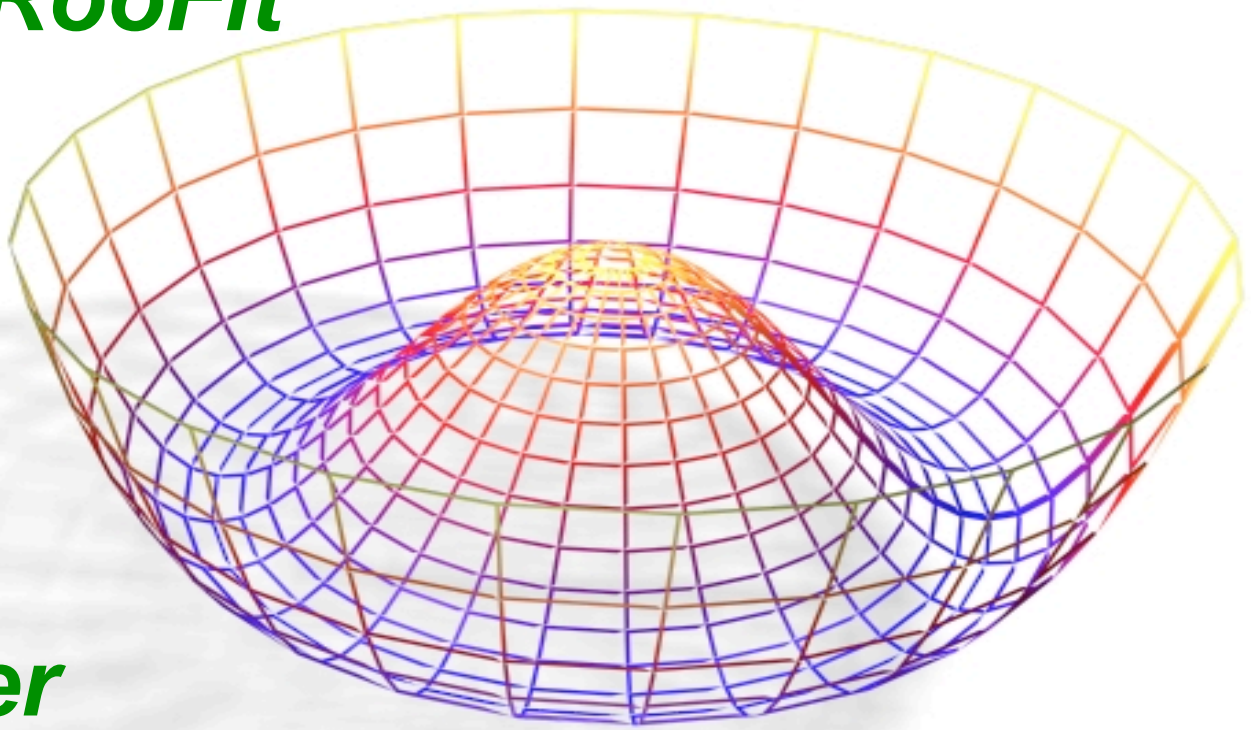
BROOKHAVEN NATIONAL LABORATORY

RooStats:
A high-level statistical framework based on RooFit



Kyle Cranmer
(BNL)

Context for discussion

BROOKHAVEN NATIONAL LABORATORY

We are a close to having real data, real analyses, and hopefully real excitement!

- if we want to converge on common tools, it will be much easier to do that now

Some discussions have begun between ATLAS and CMS related to the creation of a joint statistics committee

- to discuss conventions, tools, combinations, etc...

Next PhyStat meeting June 27-29

- will focus on LHC physics & statistical problems

After Oxford PhyStat meeting, René asked me to think about a statistical framework for ROOT

- beyond TLimit & TRolke, and interfaced with RooFit

Requirements for the Framework

BROOKHAVEN NATIONAL LABORATORY

In addition to providing tools for simple calculations, the framework should

- be able to combine the results of multiple measurements,
- be able to incorporate systematic uncertainty,
- facilitate the technical aspects of sharing code

Both LEP and Tevatron experiments have created tools that combine multiple channels and include systematic uncertainties, but

- the tools generally implement a specific technique,
- and combinations require significant manual intervention

Additional Requirements

BROOKHAVEN NATIONAL LABORATORY

There are few major classes of statistical techniques:

- Likelihood:
  - All inference from likelihood curves
- Bayesian:
  - Use prior on parameter to compute P(theory|data)
- Frequentist:
  - Restricted to statements of P(data|theory)

Even within one of these classes, there are several ways to approach the same problem.

- The framework should support each of these types of techniques, and provide common abstractions

Comparison of Methods

BROOKHAVEN NATIONAL LABORATORY

Most significant result from my PhyStat05 work was this comparison of coverage for several common methods which can incorporate systematic errors

- Clearly useful to make comparisons using same data and same assumptions
  - a nice feature of TMVA!
- If Atlas used λ_P method and CMS used Z_N, then they could "discover" with 56% less data!
  - a bad situation for ATLAS
  - a bad situation for HEP

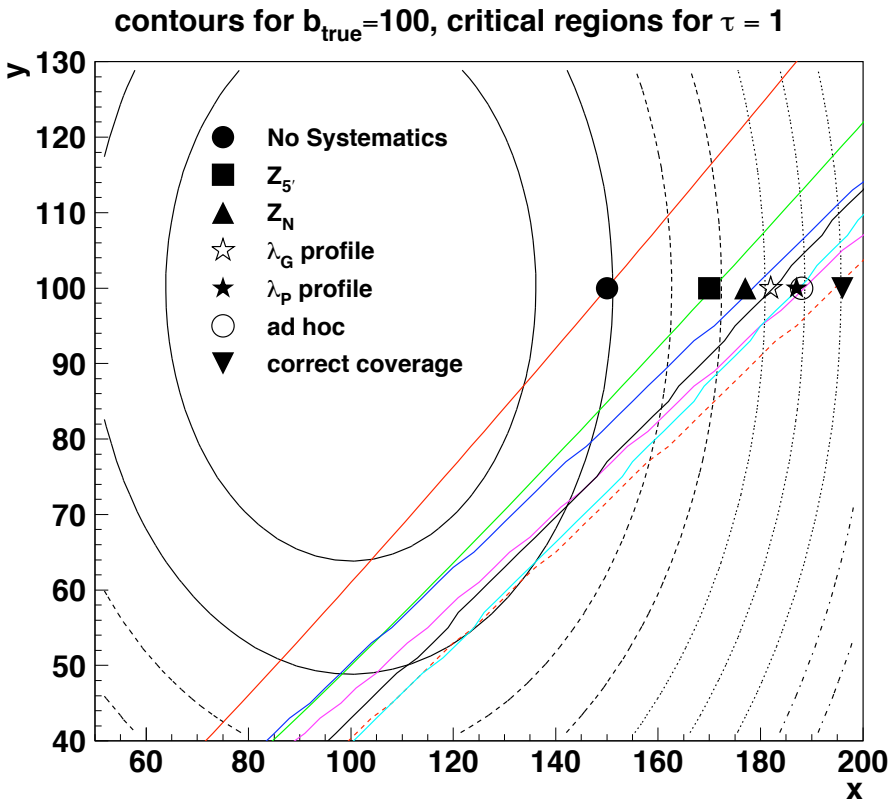


Figure 7. A comparison of the various methods critical boundary $x_{crit}(y)$ (see text). The concentric ovals represent contours of L_G from Eq. 15.

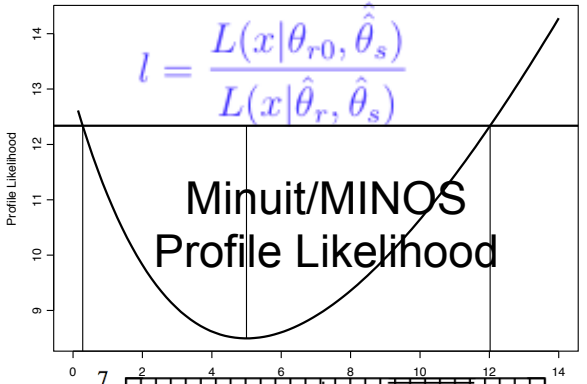
$$L_P(x, y|\mu, b) = Pois(x|\mu + b) \cdot Pois(y|\tau b).$$

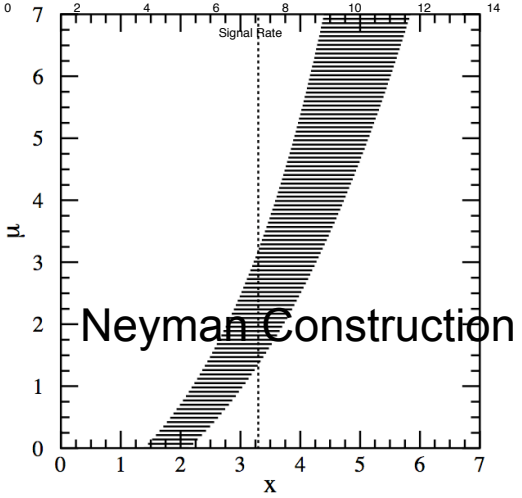
Many Methods, Many Similarities

BROOKHAVEN NATIONAL LABORATORY

Essentially all methods start with the basic probability density function or likelihood function $L(\vec{x}|\theta_r,\theta_s)$

- Building a good model is the hard part!
- want to re-use it for multiple methods
- want to interface to common tools





$$L(b|Y) = \frac{L(Y|b)\ L(b)}{L(Y)}$$

Bayes Theorem

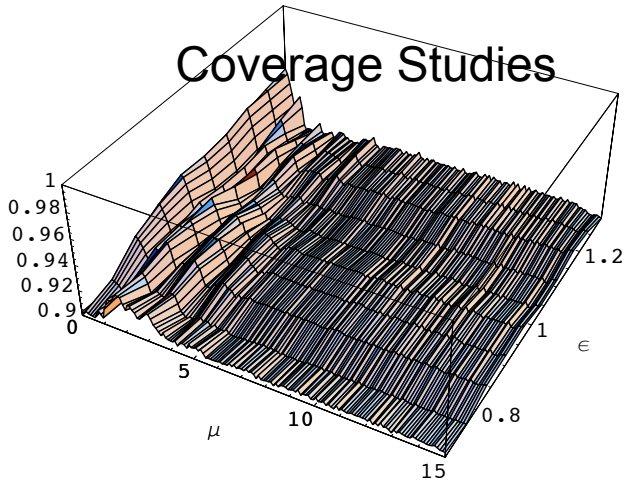
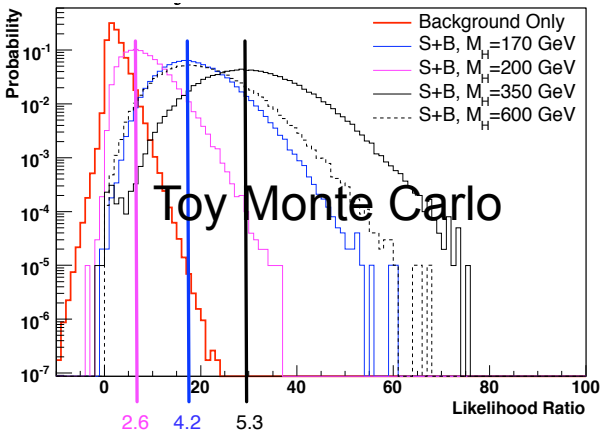


Fig. 2. Coverage plot for Unified limits, Gaussian uncertainty, $b = 3, \sigma = 0.1$.



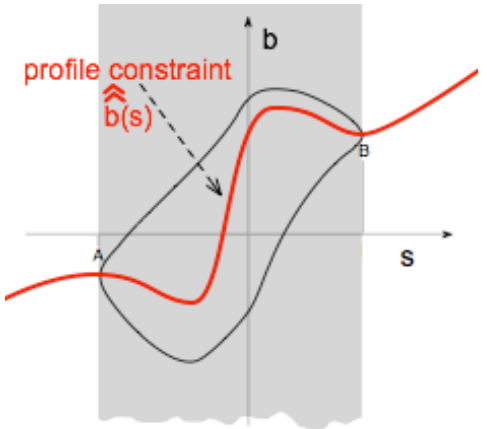


Figure 7.2: MINOS error confidence region for parameter 1

To RooFit or Not To RooFit

BROOKHAVEN NATIONAL LABORATORY

The immediate question is whether this statistical framework should be:

- parallel to RooFit (with some sort of interface?)
- on top of or a part of RooFit

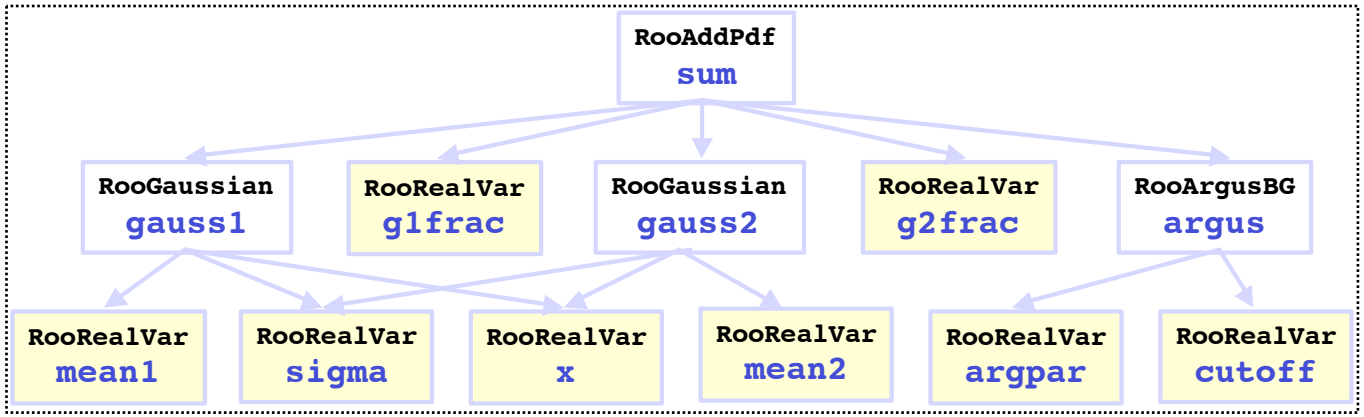
Informal survey showed significant support for RooFit, but some hesitation

- worries about RooFit doing too much for a user
  - want to maintain ability to tweak the likelihood and "dig in"
- some worries about dependencies from a software point of view
  - maybe a core part separated from graphics, etc...

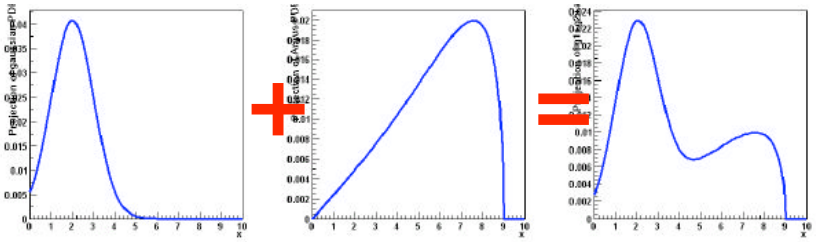
These aspects are all part of the discussion, but

- hard to deny that RooFit has a large "market share"
- RooAbsPdf has a lot of functionality that we need

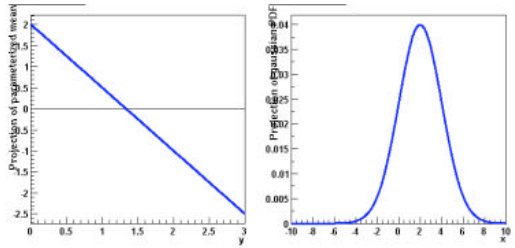
RooAbsPdf: A starting point

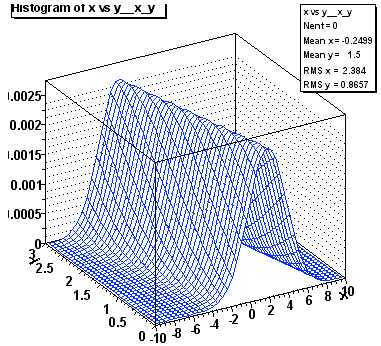


- Addition



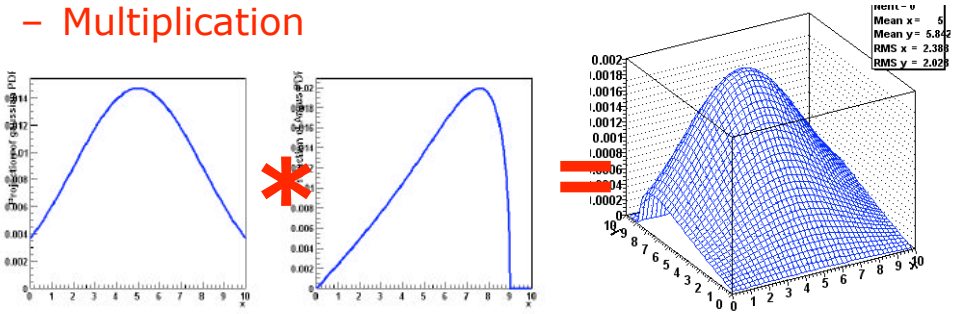
- Composition ('plug & play')





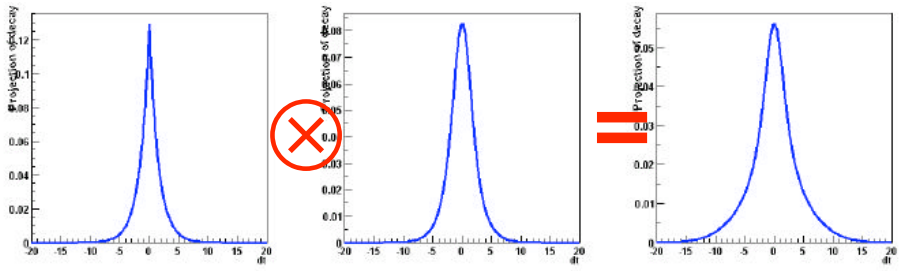
Possible in *any* PDF
No explicit support in PDF code needed

- Multiplication



Wouter Verkerke,

- Convolution



Wouter Verkerke, UCSB

The Prototype Problem Revisited

BROOKHAVEN NATIONAL LABORATORY

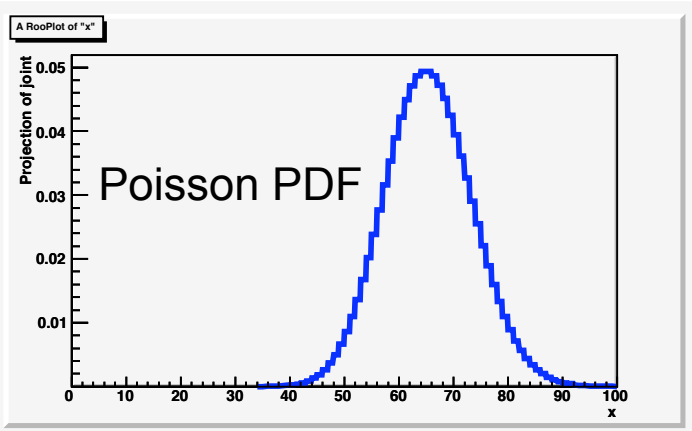
Consider this prototype problem for new physics searches:

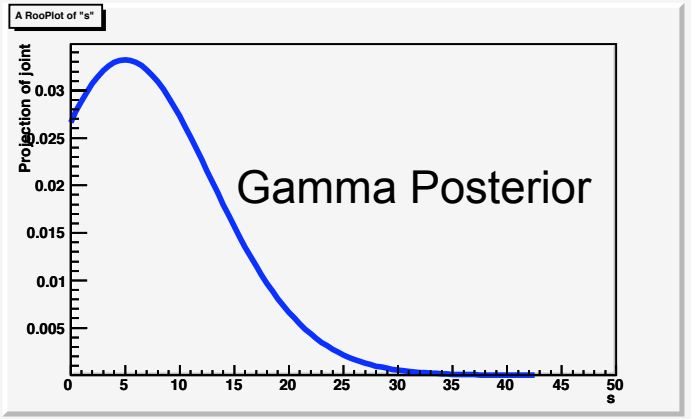
$$L_P(x, y|\mu, b) = Pois(x|\mu + b) \cdot Pois(y|\tau b).$$

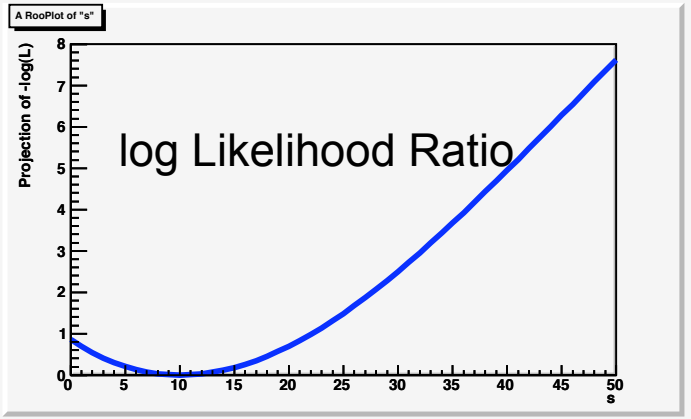
Easy to code up in RooFit:

```
RooRealVar   s("s","s",_s, 0., 100.);
RooRealVar   b("b","b",_b,  0., 200.);
RooRealVar  tau("tau","tau",_tau,0,2);
tau.setConstant(kTRUE);
RooFormulaVar  splusb("splusb","s+b",RooArgSet(s,b));
RooProduct   bTau("bTau","b*tau",RooArgSet(b, tau));
RooRealVar   x("x","x",_s+_b, 0., 200.);
RooRealVar   y("y","y",_b*_tau,  0., 200.);

RooPoisson sigRegion("sigRegion","sigRegion",x,splusb);
RooPoisson sideband("sideband","sideband",y,bTau);

RooProdPdf joint("joint","joint", RooArgSet(sigRegion,sideband) );
```

Trivial to obtain plots in three different formalisms:







Higher-Level Abstractions

BROOKHAVEN NATIONAL LABORATORY

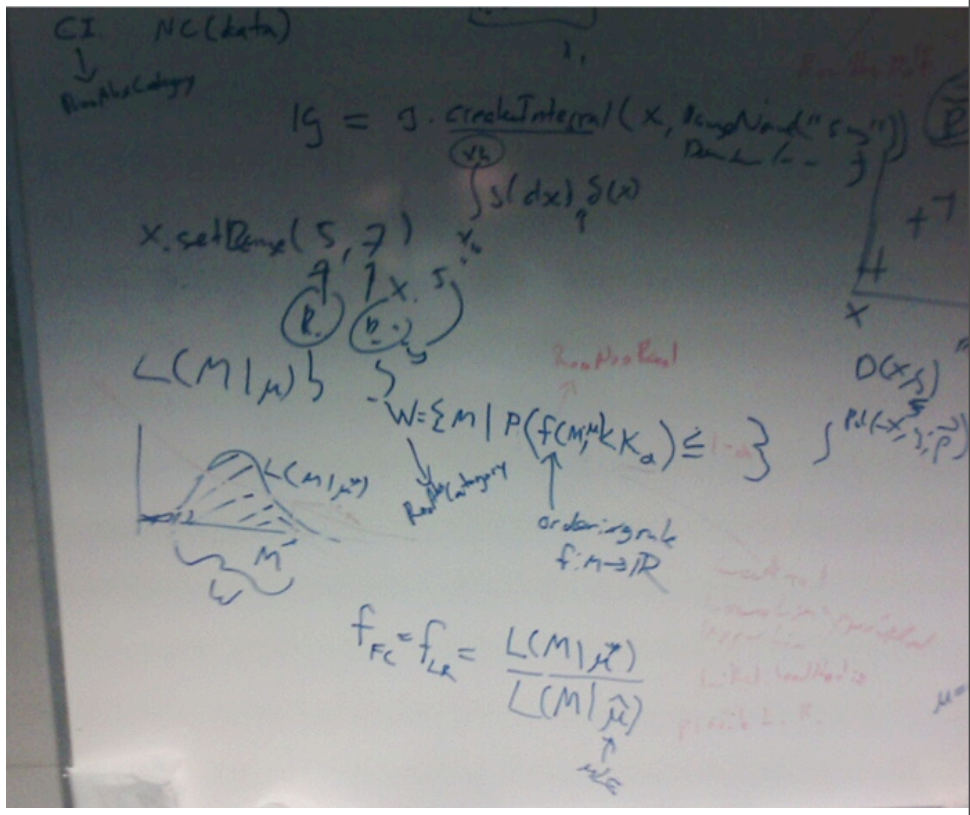
Wouter and I met a couple months ago to discuss how to implement a few statistical concepts on top of RooFit

- want class structure to maps onto statistical concepts
- successfully worked out a few of the methods

The first examples were

- Bayesian Posterior
- Profile likelihood ratio
- Acceptance Region
- Ordering Rule
- Neyman Construction
- Confidence Interval

Many concepts already have an appropriate class in RooFit

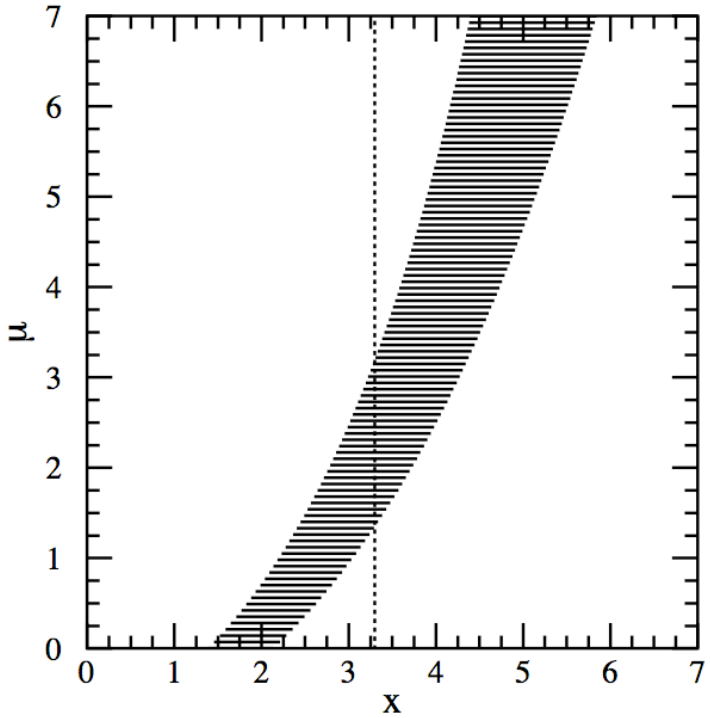


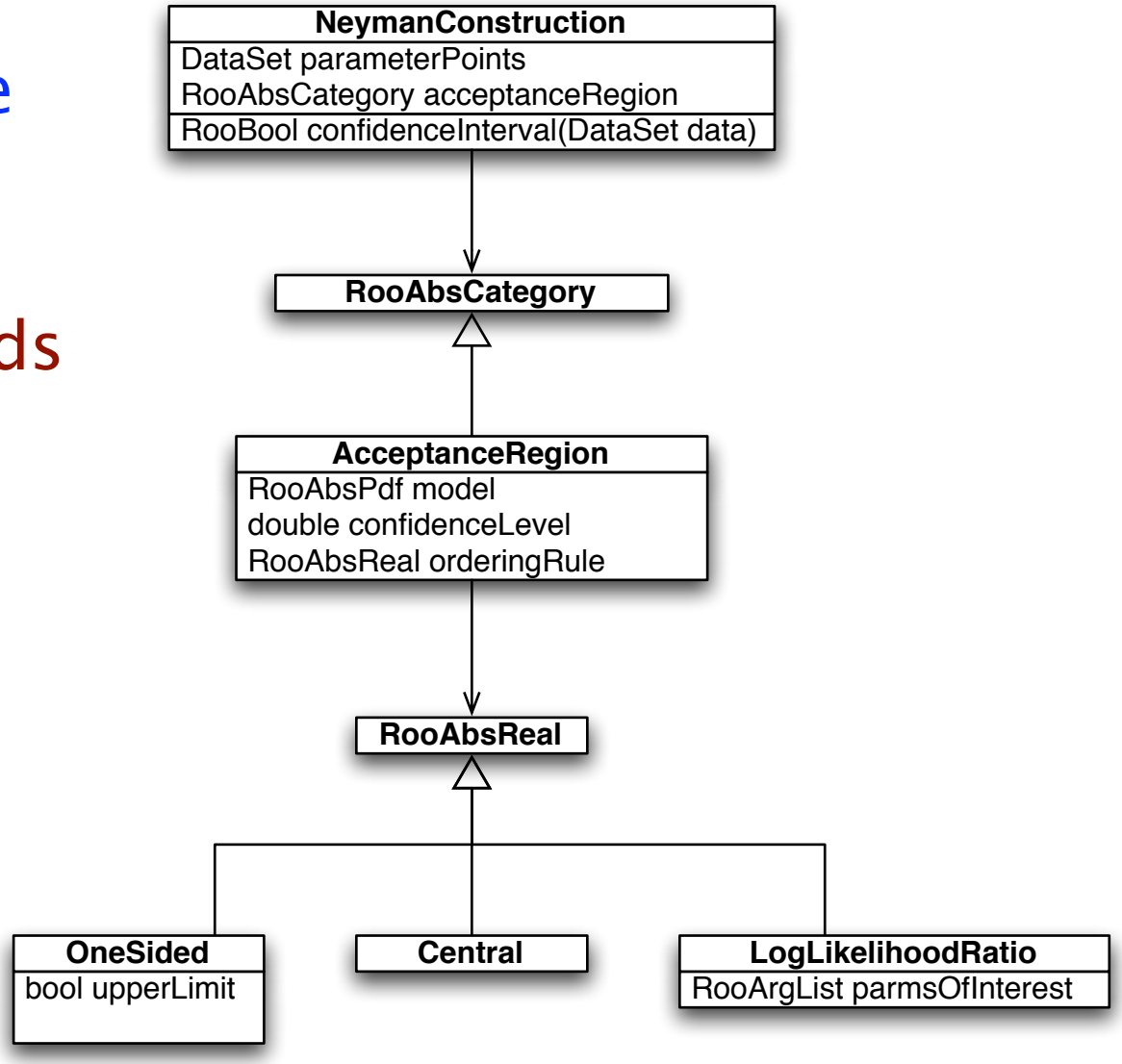
An Example: Neyman Construction

BROOKHAVEN NATIONAL LABORATORY

Neyman Construction is simply a set of acceptance regions corresponding to parameter points

- Acceptance region needs ordering rule + pdf + confidence level



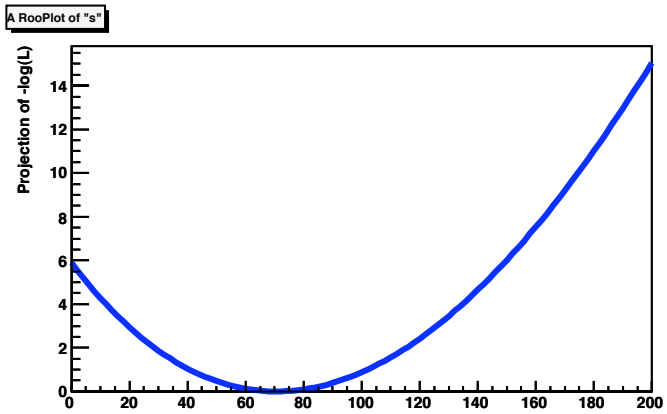


Note: Distinction between observables and parameters

From NLL to ProfileNLL

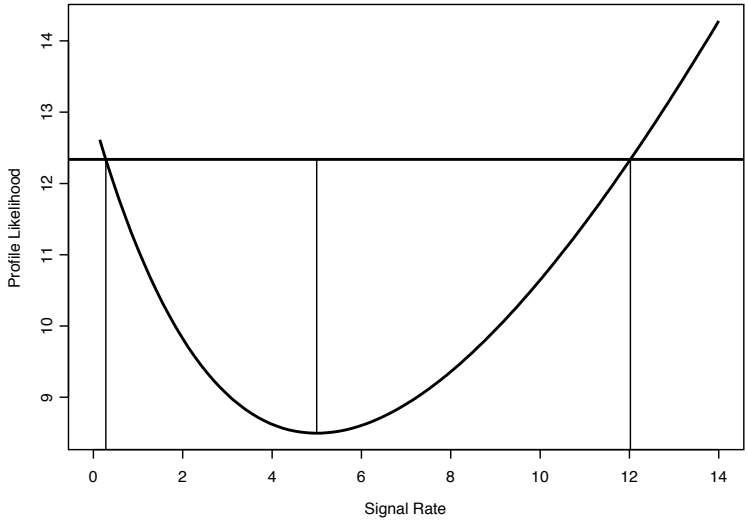
RooFit provides NLLVar, which interfaces to Minuit and provides a likelihood ratio as a function of some parameter.

- hold rest of parameters fixed



Very user friendly interface, but NLL is not so useful.

- rather have profile likelihood ratio
- needed for limits & discovery
- needed for Feldman-Cousins
- needed for profile construction
- next step on our to-do list



Dealing with Complex Models

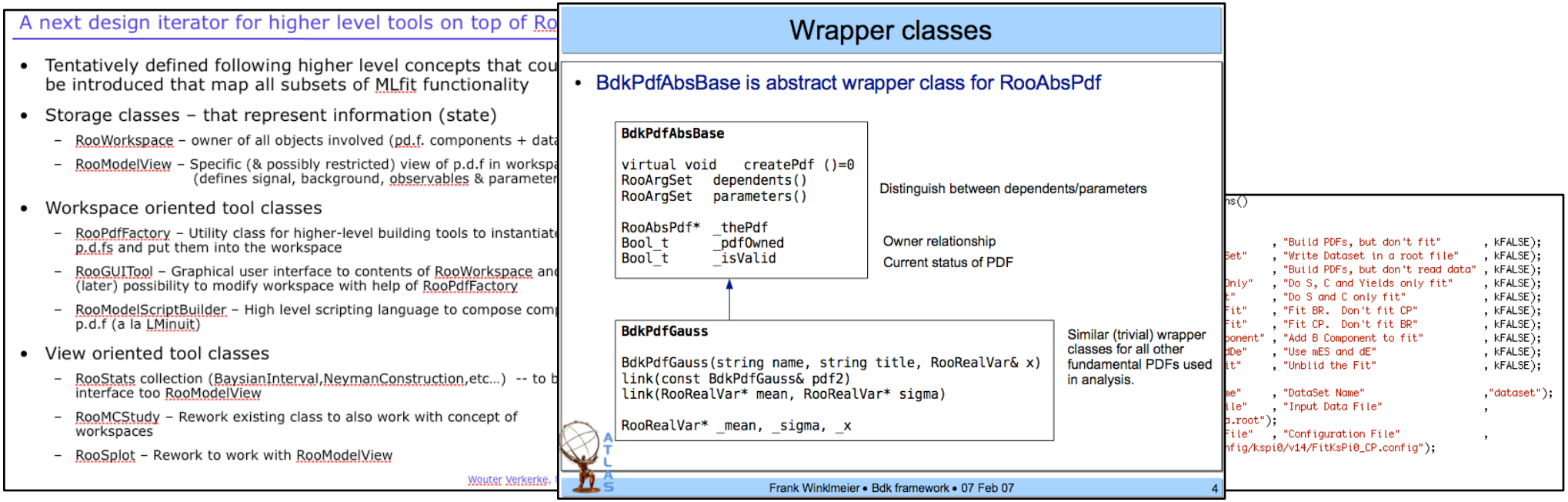
BROOKHAVEN NATIONAL LABORATORY

BaBar was a major user of RooFit, but many groups developed a layer on top of RooFit to handle complex models

- Amir Farbin developed MLFit
- Frank Winklemeier developed "Bdk framework"
- both dealt with 100's or 1000's of parameters

Discussions led to a new core RooFit concepts

- Model & Workspace + associated tools



A next design iterator for higher level tools on top of Ro

- Tentatively defined following higher level concepts that cou be introduced that map all subsets of MLfit functionality
- Storage classes – that represent information (state)
  - RooWorkspace – owner of all objects involved (pd.f. components + data
  - RooModelView – Specific (& possibly restricted) view of p.d.f in workspa (defines signal, background, observables & parameter
- Workspace oriented tool classes
  - RooPdfFactory – Utility class for higher-level building tools to instantiate p.d.fs and put them into the workspace
  - RooGUITool – Graphical user interface to contents of RooWorkspace and (later) possibility to modify workspace with help of RooPdfFactory
  - RooModelScriptBuilder – High level scripting language to compose comp p.d.f (a la LMinuit)
- View oriented tool classes
  - RooStats collection (BaysianInterval,NeymanConstruction,etc...) -- to b interface too RooModelView
  - RooMCStudy – Rework existing class to also work with concept of workspaces
  - RooSplot – Rework to work with RooModelView

Wouter Verkerke,

Wrapper classes

- BdkPdfAbsBase is abstract wrapper class for RooAbsPdf

BdkPdfAbsBase

```
virtual void    createPdf ()=0
RooArgSet   dependents()
RooArgSet   parameters()

RooAbsPdf*  _thePdf
Bool_t      _pdfOwned
Bool_t      _isValid
```

Distinguish between dependents/parameters

Owner relationship

Current status of PDF

BdkPdfGauss

```
BdkPdfGauss(string name, string title, RooRealVar& x)
link(const BdkPdfGauss& pdf2)
link(RooRealVar* mean, RooRealVar* sigma)

RooRealVar* _mean, _sigma, _x
```

Similar (trivial) wrapper classes for all other fundamental PDFs used in analysis.

Frank Winklmeier • Bdk framework • 07 Feb 07 4

```
ns()

                , "Build PDFs, but don't fit"       , kFALSE);
Set"            , "Write Dataset in a root file"    , kFALSE);
                , "Build PDFs, but don't read data" , kFALSE);
Only"           , "Do S, C and Yields only fit"     , kFALSE);
t"              , "Do S and C only fit"             , kFALSE);
Fit"            , "Fit BR.  Don't fit CP"           , kFALSE);
Fit"            , "Fit CP.  Don't fit BR"           , kFALSE);
ponent"         , "Add B Component to fit"          , kFALSE);
dDe"            , "Use mES and dE"                  , kFALSE);
it"             , "Unblid the Fit"                  , kFALSE);

me"             , "DataSet Name"                    ,"dataset");
ile"            , "Input Data File"                 ,
a.root");
File"           , "Configuration File"              ,
nfig/kspi0/v14/FitKsPi0_CP.config");
```

Model & Workspace

BROOKHAVEN NATIONAL LABORATORY

Wouter introduced these concepts in the previous talk

The RooAbsPdf remains unchanged, does not make a distinction between observable and parameter

A "Model" will make that distinction, and keep track of different categories/species/hypotheses/channels (eg. signal & background)

A "Workspace" will be a higher-level object that owns and manages all the parts of the Model

Add ability to persist the Model & Workspace to ease sharing and construction of more complex models

High-Level Scripting

BROOKHAVEN NATIONAL LABORATORY

Experience from BaBar indicated a need for some high-level scripting to

- produce complex models
- configure them

Bill Quayle's StatTools also has some XML-based layer to do this

In our approach, low-level issues dealt with by the Model & Workspace classes. We are considering both:

- dedicated classes+configuration "language"
- or do configuration with Python & PyRoot

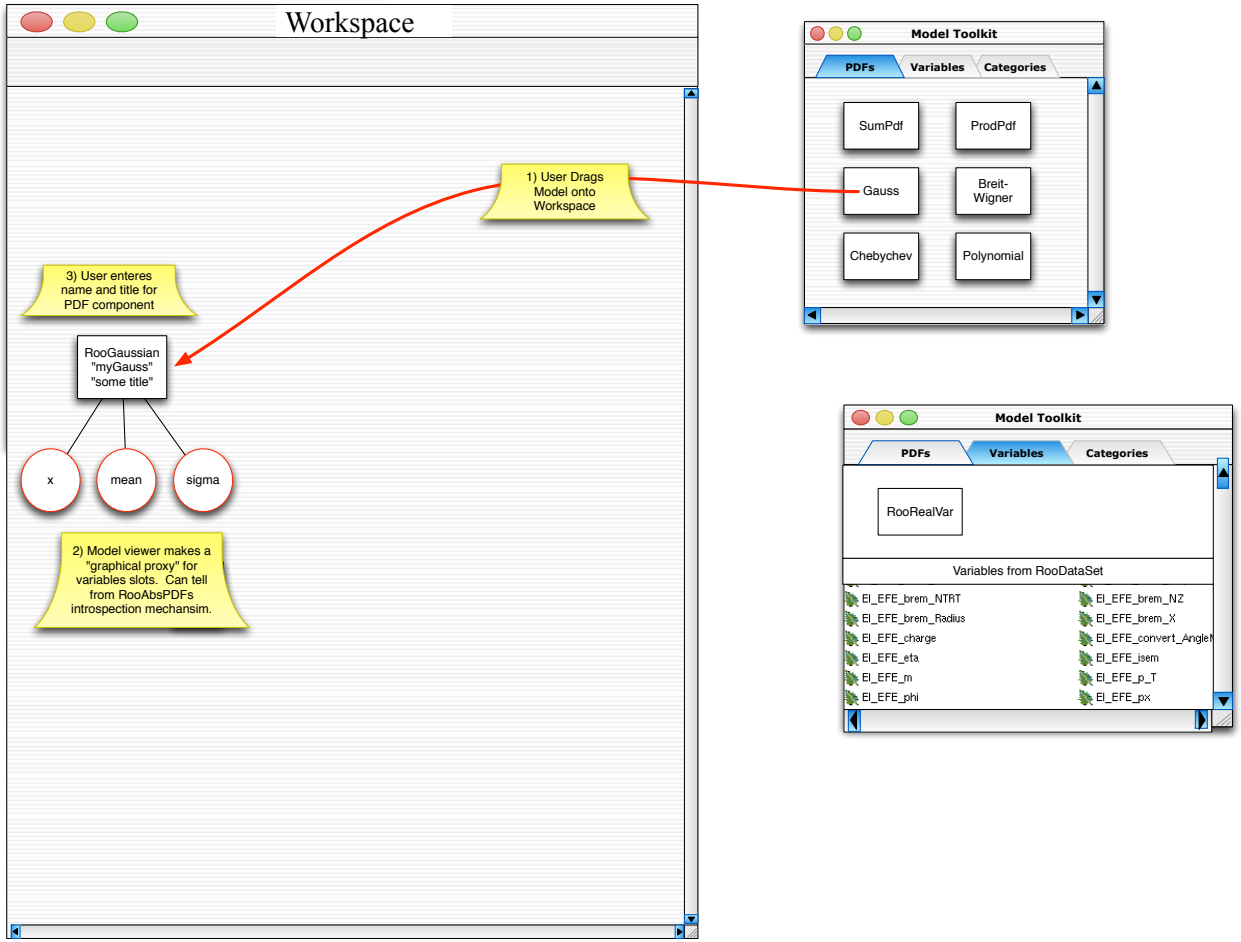
Experience from BaBar favors something like PyRoot

A Model Building GUI

BROOKHAVEN NATIONAL LABORATORY

We had some discussion about providing a GUI for constructing large PDFs

- first step was to think of how to represent the PDF
- then think about how to interact with it



A Model Building GUI

BROOKHAVEN NATIONAL LABORATORY

The representation is naturally a directed graph

- but not necessarily a Tree
- RooAbsPdf already knows how to build the graph

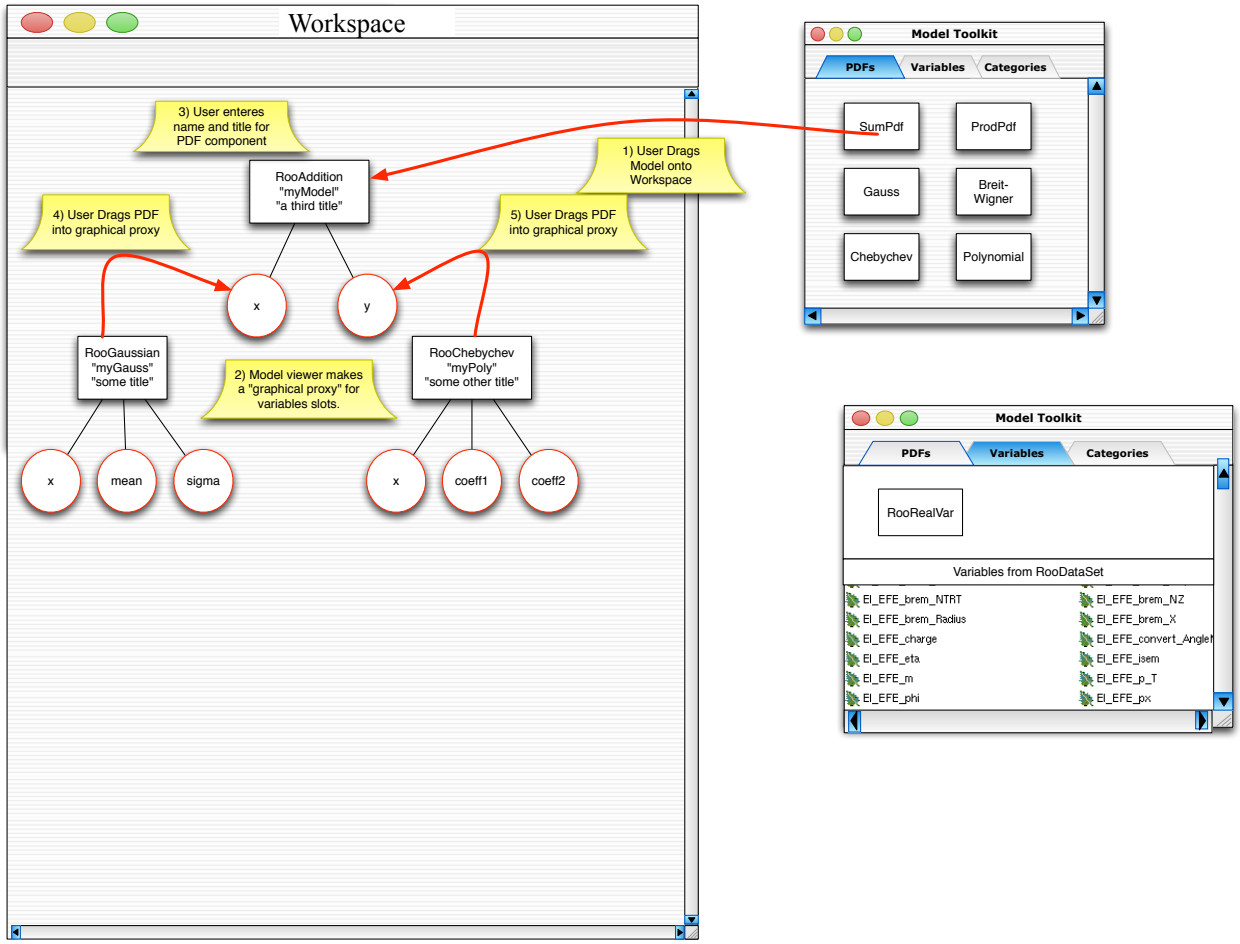
What is the structure of my composite model?

In addition to manipulation of the parameters one may also wonder what the structure of a given model is. For an easy visual inspection of the tree structure of a composite object use the method `printCompactTree()`:

```
model.printCompactTree() ;
```

The output will look like this:

```
0x9a76d58 RooAddPdf::model (model)  [Auto]
  0x9a6e698 RooGaussian::sig (signal p.d.f.)  [Auto]
    0x9a190a8 RooRealVar::x (x)
    0x9a20ca0 RooRealVar::mean (mean)
    0x9a3ce10 RooRealVar::sigma (sigma)
  0x9a713c8 RooRealVar::nsig (signal fraction)
  0x9a26cb0 RooChebychev::bkg (background p.d.f.)  [Auto]
    0x9a190a8 RooRealVar::x (x)
    0x9a1c538 RooRealVar::c0 (coefficient #0)
    0x9a774d8 RooRealVar::c1 (coefficient #1)
    0x9a3b670 RooRealVar::c2 (coefficient #2)
  0x9a66c00 RooRealVar::nbkg (background fraction)
```



A Model Building GUI

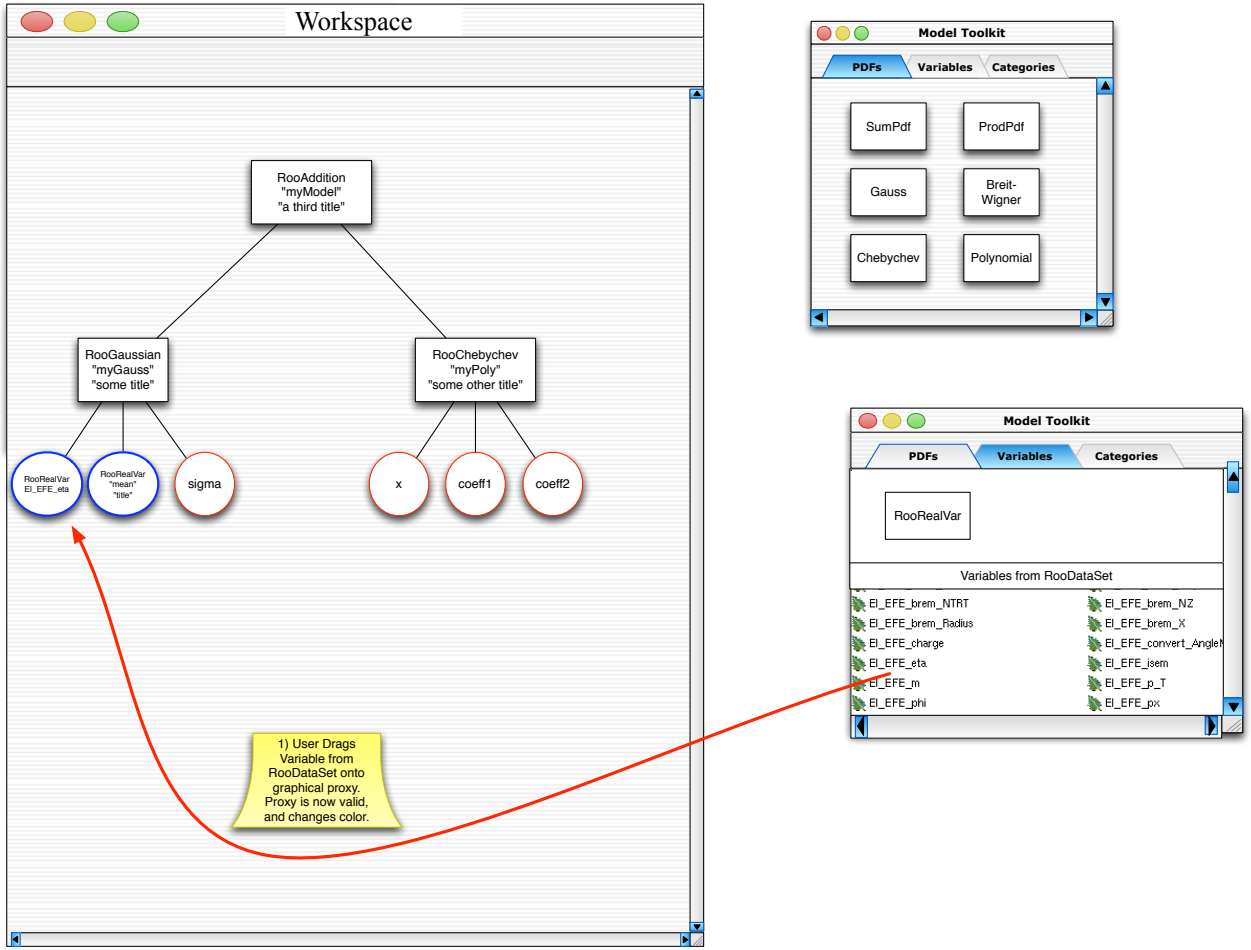
BROOKHAVEN NATIONAL LABORATORY

Want easy ability to bind variables in a TTree (or RooDataSet) to the PDF

Finally we need a way to capture the result

One approach is to auto-generate code for this PDF

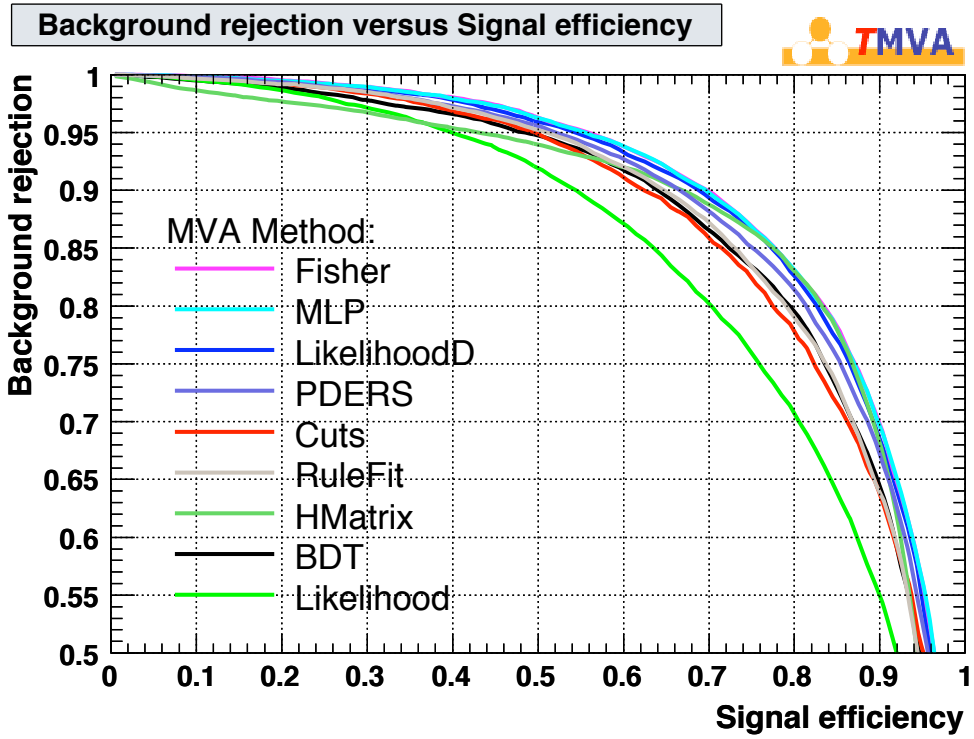
Another is to return a Workspace object that represents it



Comparison Tools

BROOKHAVEN NATIONAL LABORATORY

One of the nicest features of TMVA is that it allows one to easily compare several methods using the same data



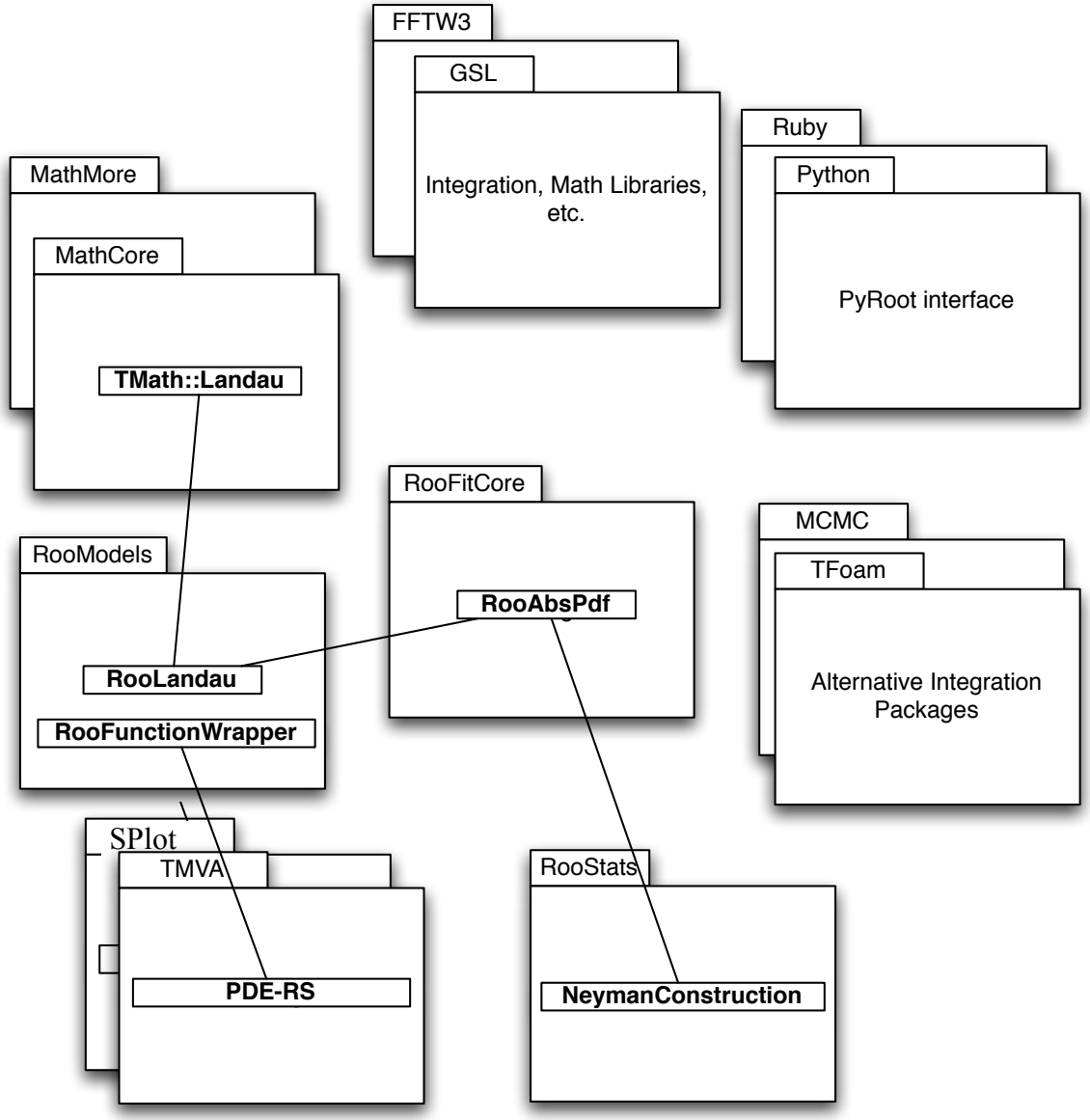
We would like to include similar functionality in RooStats so that one can easily compare the results of several methods given the same data and the same Model

- The models can be very complex, so want to avoid multiple implementations
- Coverage studies are a good means for comparison

Interoperability

BROOKHAVEN NATIONAL LABORATORY

Aim for high level of interoperability, needs thought.



Review of our Goals

BROOKHAVEN NATIONAL LABORATORY

Provide a "common language" in terms of concepts, tools, and code

- eases communication, convergence, sharing, etc.

Maintain full flexibility and extendibility

- focus on interfaces, anticipate extensions by users
  - build a framework, not an application

Provide some higher-level steering

- common at BaBar, already a few prototypes available
  - perhaps in python, via PyRoot? Needs more thought
- perhaps a graphical user interface

Provide ability to save a model and its RooFitSummary to a .root file to be shared, re-used, or combined

- the extreme form of "publishing" your result!

Conclusions

BROOKHAVEN NATIONAL LABORATORY

There seems to be a lot of support for developing common statistical tools / framework

- an important ingredient to a healthy collaboration
- if tools are also common between ATLAS & CMS, it would greatly aid our ability to compare and combine

RooFit seems to be the natural starting point

- what is there now is already very powerful!
- large user community already in ATLAS & CMS
- started to reevaluate dependencies, and re-factor?

Developing a to-do list

- should have prototype for June PhyStat07 Workshop
- could use volunteers, please contact

Now for some advice, feedback and comments....

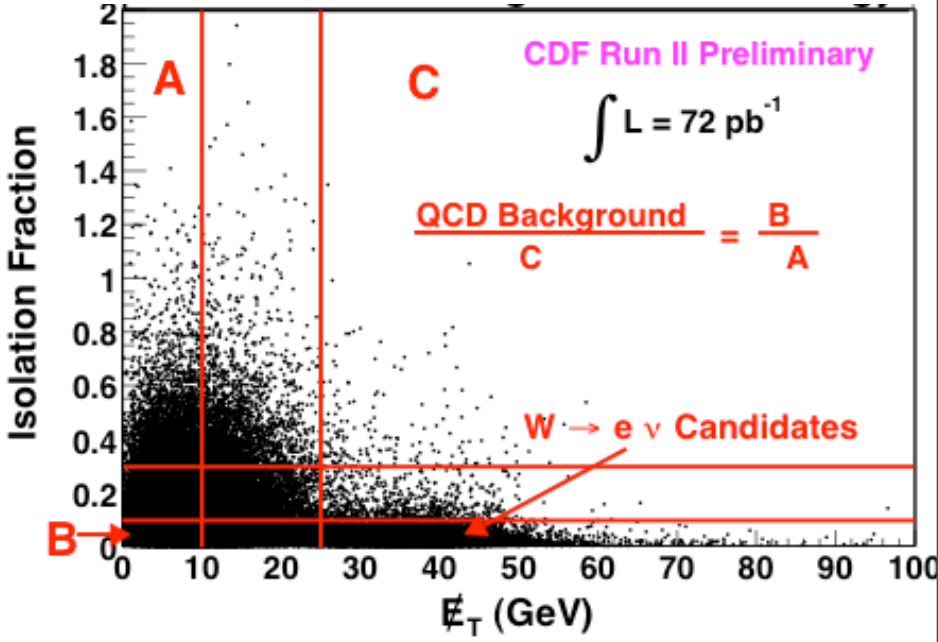
Backup

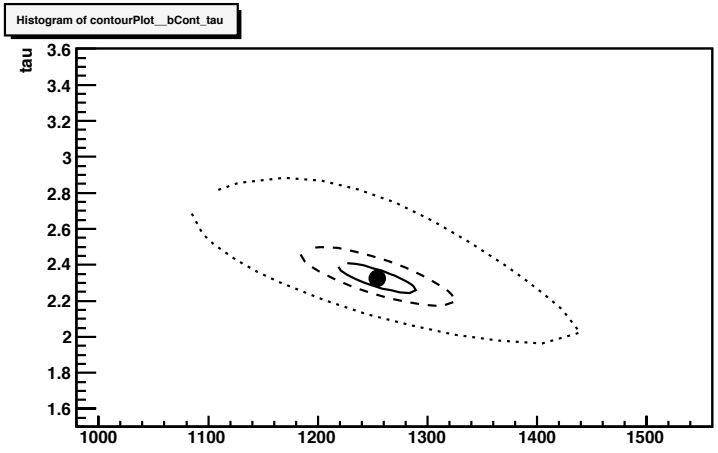
Data Driven SUSY Background Estimation

BROOKHAVEN NATIONAL LABORATORY

The estimation of the background is simply B/A*C

- but what is significance of a given x_{obs} ?
- propagation of error + C.H. is not very trustworthy
- build the model, and either use fit, Frequentist, or Bayesian
- a clear place to include correlations






```
RooRealVar   s("s","s",_x-_y, 0., 200.);
RooRealVar   b("b","b",_y*_xCont/_yCont,  0., 500.);
RooRealVar   tau("tau","tau",_yCont/_xCont,0,5);
RooRealVar   bCont("bCont","b control",_xCont,  0., 2000.);

RooFormulaVar  splusb("splusb","s+b",RooArgSet(s,b));
RooProduct   bTau("bTau","b*tau",RooArgSet(b, tau));
RooProduct   bContTau("bContTau","bCont*tau",RooArgSet(bCont, tau));
RooRealVar   x("x","x", _x, 0., 1000.);
RooRealVar   y("y","y", _y,  0., 1000.);
RooRealVar   yCont("yCont","y Control", _yCont,  0., 3000.);
RooRealVar   xCont("xCont","x Control", _xCont,  0., 3000.);

RooPoisson sigRegion("sigRegion","sigRegion",x,splusb);
RooPoisson sideband("sideband","sideband",y,bTau);
RooPoisson xContDist("xContDist","xCont dist",xCont,bCont);
RooPoisson yContDist("yContDist","yCont dist",yCont,bContTau);

RooProdPdf joint("joint","joint", RooArgSet(sigRegion,sideband,
                                            xContDist, yContDist) );
```