Status of RooFit/RooStats

W. Verkerke (NIKHEF)

RooStats Project – Overview

- Goals:
  - Standardize interface for major statistical procedures so that they can work on an arbitrary RooFit model & dataset and handle many parameters of interest and nuisance parameters.
  - Implement most accepted techniques from Frequentist, Bayesian, and Likelihood-based approaches
  - Provide utilities to perform combined measurements

- Design:
  - Essentially all methods start with the basic probability density function or likelihood function. *Building a good model is the hard part*. Want to re-use it for multiple methods → Use RooFit to construct models
  - Build series of tools that perform statistical procedures on RooFit models

RooStats Project – Structure

- **RooFit** (data modeling)
  - Data modeling language (pdfs and likelihoods). Scales to arbitrary complexity
  - Support for efficient integration, toy MC generation
  - Workspace
    - Persistent container for data models
    - Completely self-contained (including custom code)
    - Complete introspection and access to components
  - Workspace factory provides easy scripting language to populate the workspace

- **RooStats** (limits, interval calculators & utilities)
  - Profile Likelihood calculator
  - Neyman construction (FC)
  - Bayesian calculator (BAT & native MCMC)
  - Utilities (combinations, construct pdfs corresponding to standard number counting problems)

RooStats Project – Organization

- Joint ATLAS/CMS project
- Core developers
  - K. Cranmer (ATLAS)
  - Gregory Schott (CMS)
  - Wouter Verkerke (RooFit)
  - Lorenzo Moneta (ROOT)
- Open project, you are welcome to join
  - Max Baak, Mario Pelliccioni, Alfio Lazzaro contributing now
- Included since ROOT v5.22
  - Example macros in $ROOTSYS/tutorials/roostats
- Documentation
  - Code doc. via ROOT
  - Esers manual is in development

RooStats Project – Example

- Create a model - Example

$$Poisson(x \mid s \cdot r_s + b \cdot r_b) \cdot Gauss(r_s,1,0.05) \cdot Gauss(r_b,1,0.1)$$

Create workspace with above model (using factory)

```
RooWorkspace* w = new RooWorkspace("w");
w->factory("Poisson::P(obs[150,0,300],
                       sum::n(s[50,0,120]*ratioSigEff[1.,0,2.],
                              b[100,0,300]*ratioBkgEff[1.,0.,2.]))");
w->factory("PROD::PC(P, Gaussian::sigCon(ratioSigEff,1,0.05),
                        Gaussian::bkgCon(ratioBkgEff,1,0.1))");
```

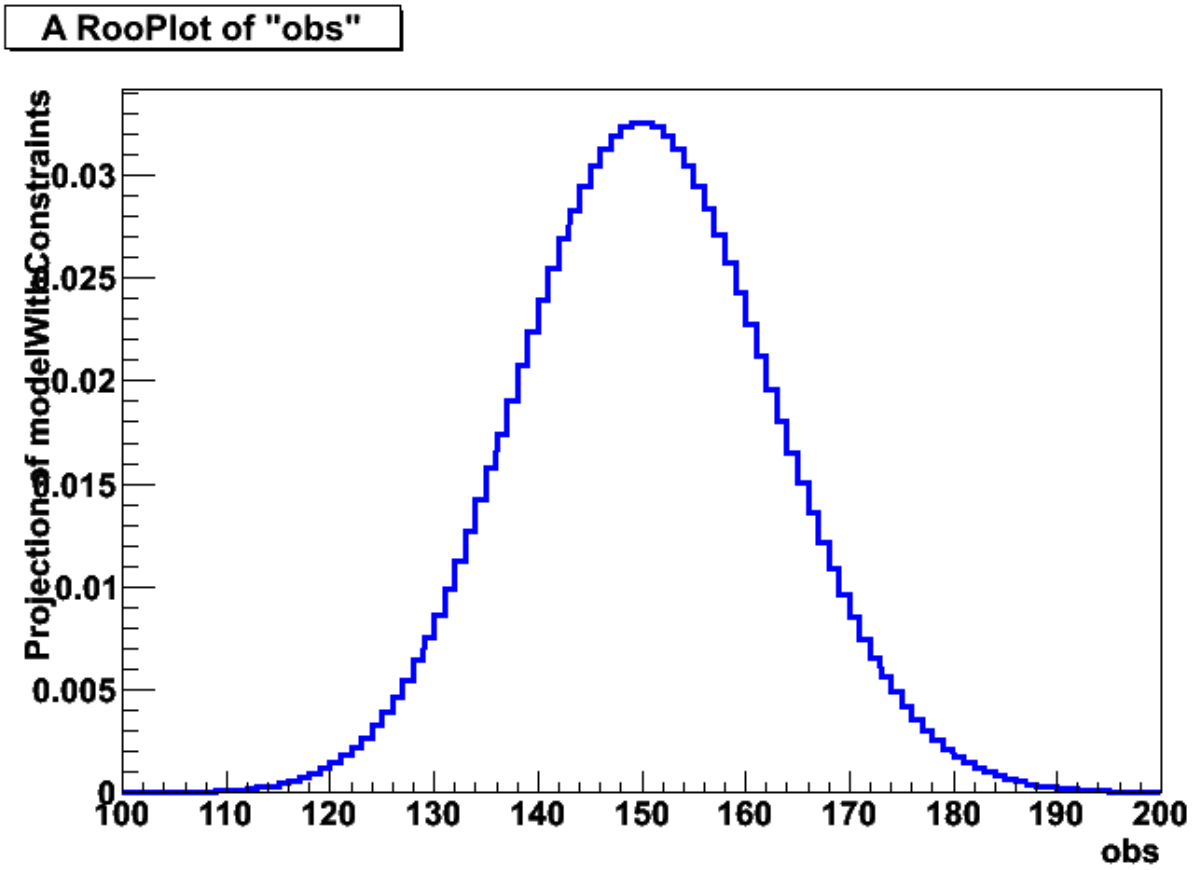
Contents of workspace from above operation

```
RooWorkspace(w) w contents

variables
---------
(b,obs,ratioBkgEff,ratioSigEff,s)

p.d.f.s
-------
RooProdPdf::PC[ P * sigCon * bkgCon ] = 0.0325554
  RooPoisson::P[ x=obs mean=n ] = 0.0325554
    RooAddition::n[ s * ratioSigEff + b * ratioBkgEff ] = 150
  RooGaussian::sigCon[ x=ratioSigEff mean=1 sigma=0.05 ] = 1
  RooGaussian::bkgCon[ x=ratioBkgEff mean=1 sigma=0.1 ] = 1
```

RooStats Project – Example

- Simple use of model

```
RooPlot* frame = w::obs.frame(100,200) ;
w::PC.plotOn(frame) ;
frame->Draw()
```



RooStats Project – Example

- Confidence intervals calculated with model
  - Profile likelihood

```
ProfileLikelihoodCalculator plc;
plc.SetPdf(w::PC);
plc.SetData(data); // contains [obs=160]
plc.SetParameters(w::s);
plc.SetTestSize(.1);
ConfInterval* lrint = plc.GetInterval(); // that was easy.
```

  - Feldman Cousins

```
FeldmanCousins fc;
fc.SetPdf(w::PC);
fc.SetData(data); fc.SetParameters(w::s);
fc.UseAdaptiveSampling(true);
fc.FluctuateNumDataEntries(false);
fc.SetNBins(100); // number of points to test per parameter
fc.SetTestSize(.1);
ConfInterval* fcint = fc.GetInterval(); // that was easy.
```

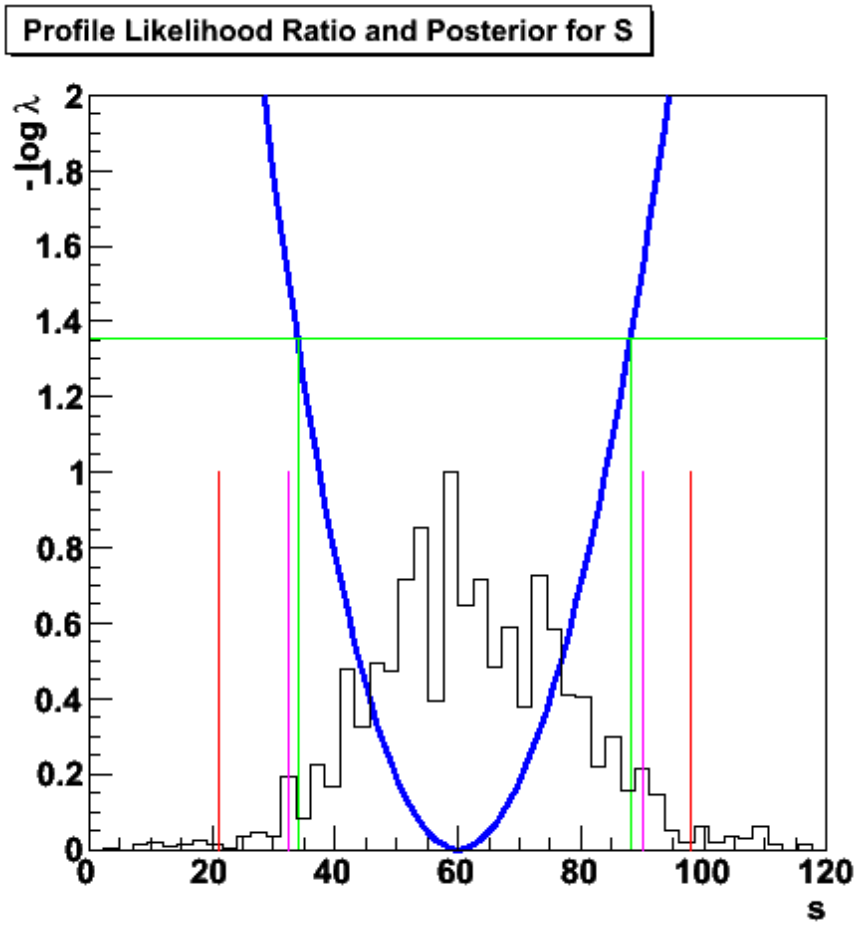
  - Bayesian (MCMC)

```
UniformProposal up;
MCMCCalculator mc;
mc.SetPdf(w::PC);
mc.SetData(data);  mc.SetParameters(s);
mc.SetProposalFunction(up);
mc.SetNumIters(100000); // steps in the chain
mc.SetTestSize(.1); // 90% CL
mc.SetNumBins(50); // used in posterior histogram
mc.SetNumBurnInSteps(40);
ConfInterval* mcmcint = mc.GetInterval();
```

RooStats Project – Example

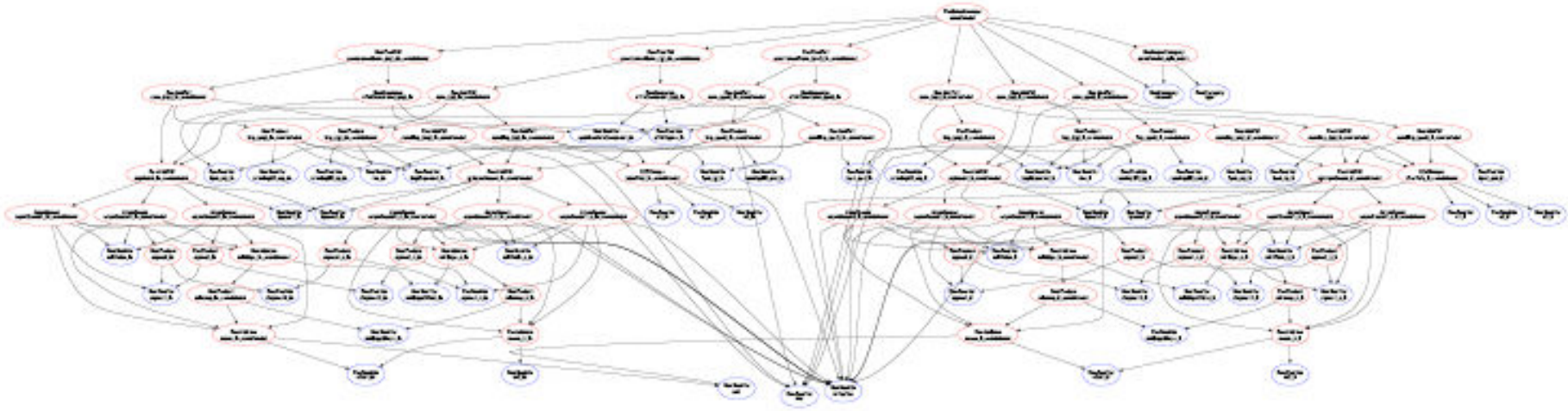
- Retrieving and visualizing output

```
double fcul = fcint->UpperLimit(w::s);
double fcll = fcint->LowerLimit(w::s);
```



RooStats Project – Example

- Some notes on example
  - Complete working example (with output visualization) shipped with ROOT distribution (*$ROOTSYS/tutorials/roofit/rs101_limitexample.C*)
  - **Interval calculators make no assumptions on internal structure of model**. Can feed model of arbitrary complexity to same calculator (computational limitations still apply!)



Recent developments in RooFit

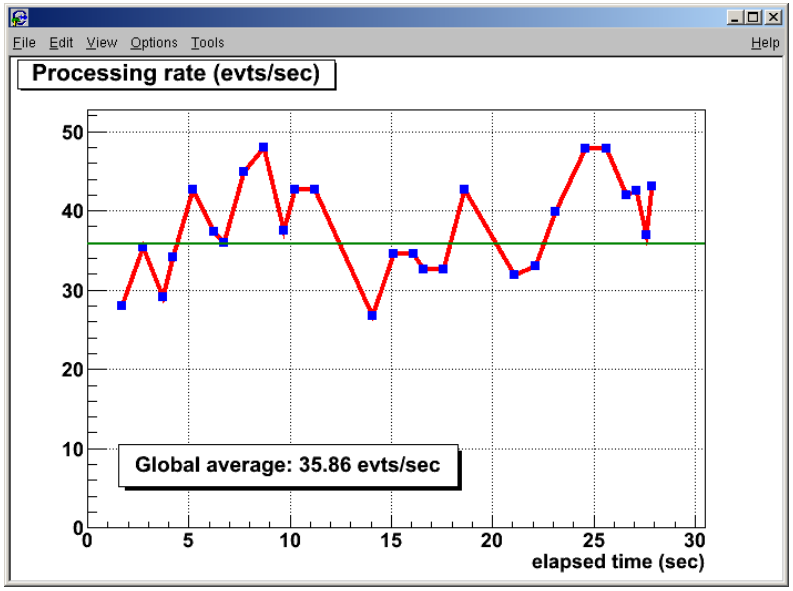
- Parallelization of toy MC generation
  - Some techniques require large amount of toy MC data to calculate result
  - Current support in RooFit has optimized handling of generation of multiple samples (initialization is only performed once), but only support for single-process generation. Present design also limits flexibility in choice of toy study
  - New framework developed in RooFit that supports multiple operation back-ends: **inline, PROOF, and batch**. Mode of operation completely(*) transparent to user (i.e. to RooStats tools)
  - Operation framework and implementation of toy study now in separate classes → Complete flexibility in definition of toy study. 'Standard' study implementation is provided with functionality similar to current framework.
  - New framework available in RooFit in next ROOT release. Adaptation of RooStats tools will probably take another cycle.

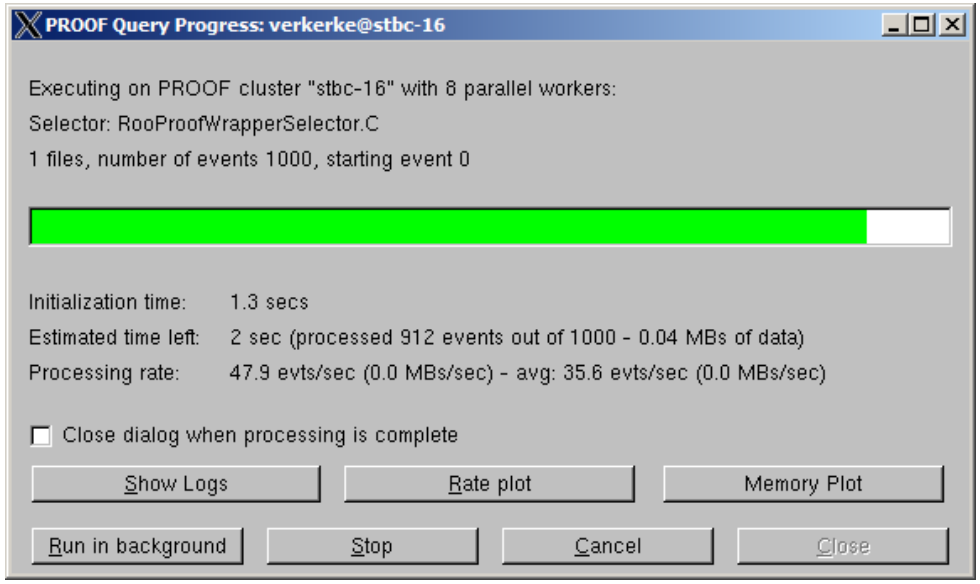
(*) Non-interactive nature of batch mode requires some user intervention

Demo of parallelization with PROOF-lite

- Example – Factor 8 speed up on a dual-quad core box.
  - Works with out-of-the box ROOT distribution
  - Also: Graceful early termination when users presses 'Stop'

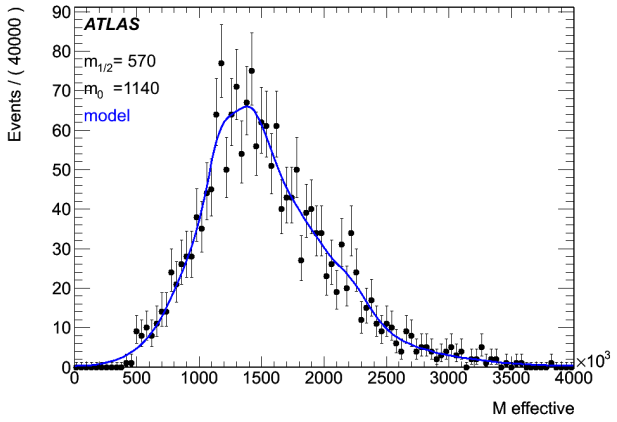
```
RooStudyManager mcs(*w,gfs) ;
mcs.run(1000) ; // inline running
mcs.runProof(1000,"") ; // empty string is PROOF-lite
mcs.prepareBatchInput("default",1000,kTRUE) ;
```

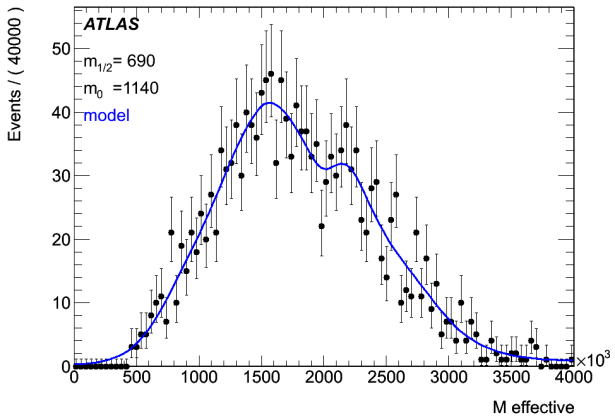


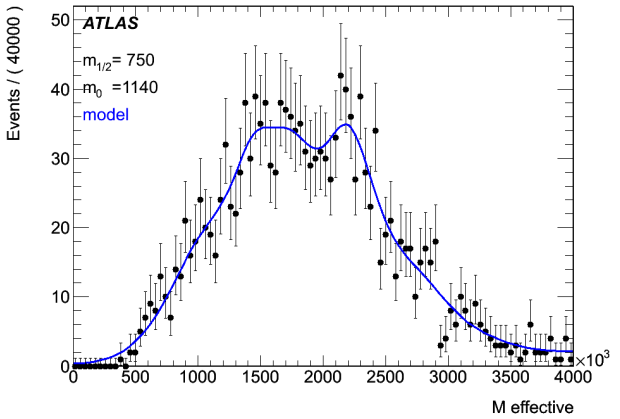


- Much larger gains can be made with 'real' PROOF farms

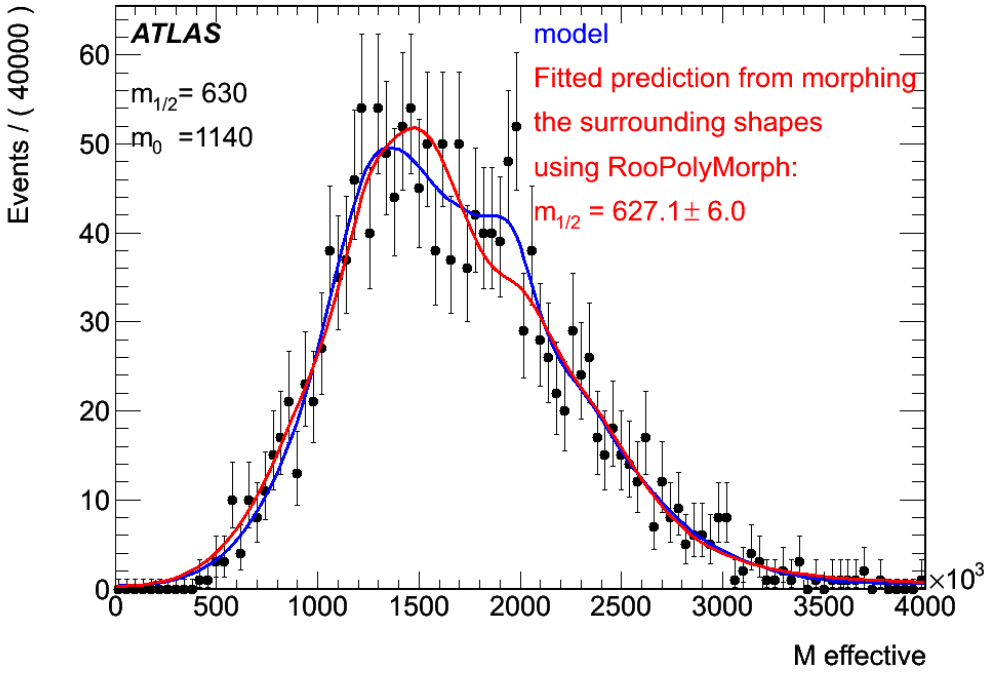
Recent developments – Morphing p.d.f.s (Max Baak & Stephan Gadatsch)





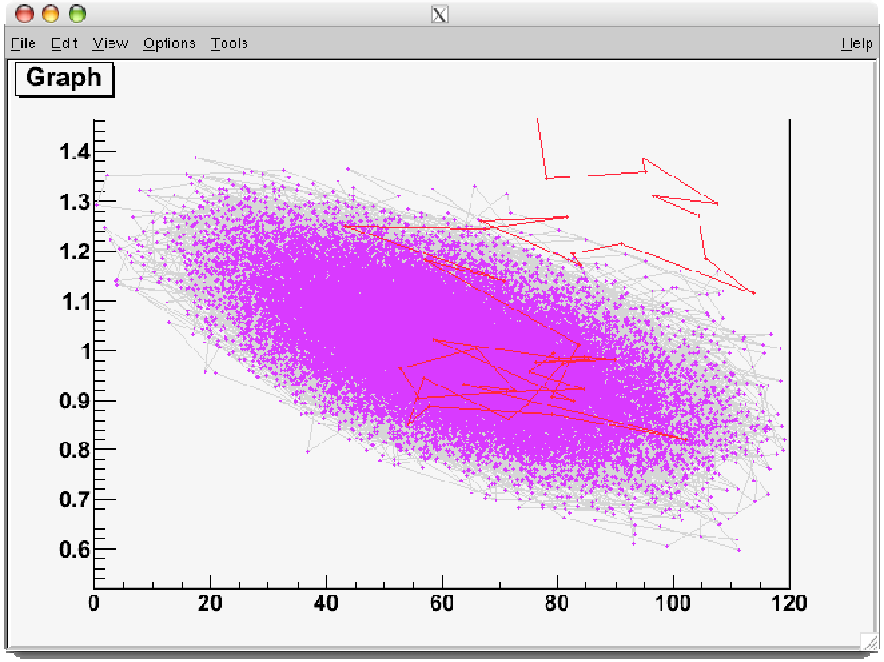


- Weighted template morphing
- Moments morph between provided input template p.d.f.s
- Morph = linear sum of shifted and scaled input templates
- Moments are non-linear functions of 1 or 2 fit-parameters
- Arbitrary number of observables per set of templates
- Computationally inexpensive algorithm
- Details in near-future presentation



Recent developments in RooStats

- New work on Markov Chain MC tools
  - Re-factorized the design so Metropolis-Hastings and MarkovChain are new classes used by MCMCCalculator and MCMCInterval
  - Now have a utility called ProposalHelper which can aids in creating a proposal function specific to your problem.
    - Can use the multivariate Gaussian from Hesse as the proposal function
    - Supports mixtures of arbitrary PDFs
    - Supports the 'bank of clues' algorithm by Lester and Allanach
  - New plotting classes to visualize the posterior, the chain itself



Recent developments – Combination tools

- Workspace concept also center of tools being developed to make combinations
  - Workspace can be persisted to file

ROOT Session #1

```
w->factory("Poisson::P(obs[150,0,300],
                       sum(s[50,0,120]*ratioSigEff[1.,0,2.],
                          b[100,0,300]*ratioBkgEff[1.,0.,2.]))");
w->factory("PROD::PC(P, Gaussian::sigCon(ratioSigEff,1,0.05),
                        Gaussian::bkgCon(ratioBkgEff,1,0.1))");
w->writeToFile("poissonExample.root") ;
```

ROOT Session #2

```
TFile f("poissonExample.root") ;
RooWorkspace* w = gDirectory->Get("w") ;
```

  - Full introspection abilities allow workspace made by you to be used by someone else without problems.
  - Introspection also allows automatic adjustments to be made (e.g. renaming of parameters, datasets...) and combinations to be built in a generic way

Recent developments – Combination tools

ROOT Session #1
```
w->writeToFile("channelA.root") ;
```

ROOT Session #2
```
w->writeToFile("channelB.root") ;
```

ROOT Session #3
```
w->writeToFile("channelC.root") ;
```

ROOT Session #4
```
RooWorkspace w("w","joint workspace") ;

// Import top-level pdfs and all their components, variables
w.import("channelA.root:w:pdfA",RenameAllVariablesExcept("A","mhiggs")) ;
w.import("channelB.root:w:pdfB",RenameVariable("mH","mhiggs")) ;
w.import("channelC.root:w:pdfC") ;

// Construct joint pdf
w.factory("SIMUL::joint(chan[A,B,C],A=pdfA,B=pdfB,C=pdfC)") ;
```

- Can also easily aggregate models and data from multiple workspaces into a single joint one
  - Tools exist to aid practical aspects of combination process (e.g. renaming of parameters upon import)
  - Tools to perform similar joining operation for data available in next ROOT release
  - **See Kyles talk**

Other new developments

- Preparations towards uniform ModelConfig interface for tools.
  - New class ModelConfig contains all 'problem definition' information, i.e. pdf, data, definition of parameters, parameters of interest, etc...
  - Simplifies interface of tools, promotes interoperability

```
ProfileLikelihoodCalculator plc(myModelConfig);
plc.SetTestSize(.1);
ConfInterval* lrint = plc.GetInterval();
```

- New RooStats tutorial macros prepared by Gregory Schott (CMS)
  - Will be bundled in forthcoming ROOT release (now have 32 macros)

Other new developments

- RooStats validation studies w.r.t. Cousins,Linnemans,Tucker paper
  - See presentation by Renaud
- Deploying new alternative RooStats release strategy through LCG SW distributions
  - Can now ship ROOT 5.22 (as used by ATLAS,CMS for production) with updated RooFit/RooStats code
  - Automated procedure
- Also monthly new RooStats releases through ROOT distributions
  - New ROOT release 5.25 due in ~2 weeks