist hypothesis test calculators using toy data (difference in treatment of nuisance parameters)
- AsymptoticCalculator
  - hypothesis tests using asymptotic properties of likelihood function

ModelConfig Class

- `ModelConfig` class input to all Roostats calculators
  - contains a reference to the RooFit workspace class
  - provides the workspace meta information needed to run RooStats calculators
    - pdf of the model stored in the workspace
    - what are observables (needed for toy generations)
    - what are the parameters of interest and the nuisance parameters
    - global observables (from auxiliary measurements) for frequentist calculators
    - prior pdf for the Bayesian tools
  - ModelConfig can be imported in workspace for storage and later retrieval

Building ModelConfig Class

- ModelConfig must be built after having the workspace
- Identify all the components which are present in the workspace

```
//specify components of model for statistical tools
ModelConfig  modelConfig("G(x|mu,1)");
modelConfig.SetWorkspace(workspace);
//set components using the name of ws objects
modelConfig.SetPdf( "normal");
modelConfig.SetParameterOfInterest("poi");
modelConfig.SetObservables("obs");
```

- Some tools (Bayesian) require to specify prior pdf

```
//Bayesian tools would also need a prior
modelConfig.SetPriorPdf( "prior");
```

- ModelConfig can be imported in a workspace to be then stored in a file

```
//can import modelConfig into workspace too
workspace.import(*modelConfig);
```

Profile Likelihood Calculator

- Method based on properties of the likelihood function
- Profile likelihood function:

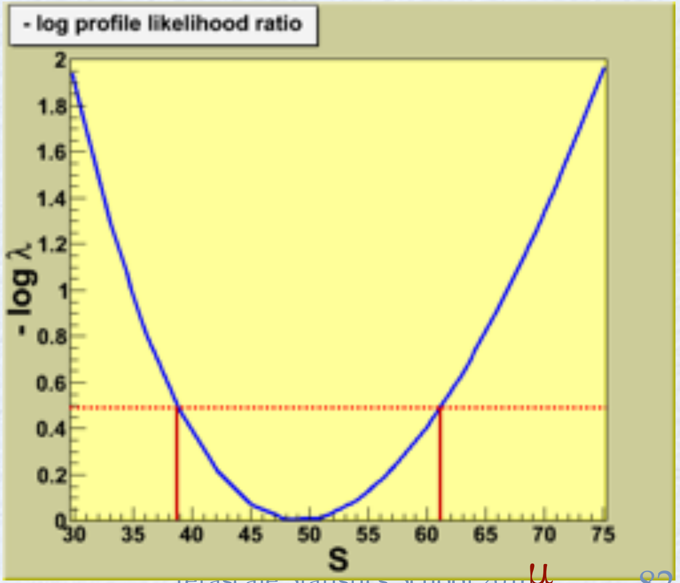
$$\lambda(\mu) = \frac{L(x|\mu, \hat{\hat{\nu}})}{L(x|\hat{\mu}, \hat{\nu})}$$

maximize w.r.t nuisance parameters ν and fix POI μ

maximize w.r.t. all parameters

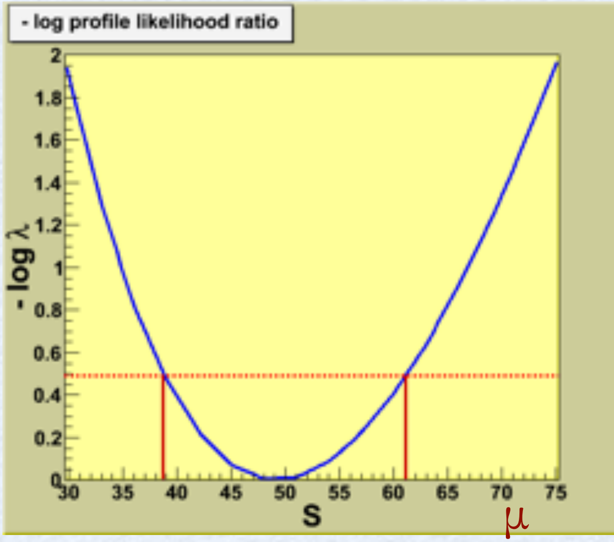
λ is a function of only the parameter of interest μ

- Uses asymptotic properties of λ based on Wilks' theorem:
- from a Taylor expansion of logλ around the minimum:
  - ➔ -2logλ is a parabola (λ is a gaussian function)
  - ➔ interval on μ from logλ values
- Method of MINUIT/MINOS
  - lower/upper limits for 1D
  - contours for 2 parameters



Using the Profile Likelihood Calculator

```
// create the class using data and model
ProfileLikelihoodCalculator plc(*data, *model);

// set the confidence level
plc.SetConfidenceLevel(0.683);

// compute the interval
LikelihoodInterval* interval = plc.GetInterval();
double lowerLimit = interval->LowerLimit(*mu);
double upperLimit = interval->UpperLimit(*mu);

// plot the interval
LikelihoodIntervalPlot plot(interval);
plot.Draw();
```



- For one-dimensional intervals:
  - 68% CL (1 σ) interval : $\Delta\log\boldsymbol{\lambda} = 0.5$
  - 95% CL interval : $\Delta\log\boldsymbol{\lambda} = 1.96$
- **`LikelihoodIntervalPlot`** can plot the 2D contours

Bayesian Analysis in RooStats

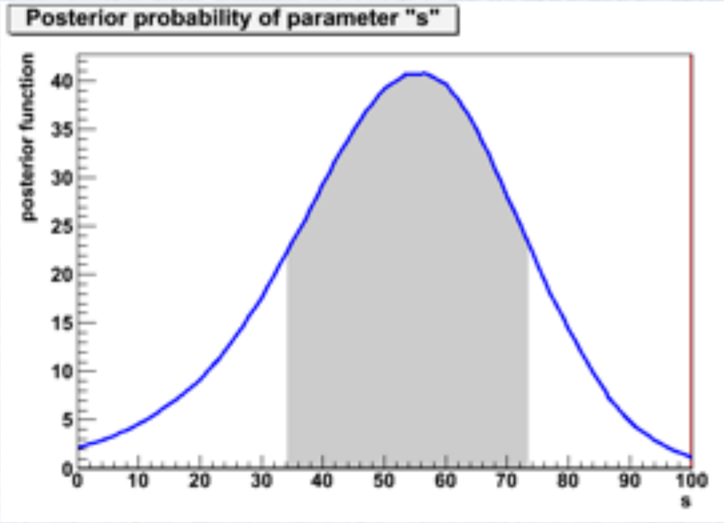
- `RooStats` provides classes for
  - marginalize posterior and estimate credible interval

posterior probability — likelihood function — prior probability — nuisance parameters marginalization

$$\underset{\text{POI} \quad \text{data}}{P(\mu|x)} = \frac{\int L(x|\mu,\nu)\Pi(\mu,\nu)d\nu}{\underbrace{\int\int L(x|\mu,\nu)\Pi(\mu,\nu)d\mu d\nu}_{\text{normalisation term}}}$$

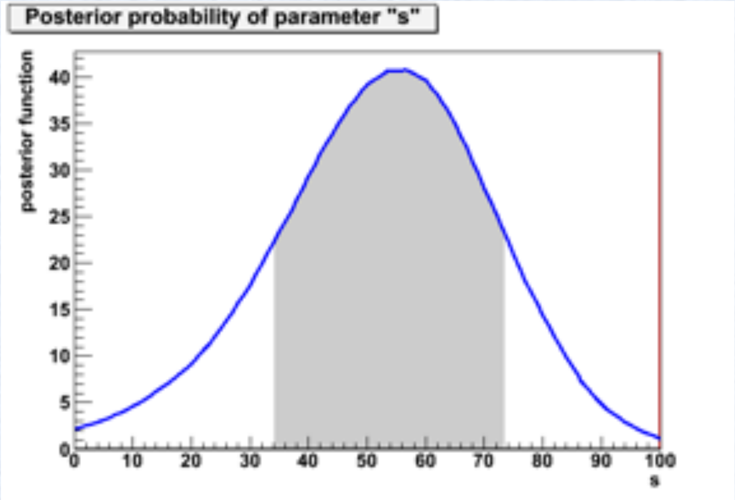
Bayesian Theorem

  - support for different integration algorithms:
    - adaptive (numerical)
    - MC integration
    - Markov-Chain
  - can work with models with many parameters (e.g few hundreds)



Bayesian Classes

- **`BayesianCalculator`** class
  - posterior and interval estimation using numerical integration
  - working only for one parameter of interest but can integrate (marginalize) many nuisance parameters
  - support for different integration algorithms, using **`BayesianCalculator::SetIntegrationType`**
    - adaptive numerical (default type), working only for few nuisances (< 10)
    - Monte Carlo integration (PLAIN, MISER, VEGAS)
    - TOYMC : average from toys where the nuisance parameters are sampled from a given p.d.f. (nuisance pdf), but can work in model with many parameters
  - can compute:
    - central interval
    - one-sided interval (upper limit)
    - a shortest interval
  - provide plot of posterior and interval



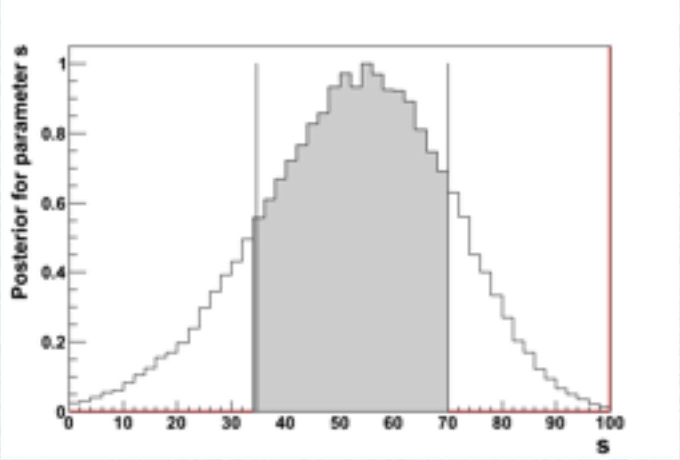
Example: 68% CL central interval

```
BayesianCalculator bc(data, model);
bc.SetConfidenceLevel(0.683);
bc.SetLeftSideTailFraction(0.5);
bc.SetIntegrationType("ADAPTIVE");
SimpleInterval* interval = bc.GetInterval();
double lowerLimit = interval->LowerLimit();
double upperLimit = interval->UpperLimit();
RooPlot * plot = bc.GetPosteriorPlot();
plot->Draw();
```

MCMC Calculator

- **`MCMCCalculator`** class
  - integration using Markov-Chain Monte Carlo (Metropolis Hastings algorithm)
  - can deal with more than one parameter of interest
  - can work with many nuisance parameters
    - e.g. used in Higgs combination with more than 300 nuisances
  - possible to specify ProposalFunction
    - multivariate Gaussian from fit result
    - Sequential proposal
  - can visualize posterior and also the chain result

MCMCCalculator




```
MCMCCalculator mc(data, model);
mc.SetConfidenceLevel(0.683);
mc.SetLeftSideTailFraction(0.5);
SequentialProposal sp(0.1);
mc.SetProposalFunction(sp);
mc.SetNumIters(1000000);
mc.SetNumBurnInSteps(50);
MCInterval* interval = bc.GetInterval();
RooRealVar * s = (RooRealVar*)
model.GetParametersOfInterest()->find("s");
double lowerLimit = interval->LowerLimit(*s);
double upperLimit = interval->UpperLimit(*s);
MCMCIntervalPlot plot(*interval);
```

Markov-Chain Monte Carlo

MCMC basics: Metropolis-Hastings algorithm

Goal: given an n-dimensional pdf $p(\vec{\theta})$,

generate a sequence of points $\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3, \ldots$

1) Start at some point $\vec{\theta}_0$

2) Generate $\vec{\theta} \sim q(\vec{\theta}; \vec{\theta}_0)$

Proposal density $q(\vec{\theta}; \vec{\theta}_0)$
e.g. Gaussian centred about $\vec{\theta}_0$

3) Form Hastings test ratio $\alpha = \min\left[1, \dfrac{p(\vec{\theta})q(\vec{\theta}_0; \vec{\theta})}{p(\vec{\theta}_0)q(\vec{\theta}; \vec{\theta}_0)}\right]$

4) Generate $u \sim \text{Uniform}[0, 1]$

5) If $u \leq \alpha$, $\vec{\theta}_1 = \vec{\theta}$, $\leftarrow$ move to proposed point

else $\vec{\theta}_1 = \vec{\theta}_0$ $\leftarrow$ old point repeated

6) Iterate

RooStats Standard Macros

- RooStats provides standard tutorials taking all as input workspace, ModelConfig and data set names
- StandardProfileLikelihoodDemo.C

  run ProfileLikelihoodCalculator - get interval and produce plot

  root[]StandardProfileLikelihoodDemo("ws.root","w","ModelConfig","data")

- StandardBayesianNumericalDemo.C

  run Bayesiancalculator: get a credible interval and produce plot of posterior function

  root[]StandardBayesianNumericalDemo("ws.root","w","ModelConfig","data")

- StandardBayesianMCMCDemo.C

  run bayesian MCMCCalculator: get a credible interval and produce plot of posterior function

  root[]StandardBayesianMCMCDemo("ws.root","w","ModelConfig","data")

Time For Exercises !

RooStats Exercises

- Model building example
  - **CountingModel** notebook for a Poisson model (signal plus background)
    - following examples at this link: https://twiki.cern.ch/twiki/bin/view/RooStats/RooStatsTutorialsAugust2012#Create_Poisson_Counting_model
      1. use different parameterisation for the systematics in the background events (e.g. log-normal or gamma)
      2. add an extra systematic contribution (e.g. in the signal efficiency)
- ProfileLikelihood example
  - **ProfileLikelihood** notebook
- Bayesian examples
  - **BayesianNumerical**
  - **BayesianMCMC**
- Can also use the Standard tutorial macros to run the RooStats calculators
  - example is **StandardDemos** notebook

Useful Terminology

- **Observable** (or random variable): quantities that are directly measured by an experiment (eg. candidates mass, helicity angle, NNet output) – they form a dataset
- **Model:** based on probability density function (PDF) that describes one or multiples observables – parametric or non-parametric. PDF are normalized such that their integral over any observable is 1
- **Parameters of interest:** parameters of the model that one wishes to estimate or constrain (eg. particle mass, cross-section)
- **Nuisance parameters:** parameters of the model that are uncertain but not “of interest” (systematics-associated normalization or shape parameters)
    - treatment of systematic uncertainties depends on the statistical method used

RooStats
Part2

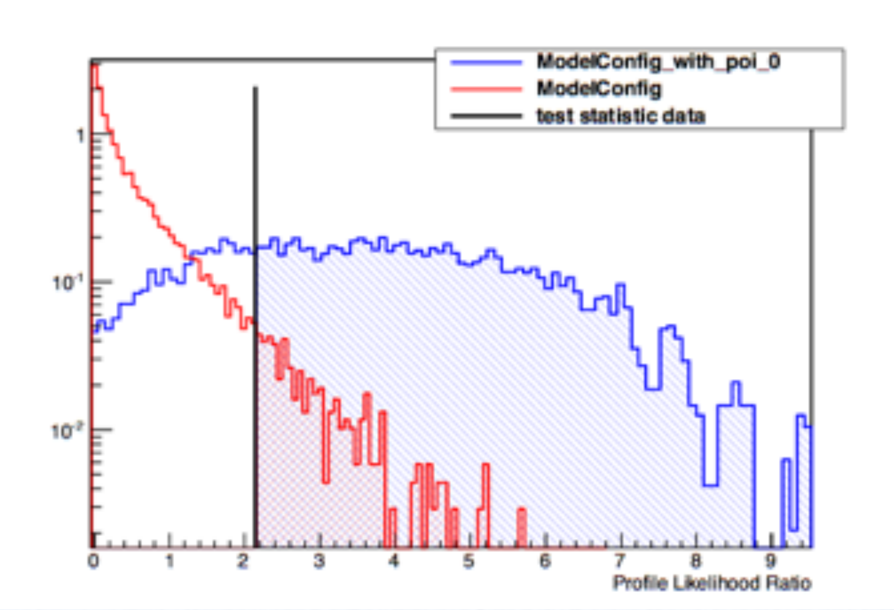
- Hypothesis tests in RooStats using toys and asymptotic formulae
- Hypothesis test inversion
  - Limit and interval calculators
    - CLs, Feldman-Cousins

Frequentist Hypothesis Tests

- Ingredients:
  - Null Hypothesis: the hypothesis being tested (e.g. $\theta = \theta_0$), assumed to be true and one tries to reject it
    - e.g. the data consists only of background events
  - Alternate Hypothesis: the competitive hypothesis (e.g. $\theta \neq \theta_0$)
    - e.g. the data consists of signal and background
  - w is the critical region, a subspace of all possible data used to define if hypothesis is rejected
    - size of test : $\alpha = P(X \in w \mid H_0)$ H_0 is rejected while is true
    - power of test : $1- \beta = P(X \in w \mid H_1)$
  - Test statistics: a function of the data, t(X) ,used for defining the critical region in multidimensional data: $X \in w \rightarrow t(X) \in w_t$

RooStats Hypothesis Test

- Define null and alternate model using ModelConfig
  - can use ModelConfig::SetSnapshot(const RooArgSet &) to define parameter values for the null in case of a common model (e.g. $\mu = 0$ for the B model)
- Select test statistics to use
- Select calculator
  - Use toys or asymptotic formula to get sampling distribution of test statistics
  - FrequentistCalculator or HybridCalculator have different treatment of nuisance parameters



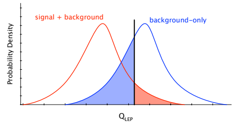
Test Statistics

- Test statistics maps multidimensional space in one, in a way relevant to the hypothesis being tested

RooStats has the three common test statistics used in the field (and more)

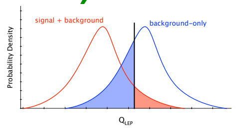
- simple likelihood ratio (used at LEP, nuisance parameters fixed)

$$Q_{LEP} = L_{s+b}(\mu = 1)/L_b(\mu = 0)$$



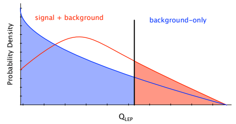
- ratio of profiled likelihoods (used commonly at Tevatron)

$$Q_{TEV} = L_{s+b}(\mu = 1, \hat{\hat{\nu}})/L_b(\mu = 0, \hat{\hat{\nu}}')$$



- profile likelihood ratio (related to Wilks's theorem)

$$\lambda(\mu) = L_{s+b}(\mu, \hat{\hat{\nu}})/L_{s+b}(\hat{\mu}, \hat{\nu})$$



- preferred choice is profile likelihood ratio which has known asymptotic distribution

FrequentistCalculator

- Generate toys using nuisance parameter at their conditional ML estimate ($\theta = \hat{\theta}_{\mu}$) by fitting them to the observed data
- Treat constraint terms in the likelihood (e.g. systematic errors) as auxiliary measurements
  - introduce global observables which will be varied (tossed) for each pseudo-experiment
  - $L = \text{Poisson}(n_{obs} \mid \mu + b)\ \text{Gaussian}(b_0 \mid b, \sigma_b)$
    - b_0 is a global observables, varied for each toys but it needs to be considered constant when fitting
    - n_{obs} is the observable which is part of the data set
    - μ is the parameter of interest (poi)
    - b is the nuisance parameter

HybridCalculator

- Nuisance parameters are integrated using their pdf (the constraint term) which is interpreted as a Bayesian prior
  - integration is done by generating for each toys different nuisance parameters values
  - need to have a pdf for the nuisance parameters (often it can be derived automatically from the model)

$$L = \text{Poisson}(\, n_{obs} \mid \mu + b)\ \text{Gaussian}(\, b \mid b_0, \sigma_b)$$

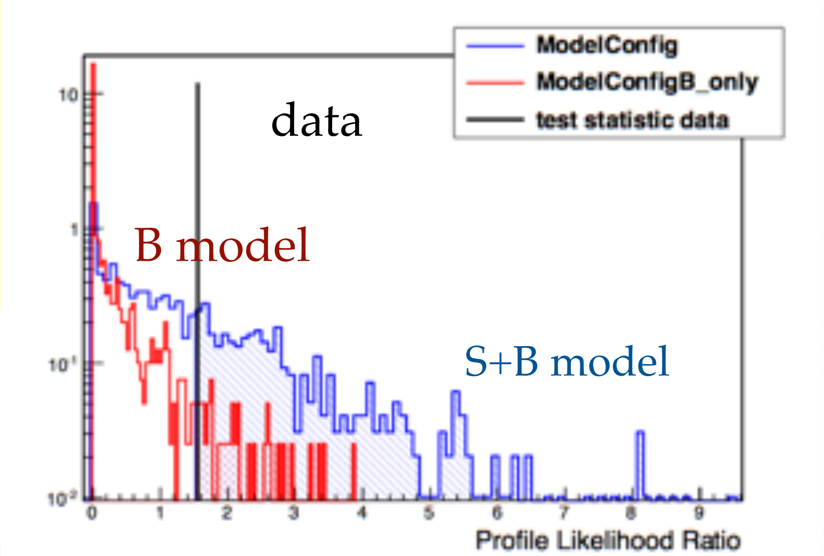
$$L = \int \text{Poisson}(\, n_{obs} \mid \mu + b)\ \text{Gaussian}(\, b \mid b_0, \sigma_b)\, db$$

Example: FrequentistCalculator

- Define the models
  - N.B for discovery significance null is B model and alt is S+B

```
// create first HypoTest calculator (data, alt model , null model)
FrequentistCalculator fcalc(*data, *sbModel, *bModel);

// create the test statistics
ProfileLikelihoodTestStat profll(*sbModel->GetPdf());
// use one-sided profile likelihood for discovery tests
profll.SetOneSidedDiscovery(true);

// configure  ToyMCSampler and set the test statistics
ToyMCSampler *toymcs = (ToyMCSampler*)fcalc.GetTestStatSampler();
toymcs->SetTestStatistic(&profll);

fcalc.SetToys(1000,1000);  // set number of toys for (null, alt)

// run the test
HypoTestResult * r = fcalc.GetHypoTest();
r->Print();

// plot test statistic distributions
HypoTestPlot * plot = new HypoTestPlot(*r);
plot->Draw();
```

```
Results HypoTestCalculator_result:
 - Null p-value = 0.034 +/- 0.00573097
 - Significance = 1.82501 sigma
 - Number of Alt toys: 1000
 - Number of Null toys: 1000
```

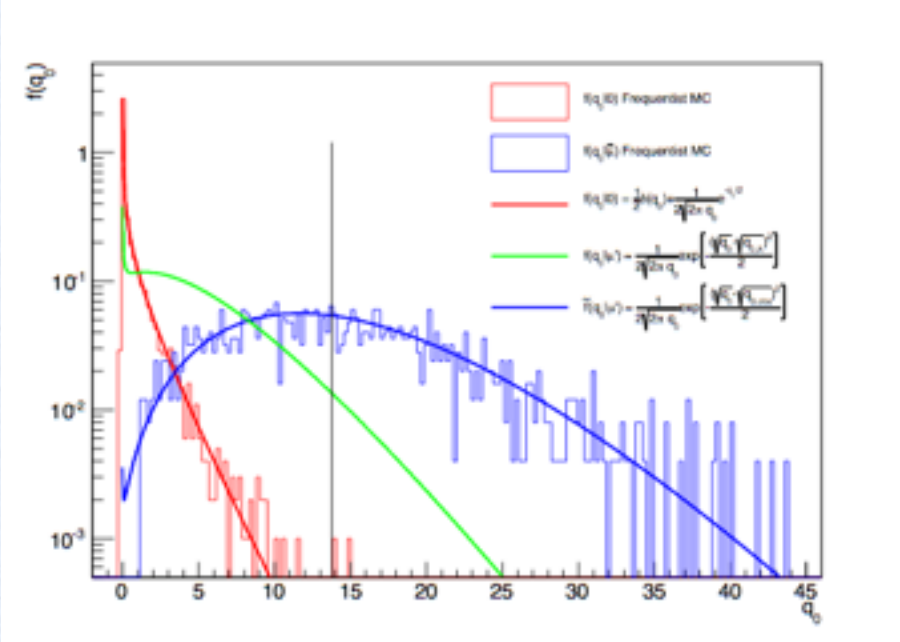


AsymptoticCalculator

- Use the asymptotic formula for the test statistic distributions
- one-sided profile likelihood test statistic:
  - null model ($\mu = \mu_{\mathrm{TEST}}$)
    - half X^2 distribution
  - alt model ($\mu \neq \mu_{\mathrm{TEST}}$)
    - non-central X^2
    - use Asimov data to get the non centrality parameter $\Lambda = (\mu\text{-}\mu_{\mathrm{TEST}})/\sigma$
- p-values for null and alternate can be obtained without generating toys

$$\lambda(\mu) = \frac{L(x|\mu,\hat{\hat{\nu}})}{L(x|\hat{\mu},\hat{\nu})}$$

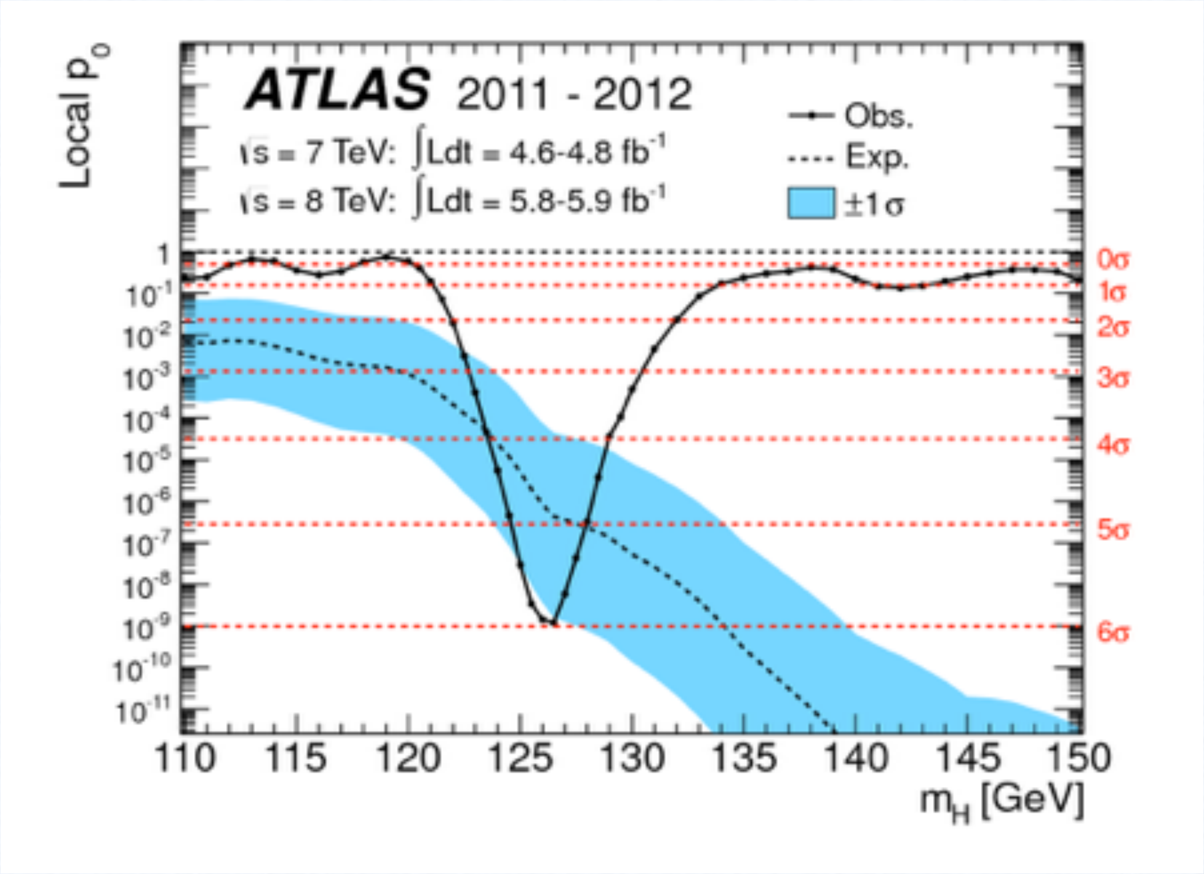
$\lambda(\mu) = 0$ for
$\hat{\mu} < 0$ (discovery)
$\hat{\mu} < \mu_{\mathrm{TEST}}$ (limits)



➡ see Cowan, Cranmer, Gross, Vitells, arXiv:1007.1727,EPJC 71 (2011) 1-1

Example: Discovery Significance

- Performing the tests for different mass hypotheses (*i.e* different signal models):



Inversion of Hypothesis Tests

- one-to-one mapping between hypothesis tests and confidence intervals

Table 20.1 Relationships between hypothesis testing and interval estimation

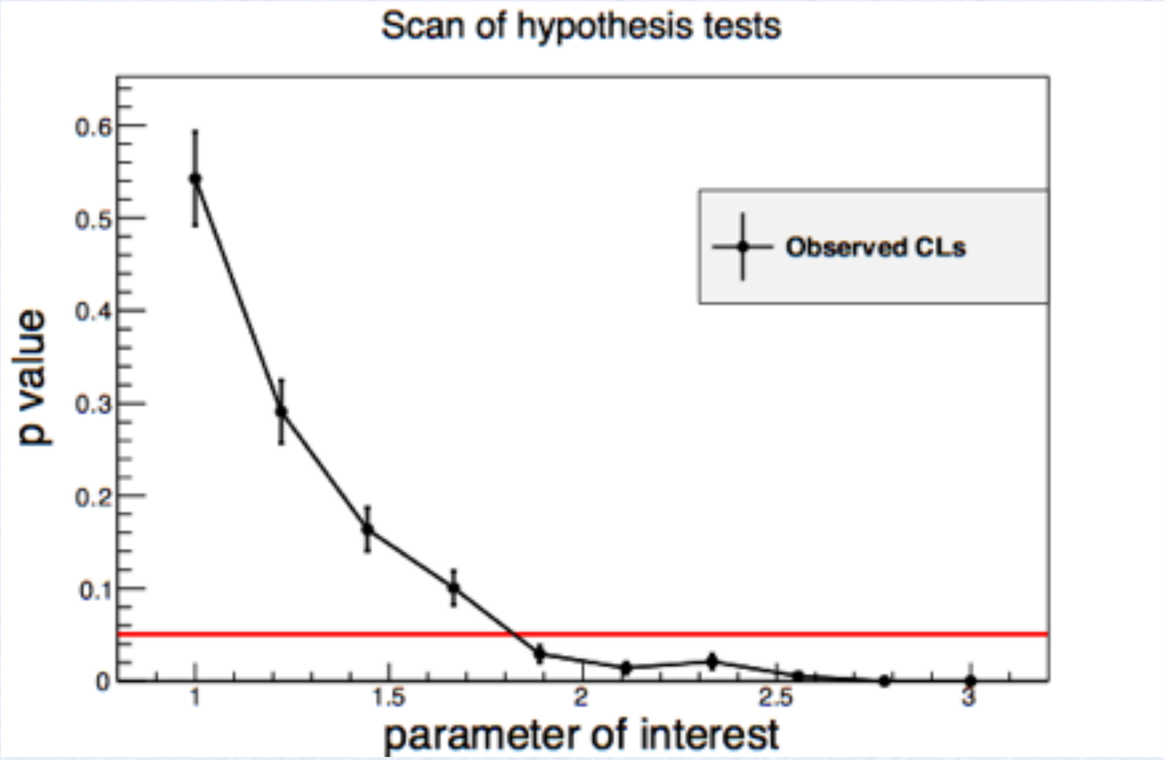
Property of test		Property of corresponding confidence interval
Size $= \alpha$		Confidence coefficient $= 1 - \alpha$
Power $=$ probability of rejecting a false value of $\theta = 1 - \beta$		Probability of not covering a false value of $\theta = 1 - \beta$
Most powerful		Uniformly most accurate
$\longleftarrow$	$\left\{ \begin{matrix} \textit{Unbiased} \\ 1 - \beta \geq \alpha \end{matrix} \right\}$	$\longrightarrow$
Equal-tails test $\alpha_1 = \alpha_2 = \frac{1}{2}\alpha$		Central interval

from G. Feldman visiting Harvard statistics department

They explained that in statistical theory there is a one-to-one correspondence between a hypothesis test and a confidence interval. (The confidence interval is a hypothesis test for each value in the interval.) The Neyman-Pearson Theorem states that the likelihood ratio gives the most powerful hypothesis test. Therefore, it must be the standard method of constructing a confidence interval.

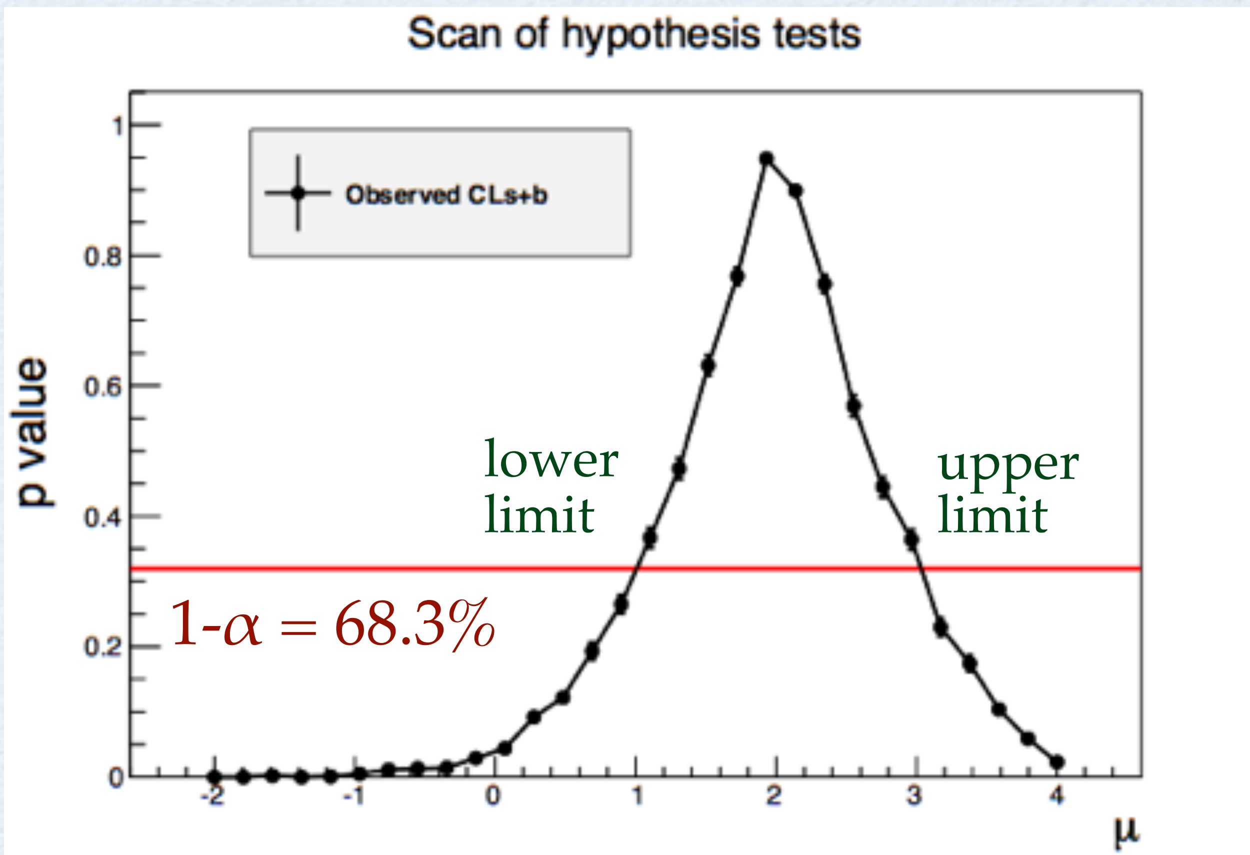
Hypothesis Test Inversion

- Performing an hypothesis test at each value of the parameter
- Interval can be derived by inverting the p-value curve, function of the parameter of interest (μ)
    - value of μ which has p-value α (e.g. 0.05), is the upper limit of 1-α confidence interval (e.g. 95%)



Hypothesis Test Inversion

- use one-sided test for upper limits (e.g. one-side profile likelihood test statistics)
- use two-sided test for a 2-sided interval



Example: 1-σ interval for a Gaussian measurement

HypoTestInverter class

- Input is an Hypothesis Test calculator:
    - Frequentist/Hybrid/AsymptoticCalculator
    - possible to customize test statistic, number of toys, etc..
        - N.B: null model is S+B, alternate is B only model
- Compute an Interval (result is a **`ConfInterval`** object):
    - scan given interval of μ and perform hypothesis tests
    - compute upper/lower limit from scan result
        - can use $CL_s = CL_{s+b} / CL_b$ for the p-value
    - result (**`HypoTestInverterResult`**) contains all the hypothesis test results for each scanned μ value
    - can compute expected limits and bands

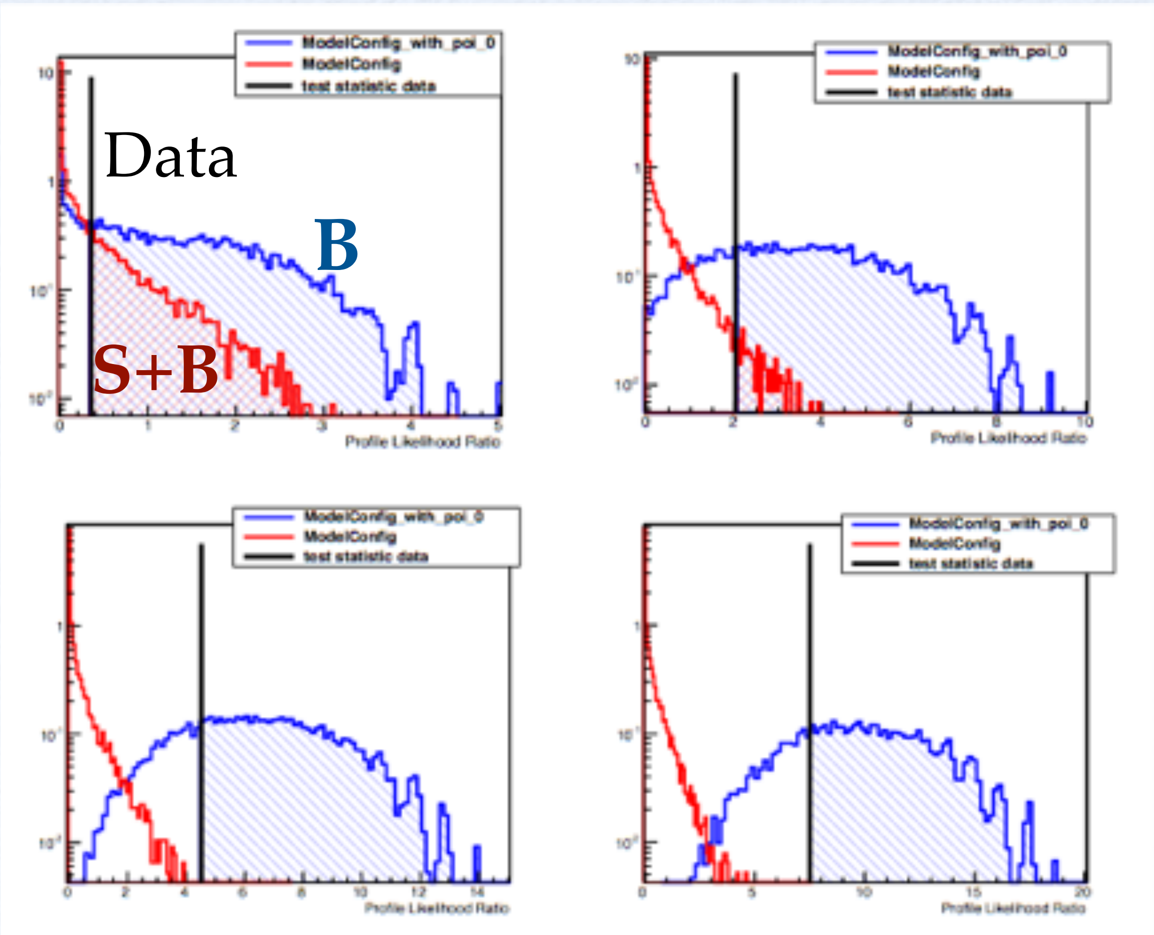
HypoTestInverter

- **`HypoTestInverter`** class in RooStats

```
// create first HypoTest calculator (N.B null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);

HypoTestInverter calc(*fc);
calc.UseCLs(true);

// configure  ToyMCSampler and set the test statistics
ToyMCSampler *toymcs = (ToyMCSampler*)fc.GetTestStatSampler();

ProfileLikelihoodTestStat profll(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood
profll.SetOneSided(true);
toymcs->SetTestStatistic(&profll);

// configure and run the scan
calc.SetFixedScan(npoints,poimin,poimax);
HypoTestInverterResult * r = calc.GetInterval();

// get result and plot it
double upperLimit = r->UpperLimit();
double expectedLimit = r->GetExpectedUpperLimit(0);

HypoTestInverterPlot *plot = new HypoTestInverterPlot("hi","",r);
plot->Draw();
```

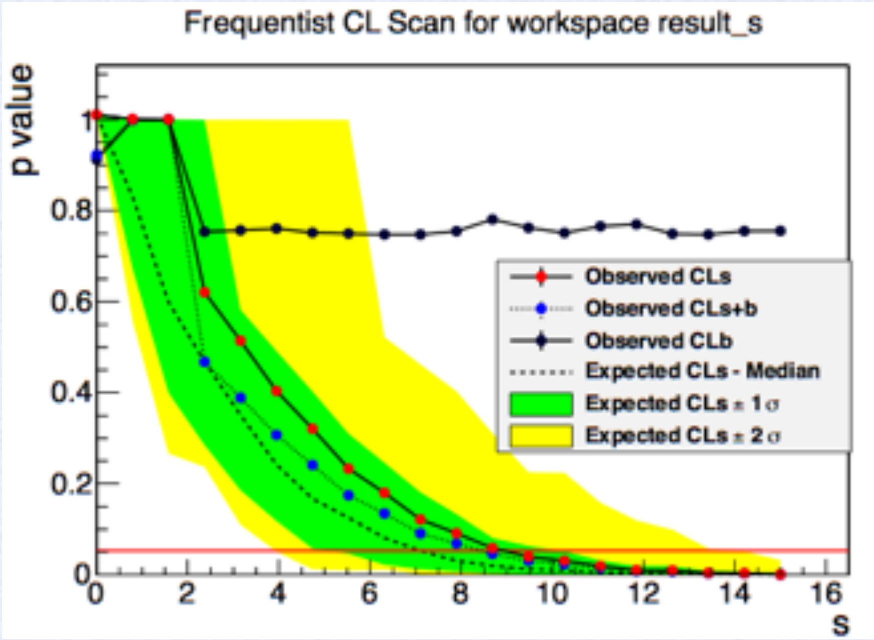
Running the HypoTestInverter

Hypothesis test results for each scanned point



p-value, CL_{s+b} (or CL_b) is integral of S+B (or B) test statistic distribution from data value

Scan result



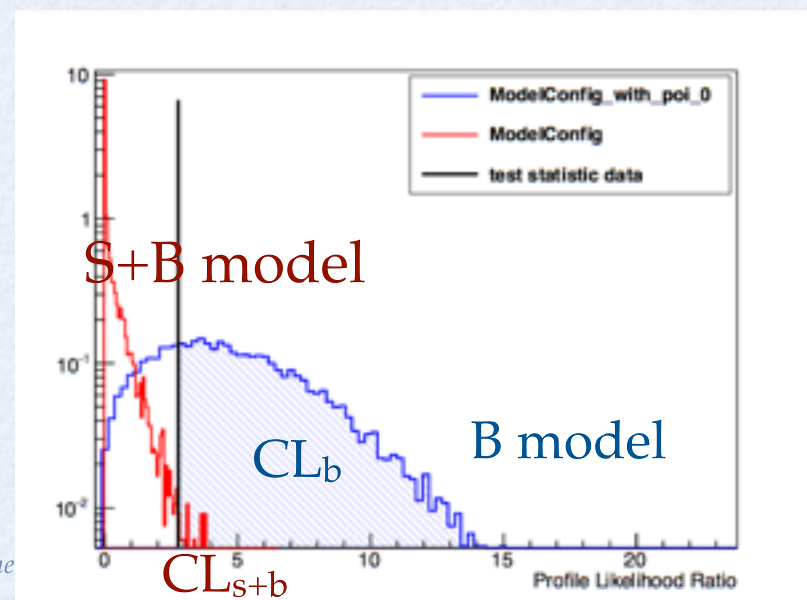
How expected limit and bands are obtained ?

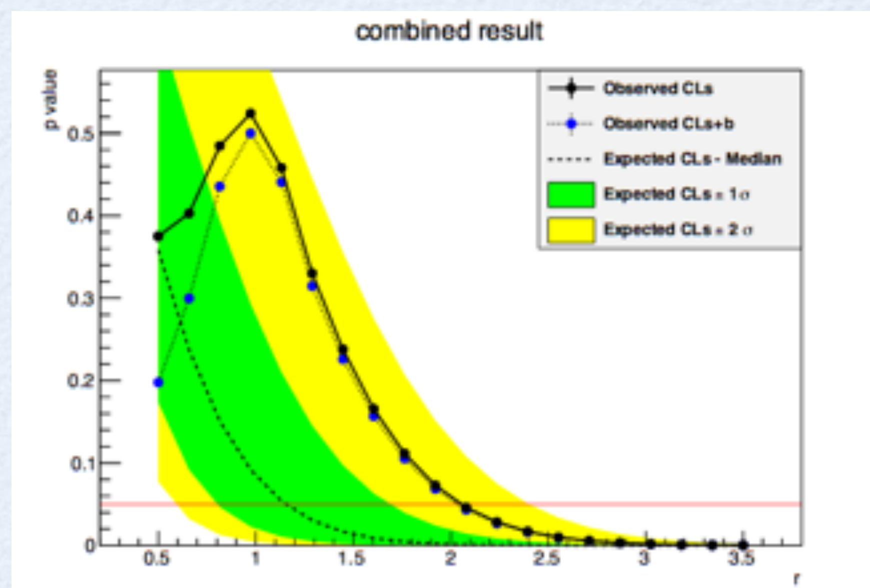
- compute p-value for quantiles (median, +/1,2 sigma) of the B model test statistic distribution (*i.e.* use quantile as the observed value)

Asymptotic Limits

- **`AsymptoticCalculator`** class for HypoTestInverter
  - use the asymptotic formula for the test statistic distributions
    - χ^2 approximation for the profile likelihood ratio
      - see G. Cowan *et al.*, arXiv:1007.1727,EPJC 71 (2011) 1-1
  - p-values CL_{s+b} (null) and CL_b (alt) obtained without generating toys
  - also expected limits from the alt distribution

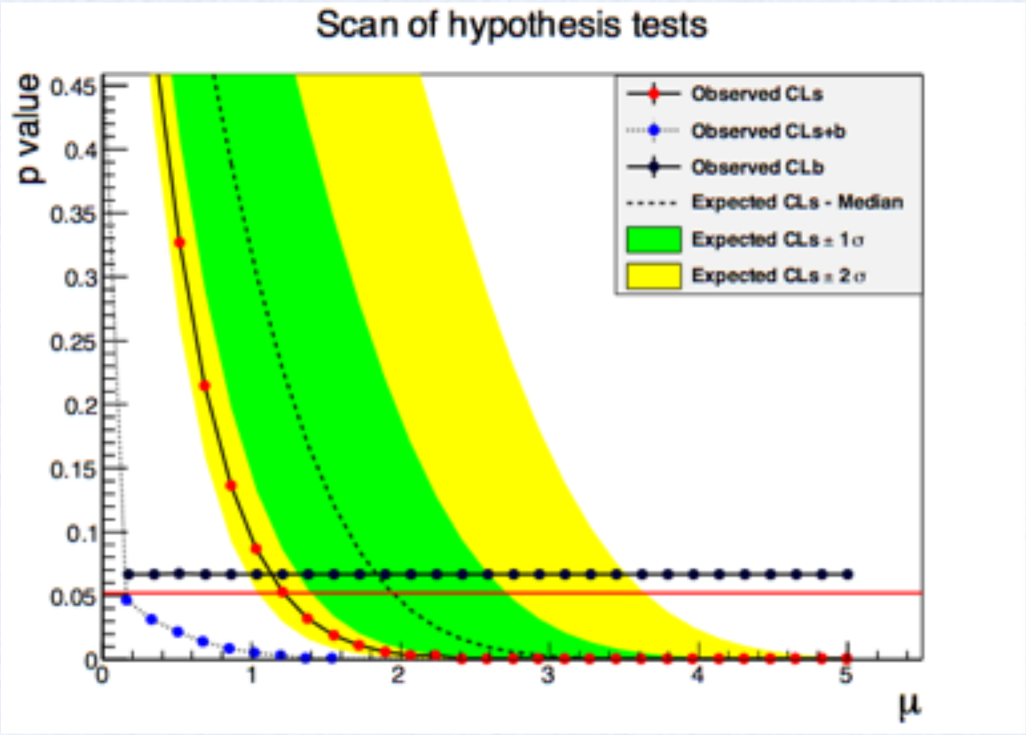
```
// create first HypoTest calculator (N.B null is s+b model)
AsymptoticCalculator ac(*data, *bModel, *sbModel);

HypoTestInverter calc(*ac);
// run inverter same as using other calculators
........
```

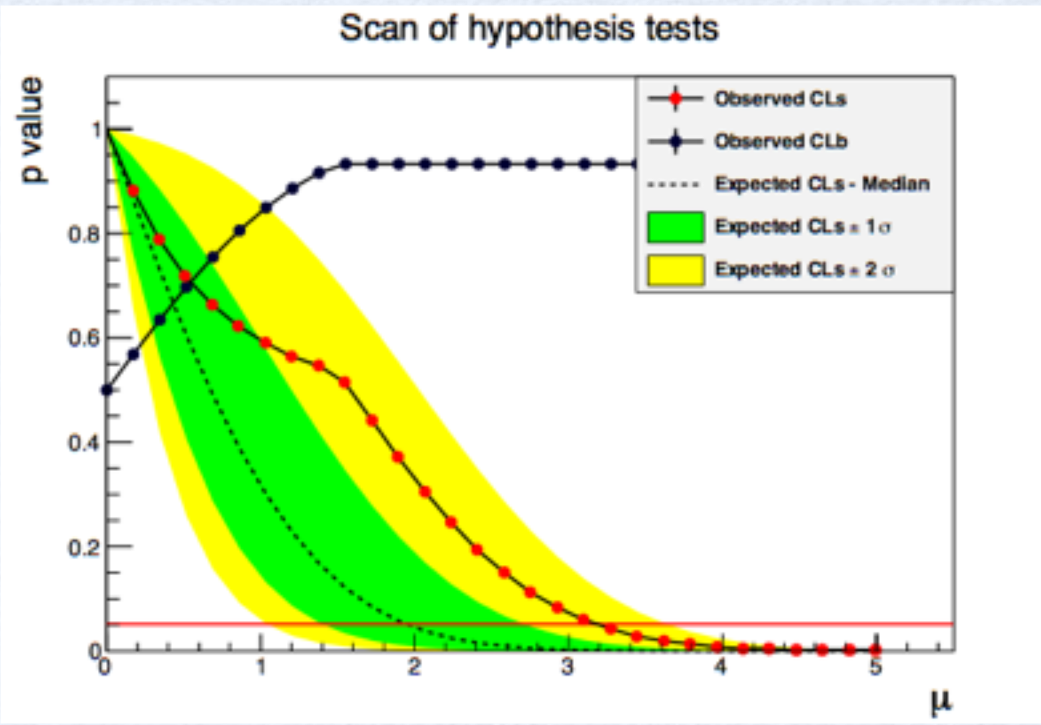




Example of Scan

- 95% CL limit on a Gaussian measurement:
  - Gauss(x,μ,1), with μ≥0





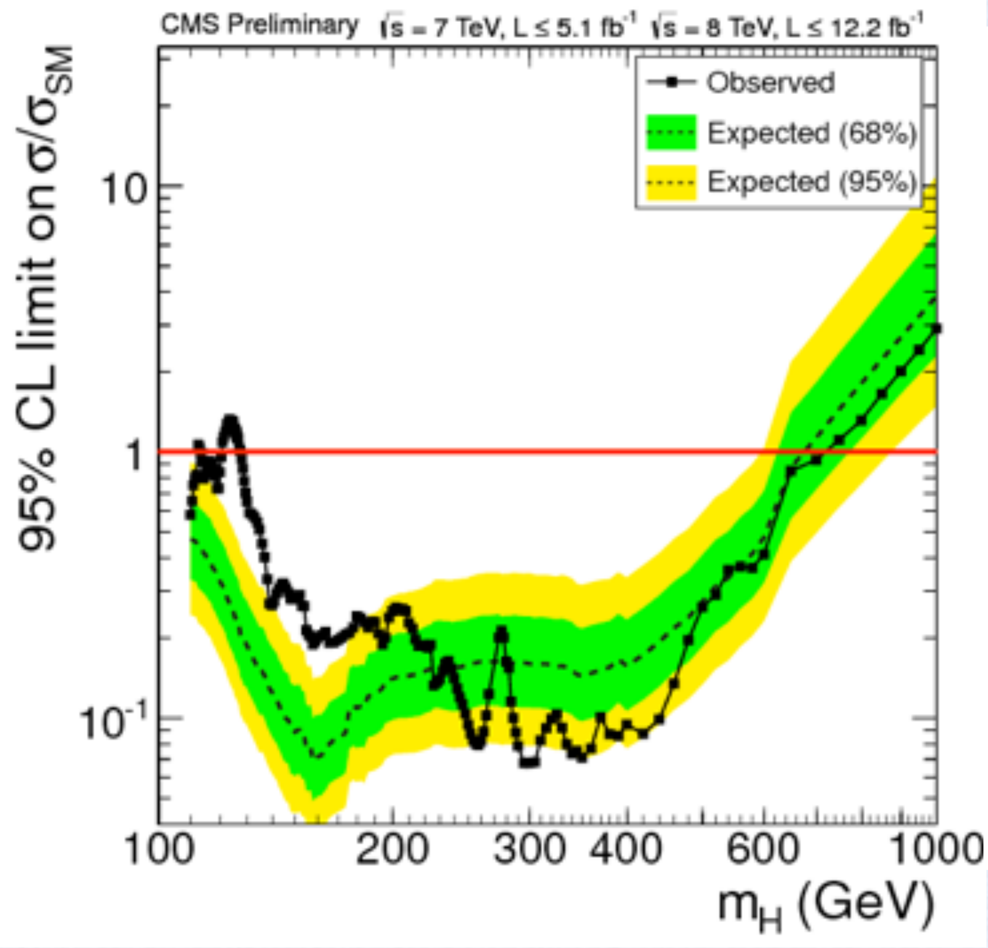
deficit, observation x = -1.5

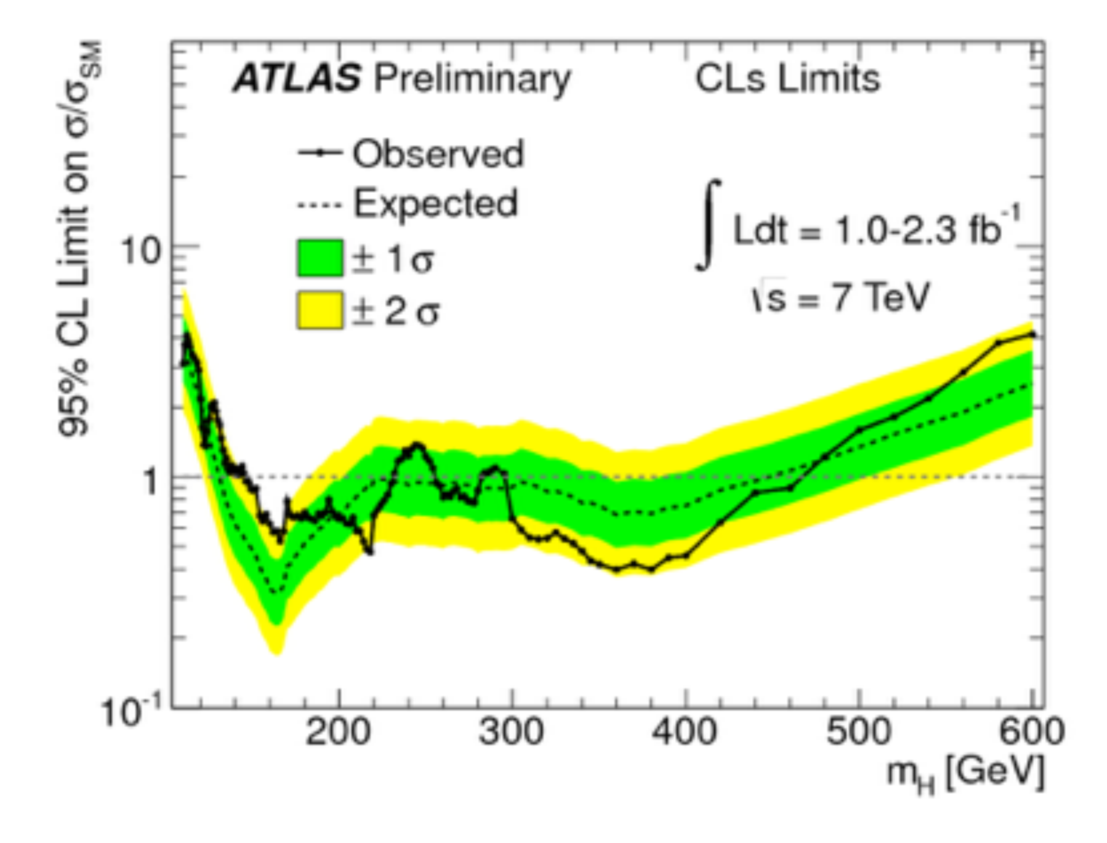
excess, observation x = 1.5

use CL_s as p-value to avoid setting limits which are too good

Example: Computing Limits

- By computing limits for different mass hypothesis:



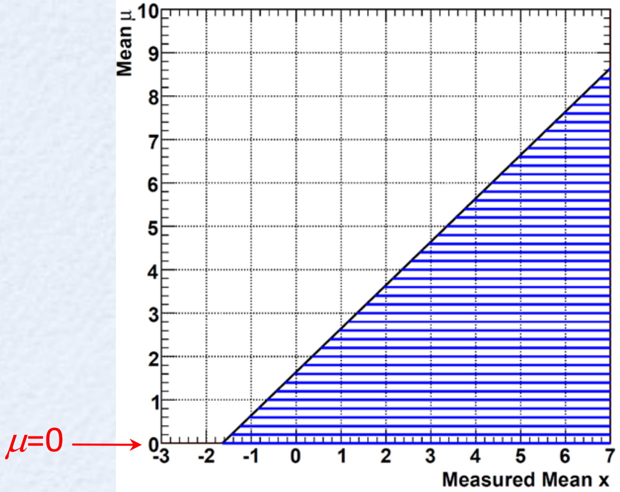


Limits on bounded measurements

from Bob Cousins:

Downward fluctuations in searches for excesses

Classic example: Upper limit on mean μ of Gaussian based on measurement *x* (in units of σ).



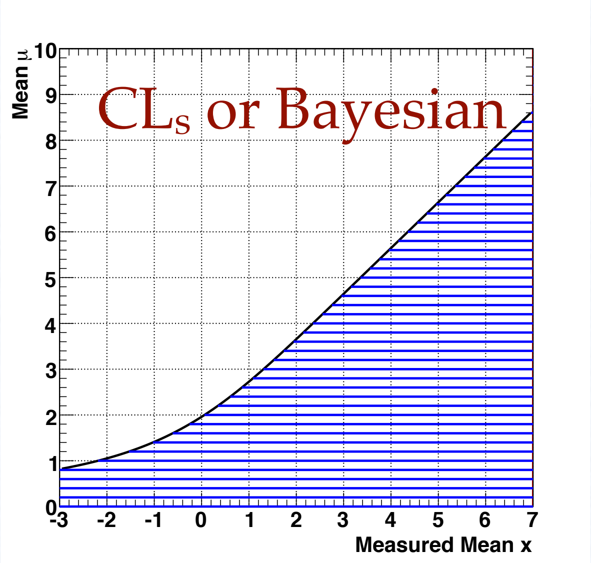
Frequentist 1-sided 95% C.L. Upper Limits, based on α = 1 – C.L. = 5% (called CL_{sb} at LEP).

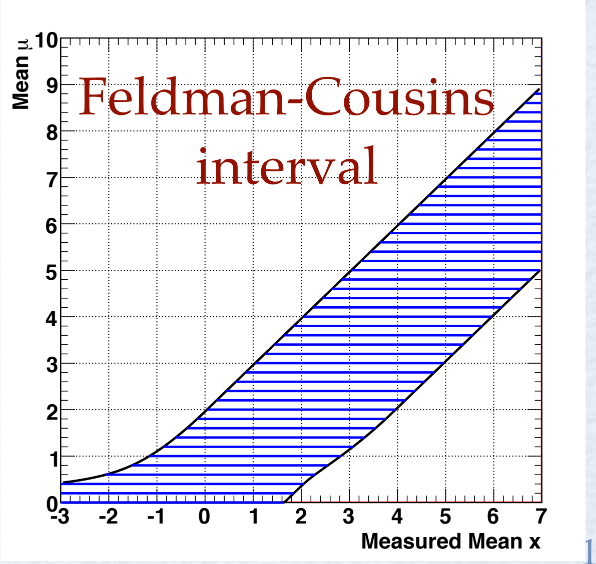
For *x* < –1.64 σ the confidence interval is the *null* set!

Bob Cousins, CMSDAS, 1/2012

If $\mu \geq 0$ in model, as measured x becomes increasingly negative, standard classical upper limit becomes small and then null.

Issue acute 15-25 years ago in expts to measure ν_e mass in (tritium β decay): several measured $m_\nu^2 < 0$.

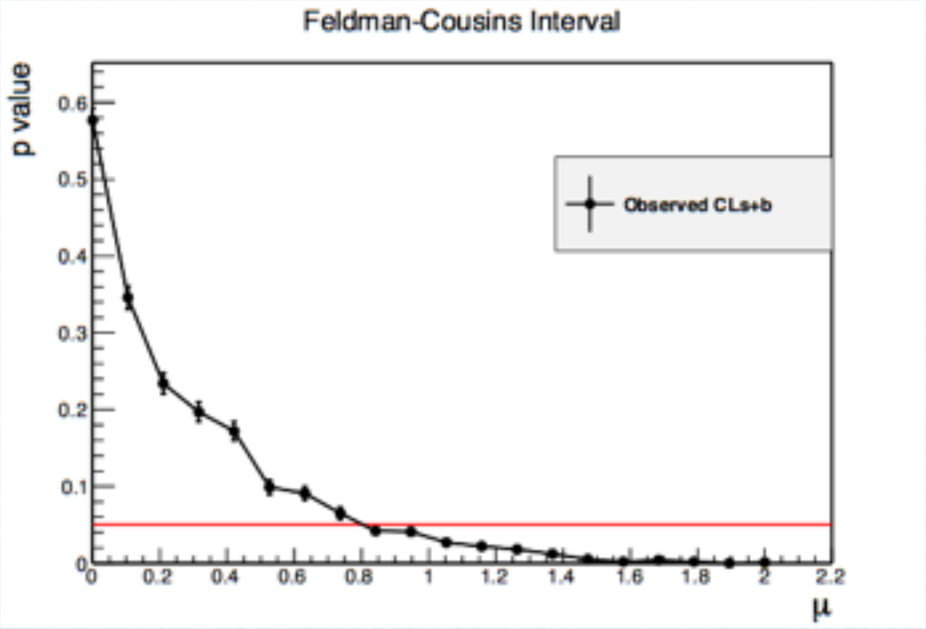




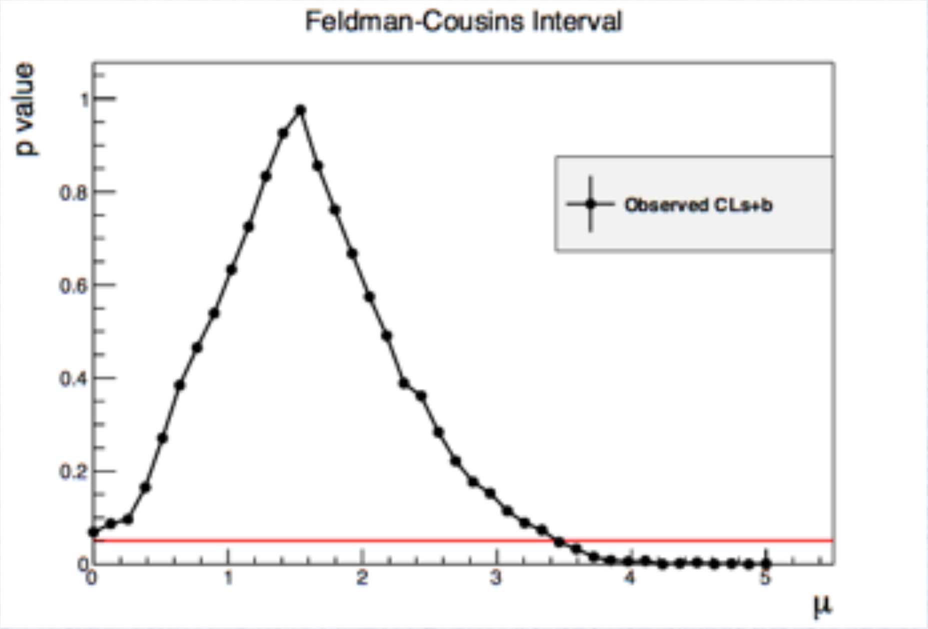
Feldman-Cousins intervals

- HypoTestInverter class can compute also a Feldman-Cousins interval
  - need to use FrequentistCalculator and CL_{s+b} as p-value
  - use the 2-sided profile likelihood test statistic

$$\lambda(\mu) = \frac{L(x|\mu, \hat{\hat{\nu}})}{L(x|\hat{\mu}, \hat{\nu})}$$



observation x = -1.5



observation x = 1.5

Feldman-Cousins Interval

from Kyle Cranmer:

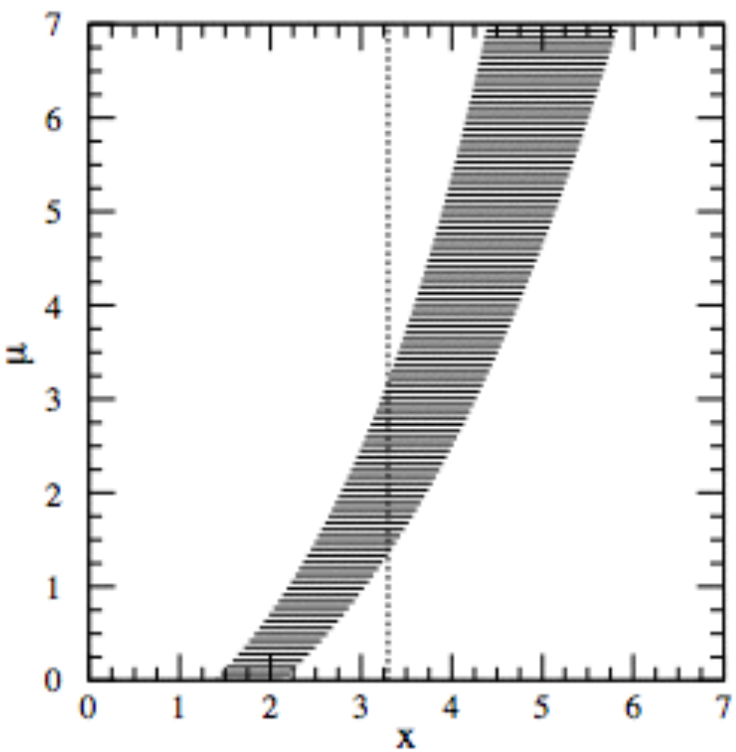
A different way to picture Feldman-Cousins

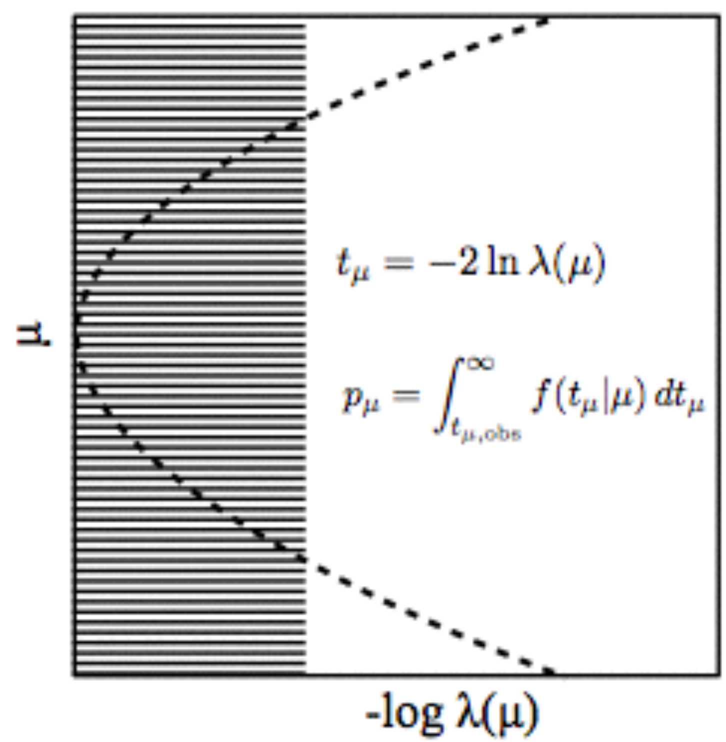
Most people think of plot on left when thinking of Feldman-Cousins

- bars are regions "ordered by" $R = P(n|\mu)/P(n|\mu_{\text{best}})$, with $\int_{x_1}^{x_2} P(x|\mu)dx = \alpha.$

But this picture doesn't generalize well to many measured quantities.

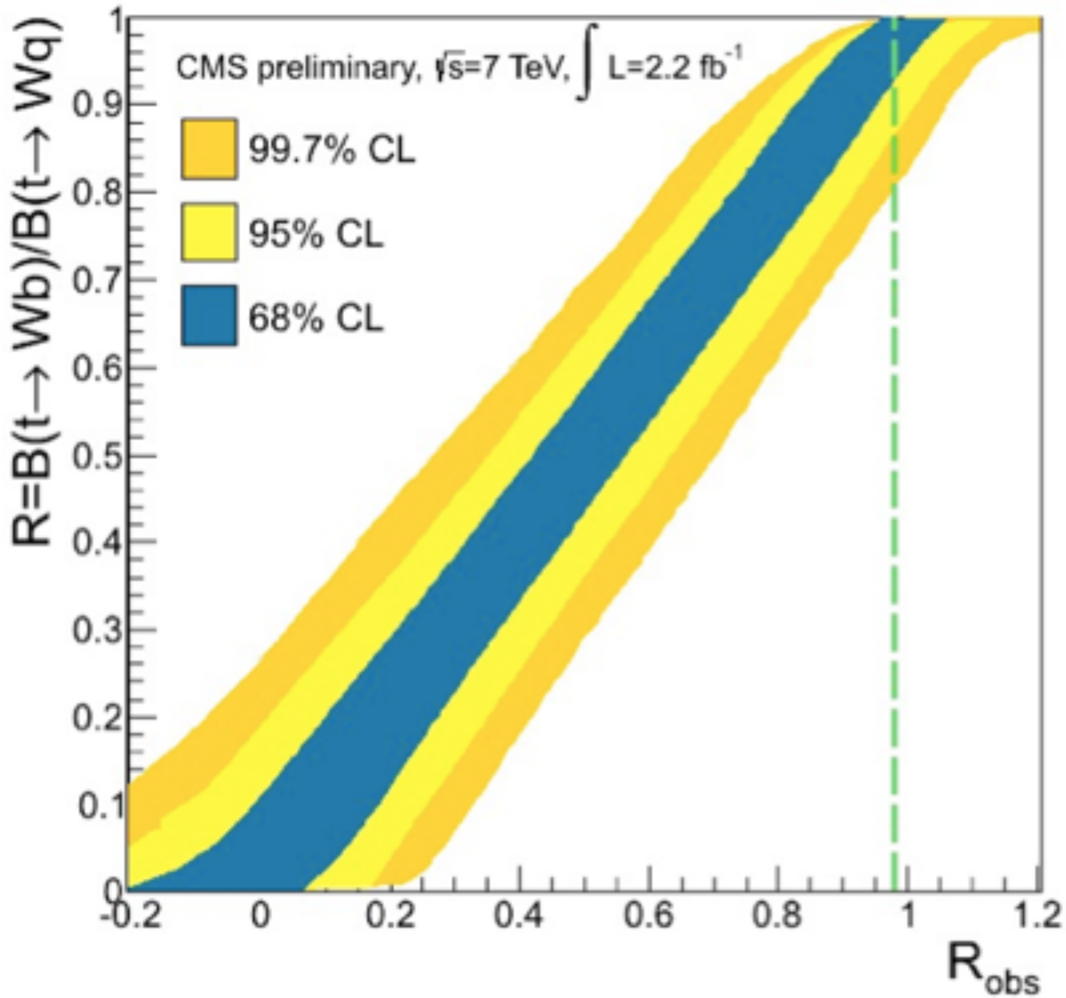
- Instead, just use R as the test statistic... and R is $\lambda(\mu)$





Example: Feldman-Cousins interval

- Same RooStats code but with different configuration can compute also a Feldman-Cousins interval



StandardHypoTestInvDemo.C

- Standard ROOT macro to run the Hypothesis Test inversion.
- Inputs to the macro:
  - workspace file, workspace name
  - name of S+B model (null) and for B model (alt)
    - if no B model is given, use S+B model with poi = 0
  - data set name
  - calculator type: frequentist (= 0), hybrid (=1), or asymptotic (=2)
  - test statistics
- options:
  - use CL_s or CL_{s+b} for computing limit
  - number of points to scan and min, max of interval

load the macro after having created the workspace and saved in file SPlusBExpoModel.root

```
root[] .L StandardHypoTestInvDemo.C
```

run for CLs (with frequentist calculator (type = 0) and one-side PL test statistics (type = 3) scan 10 points in [0,100]

```
root[] StandardHypoTestInvDemo("SPlusBExpoModel.root","w","ModelConfig","","data",0,3, true, 10, 0, 100)
```

run for Asymptotic CLs (scan 20 points in [0,100])

```
root[] StandardHypoTestInvDemo(SPlusBExpoModel.root","w","ModelConfig","","data",2,3, true, 20, 0, 100)
```

run for Feldman-Cousins (scan 10 points in [0,100])

```
root[] StandardHypoTestInvDemo(SPlusBExpoModel.root","w","ModelConfig","","data",0,2, false, 10, 0, 100)
```

Time For Exercises !

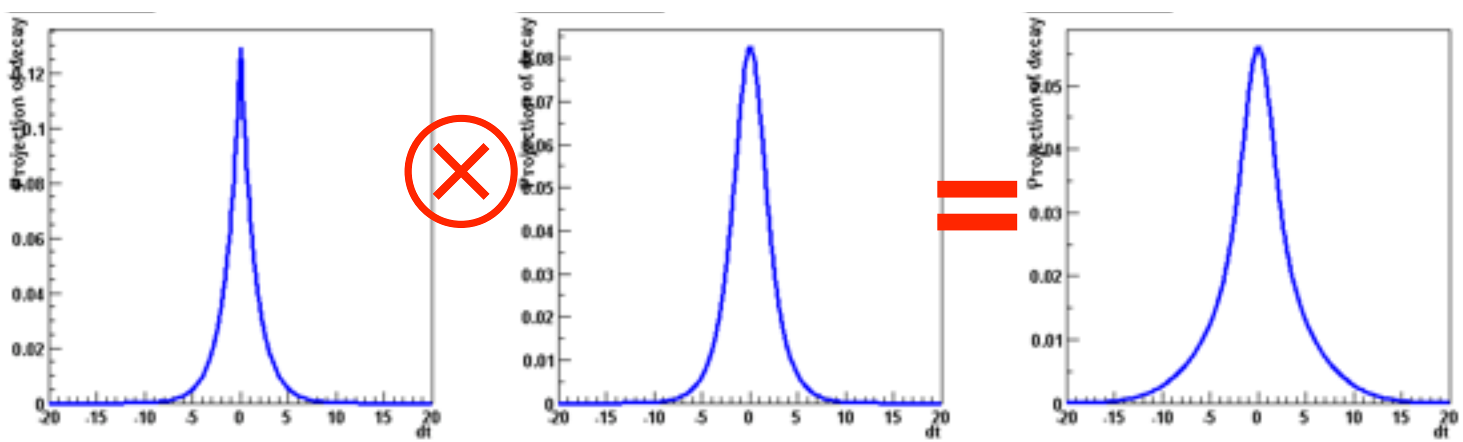
Advanced RooStats Examples

- Hypothesis test example (**HypothesisTest** notebook)
  - run on the Higgs un-binned or binned model (HiggsModel.root or HiggsBinModel.root)
  - **p0Plot** for computing the significance for different mass values
- Frequentist interval example (**HypoTestInversion** notebook)
  - e.g. run on Counting workspace or any others
  - be careful when using toys (not using the asymptotic calculator). It might need a long time
- Can also use the Standard tutorial macros to run on any workspace
  - examples are in **StandardDemos** notebook

Convolution

- Model representing a convolution of a theory model and a resolution model often useful

$$f(x) \otimes g(x) = \int_{-\infty}^{+\infty} f(x) g(x - x') dx'$$



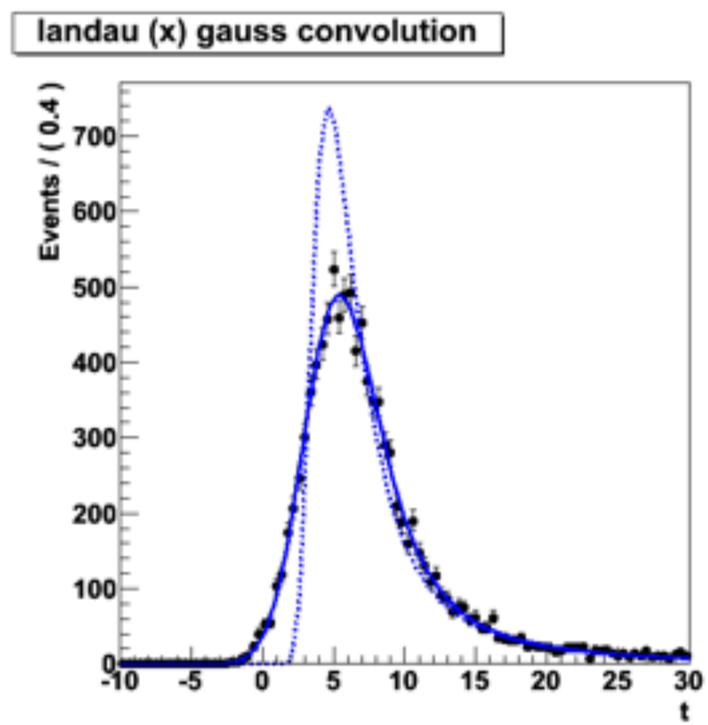
- But numeric calculation of convolution integral can be challenging. No one-size-fits-all solution, but 3 options available
  - Analytical convolution (BW⊗Gauss, various B physics decays)
  - Brute-force numeric calculation (slow)
  - FFT numeric convolution (fast, but some side effects)

Convolution

- Example

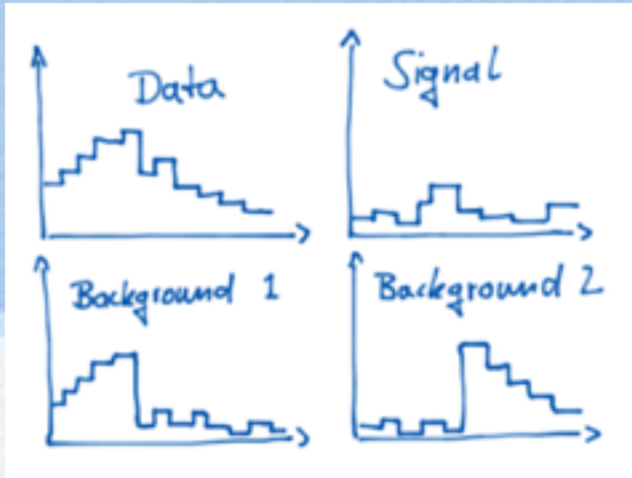
```
w.factory("Landau::L(x[-10,30],5,1)") :
w.factory("Gaussian::G(x,0,2)") ;

w.var("x")->setBins("cache",10000) ; // FFT sampling density
w.factory("FCONV::LGf(x,L,G)") ;      // FFT convolution

w.factory("NCONV::LGb(x,L,G)") ;     // Numeric convolution
```

- FFT usually best
  - Fast: unbinned ML fit to 10K events take ~5 seconds
  - NB: Requires installation of FFTW package (free, but not default)
  - Beware of cyclical effects (some tools available to mitigate)



Open Issues

HistFactory

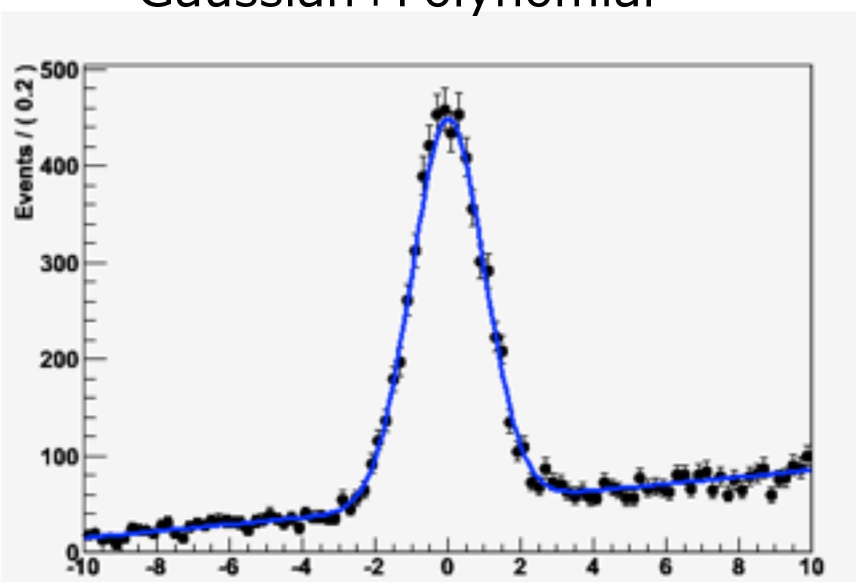


see also HistFactory doc (https://cdsweb.cern.ch/record/1456844/files/CERN-OPEN-2012-016.pdf*)*

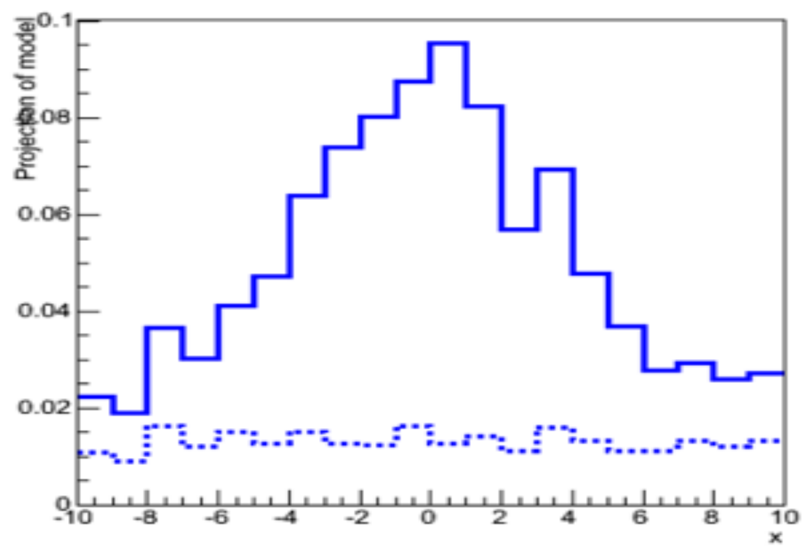
HistFactory – a new class of pdfs

- Focus of RooFit traditionally on analytical models
  - Assumes you can formulate signal/background in an analytical form
  - Often possible in e+e- experiments,
    shapes for hadron colliders cumbersome

Analytical form:
Gaussian+Polynomial



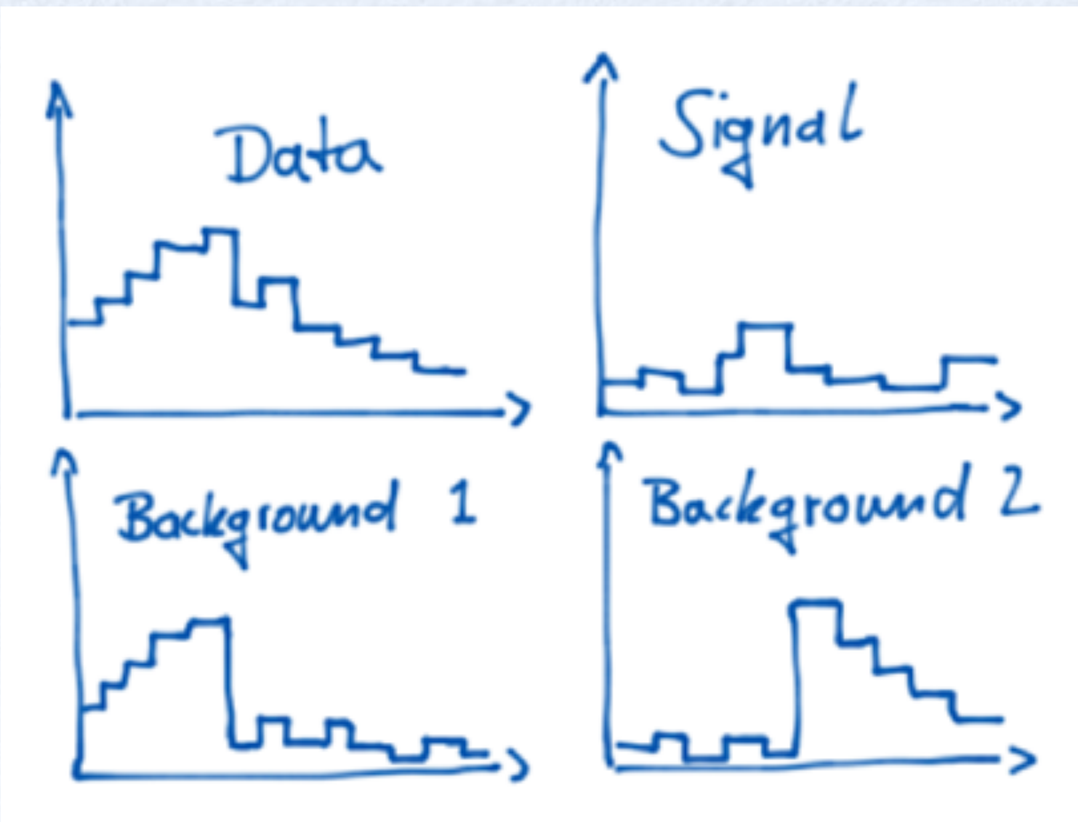
Template form:
Histogram (discrete)



K. Cranmer, G. Lewis, L. Moneta, A. Shibata, and W. Verkerke, *HistFactory: A tool for creating statistical models for use with RooFit and RooStats*, CERN-OPEN-2012-016 (2012). `http://cdsweb.cern.ch/record/1456844`.

Model Building with HistFactory

- Tool to build models from input histograms

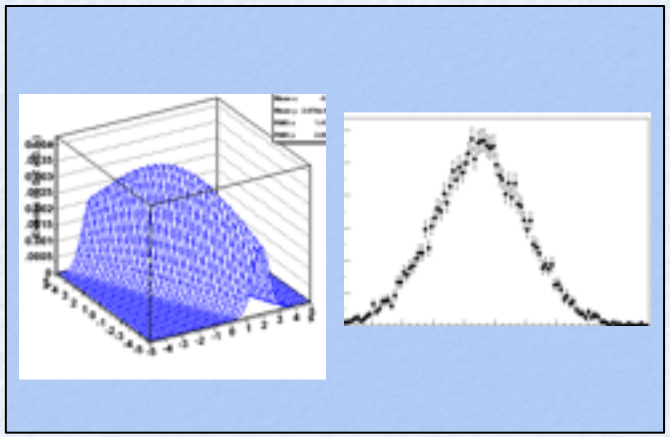


RooFit
Workspace

RooFit/RooStats at LHC (Higgs analysis)

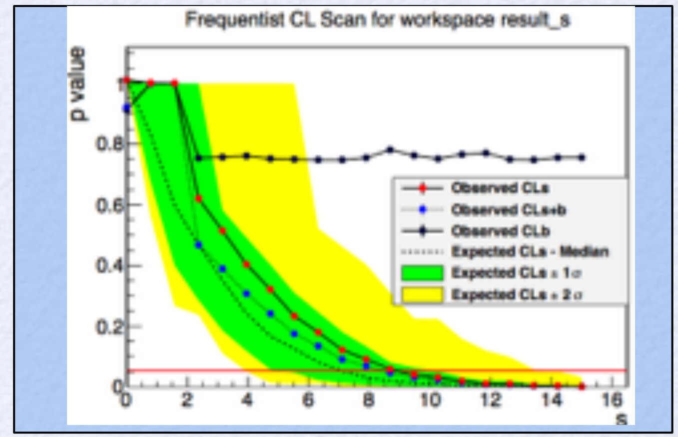
Class RooWorkspace

Simplify packaging and sharing of models



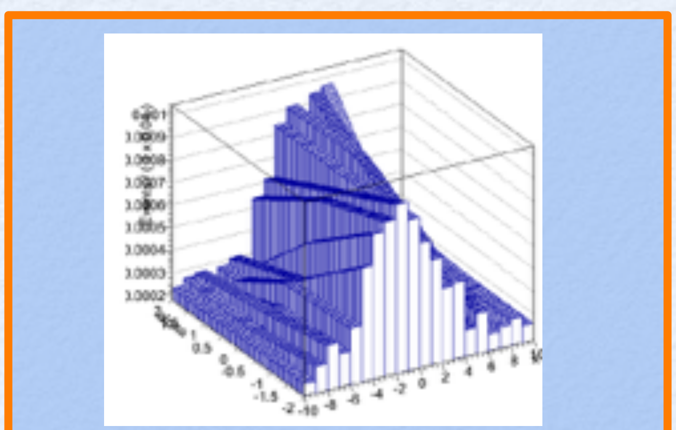
RooStats toolkit

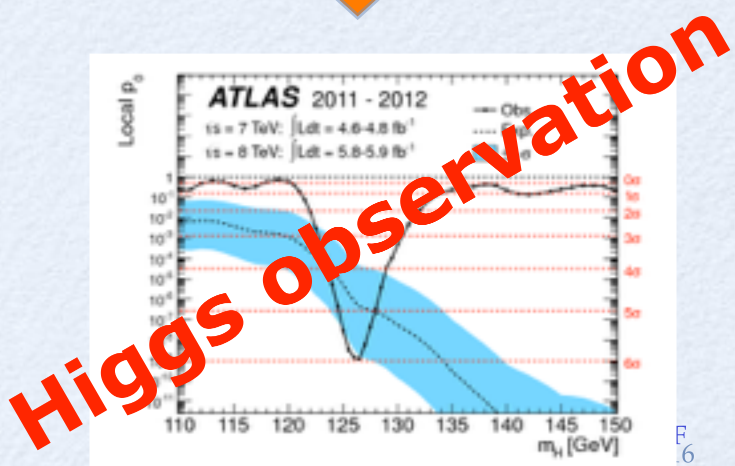
Statistical tests based on likelihoods from RooFit models



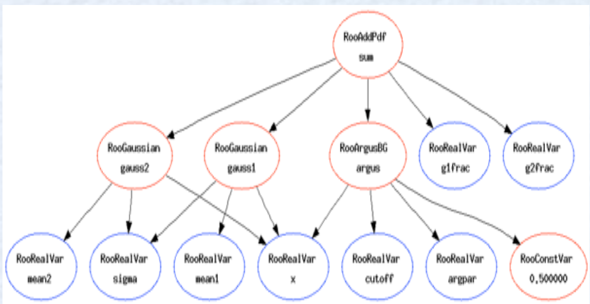
HistFactory package

Constructing models from Monte Carlo templates

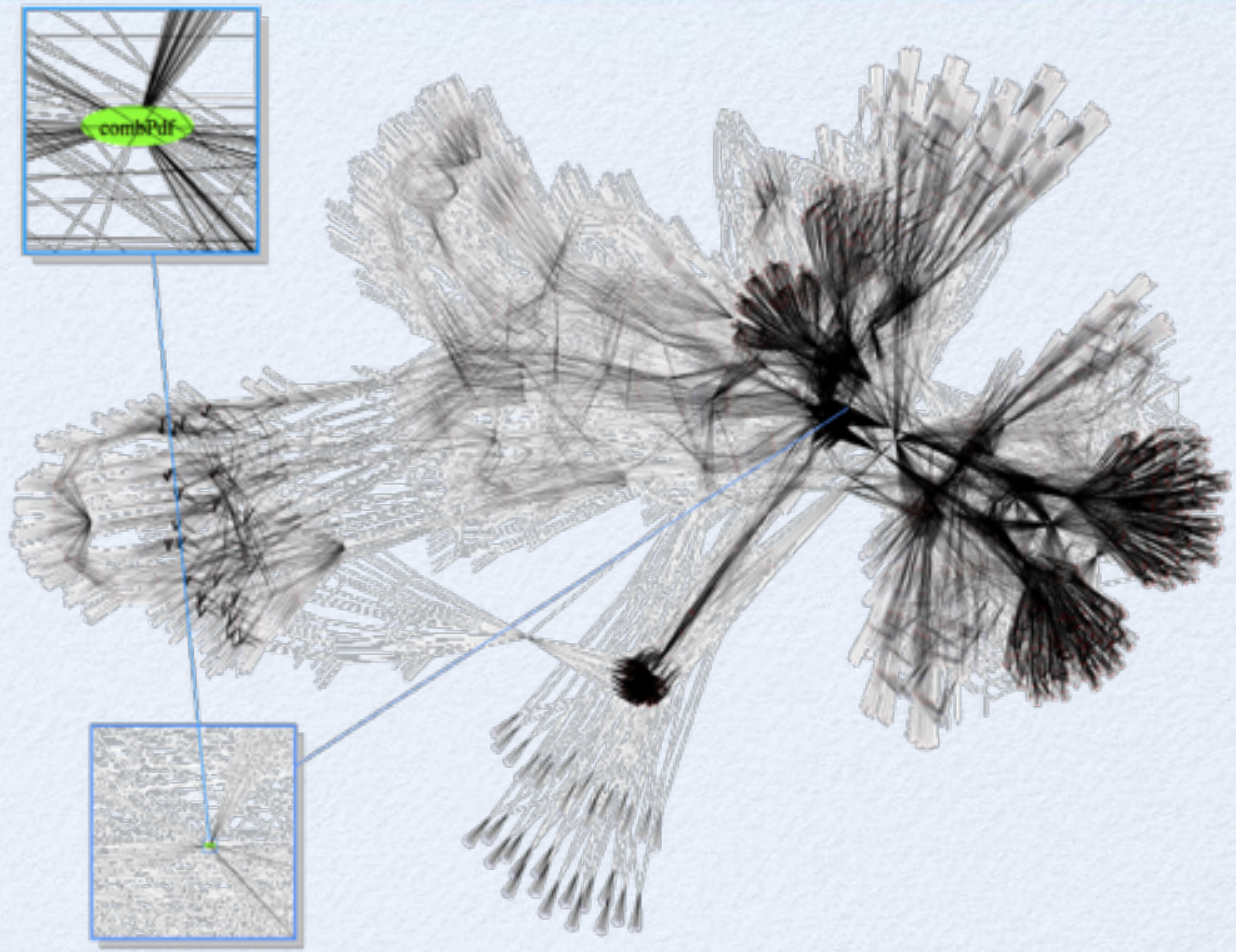




How well does it scale?



Graph of the full ATLAS Higgs combination model



Model has ~23.000 function objects, ~1600 parameters

Reading/writing of full model takes ~4 seconds

ROOT file with workspace is ~6 Mb

HistFactory concept

- Measurement
  - used to give global description of the model
  - can contain one or several channels
- Channel
  - disjoints selected regions of events
- Sample
  - set of process contributions to a channel

HistFactory

- Generalization of number counting models

$$\mathcal{P}(n_b|\mu) = \text{Pois}(n_{\text{tot}}|\mu S + B)\left[\prod_{b\in\text{bins}} \frac{\mu\nu_b^{\text{sig}} + \nu_b^{\text{bkg}}}{\mu S + B}\right]$$

where n_b is the data histogram

in general HistFactory produces model of this form

$$\mathcal{P}(n_c, x_e, a_p \mid \phi_p, \alpha_p, \gamma_b) = \prod_{c\in\text{channels}} \left[\text{Pois}(n_c|\nu_c) \prod_{e=1}^{n_c} \underbrace{f_c(x_e|\boldsymbol{\alpha})}_{\text{p.d.f}}\right] \cdot \underbrace{G(L_0|\lambda, \Delta_L)}_{\text{luminosity constraint}} \cdot \prod_{p\in\mathbb{S}+\boldsymbol{\Gamma}} \underbrace{f_p(a_p|\alpha_p)}_{\text{parameter constraint}}$$

$$f_c(x_e|\phi_p, \alpha_p, \gamma_b) = \underbrace{\frac{\nu_{cb_e}}{\nu_c}}_{\text{expected events/bin}} \qquad \nu_c = \sum_{b\in\text{bins of channel } c} \nu_{cb}$$

HistFactory Model

$$\mathcal{P}(n_{cb}, a_p \mid \phi_p, \alpha_p, \gamma_b) = \prod_{c \in \text{channels}} \prod_{b \in \text{bins}} \text{Pois}(n_{cb}|\nu_{cb}) \cdot G(L_0|\lambda, \Delta_L) \cdot \prod_{p \in \mathbb{S}+\mathbf{\Gamma}} P_p(a_p|\alpha_p)$$

luminosity constraint

parameter constraint

expected number of events in a bin

$$\nu_{cb}(\phi_p, \alpha_p, \gamma_b) = \lambda_{cs}\, \gamma_{cb}\, \phi_{cs}(\boldsymbol{\alpha})\, \eta_{cs}(\boldsymbol{\alpha})\, \sigma_{csb}(\boldsymbol{\alpha})$$

λ_{cs}	luminosity parameter for each sample of a channel
γ_{cb_e}	bin by bin scale factor (statistical + systematics)
$\phi_{cs} = \prod_{p \in \mathbb{N}_c} \phi_p$	product of unconstrained normalisation. Depend on P.O.I. (e.g. signal rate)
$\eta_{cs}(\boldsymbol{\alpha})$	normalisation uncertainty for each sample of a channel
σ_{csb_e}	nominal bin content and its uncertainty (from input histograms)

HistFactory Capabilities

- HistFactory can include:
  - multiple channels and samples
  - unconstrained normalisation for any sample
  - parametrize variation in normalization due to systematic effects
  - bin by bin statistical uncertainty (overall for all samples)
  - parametrize systematic variation of a single bin

	Constrained	Unconstrained
Normalization Variation	`OverallSys` (η_{cs})	`NormFactor` (ϕ_p)
Coherent Shape Variation	`HistoSys` σ_{csb}	–
Bin-by-bin variation	`ShapeSys` & `StatError` γ_{cb}	`ShapeFactor` γ_{csb}

HistFactory Capabilities (2)

- In addition the HistFactory can
  - can combine multiple channels
  - produce a RooFit workspace which can be used in RooStats
    - can be used to combine several measurements
- Configuration can be done in XML or directly in C++ or Python

How To Create a Model

- Simple counting model

$$\text{Poisson}(\, n_{obs} \mid \mu + b)\ \text{Gaussian}(\, b \mid b_0, \sigma_b)$$

```
// create first input histograms
int nobs = 3; double b = 1; double errb = 0.2;

// observed histogram
TH1D * hobs = new TH1D(“hobs","hobs",1,0,1);
hobs->SetBinContent(1,nobs);

//signal histogram (assume expected one is 1)
TH1D * hs = new TH1D("hs","signal  histo",1,0,1);
hs->SetBinContent(1,1);


TH1D * hb = new TH1D("hb","bkg  histo",1,0,1);
hb->SetBinContent(1,b);
```

How To Create a Model (2)

- Create HistFactory Measurement class

```
HistFactory::Measurement meas("CountingModel","CountingModel");
meas.SetPOI("mu");

meas.SetLumi(1.0);
meas.SetLumiRelErr(0.1);  // not relevant
// this does not make lumi varying
meas.AddConstantParam("Lumi");
```

- Create Channels and Sample

```
HistFactory::Channel channel("SignalRegion");
channel.SetData(hobs);

HistFactory::Sample signal("signal");
signal.AddNormFactor("mu",1,0,30);
//signal.AddOverallSys("sig_unc",0.9, 1.1);
signal.SetHisto(hs);
channel.AddSample(signal);

HistFactory::Sample backg("background");
backg.SetHisto(h1_b);
backg.AddOverallSys("b_unc",1.-errb, 1+errb);  // b uncertainty
channel.AddSample(backg);

meas.AddChannel(channel);
```

How To Create a Model (3)

- Creating a RooWorkspace given the Measurement

```
RooWorkspace * w = HistFactory::MakeModelAndMeasurementFast(meas);
```

```
RooWorkspace(SignalRegion) SignalRegion workspace contents

variables
---------
(Lumi,alpha_b_unc,binWidth_obs_x_SignalRegion_0,binWidth_obs_x_SignalRegion_1,mu,nom_alpha_b_unc,nominalLumi,obs_x_S
nalRegion,weightVar)

p.d.f.s
-------
RooRealSumPdf::SignalRegion_model[ binWidth_obs_x_SignalRegion_0 * L_x_signal_SignalRegion_overallSyst_x_Exp +
binWidth_obs_x_SignalRegion_1 * L_x_background_SignalRegion_overallSyst_x_Exp ] = 2/2
RooGaussian::alpha_b_uncConstraint[ x=alpha_b_unc mean=nom_alpha_b_unc sigma=1 ] = 1
RooGaussian::lumiConstraint[ x=Lumi mean=nominalLumi sigma=0.001 ] = 1
RooProdPdf::model_SignalRegion[ lumiConstraint * alpha_b_uncConstraint * SignalRegion_model(obs_x_SignalRegion) ] =

functions
--------
RooProduct::L_x_background_SignalRegion_overallSyst_x_Exp[ Lumi * background_SignalRegion_overallSyst_x_Exp ] = 1
RooProduct::L_x_signal_SignalRegion_overallSyst_x_Exp[ Lumi * signal_SignalRegion_overallSyst_x_Exp ] = 1
RooStats::HistFactory::FlexibleInterpVar::background_SignalRegion_epsilon[ paramList=(alpha_b_unc) ] = 1
RooHistFunc::background_SignalRegion_nominal[ depList=(obs_x_SignalRegion) ] = 1
RooProduct::background_SignalRegion_overallSyst_x_Exp[ background_SignalRegion_nominal *
background_SignalRegion_epsilon ] = 1
RooHistFunc::signal_SignalRegion_nominal[ depList=(obs_x_SignalRegion) ] = 1
RooProduct::signal_SignalRegion_overallNorm_x_sigma_epsilon[ mu * signal_SignalRegion_epsilon ] = 1
RooProduct::signal_SignalRegion_overallSyst_x_Exp[ signal_SignalRegion_nominal *
signal_SignalRegion_overallNorm_x_sigma_epsilon ] = 1
```

HistFactory Output

- makes a combined workspace with data

```
RooWorkspace(combined) combined contents

variables
---------
(channelCat,nom_alpha_b_unc,obs_x_SignalRegion,weightVar)

datasets
--------
RooDataSet::asimovData(obs_x_SignalRegion,weightVar,channelCat)
RooDataSet::obsData(channelCat,obs_x_SignalRegion)

named sets
----------
ModelConfig_GlobalObservables:(nom_alpha_b_unc)
ModelConfig_Observables:(obs_x_SignalRegion,weightVar,channelCat)
globalObservables:(nom_alpha_b_unc)
observables:(obs_x_SignalRegion,weightVar,channelCat)
```

- create also a ModelConfig

```
=== Using the following for ModelConfig ===
Observables:              RooArgSet:: = (obs_x_SignalRegion,weightVar,channelCat)
Parameters of Interest:   RooArgSet:: = (mu)
Nuisance Parameters:      RooArgSet:: = (alpha_b_unc)
Global Observables:       RooArgSet:: = (nom_alpha_b_unc)
PDF:                      RooSimultaneous::simPdf[ indexCat=channelCat SignalRegion=model_SignalRegion ] = 2
```

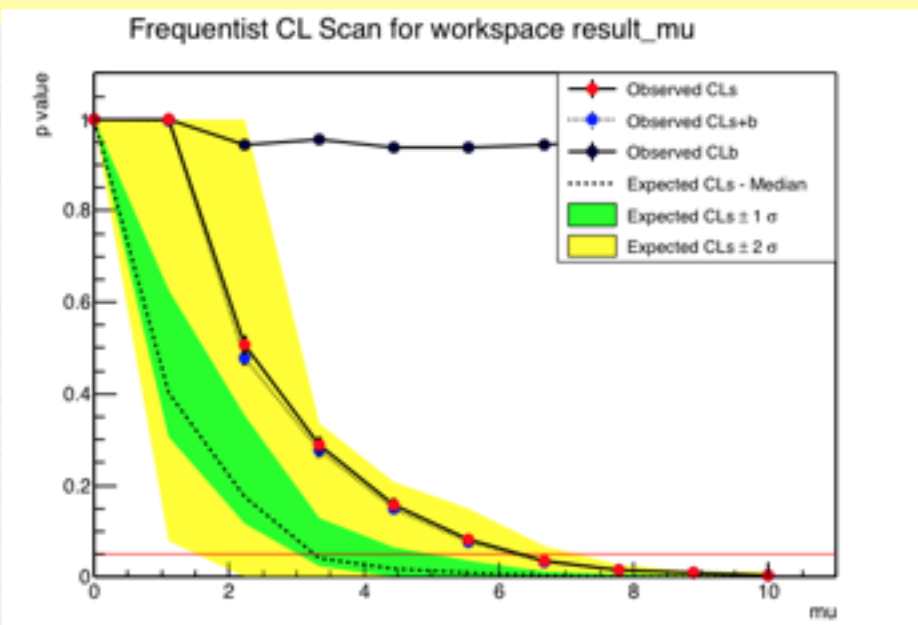
Using HistFactory Models

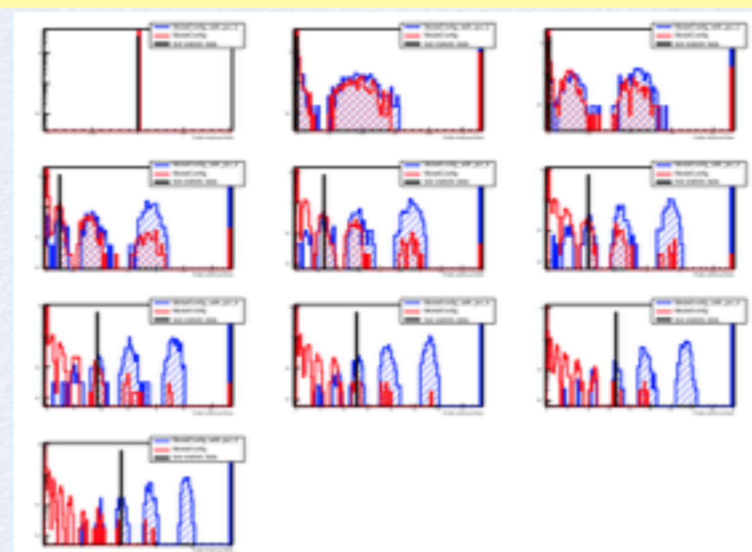
- Combined model saved in a ROOT file
- Model can be used directly in RooStats tools

```
root[] .L StandardHypoTestInvDemo.C
```

run for CLs (with frequentist calculator (type = 0) and one-side PL test statistics (type = 3) scan 10 points in [0,10]

```
root[] StandardHypoTestInvDemo(“model.root","combined","ModelConfig","","obsData",0,3, true, 10, 0, 10)
```





Interpolation Options

HistFactory has different option for interpolating the systematic variations : $\eta(\alpha)$

- 0) Linear
- 1) Exponential
- 2) Quadratic interp. linear extrapolation
- 4) Polynomial interpolation Exponential extrapolation (default)

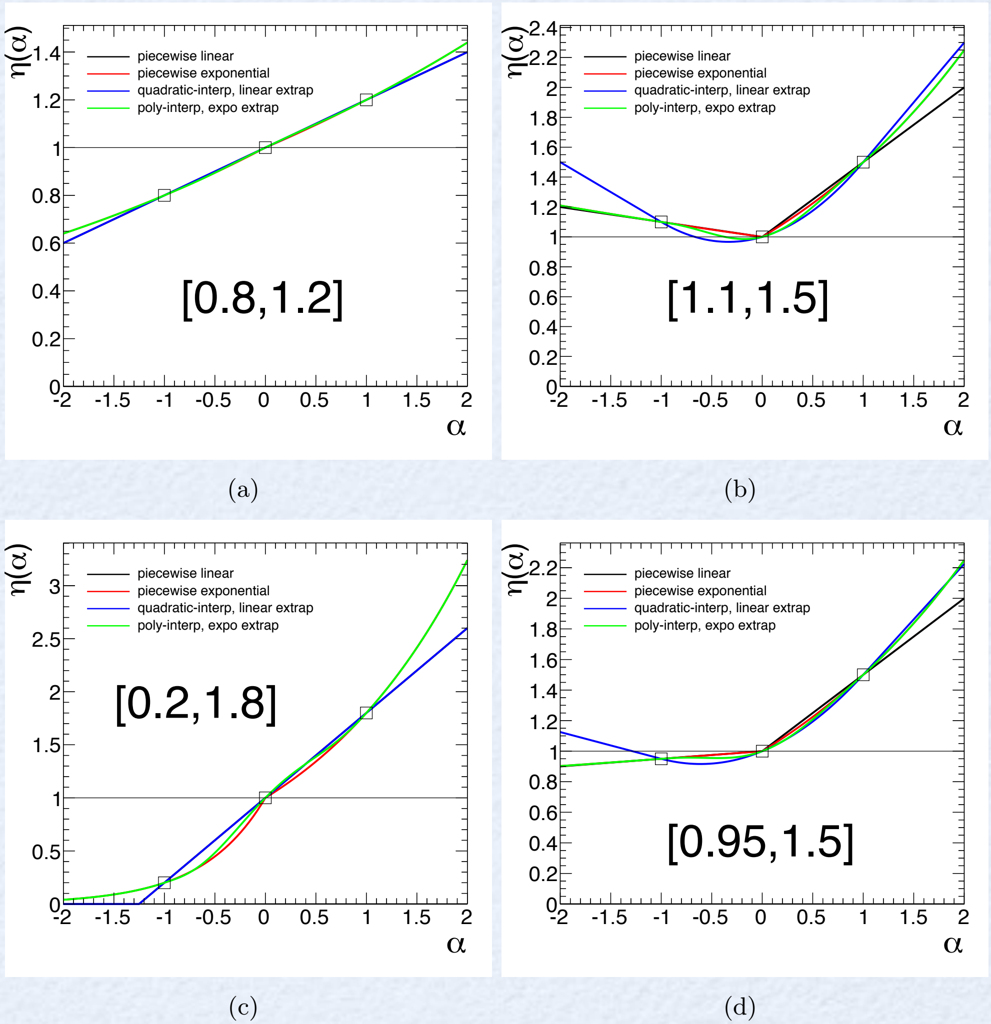


Figure 3: Comparison of the three interpolation options for different $\eta^{\pm}$. (a) $\eta^- = 0.8$, $\eta^+ = 1.2$, (b) $\eta^- = 1.1$, $\eta^+ = 1.5$, (c) $\eta^- = 0.2$, $\eta^+ = 1.8$, and (d) $\eta^- = 0.95$, $\eta^+ = 1.5$

Time For Last Exercise !

- Simple Model building example (**HistFactoryModel** notebook)
  - build of a counting model using the HistFactory

Summary

- RooFit/RooStats allow you to perform advanced statistical data/analysis
  - LHC results (*e.g.* Higgs observation)
- Capable of using different tools and interpretations (Frequentist/Bayesian) on the same model
- Generic tools capable to deal with large variety of models
  - based on histograms or un-binned data
  - multi-dimensional observations
- Provide tools to facilitate complex model building
  - HistFactory for histogram based analysis

Documentation

- RooStats TWiki: https://twiki.cern.ch/twiki/bin/view/RooStats/WebHome
- RooStats users guide (not really completed)
  - http://root.cern.ch/viewcvs/branches/dev/roostats/roofit/roostats/doc/usersguide/RooStats_UsersGuide.pdf
- For reference and citation: ACAT 2010 proceedings papers: http://arxiv.org/abs/1009.1003
- RooStats tutorial macros: http://root.cern.ch/root/html534/tutorials/roostats/index.html
- HistFactory document: https://cdsweb.cern.ch/record/1456844/files/CERN-OPEN-2012-016.pdf
- RooStats user support:
  - Request support via ROOT talk forum: http://root.cern.ch/phpBB2/viewforum.php?f=15 (questions on statistical concepts accepted)
  - contact me directly (email: Lorenzo.Moneta at cern.ch)
- Contacts for statistical questions:
  - ATLAS statistics forum:
    - TWiki: https://twiki.cern.ch/twiki/bin/view/AtlasProtected/StatisticsTools
  - CMS statistics committee:
    - TWiki: https://twiki.cern.ch/twiki/bin/view/CMS/StatisticsCommittee

Thank you !