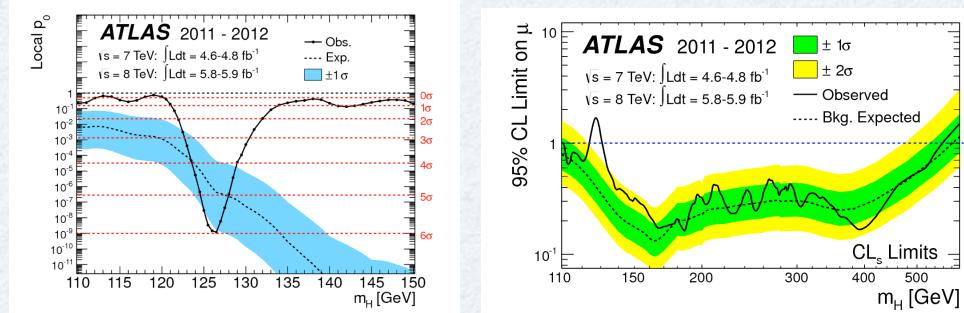


RooStats

Lecture and Tutorials



Outline

- Introduction to RooFit
 - Basic functionality
 - Model building using the workspace
 - Composite models
- Exercises on RooFit:
 - building and fitting models
- Introduction to **RooStats**
 - **Interval** estimation tools (**Likelihood/Bayesian**)
 - **Hypothesis** tests
 - Frequentist interval/limit calculator (CLs)
- Exercises on interval/limit estimation and discovery significance (hypothesis test)

RooStats Project

- Collaborative project to provide and consolidate advanced statistical tools needed by LHC experiments
- Joint contribution from ATLAS, CMS, ROOT and RooFit
 - developments over-sighted by ATLAS and CMS statistics committees
 - initiated from previous code developed in ATLAS and CMS
 - used by both collaborations

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 easily compare different statistical methods
 - provide utility for combinations of results
 - using same tools across experiments facilitates the combinations of results

Statistical Applications

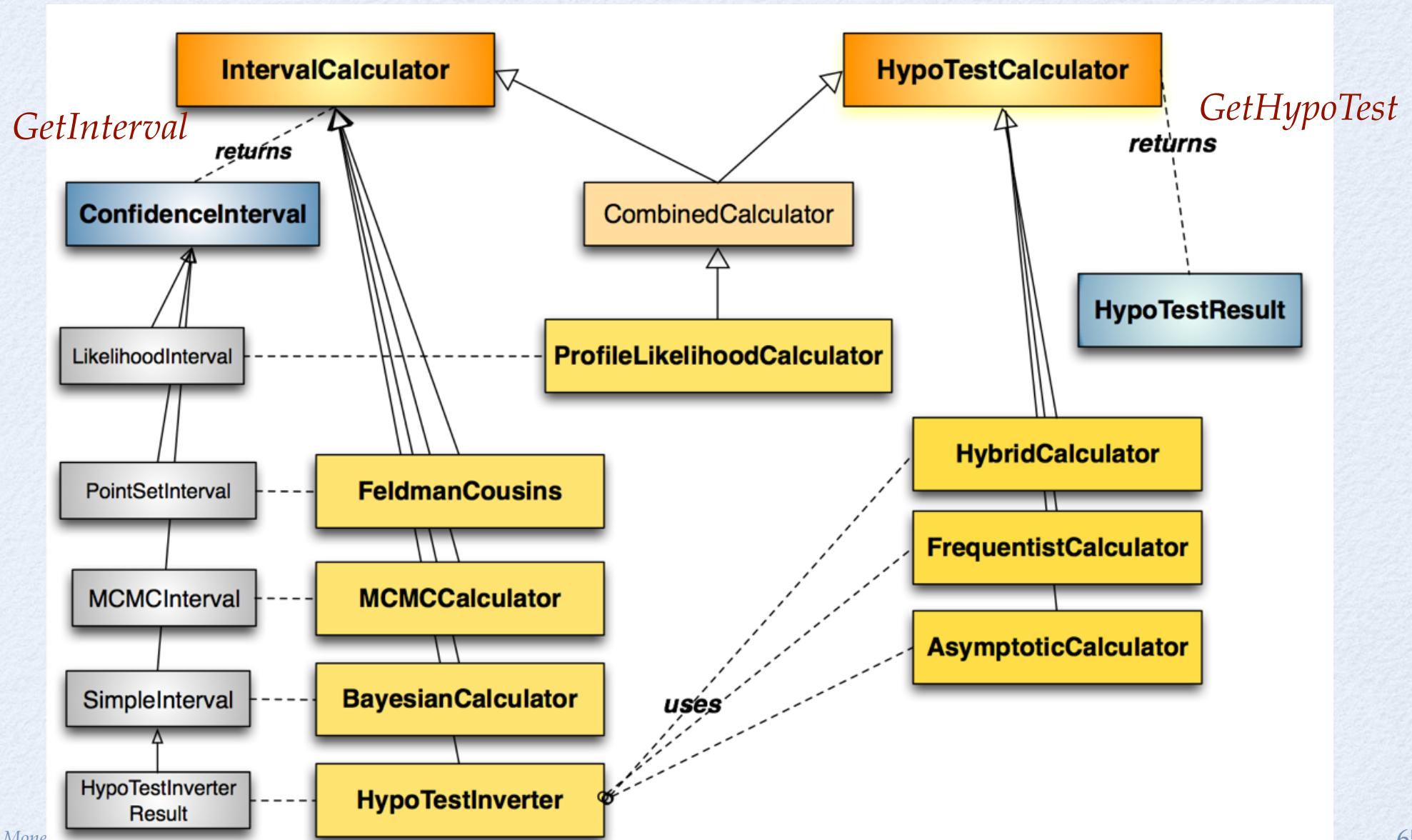
- Statistical problems:
 - point estimation (covered by RooFit)
 - estimation of confidence (credible) intervals
 - hypothesis tests
 - goodness of fit (not yet addressed)

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 Calculator classes

Interval Calculators

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

HypoTest Calculators

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(xlmu,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 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:

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

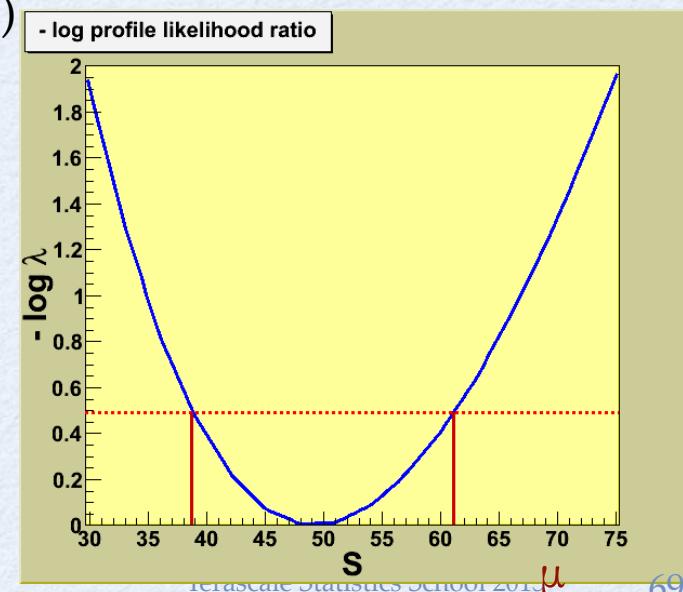
maximize w.r.t nuisance parameters ν and fix POI μ

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 $\log\lambda$ around the minimum:
 - $-2\log\lambda$ is a parabola (λ is a gaussian function)
 - interval on μ from $\log\lambda$ values

- Method of **MINUIT/MINOS**
 - lower/upper limits for 1D
 - contours for 2 parameters



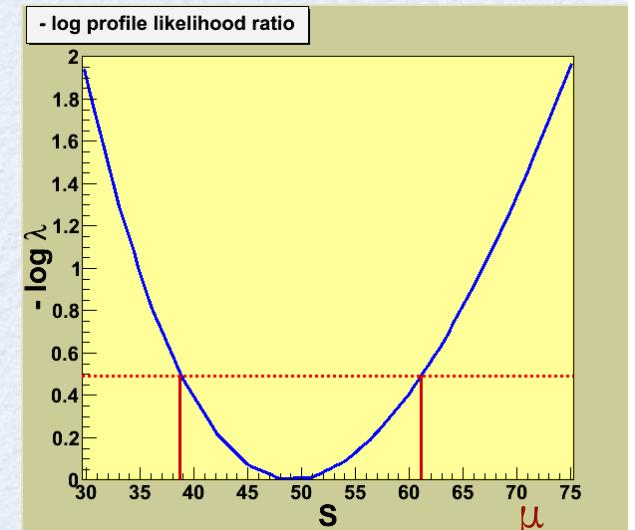
Using the Profile Likelihood Calculator

```
// 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 (1σ) interval : $\Delta \log \lambda = 0.5$
 - 95% CL interval : $\Delta \log \lambda = 1.96$
- **LikelihoodIntervalPlot** can plot the 2D contours

Bayesian Analysis in RooStats

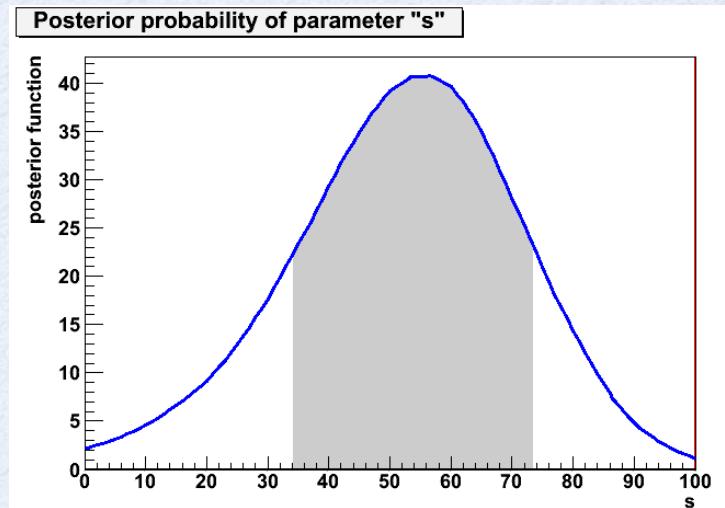
- **RooStats** provides classes for
 - marginalize posterior and estimate credible interval

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

posterior probability likelihood function prior probability nuisance parameters
 POI data marginalization

Bayesian Theorem

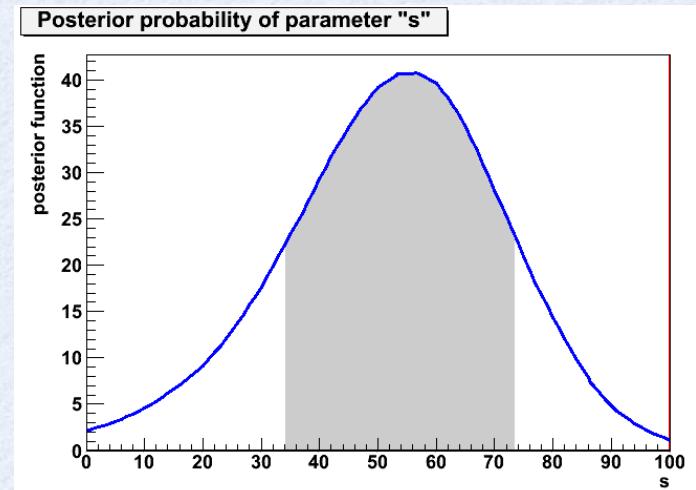
- support for different integration algorithms:
 - adaptive (numerical)
 - MC integration
 - Markov-Chain
- can work with models with many parameters (e.g few hundreds)



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

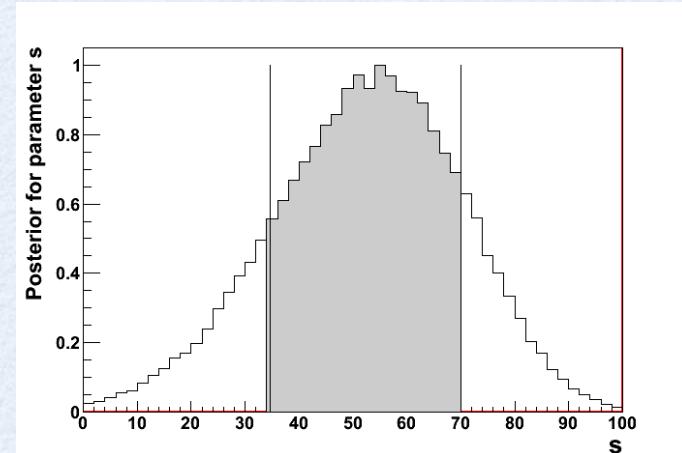
```
BayesianCalculator bc(data, model);
bc.SetConfidenceLevel(0.683);
bc.SetLeftSideTailFraction(0.5);
bc.SetIntegrationType("ADAPTIVE");
SimpleInterval* interval = bc.GetInterval();
double lowerLimit = interval->LowerLimit();
double upperLimit = interval->UpperLimit();
RooPlot * plot = bc.GetPosteriorPlot();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

MCMC Calculator

MCMCCalculator

- **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 mc(data, model);
mc.SetConfidenceLevel(0.683);
mc.SetLeftSideTailFraction(0.5);
SequentialProposal sp(0.1);
mc.SetProposalFunction(sp);
mc.SetNumIter(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);
```

Running RooStats

- 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 !

Follow the Twiki page at

<https://twiki.cern.ch/twiki/bin/view/RooStats/RooStatsTutorialsMarch2015>

RooStats

Part2

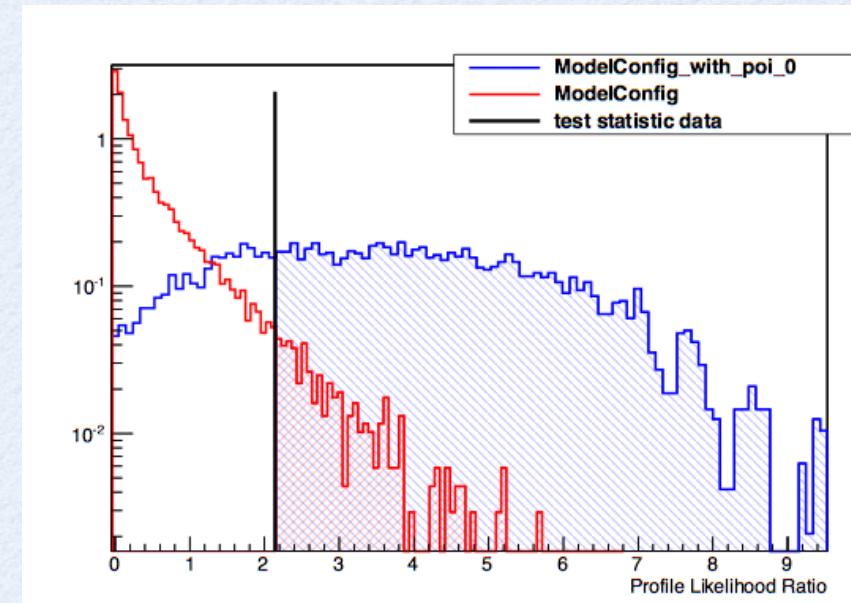
- Hypothesis tests in RooStats using toys and asymptotic formulae
- Hypothesis test inversion
 - Limit and interval calculators
 - CLs, Feldman-Cousins

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
 - **Alternate Hypothesis:** the competitive hypothesis (e.g. $\theta \neq \theta_0$)
 - w is the **critical region**, a subspace of all possible data:
 - **size of test :** $\alpha = P(X \in w \mid H_0)$
 - **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



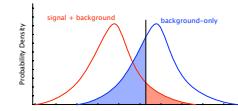
Test Statistics

- 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)

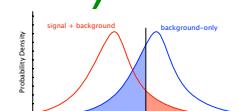
- simple likelihood ratio (used at LEP, nuisance parameters fixed)

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



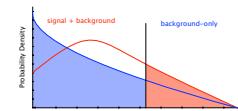
- ratio of profiled likelihoods (used commonly at Tevatron)

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



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

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



- 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
 - $L = \text{Poisson}(n_{\text{obs}} \mid \mu + b) \text{ Gaussian}(b_0 \mid b, \sigma_b)$
 - 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 toy different nuisance parameters values
 - need to have a pdf for the nuisance parameters (often it can be derived automatically from the model)

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



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

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);

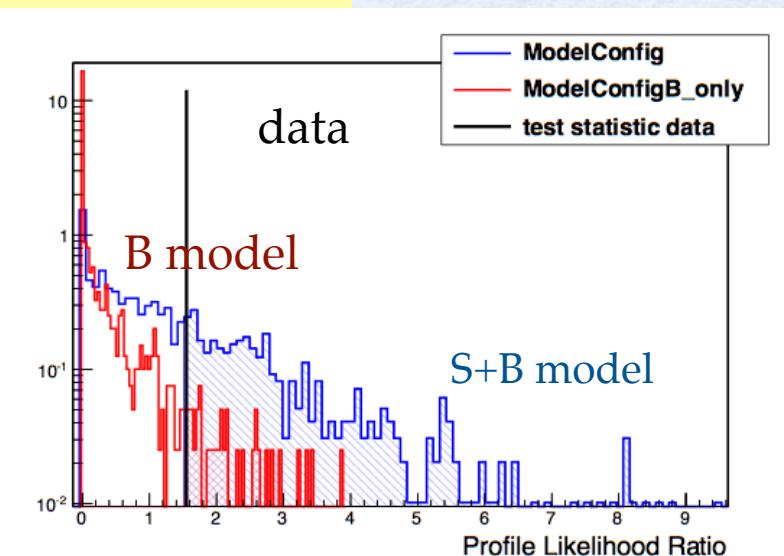
fcalc.SetToys(1000,1000); // set number of toys for (null, alt)

// 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

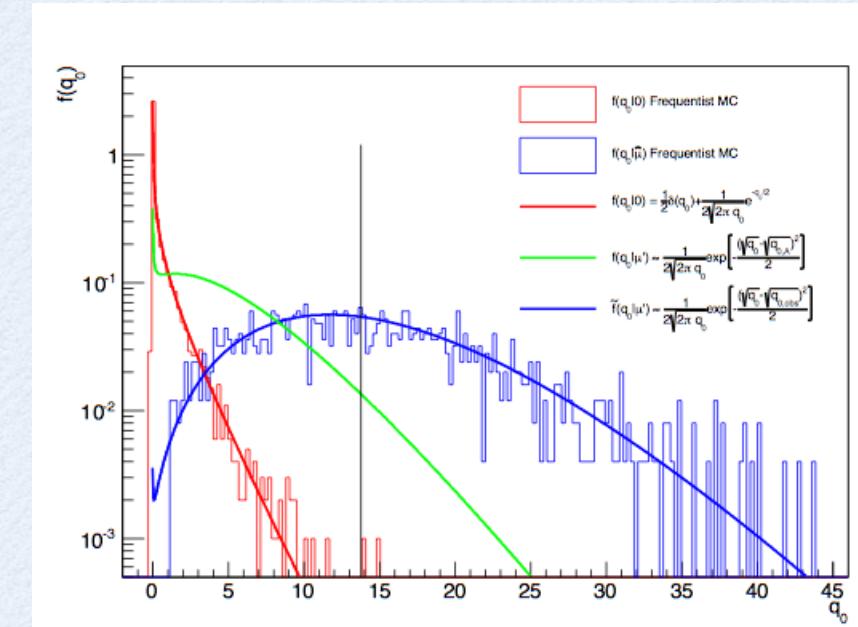


Asymptotic Calculator

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

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

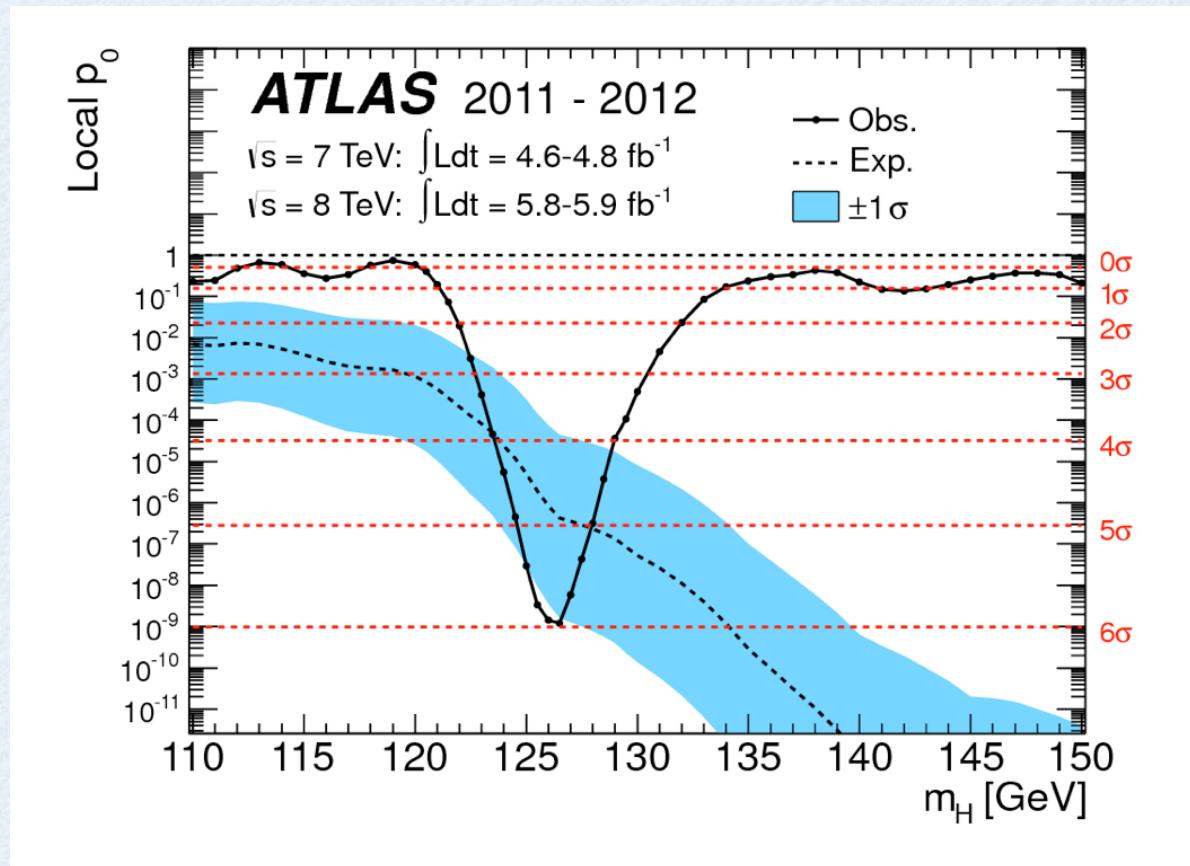
$\lambda(\mu) = 0$ for
 $\hat{\mu} < 0$ (discovery)
 $\hat{\mu} < \mu_{\text{TEST}}$ (limits)



→ 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

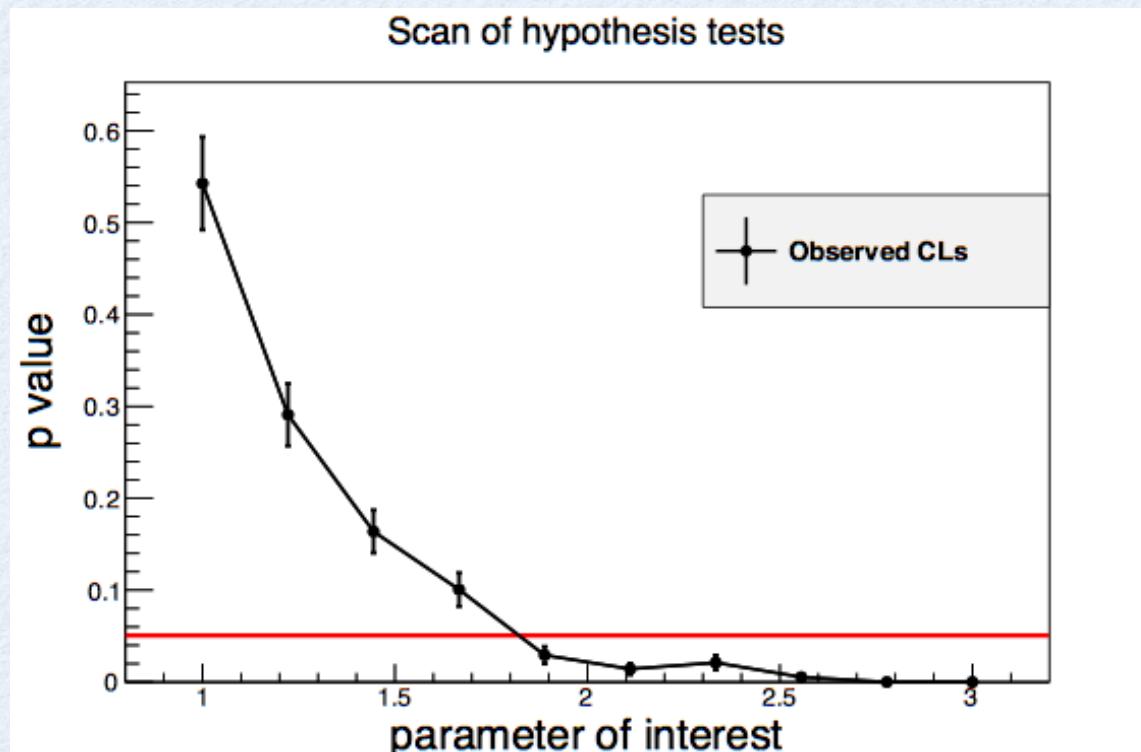
Property of test	Property of corresponding confidence interval
Size = α	Confidence coefficient = $1 - \alpha$
Power = probability of rejecting a false value of $\theta = 1 - \beta$	Probability of not covering a false value of $\theta = 1 - \beta$
Most powerful	Uniformly most accurate
Equal-tails test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

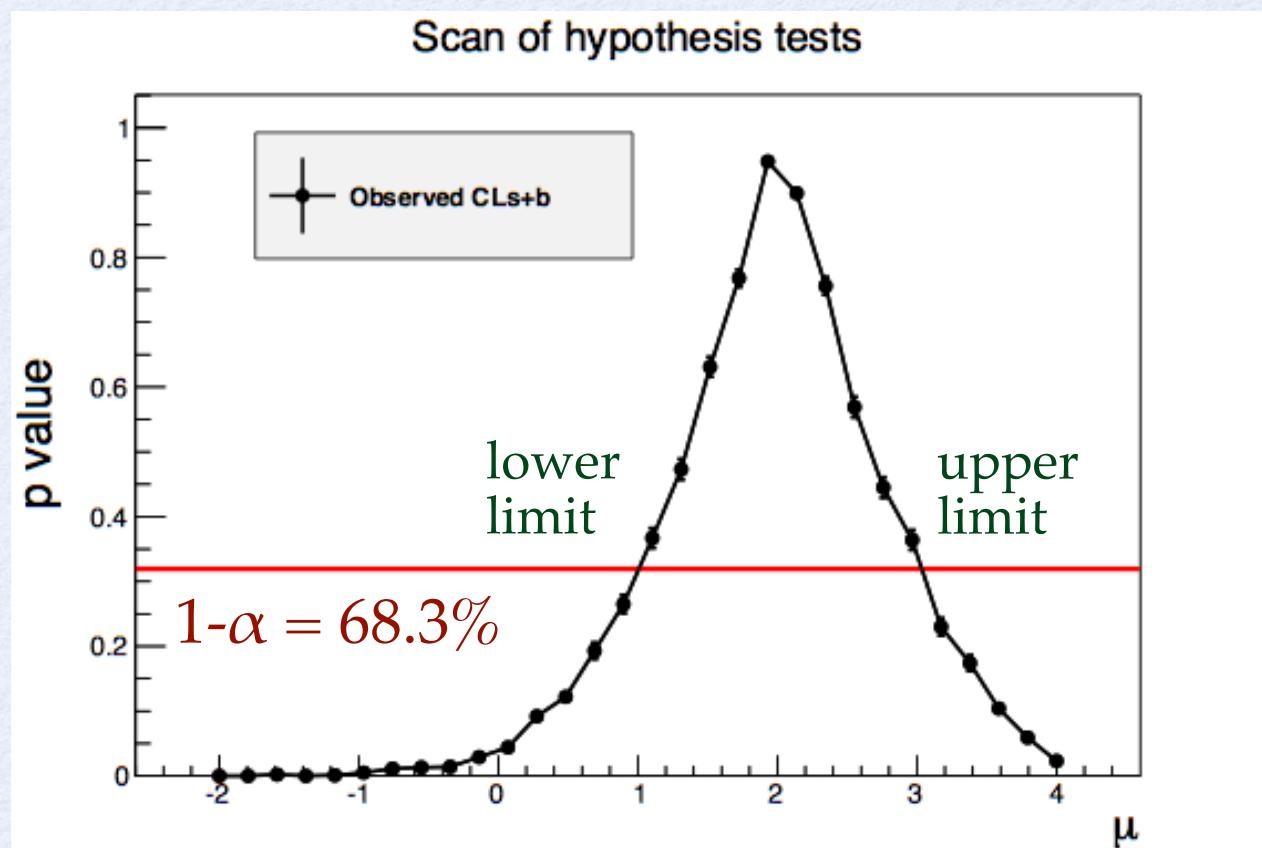
Hypothesis Test Inversion

- 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 $1-\alpha$ 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: $1-\sigma$ 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 prof11(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood
prof11.SetOneSided(true);
toymcs->SetTestStatistic(&prof11);

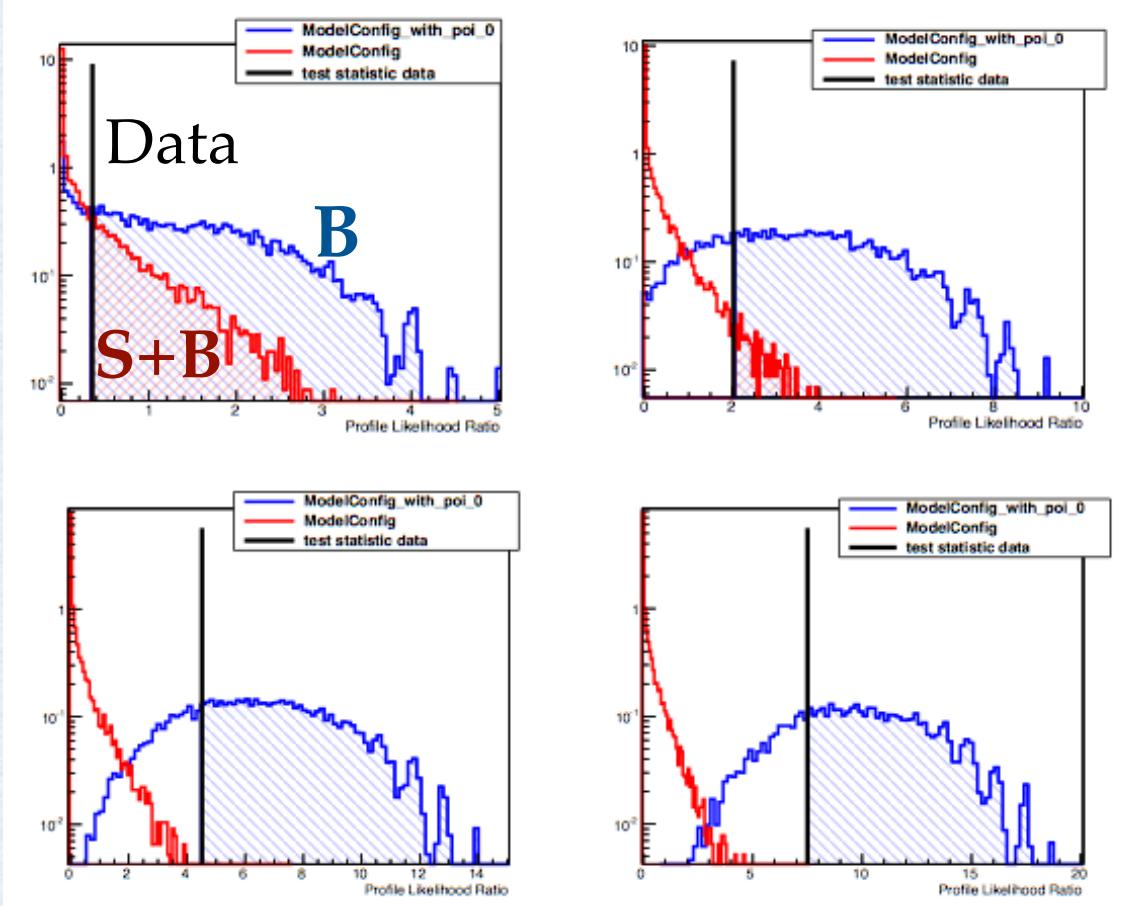
// configure and run the scan
calc.SetFixedScan(npoints,poimin,poimax);
HypoTestInverterResult * r = calc.GetInterval();

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

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

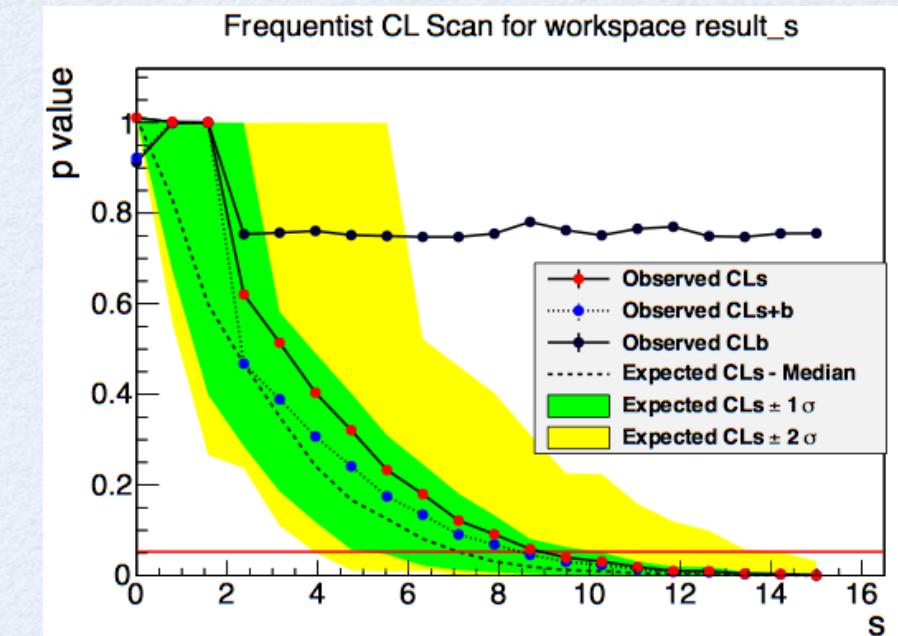
Running the HypoTestInverter

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 ?

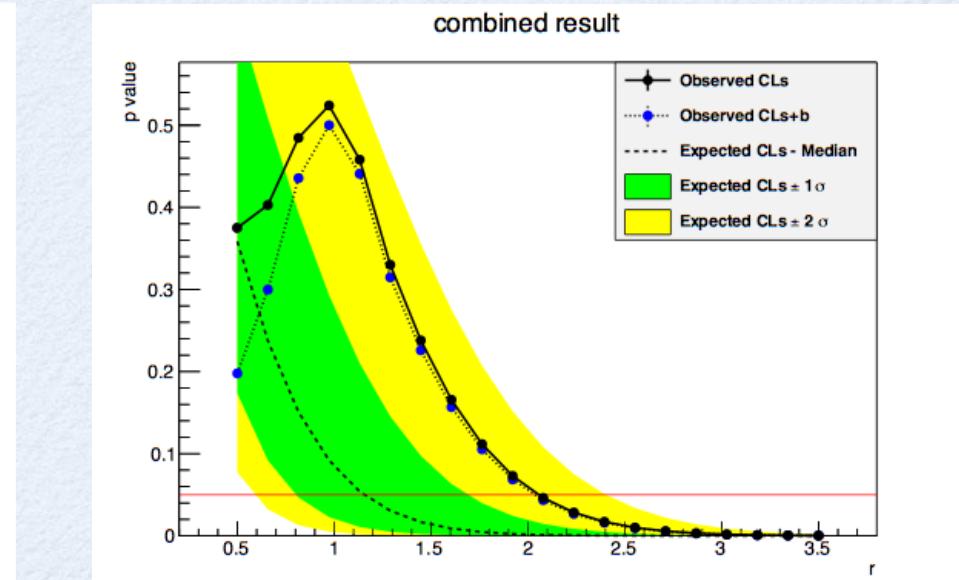
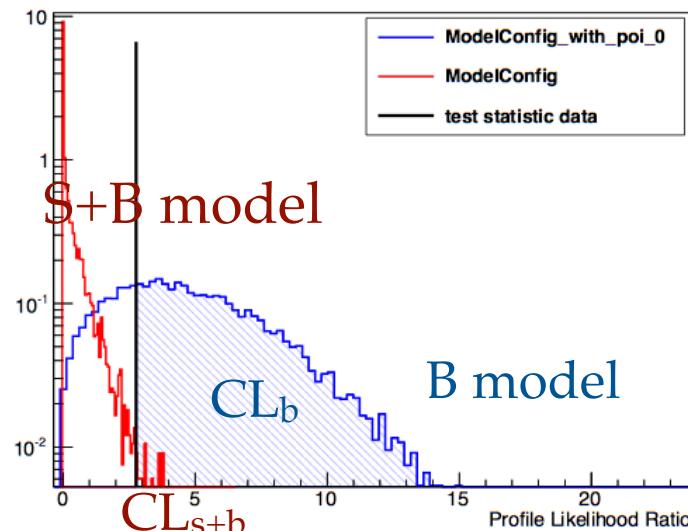
- compute p-value for quantiles (median, +/1,2 sigma) of the B model test statistic distribution (*i.e.* use quantile as the observed value)

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

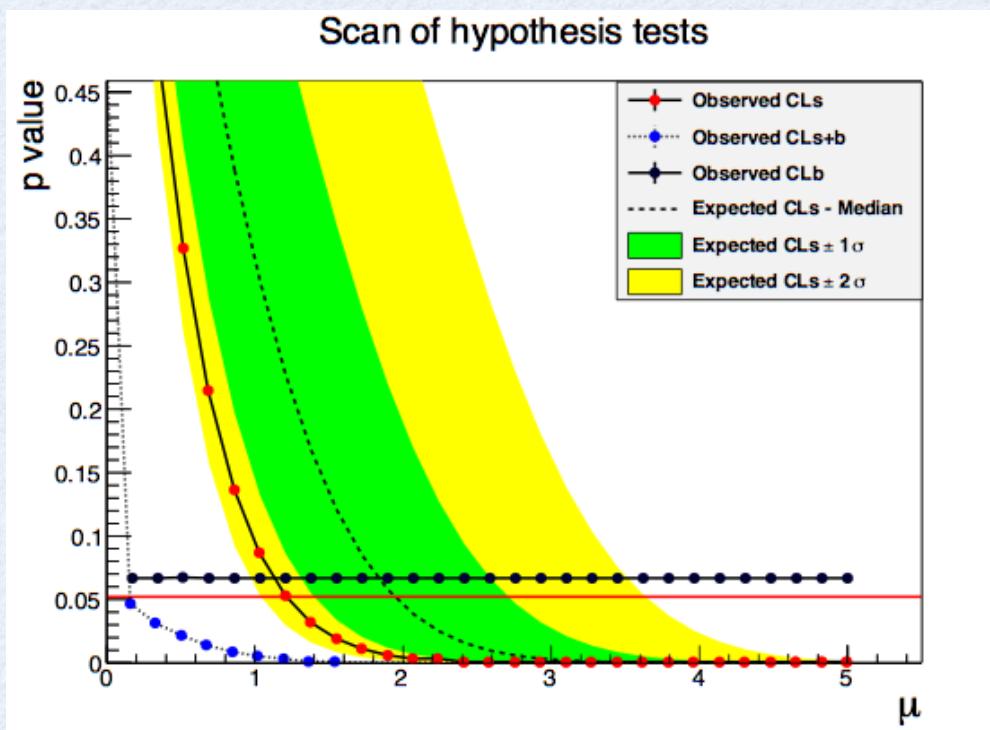
```
// create first HypoTest calculator (N.B null is s+b model)
AsymptoticCalculator ac(*data, *bModel, *sbModel);

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



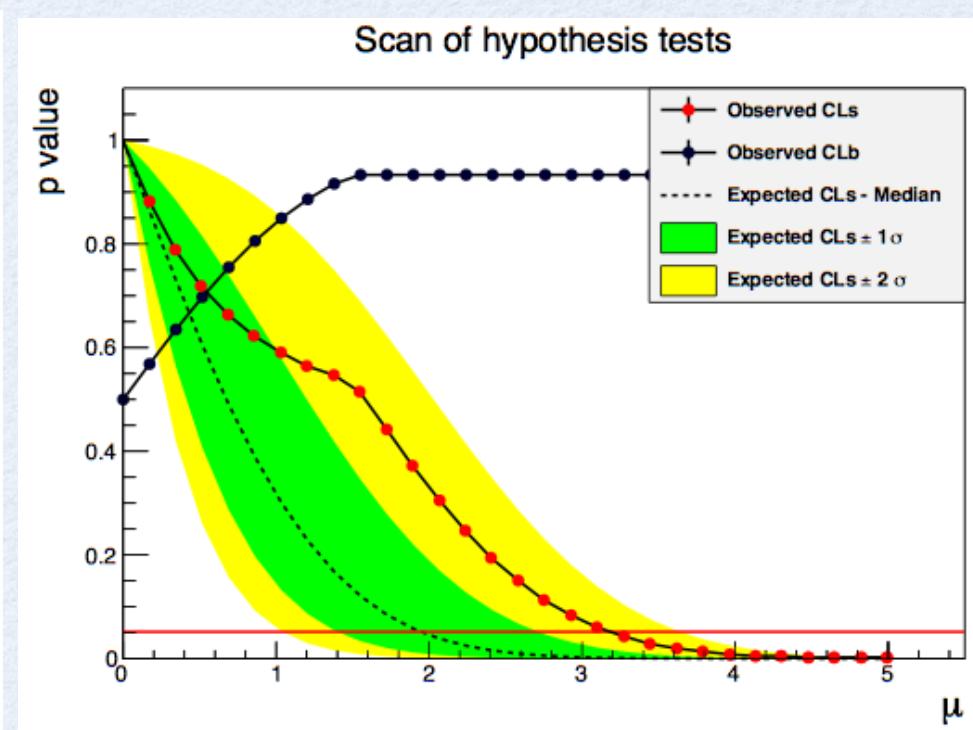
Example of Scan

- 95% CL limit on a Gaussian measurement:
 - $\text{Gauss}(x, \mu, 1)$, with $\mu \geq 0$



deficit, observation $x = -1.5$

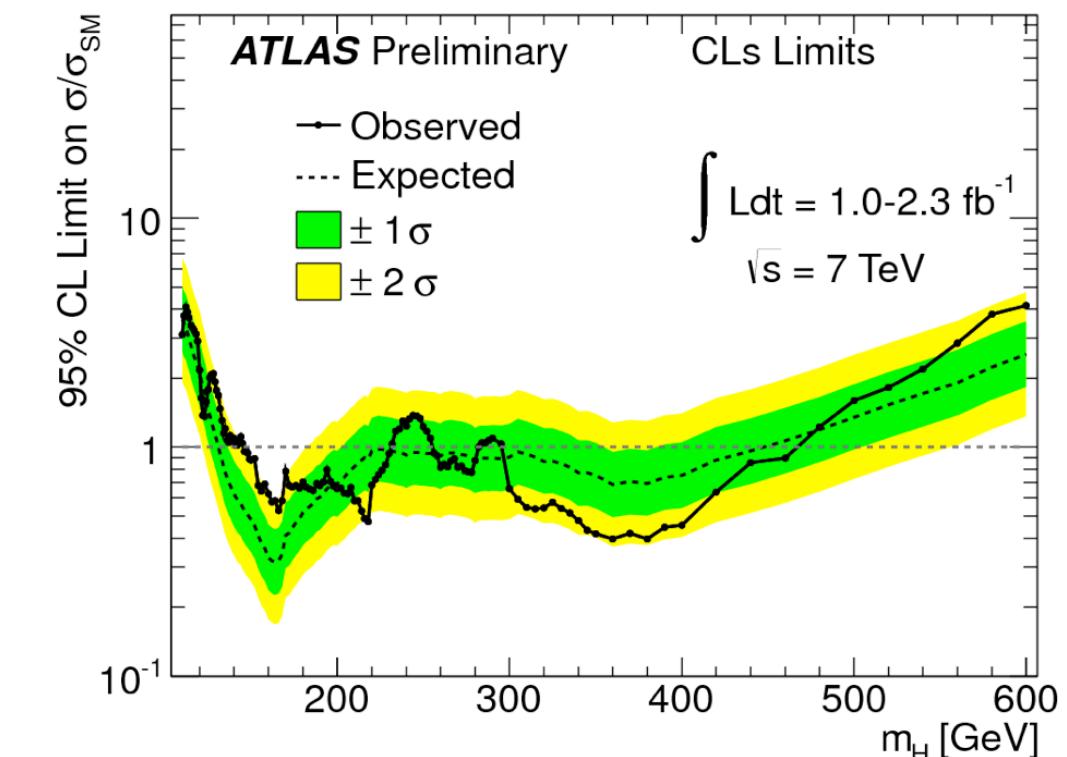
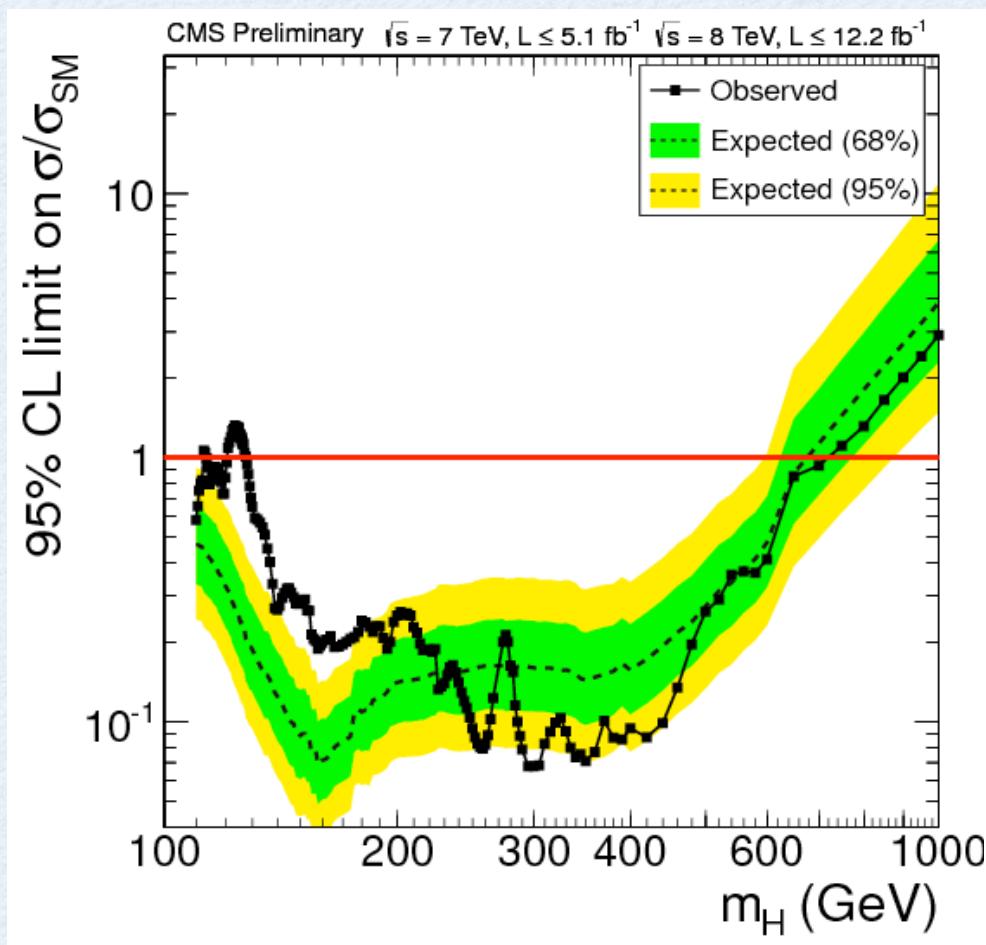
use CL_s as p-value to avoid setting limits which are too good



excess, observation $x = 1.5$

Example: Computing Limits

- By computing limits for different mass hypothesis:

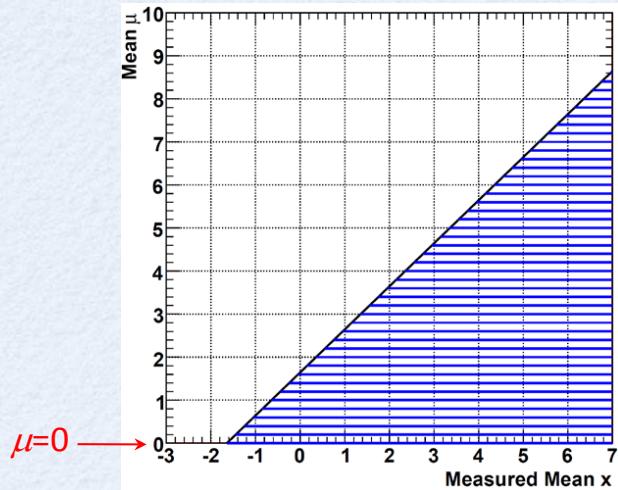


Limits on bounded measurements

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).



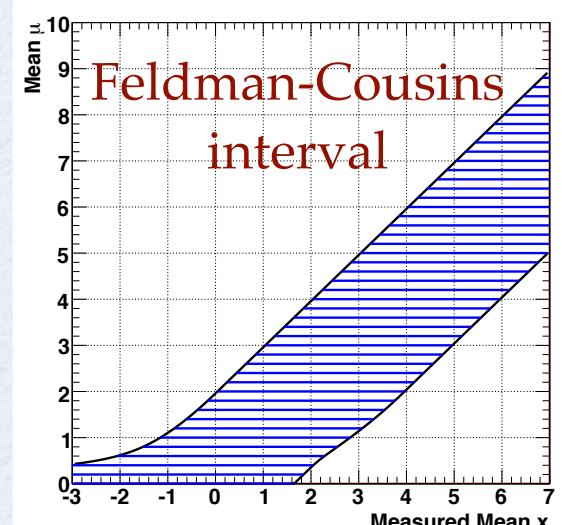
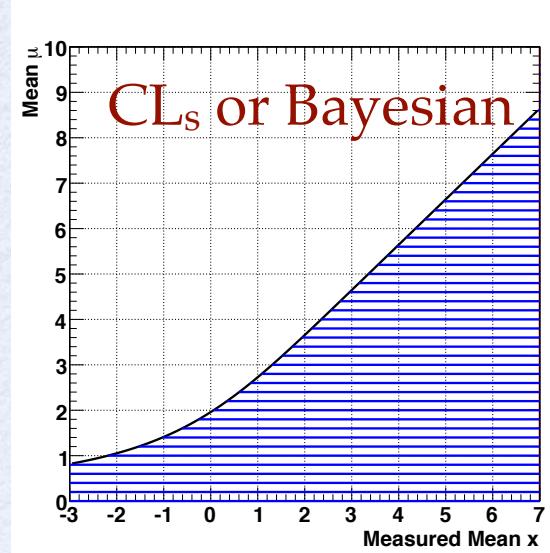
Frequentist 1-sided 95% C.L. Upper Limits, based on $\alpha = 1 - \text{C.L.} = 5\%$ (called CL_{sb} at LEP).

For $x < -1.64 \sigma$ 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.

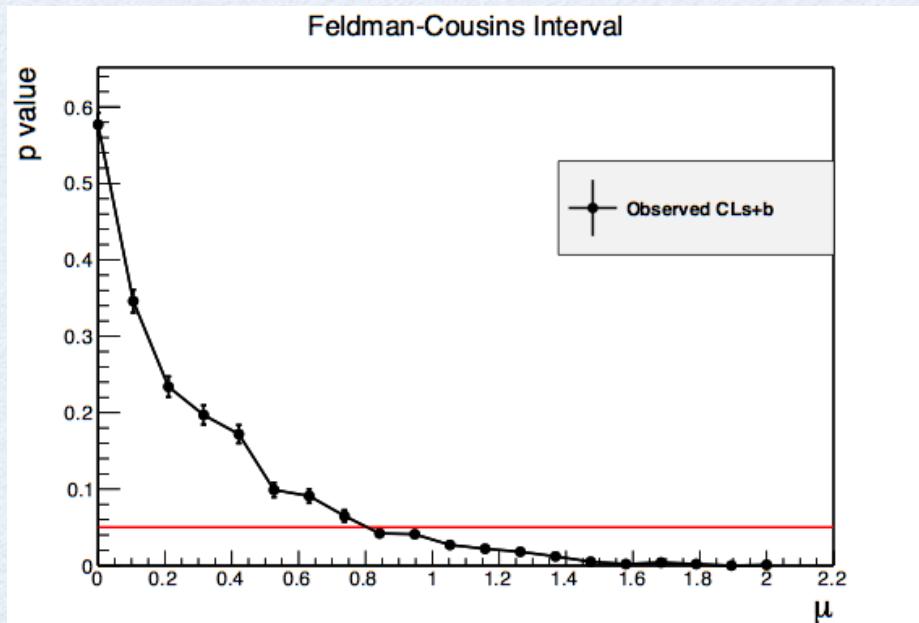
Issue acute 15-25 years ago in expts to measure ν_e mass in (tritium β decay): several measured $m_\nu^2 < 0$.



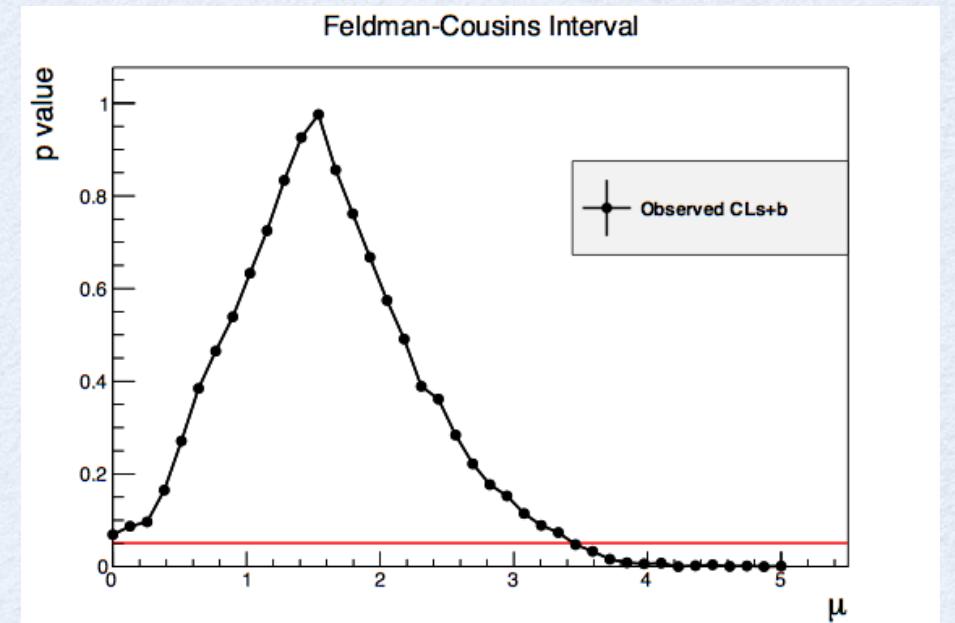
Feldman-Cousins intervals

- 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

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



observation $x = -1.5$

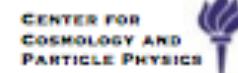


observation $x = 1.5$

Feldman-Cousins Interval

from Kyle Cranmer:

A different way to picture Feldman-Cousins

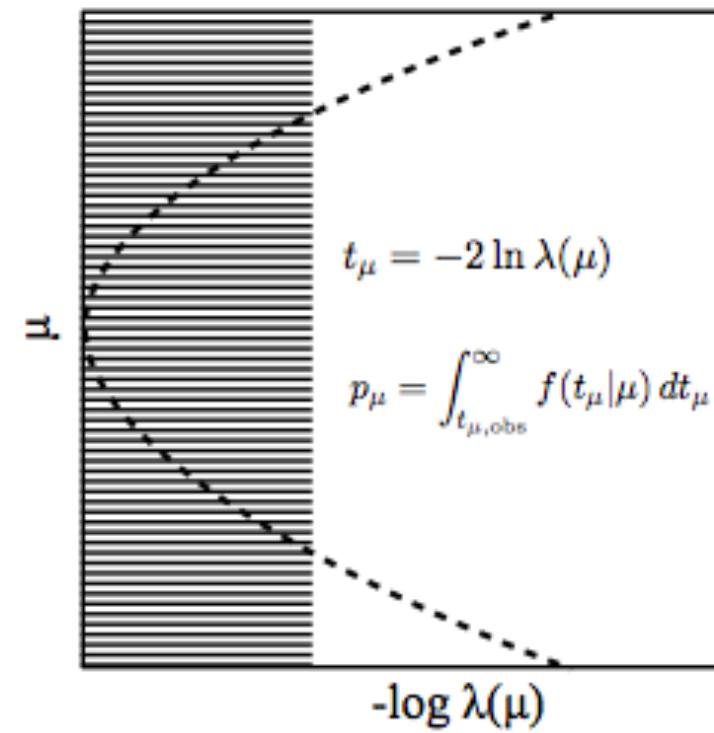
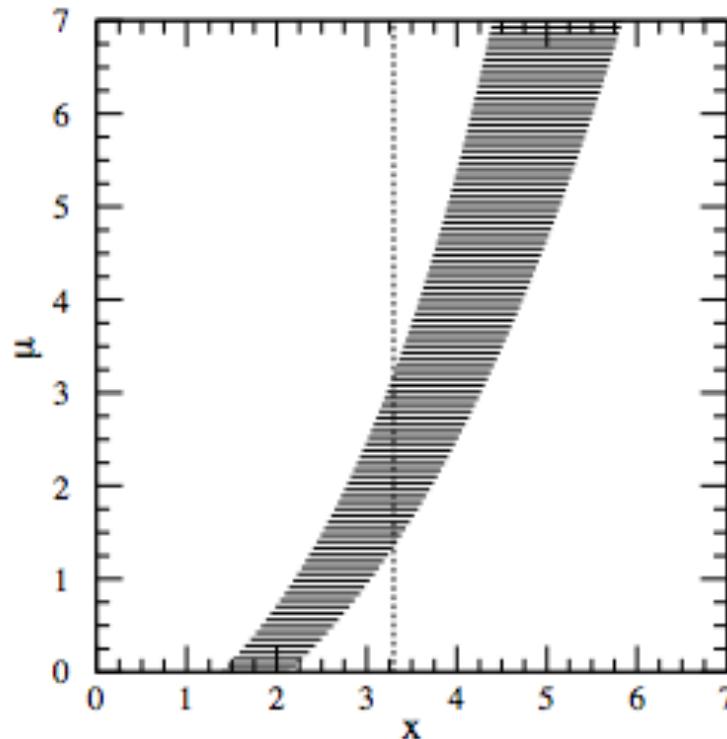


Most people think of plot on left when thinking of Feldman-Cousins

- 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 $\lambda(\mu)$

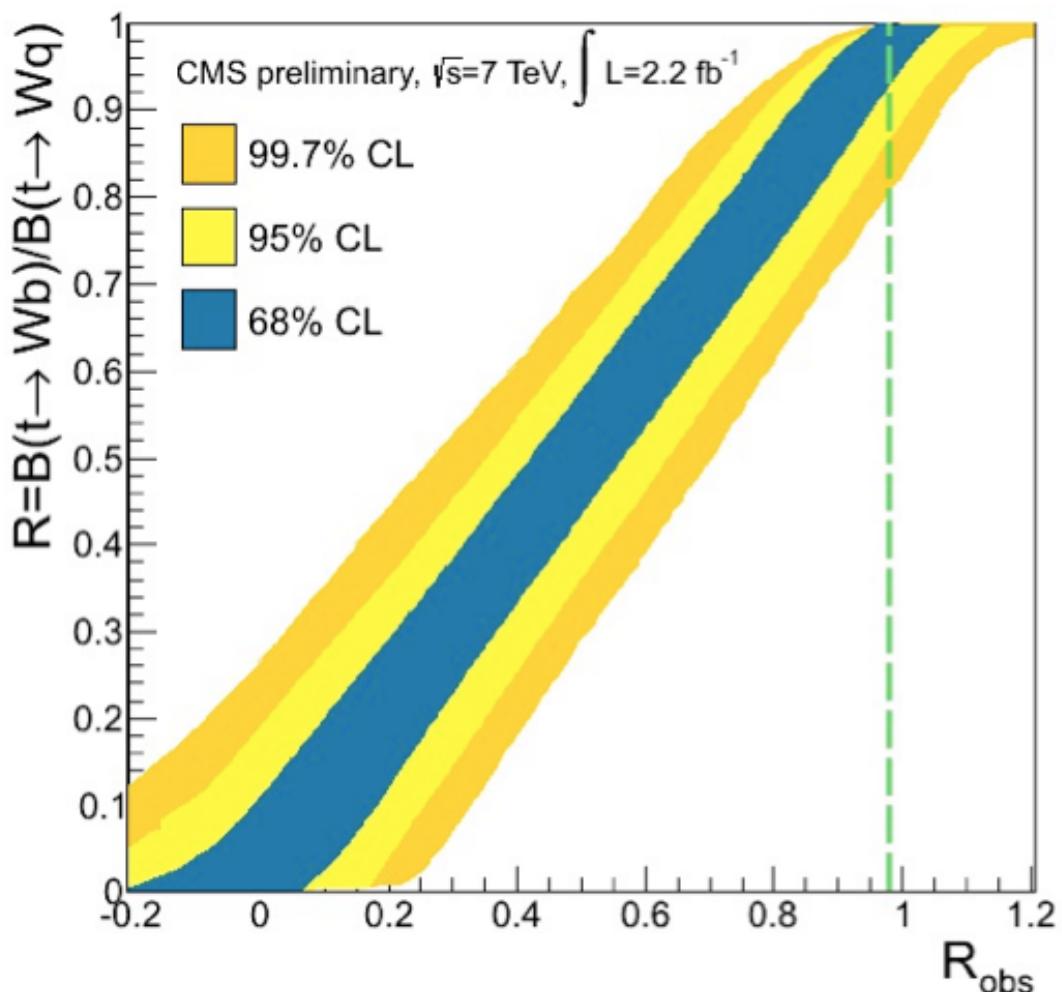




Example: Feldman-Cousins interval



- 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 CL_s (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 CL_s (scan 20 points in [0,100])

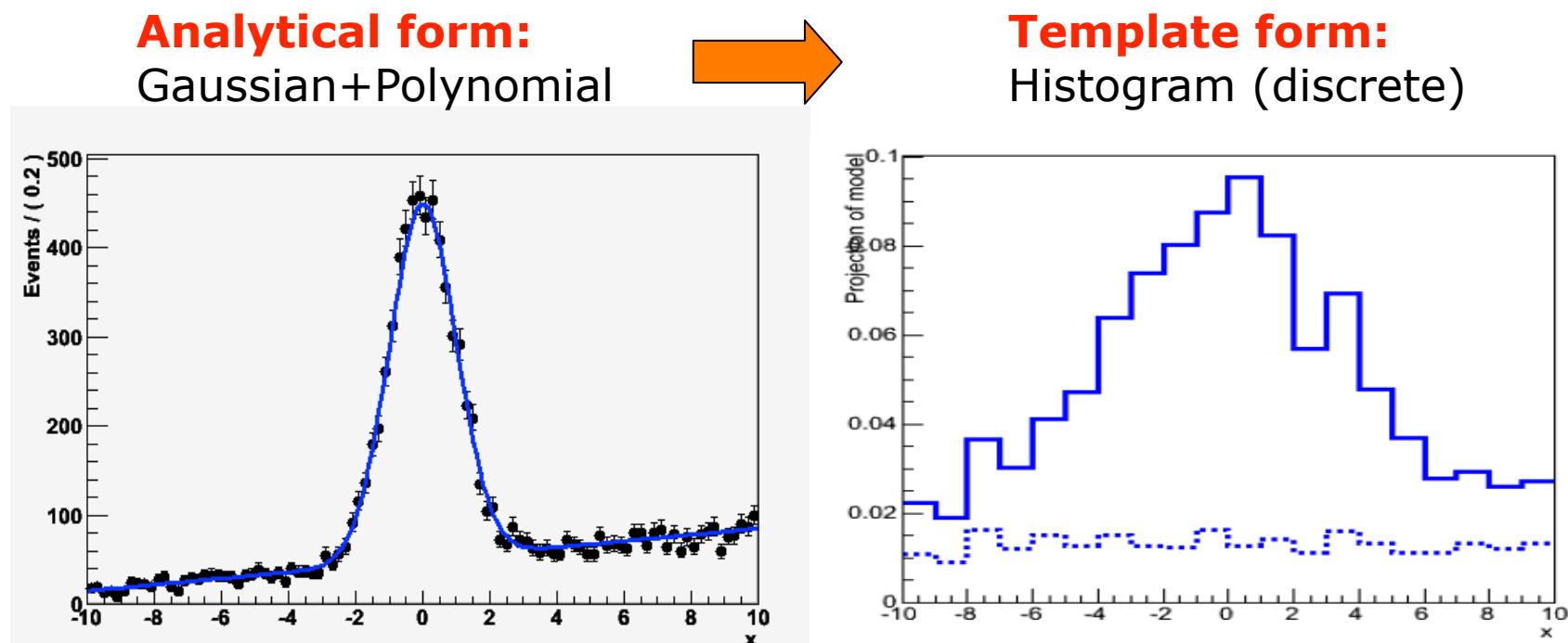
```
root[] StandardHypoTestInvDemo(SPlusBExpoModel.root,"w","ModelConfig","", "data",2,3, true, 20, 0, 100)
```

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

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

HistFactory – a new class of pdfs

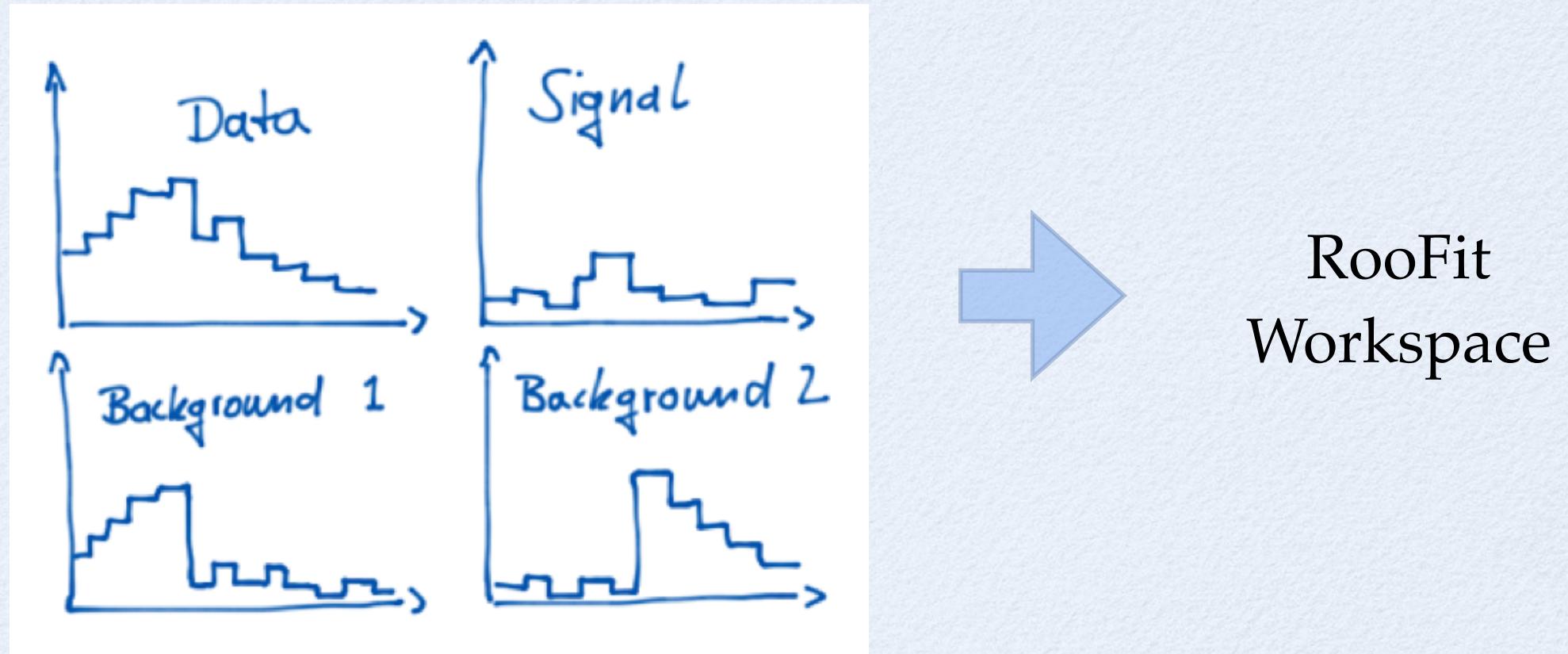
- Focus of RooFit traditionally on analytical models
 - Assumes you can formulate signal/background in an analytical form
 - Often possible in e+e- experiments,
shapes for hadron colliders cumbersome → **rely on MC simulation**



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

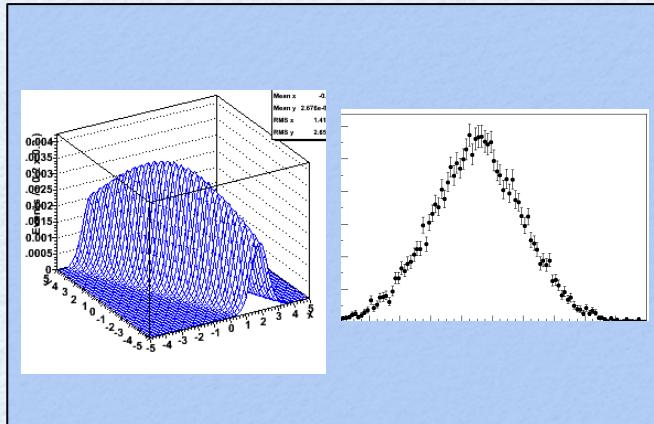
- Tool to build models from input histograms



RooFit/RooStats at LHC (Higgs analysis)

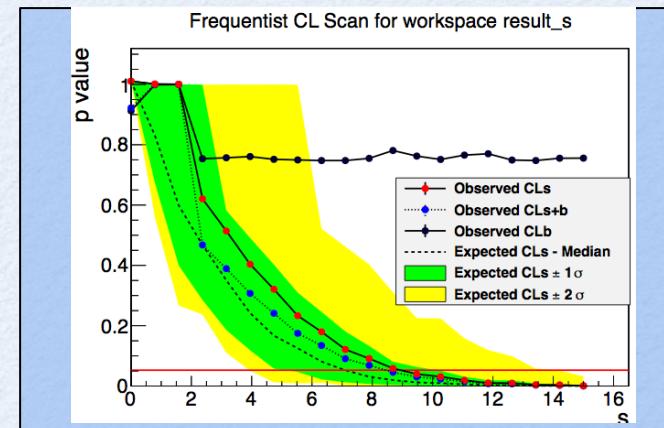
Class RooWorkspace

*Simplify packaging
and sharing of models*



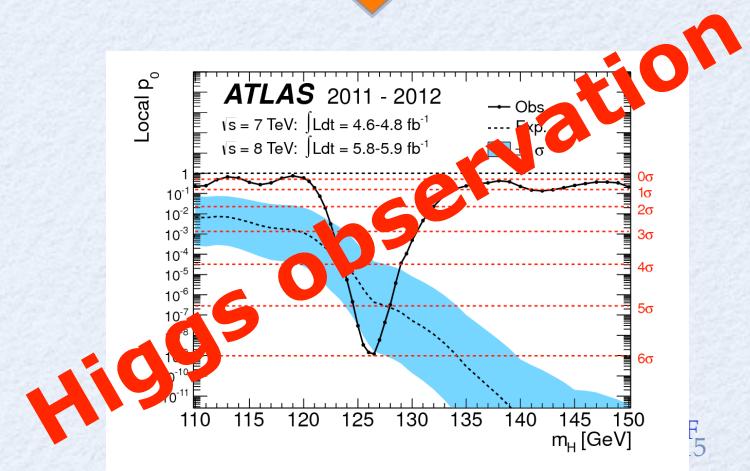
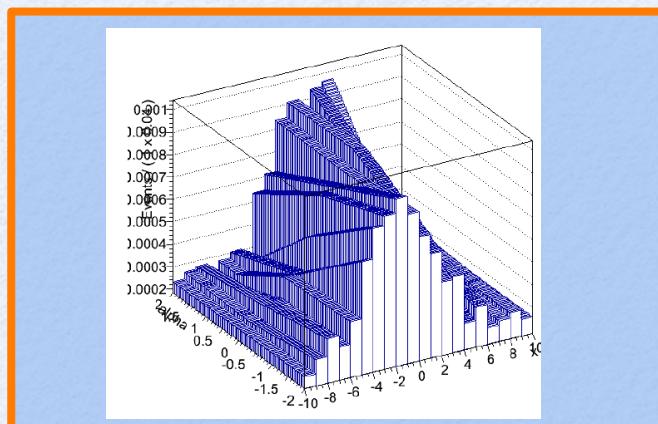
RooStats toolkit

*Statistical tests based on
likelihoods from RooFit models*

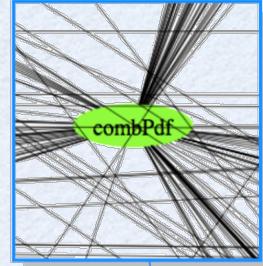
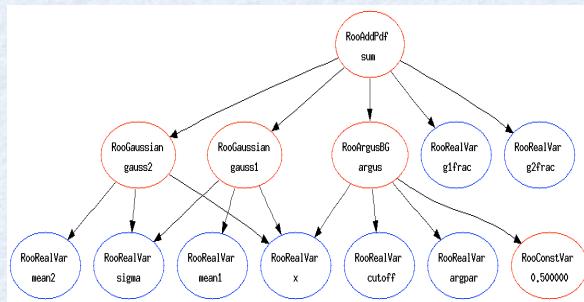


HistFactory package

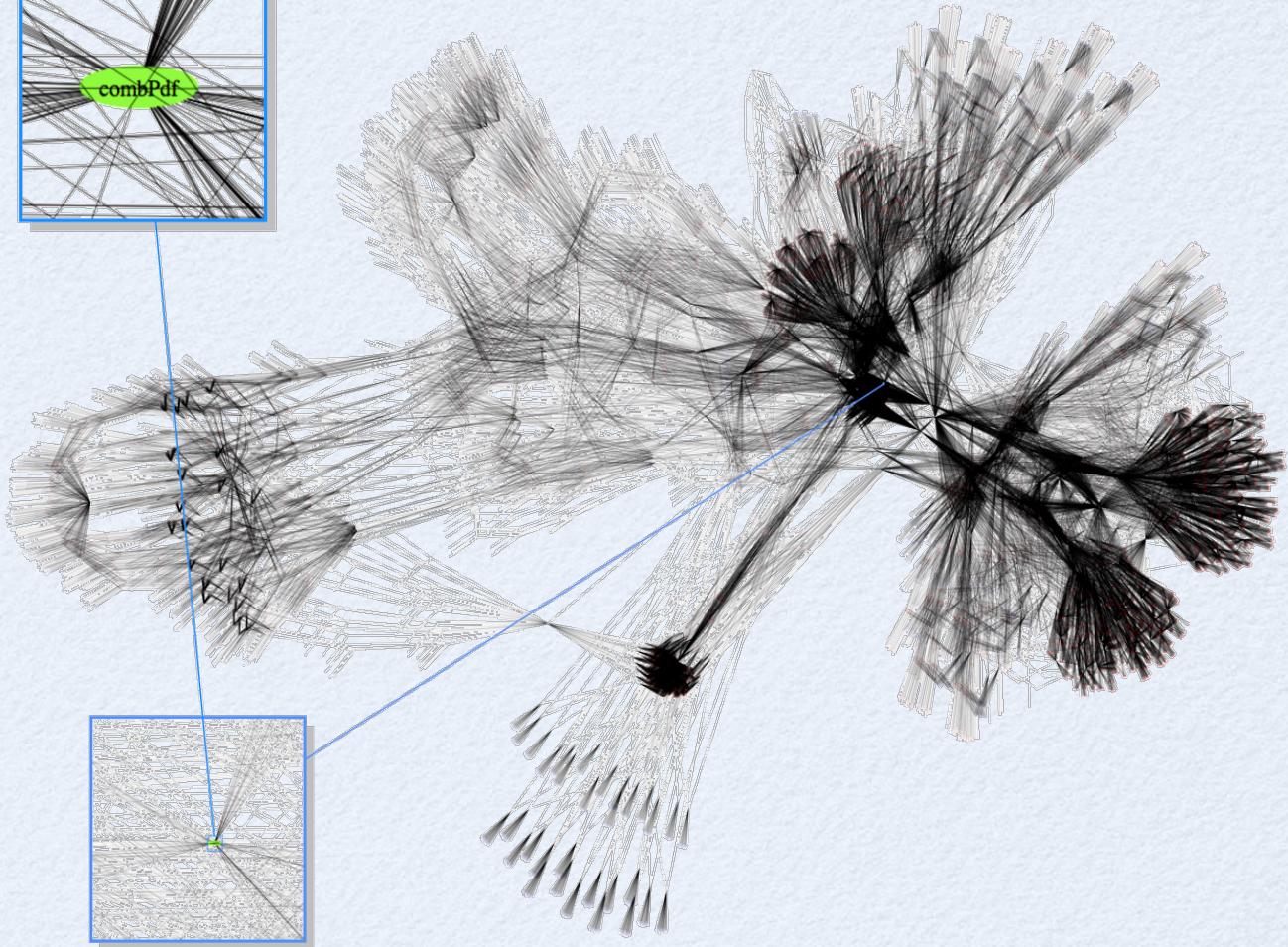
*Constructing models from
Monte Carlo templates*



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

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>

Time For Exercises !

Follow the Twiki page at

<https://twiki.cern.ch/twiki/bin/view/RooStats/RooStatsTutorialsMarch2015>