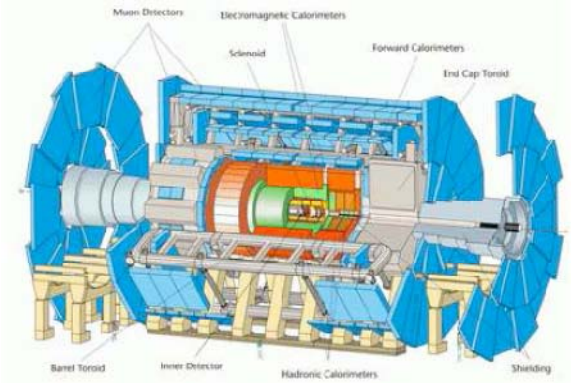
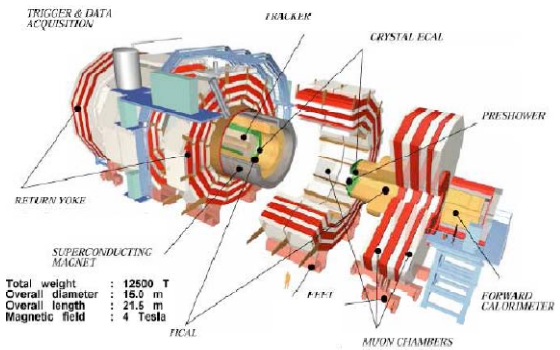
Statistics software for the LHC

Wouter Verkerke (NIKHEF)

Introduction

- LHC will start taking data in 2008
  - Expect enormous amount of data to be processed and analyzed (billions of events, Tb's of data in most condensed format)





- Central themes in LHC data analysis
  - Separating signal from background (e.g. finding the Higgs, SUSY)
  - Complex fits to extract physics parameters from data (e.g. CP violation parameters in B meson decays)
  - Determining statistical significance of your signal
  - Lots of interesting statistics issues with nuisance parameters (measurements in 'control' regions of data)

Introduction

- Statistics tools for LHC analyses
  - Potentially a very broad subject!
  - Need to make some choices here
    (apologies to software unmentioned)

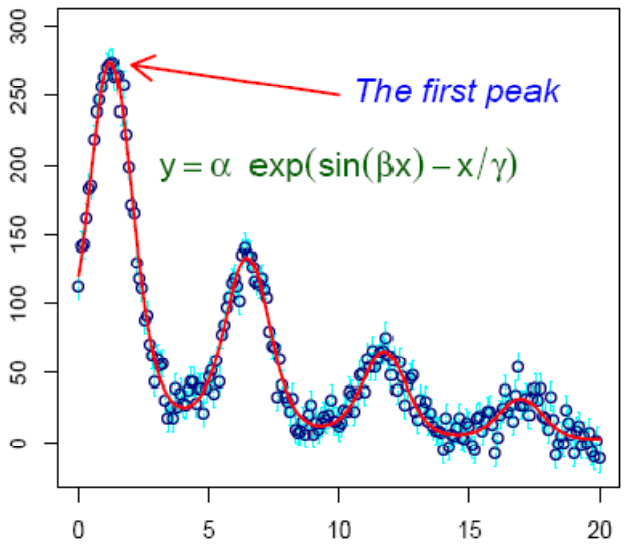
- Will focus on following areas, breakdown roughly following practical steps taken in LHC data analysis
  - Working environment (R/ROOT)
  - Separating signal and background
  - Fitting, minimization and error analysis
  - Practical aspects of defining complex data models
  - Tools to calculate limits, confidence intervals etc..

Working Environment

Choice of working environment R vs. ROOT

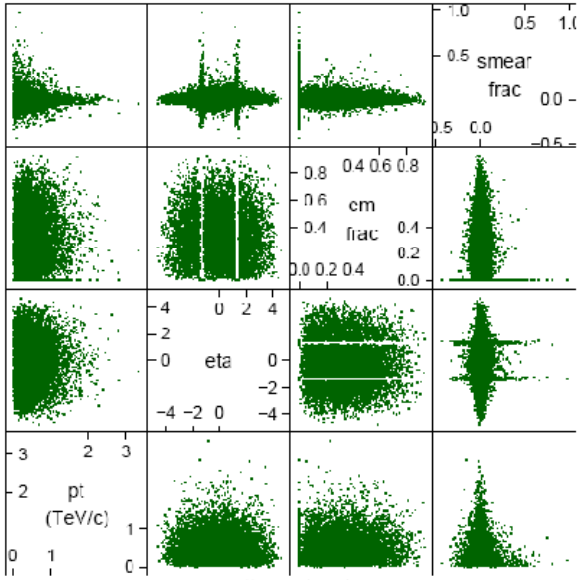
- Several very nice presentations on ***R*** at PhyStat2005
  - "R is a language and environment for statistical computing and graphics." R is "not unlike S"
  - The author of S, John Chambers, received 1998 ACM Software System Award:
    - The ACM's citation notes that Dr. Chambers' work "will forever alter the way people analyze, visualize, and manipulate data . . . "
  - Popular among statisticians

- "Why Use ***R***"
  - R provides excellent graphical tools
  - The R language is convenient and powerful for data manipulation
  - Many modern data analysis techniques are available as R packages



Choice of working environment R vs. ROOT

- Many nice tools for data visualization
- Can read many data formats
  - Included ROOT Trees
- An R session can be saved to disk, and the application state recovered at a later time



- *Extension of R come in packages*
  - Distributed package management integrated in R
  - Like Perl's CPAN, the Linux yum utility. E.g to download and install package from central repository

```
> install.packages("Rcmdr", dependencies = TRUE)
```

Choice of working environment R vs. ROOT

- But HEP Community very focused on homegrown platform
  - ROOT (1995-now)
  - And before that PAW (1985-1995)
- Large project with many developers, contributors, workshops

ROOT Team today
(working 50% or more)

Interesting talks about all these topics

- CORE: Fons Rademakers, Leo Franco, Diego Marcos
- DICT: Axel Naumann, Philippe Canal, (Stefan Roiser)
- I/O: Philippe Canal(50%), Paul Russo
- MATH: Lorenzo Moneta, (Anna Kreshuk)
- GEOM: Andrei Gheata, Mihaela Gheata
- GUI: Ilka Antcheva, Bertrand Bellenot, (V.Onuchin(yy%))
- GRAF: Olivier Couet, Timur Potcheptsov, Matev Tadel(50%)
- PROOF: Fons, Gerri Ganis, Jan Iwaszkiewicz
- PYROOT: Wim Lavrijsen (xx%)

5

Choice of working environment R vs. ROOT

- ROOT has become *de facto* HEP standard analysis environment
  - Available and actively used for analyses in running experiments (Tevatron, B factories etc..)
  - ROOT is integrated LHC experimental software releases
  - Data format of LHC experiments is (indirectly) based on ROOT → Several experiments have/are working on summary data format directly usable in ROOT
  - Ability to handle *very* large amounts of data
- ROOT brings together a lot of the ingredients needed for (statistical) data analysis
  - C++ command line, publication quality graphics
  - Many standard mathematics, physics classes: Vectors, Matrices, Lorentz Vectors Physics constants…
- Line between 'ROOT' and 'external' software not very sharp
  - Lot of software developed elsewhere, distributed with ROOT (TMVA, RooFit)
  - Or thin interface layer provided to be able to work with external library (GSL, FFTW)
  - Still not quite as nice & automated as 'R' package concept

(Statistical) software repositories

- ROOT functions as moderated repository for statistical & data analysis tools
  - Examples TMVA, RooFit, TLimit, TFractionFitter

- Several HEP repository initiatives, some contain statistical software
  - PhyStat.org (StatPatternRecognition, TMVA,LepStats4LHC)
  - HepForge (mostly physics MC generators),
  - FreeHep

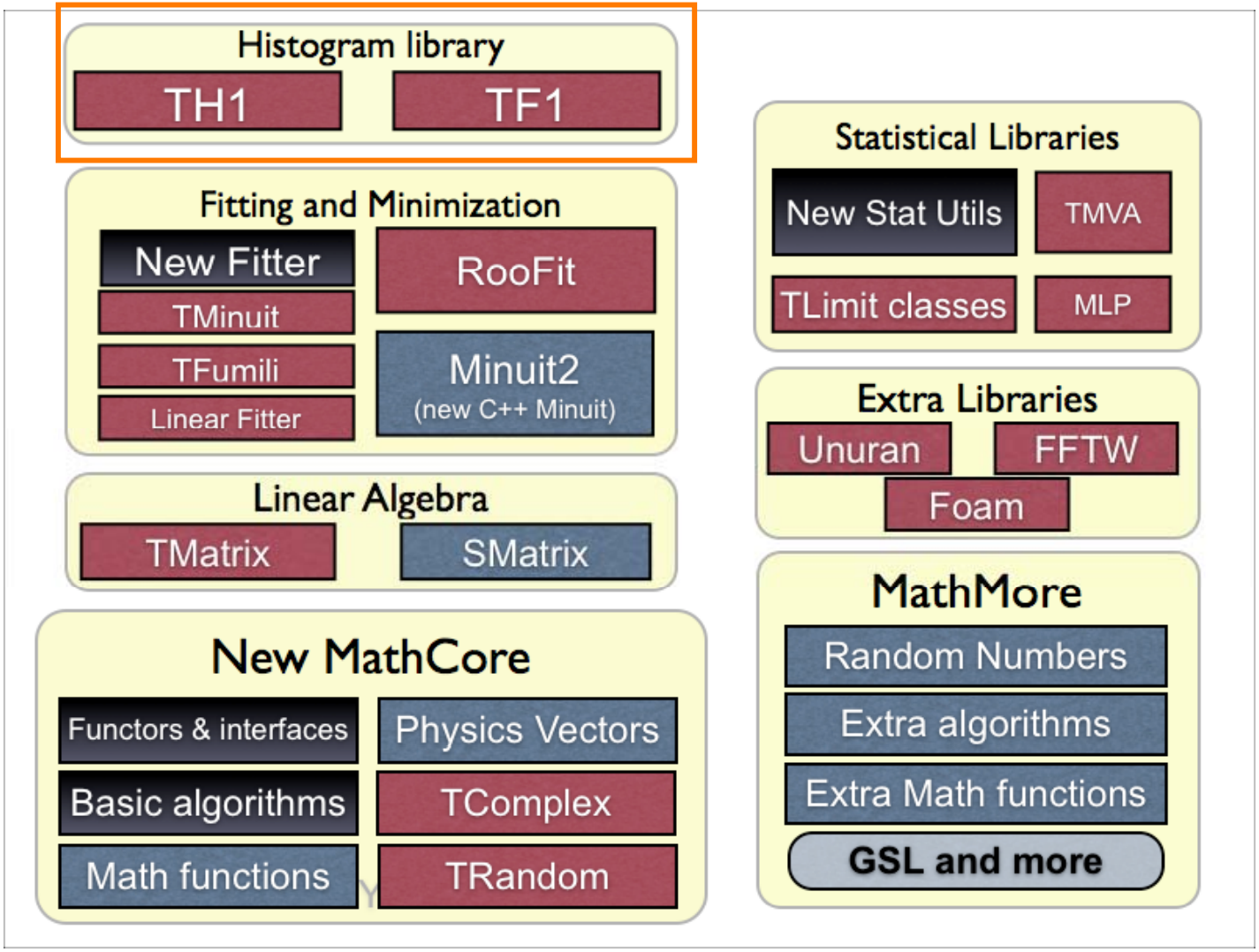
- Excellent summary of non-HEP statistical repositories on Jim Linnemans statistical resources web page
  - From Phystat 2005
  - http://www.pa.msu.edu/people/linnemann/stat_resources.html

Core ROOT software specifically related to *statistical analysis*

- Lots of statistics related software in MATH works package provided by ROOT team (L. Moneta)
  - See also presentation by Lorenzo in this session
- Main responsibilities for this work package:
  - Basic mathematical functions
  - Numerical algorithms
  - Random numbers
  - Linear algebra
  - Physics and geometry vectors (3D and 4D)
  - Fitting and minimization
  - Histograms (math and statistical part)
  - Statistics (confidence levels, multivariate analysis)

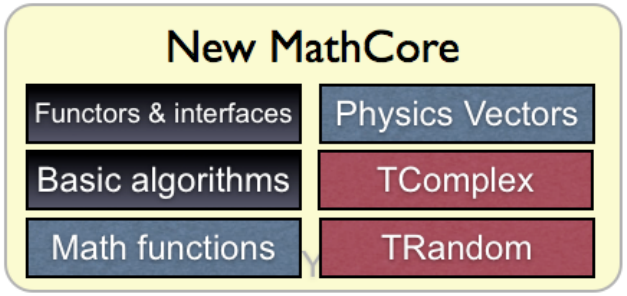
(Re)organization of ROOT math/statistics tools

Almost every recent HEP histogram/curve started its life as TH1 or TF1 object

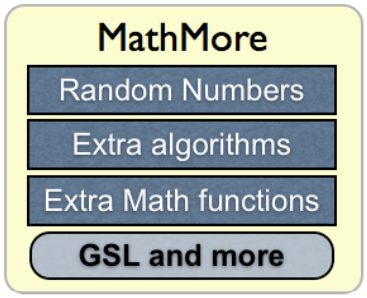


ROOT Math Libraries

- **MathCore** provides
  - **function interfaces,**
  - **math functions**
  - **basic algorithms,**
  - **random number generation**
  - Provides coherent interface for algorithms
  - Several random number generators available

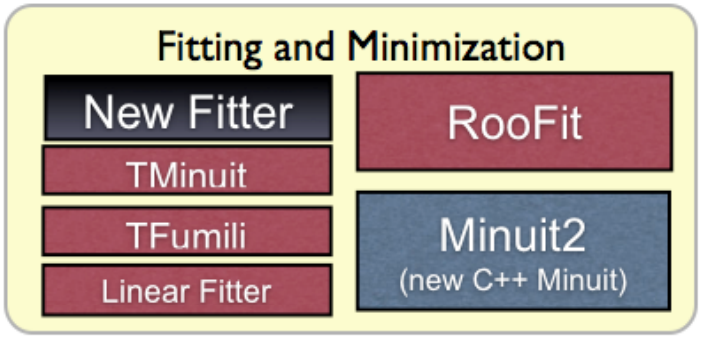


- **MathMore** provides C++ interface to GSL algorithms and functions
  - requires a GSL (version >= 1.8) already installed



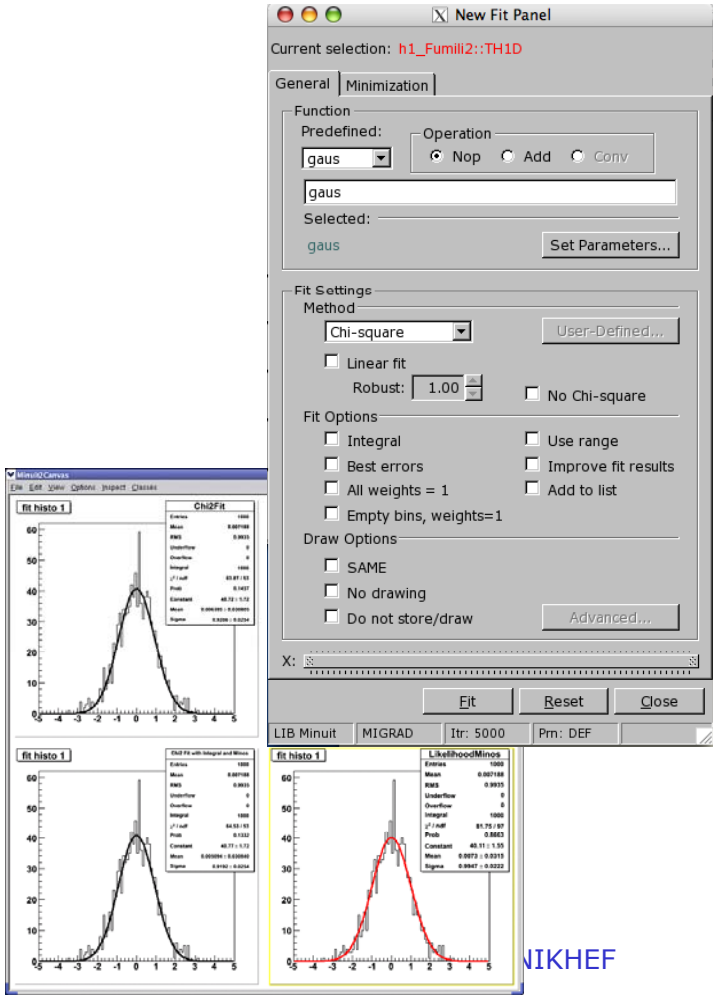
- Numerical algorithms for 1D functions:
  - Numerical Derivation
  - Numerical Integration
  - Root Finders
  - Minimization
  - Interpolation
  - Chebyshev polynomials (for function approximation)

ROOT – Fitting & Minimization



- Fitting involves
  - Data modeling
  - Minimization
  - Error Analysis

- Simplest use case: ROOT native fitting interface
  - Model = TF1/TF2/TF3 function
  - Data is TH1/TH2/TH3 histogram
  - Minimization & error analysis by MINUIT (χ^2 or –logL)
  - Command line or graphical interface
  - Quick & easy, mostly for 1D problems, but can handle 2/3D
  - But does not scale well to very complex problems



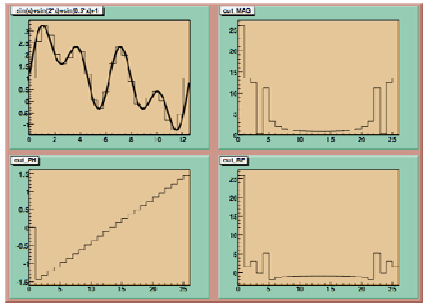
ROOT – Choices for Minimization & Error analysis

- Industry Standard for minimization and error analysis for nearly 40 years: MINUIT
  - From the manual
    "*It is especially suited to handle difficult problems, including those which may require guidance in order to find the correct solution.*"
  - Multi-decade track record Started in CERNLIB, then interfaced in PAW, now in ROOT
  - MINUIT provides two options for analysis:
    HESSE (2nd derivative) and MINOS (hill climbing)
  - Ported to C++ by authors in 2002, extensively tested again

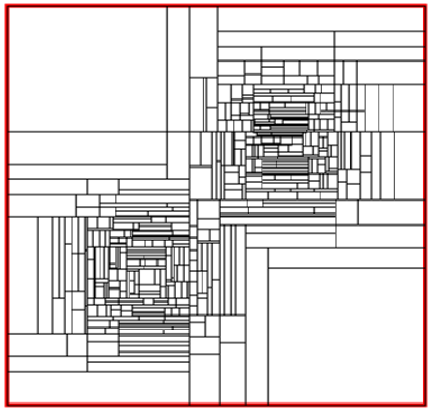
- A number of alternatives available
  - `TFumili`, faster convergence for certain type of problems
  - `TLinearFitter` analytical solution for linear problems
  - `TFractionFitter` , `TBinomialEfficiencyFitter` for fitting of MC templates, efficiencies from ratios (F. Filthaut)

Examples of imported/interfaced external software in ROOT

- Fast Fourier Transforms
  - ROOT design has abstract base class `TVirtualFFT`.
  - Easy interface to `TH1` histogram class
  - Currently one implementation:
    interface to external FFTW package
    - Industrial strength packages, details see `www.fftw.org`
    - Interface requires pre-installed library
    - `TFFTComplex` for complex input/output transforms
    - `TFFTRealComplex` for real input/complex output
    - `TFFTComplexReal` for complex input/real output
    - `TFFTReal` for real input/output



- Class `TFoam` is slimmed down version of FOAM MC generator
  - Distributed with ROOT, no external libraries needed
  - Synergy with ROOT – Persist grids using ROOT I/O



Signal and Background
Tools for Multivariate Analysis

Wouter Verkerke, NIKHEF

HEP tradition & multivariate analysis

- HEP traditionally quite conservative
  - Cut based analysis or simple NN applications most common past
- Simple NN tools integrated in PAW/ROOT since long time
  - PAW: MLPfit (MultiLayer Percetron. J. Schwindling)
  - ROOT: `TMultilayerPerceptron` (inspired on MLPFIT)
  - Also standalone tools used (JETNET etc..)
- Use of other (newer) MVA taking off in recent years
- Easy access other/newer MVA techniques crucial to acceptance & good uses of these techniques in HEP
  - E.g overheard over coffee at CERN after seminar on an analysis using Boosted Decision Trees

    "Of course you can find a signal anywhere with a BDT!
    I'll believe it when a cut based analysis see the same signal"

  - Clear need for tools to understand & compare MVA techniques

Multivariate Classification Algorithms

- A large variety of multivariate classifiers (MVAs) exists

Traditional

Rectangular cuts (optimisation often "by hand")
Projective likelihood (up to 2D)
Linear discriminants (χ^2 estimators, Fisher, …)
Nonlinear discriminants (Neural nets, …)

Variants

Prior decorrelation of input variables (input to cuts and likelihood)
Principal component analysis of input variables
Multidimensional likelihood (kernel nearest neighbor methods)

New

Decision trees with boosting and bagging, Random forests
Rule-based learning machines
Support vector machines
Bayesian neural nets, and more general *Committee* classifiers

Software for Multivariate analysis

- Case for use of Multivariate analysis in HEP is clear
  - Additional discrimination power
  - Especially if information is contained in correlations (which are difficult to exploit with a cut-based analysis)
  - New techniques more robust when many variable are added that not carry a lot of discriminating information
- Active development of HEP MVA training & evaluation & validation tools since 2005
  - **TMVA – Toolkit for MultiVariate Analysis (see talk by F.Tegelfeld)**
  - **StatPatternRecognition (see talk by I. Narksy)**
- Both developed in C++ on SourceForge
  - Implement large variety of MVA techniques
  - Provide training & validation tools
  - Work with ROOT native data
- The average HEP physicist can now get started with e.g. a Boosted Decision Tree in <1 day of work

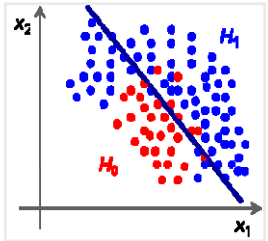
Introduction to *T*MVA & StatPatternRecognition

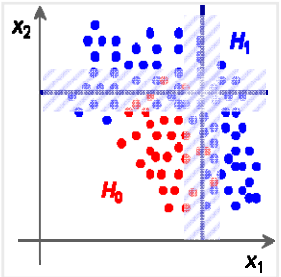
- *T*MVA — Toolkit for multivariate data analysis
  - Distributed with ROOT / read & write ROOT native data format
  - Framework for *parallel* **training**, **testing**, **evaluation** and **application** of MV classifiers
  - Each classifier provides a ranking of the input variables
  - The input variables can be de-correlated or projected upon their principal components
  - Training results and full configuration are written to weight files and applied by a **Reader**

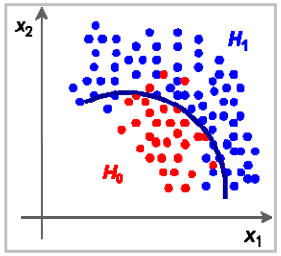
- StatPatternRecognition
  - Standalone tool with ROOT I/O capability
  - Production tool, rather than Interactive Analysis tools
  - Ability to process large datasets in many dimensions
  - Reasonable CPU and memory consumption, scalability
  - Use in production code developed by a HEP collaboration

Classifiers implemented in TMVA

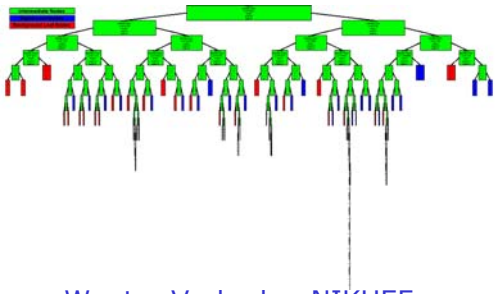
- Currently implemented classifiers :
  - Rectangular cuts (optimized)
  - Projective and multi-D likelihood estimator
  - Fisher and H-Matrix discriminants
  - Artificial neural networks
    (three different *multilayer perceptrons*)
  - Boosted/bagged decision trees
    with automatic node pruning
  - RuleFit
  - Support Vector Machine







- Coming:
  - *Generalised non-linear discriminant (NLD)*
  - *Committee classifier*



Classifiers implemented in StatPatternRecognition

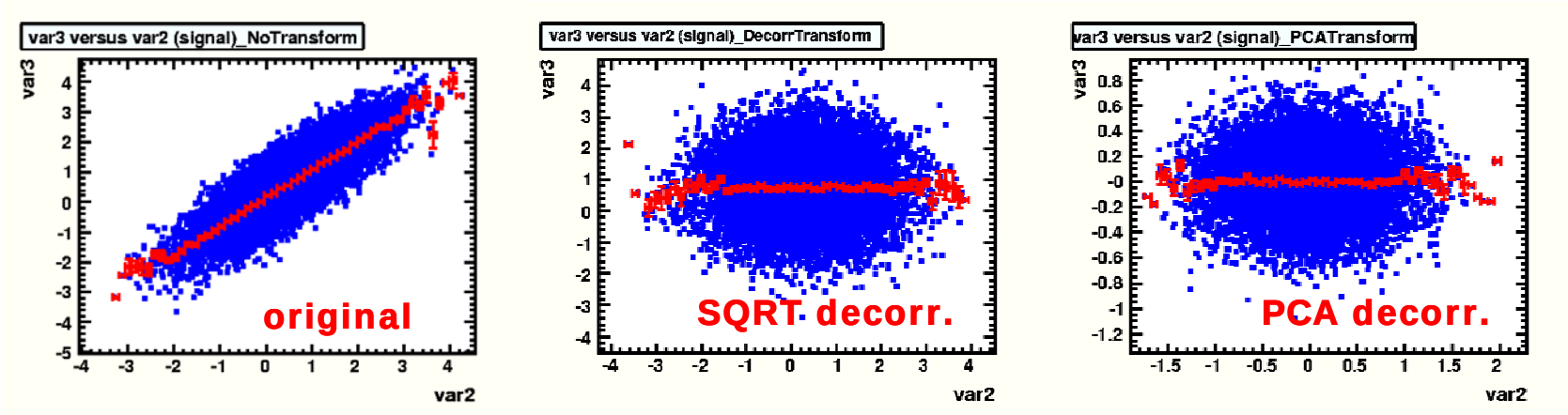
- Classifiers implemented in StatPatternRecognition
  - Decision split, or stump
  - Decision trees (2 flavors: regular tree and top-down tree)
  - Bump hunter (PRIM, Friedman & Fisher)
  - LDA (aka Fisher) and QDA
  - Logistic regression
  - Boosting: discrete AdaBoost, real AdaBoost, and epsilon-Boost.
    - Can boost any sequence of classifiers.
  - Arc-x4 (a variant of boosting from Breiman)
  - Bagging.
    - Can bag any sequence of classifiers.
  - Random forest
  - Backprop neural net with a logistic activation function
  - Multi-class learner (Allwein, Schapire and Singer)
  - Interfaces to SNNS neural nets (without training):
  - Backprop neural net, and Radial Basis Functions

Example of TMVA infrastructure – Decorrelation of input variables

- Removal of linear correlations by rotating input variables
  - Commonly realized for all methods in *T*MVA centrally in **DataSet** class

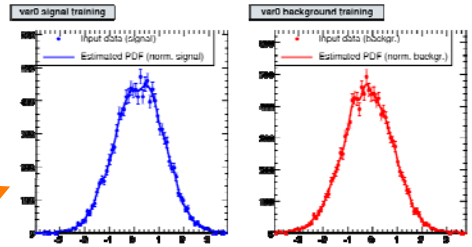
- Determine *square-root* C' of covariance matrix C, *i.e.*, $C = C'C'$
  - Compute C' by diagonalising C:
  - Transform original (x) into decorrelated variable space (x') by: $x' = C'^{-1}x$

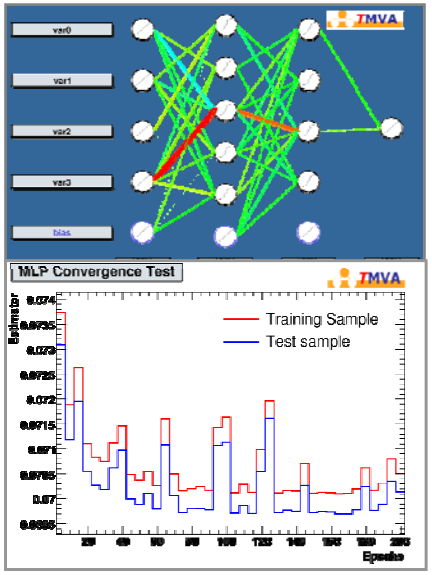
- Also implemented Principal Component Analysis

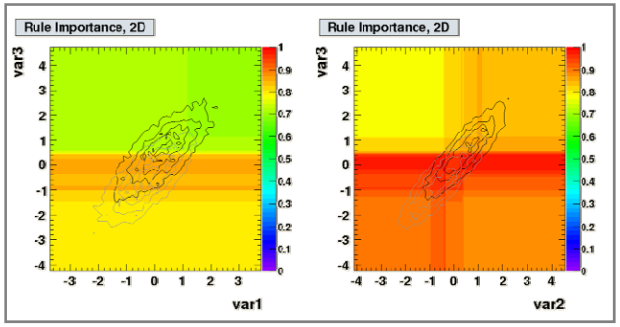


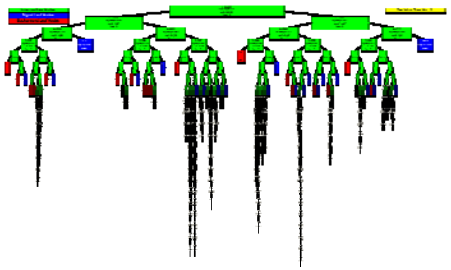
TMVA – Classifier validation

- Lot of training level output provided for validation of classifier training
- Examples shown here
  - PDFs of Projective Likelihood method
  - Topology of Multilayer Perceptron and training progress indicator
  - Graphical representation of trees constructed for Boosted Decision Tree classifier
  - Rule importance for signal and background for RuleFitter method





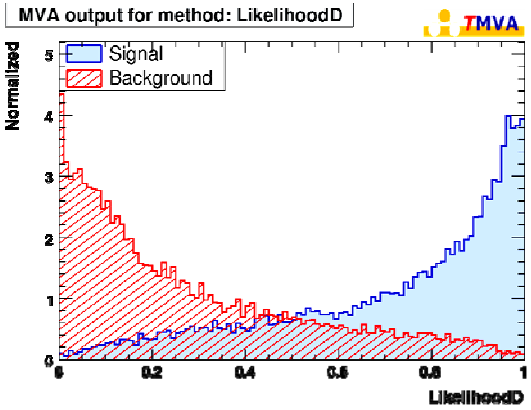




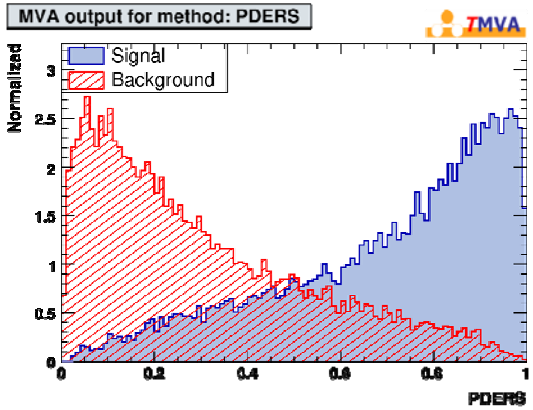
TMVA Classifier output comparison

- TMVA Strong point: Easy comparison of performance of various methods

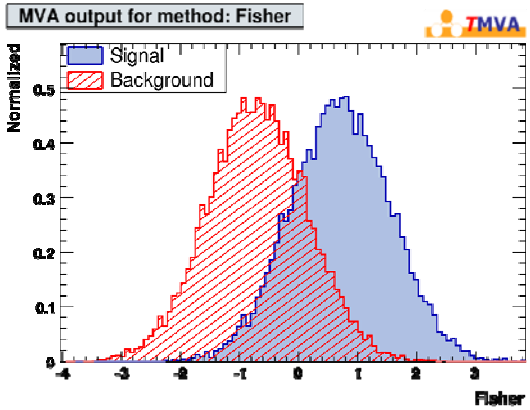
Likelihood



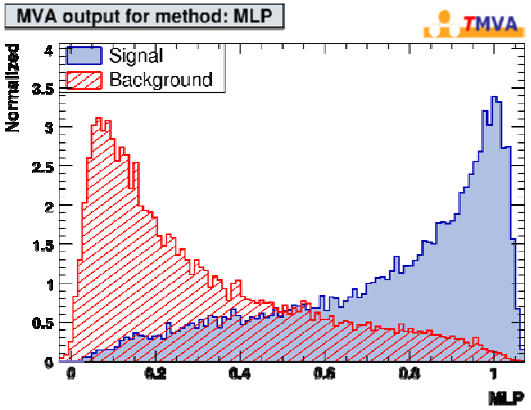
Prob.Dens.Est



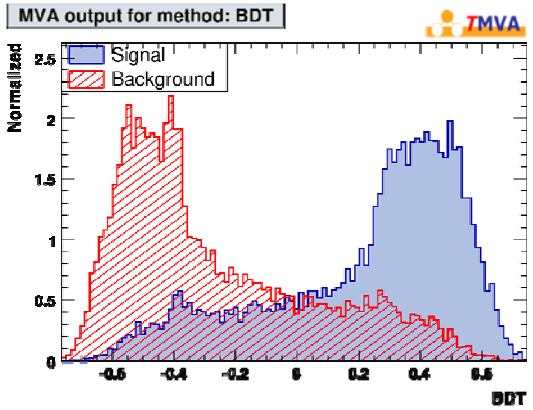
Fisher



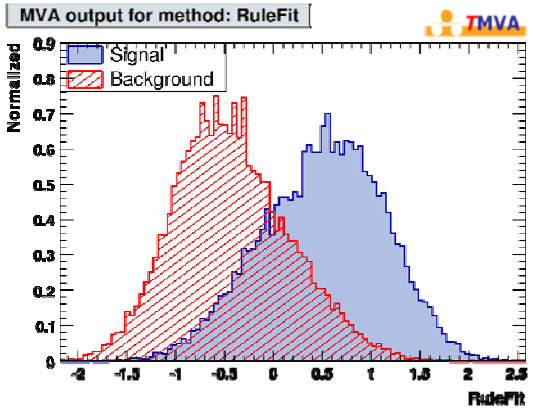
MultiLayerPerceptron



BoostedDecisionTrees

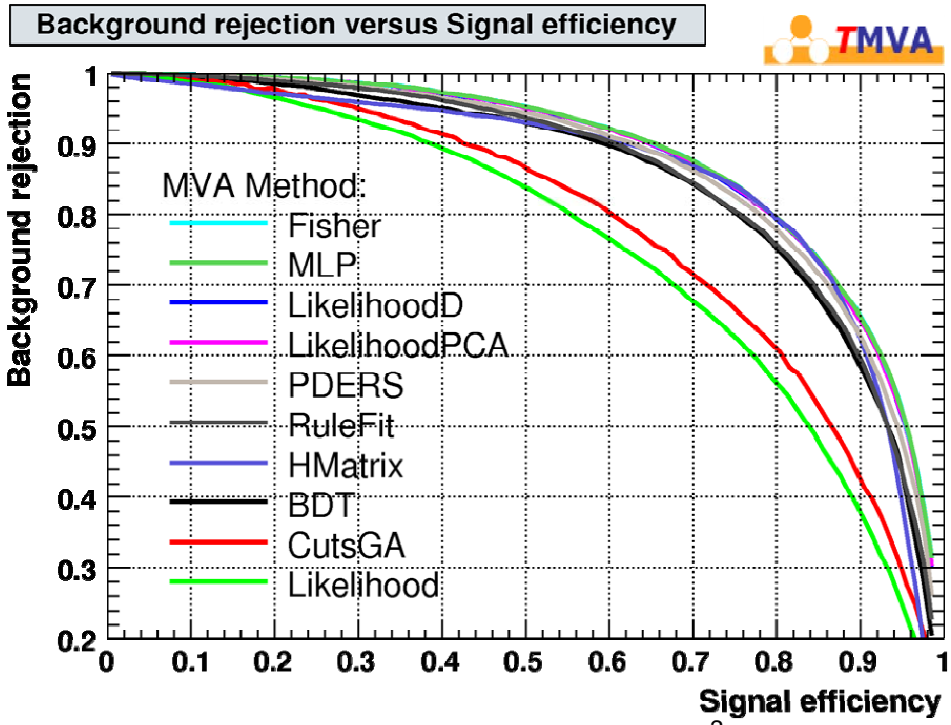


RuleFitter



TMVA Classifier evaluation & comparison

- Several benchmark quantities computed by *T*MVA
  - e(sig) vs e(bkg) when cutting on classifier output



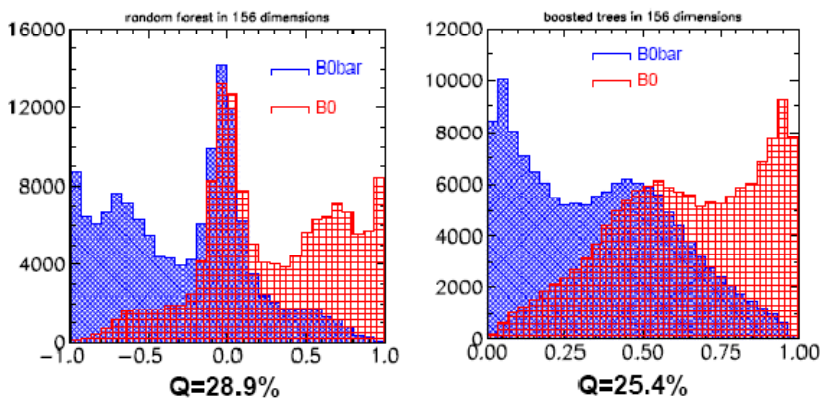
  - The *Separation* $\frac{1}{2}\int \frac{\left(\hat{y}_S(y)-\hat{y}_B(y)\right)^2}{\hat{y}_S(y)+\hat{y}_B(y)}dy$
  - The discrimination *Significance* $\left(\bar{y}_S-\bar{y}_B\right)\Big/\sqrt{\sigma^2_{y,S}+\sigma^2_{y,B}}$

TMVA Classifier evaluation & comparison

- Overtraining detection
  - Compare performance between training and test sample to detect overtraining
- Classifiers provide specific ranking of input variables
- Correlation matrix and classification "overlap" :
  - if two classifiers have similar performance, but significant non-overlapping classifications → combine them!

StatPatternRecognition – Training & evaluation tools

- Cross-validation
  - Split data into M subsamples
  - Remove one subsample, optimize your model on the kept M-1 subsamples and estimate prediction error for the removed subsample
  - Repeat for each subsample
- Bootstrap
  - Randomly draw N events with replacement out of the data sample of size N
  - Optimize model on a bootstrap replica and see how well it predicts the behavior of the original sample
  - Reduce correlation between bootstrap replica and the original sample
  - *To assess estimator bias and variance on small training samples*
- Tools for variable selection, computation of data moments
- Arbitrary grouping of input classes in two categories (signal and background)
- Tools for combining classifiers: training a classifier in the space of outputs of several subclassifiers



StatPatternRecognition Distribution & Use

- Distribution
  - Introduced in early 2005 at BaBar and committed to BaBar CVS.
  - In September 2006 SPR was posted at Sourceforge: https://sourceforge.net/projects/statpatrec
  - Two versions
    - Standalone with ASCII I/O; no external dependencies
    - With Root I/O (ROOT dependency)
- Use
  - Currently used by several HEP collaborations (BaBar, D0, MINOS, SNO, searches for supernovae), being promoted within CMS.
  - There are users outside HEP as well. Downloaded off Sourceforge ~310 times (as of mid April 2007).
  - O(10) analyses and O(20) people using the package in BaBar alone, mostly in rare leptonic and particle ID groups.

*T*MVA Development and Distribution

- *T*MVA is a SourceForge (SF) package for world-wide access
  - Home page ..... `http://tmva.sf.net/`
  - SF project page ..... `http://sf.net/projects/tmva`
  - View CVS ..... `http://tmva.cvs.sf.net/tmva/TMVA/`
  - Mailing list ..... `http://sf.net/mail/?group_id=152074`
  - ROOT class index .....`http://root.cern.ch/root/htmldoc/TMVA_Index.html`

- Active project
  - Currently 6 main developers, and 26 registered contributors at SF
  - >1000 downloads since March 2006 (not accounting cvs checkouts and ROOT users)

- Written in C++, relying on core ROOT functionality
  - Full examples distributed with *T*MVA, including analysis macros and GUI
  - Scripts are provided for *T*MVA use in ROOT macro, as C++ executable or with python

- Integrated and distributed with ROOT since ROOT v5.11/03

Constructing models to describe your data

Wouter Verkerke, NIKHEF

Modeling your data – Introduction

- A *technical* though important ingredient for fitting, calculation of limit, confidence intervals and significances is a language to model your data
  - For example how do you calculate the likelihood associated with a probability density function given by the following mathematical expression

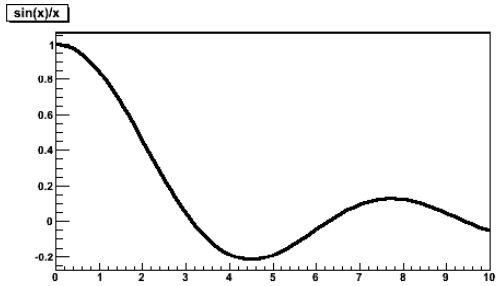
$$f_{sig} \cdot \left[\mathrm{SigSel}(m;\bar{p}_{sig}) \cdot \left(\mathrm{SigDecay}(t;\vec{q}_{sig},\sin(2\beta)) \otimes \mathrm{SigResol}(t \mid dt;\vec{r}_{sig})\right)\right] + (1-f_{sig})\left[\mathrm{BkgSel}(m;\bar{p}_{bkg}) \cdot \left(\mathrm{BkgDecay}(t;\vec{q}_{bkg}) \otimes \mathrm{BkgResol}(t \mid dt;\vec{r}_{bkg})\right)\right]$$

- Not easy…
  - Functionality of ROOT TF1 class too limited for complex physics use cases
  - Common alternative: custom likelihood function implement in O(100-1000) lines of FORTRAN or C++ code.
  - *Cumbersome & manpower intensive because flexibility / modularity is difficult to achieve*

Modeling your data – ROOT TF1

- ROOT TF1 ideal for cases of lower complexity
  - **Interpreted C++ expressions**

```
TF1 *fa1 = new TF1("fa1","sin(x)/x",0,10);
fa1->Draw();
```



  - **For expression of >1 line: binding of generic C++ functions (shown), functors, or classes**

```
Double_t myfunction(Double_t *x, Double_t *par) {
  Float_t xx =x[0];
  Double_t f = TMath::Abs(par[0]*sin(par[1]*xx)/xx);
  return f;
}

TF1 *f1 = new TF1("myfunc",myfunction,0,10,2);
f1->SetParameters(2,1);
f1->SetParNames("constant","coefficient");
f1->Draw(); }
```

Modeling your data – ROOT TF1

- ROOT Class TF1 supports
  - Toy MC generation using accept/reject sampling

```
TH1F *h1=new TH1F("h1","test",100,0,10);
h1->FillRandom("myfunc",20000);
```

  - Fitting to TH1

```
f1->SetParameters(800,1);
h1.Fit("myfunc");
```

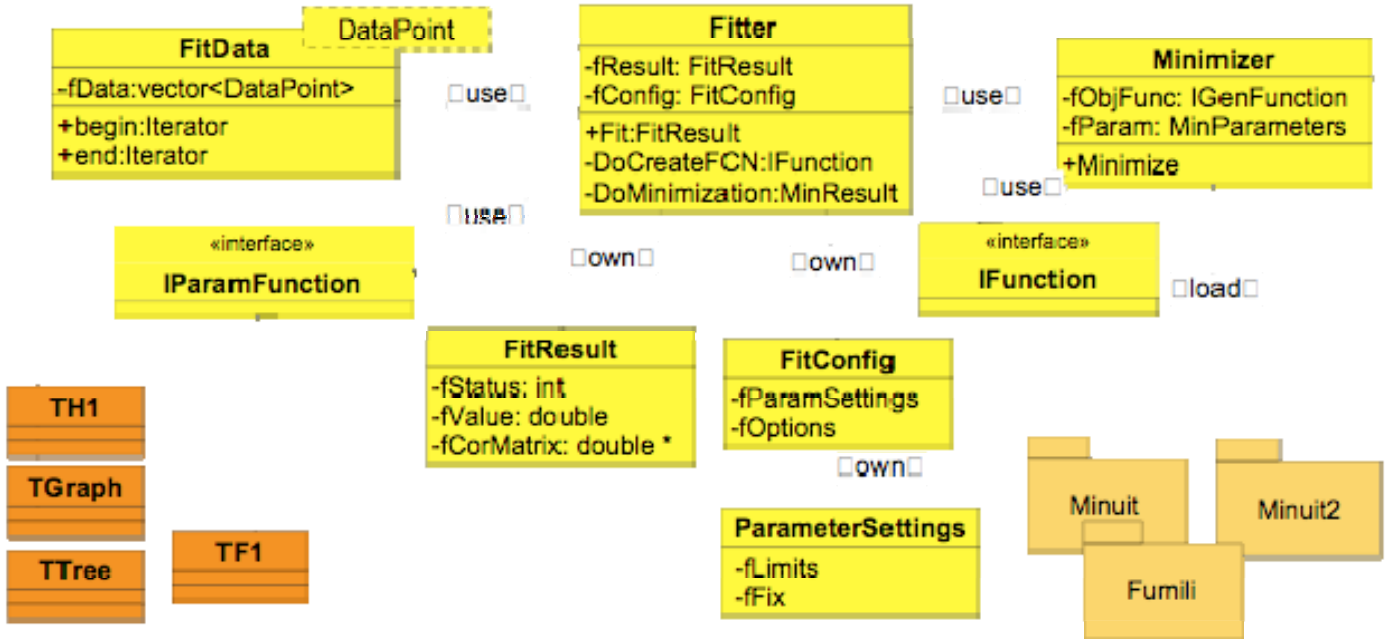
  - Numeric integration / derivation

- Still several open issues on the road to complex problems
  - Scalability issue to very complex models (e.g. sin2β p.d.f shown before)
  - Really need *probability density* functions for many physics problems. Brings in many issues with normalization and performance and (numerical) integration

ROOT Data modeling – Beyond TH1

- Two projects aiming for more formal and modular representation of data, (probability density) functions, fit results
- RooFit – Originally developed for BaBar in 1999
- New ROOT fitter framework

Design proposal



RooFit – Development history

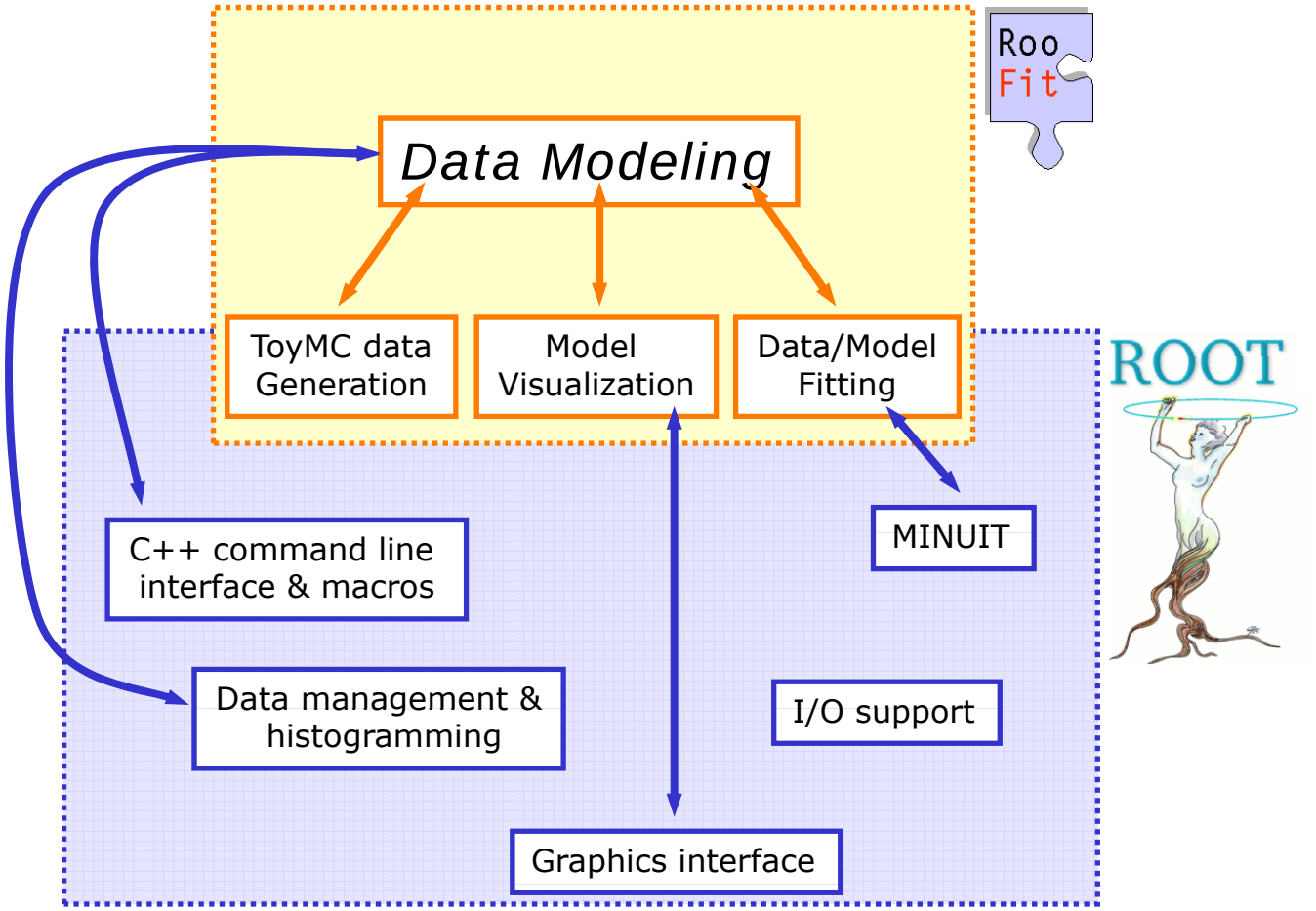
- Initial approach complex unbinned ML fits in BaBar: write it from scratch (in FORTRAN!)
  - Does its designated job quite well, but took a long time to develop
  - Possible because sin(2β) effort supported by O(50) people.
  - Optimization of ML calculations hand-coded → error prone and not easy to maintain
  - Difficult to transfer knowledge/code from one analysis to another.

- A better solution: A modeling language in C++ that integrates seamlessly into ROOT
  - Recycle code *and* knowledge

- Development of RooFit package
  - Started 7 years ago in BaBar CVS repository
  - Very successful in BaBar virtually everybody uses it
  - Now in standard ROOT distribution

RooFit – a data modeling language for ROOT

Extension to ROOT – (Almost) no overlap with existing functionality



Data modeling – OO representation

- Idea: represent math symbols as C++ objects

Mathematical concept		RooFit class
variable	x, p	`RooRealVar`
function	$f(\vec{x})$	`RooAbsReal`
PDF	$F(\vec{x}; \vec{p}, \vec{q})$	`RooAbsPdf`
space point	$\vec{x}$	`RooArgSet`
integral	$\int_{x_{\min}}^{x_{\max}} f(x)dx$	`RooRealIntegral`
list of space points	$\vec{x}_k$	`RooAbsData`

  - Result: 1 line of code per symbol in a function (the C++ constructor) rather than 1 line of code per function

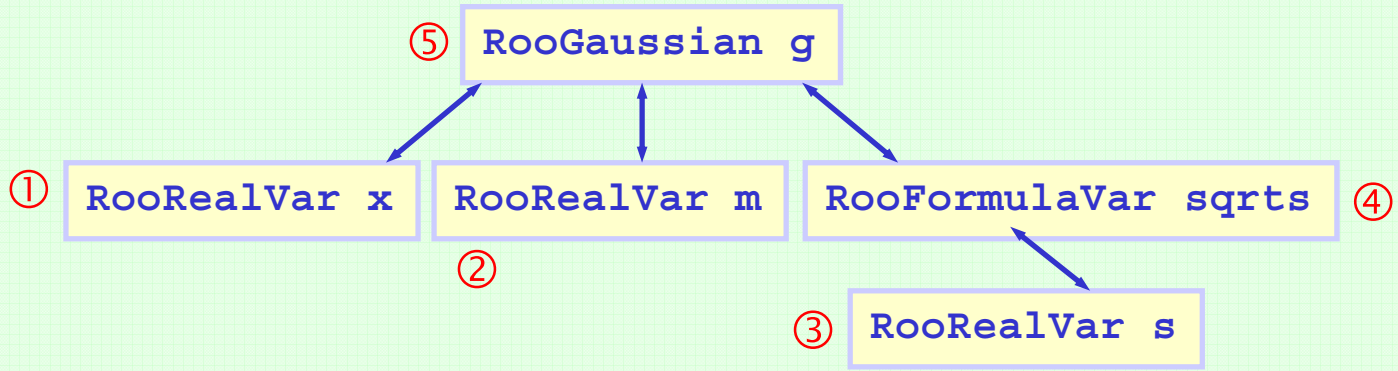
Data modeling – Constructing composite objects

- Straightforward correlation between mathematical representation of formula and RooFit code

Math

$$gauss(x, m, \sqrt{s})$$

RooFit diagram

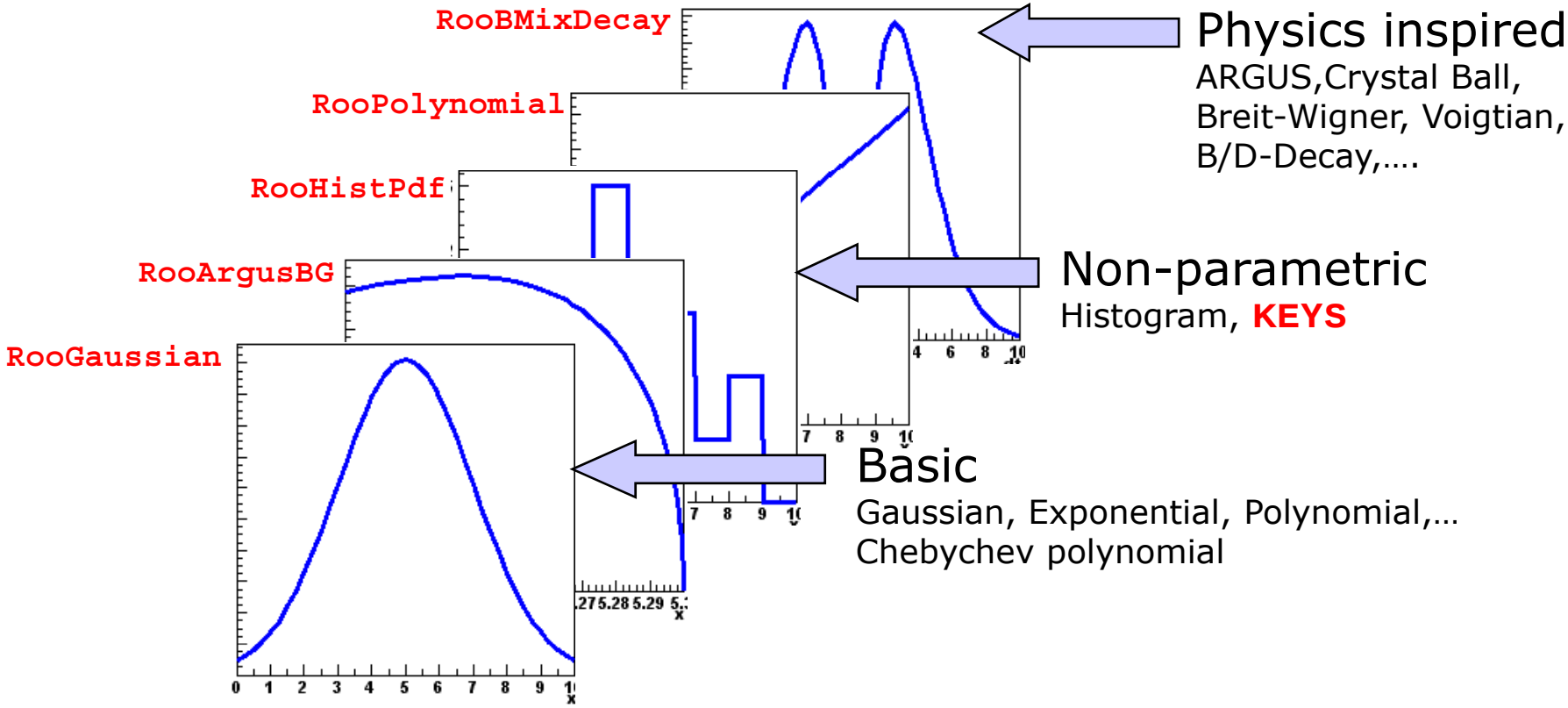


RooFit code

```
① RooRealVar x("x","x",-10,10) ;
② RooRealVar m("m","mean",0) ;
③ RooRealVar s("s","sigma",2,0,10) ;
④ RooFormulaVar sqrts("sqrts","sqrt(s)",s) ;
⑤ RooGaussian g("g","gauss",x,m,sqrts) ;
```

Model building – (Re)using standard components

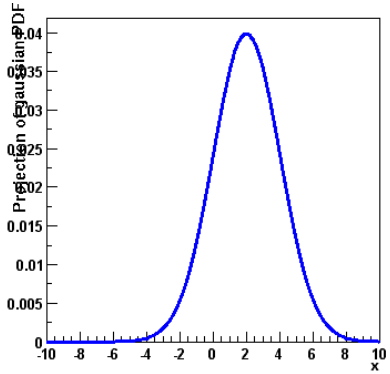
- RooFit provides a collection of compiled standard PDF classes



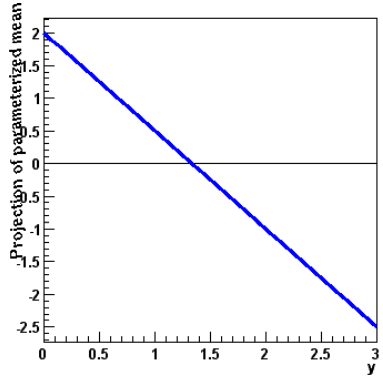
Easy to extend the library: each p.d.f. is a separate C++ class

Model building – (Re)using standard components

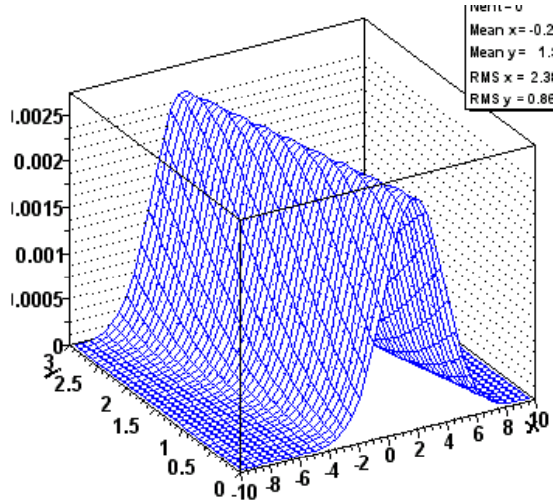
- P.d.f. arguments can be parameters, observables, or even functions themselves
  - No static notion of interpretation → Maximum flexibility



$g(x;m,s)$



$m(y;a_0,a_1)$



$g(x,y;a_0,a_1,s)$

```
RooPolyVar  m("m",y,RooArgList(a0,a1)) ;
RooGaussian g("g","gauss",x,m,s) ;
```

  - Works for any RooFit object, including classes you write yourself

Handling of p.d.f normalization

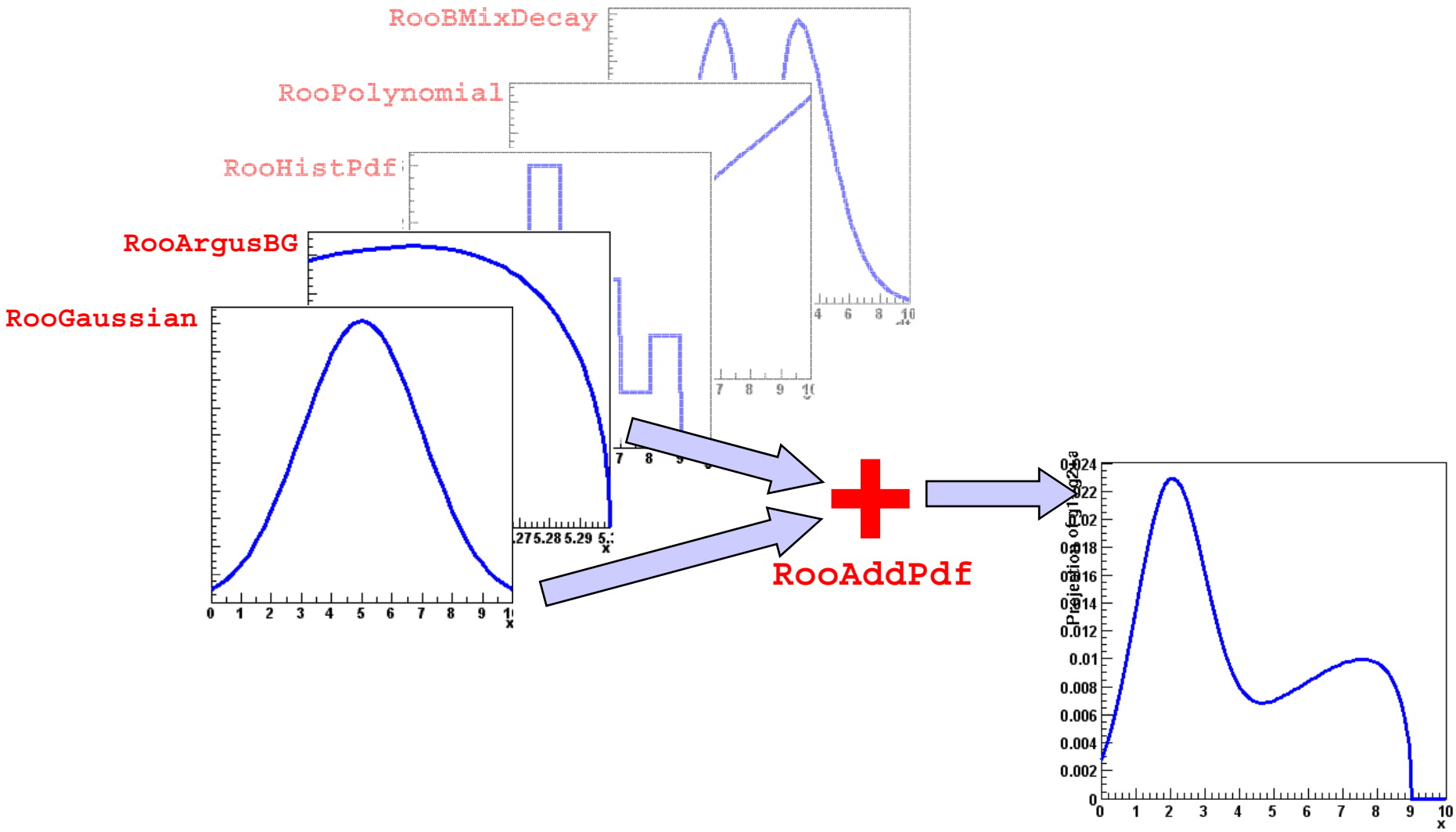
- Normalization of (component) p.d.f.s to unity is often a good part of the work of writing a p.d.f.

- RooFit handles most normalization issues transparently to the user
  - P.d.f can advertise (multiple) analytical expressions for integrals
  - When no analytical expression is provided, RooFit will automatically perform numeric integration to obtain normalization
  - More complicated that it seems: even if *gauss(x,m,s)* can be integrated analytically over x, *gauss(f(x),m,s)* cannot. Such use cases are automatically recognized.
  - Multi-dimensional integrals can be combination of numeric and p.d.f-provided analytical partial integrals

- Variety of numeric integration techniques is interfaced
  - Adaptive trapezoid, Gauss-Kronrod, VEGAS MC…
  - User can override configuration globally or per p.d.f. as necessary

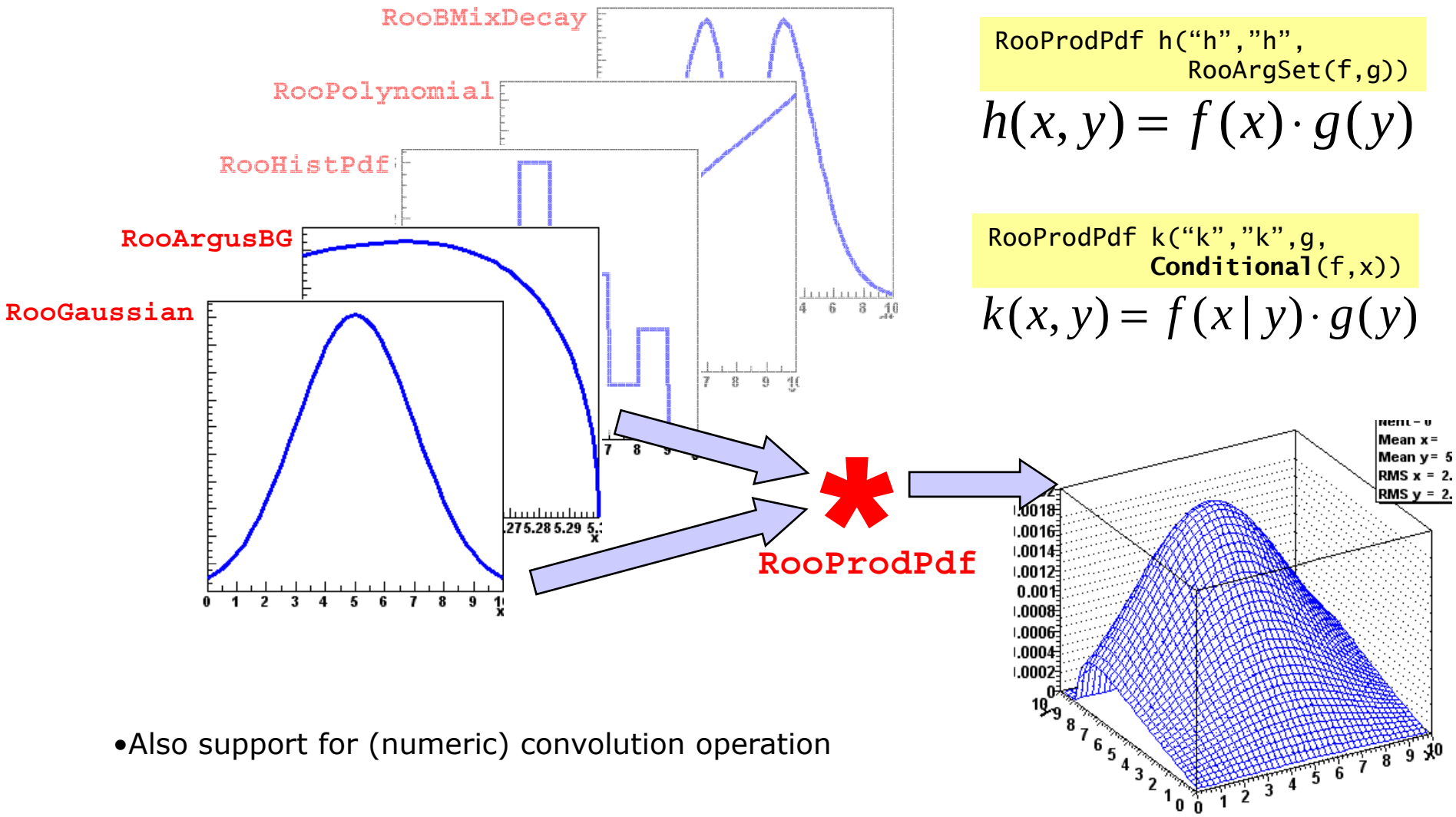
Model building – (Re)using standard components

- Most physics models can be composed from 'basic' shapes



Model building – (Re)using standard components

- Most physics models can be composed from 'basic' shapes




```
RooProdPdf h("h","h",
             RooArgSet(f,g))
```

$$h(x,y) = f(x) \cdot g(y)$$

```
RooProdPdf k("k","k",g,
             Conditional(f,x))
```

$$k(x,y) = f(x \mid y) \cdot g(y)$$

•Also support for (numeric) convolution operation

Model building – (Re)using standard components

- Integral of p.d.f is also p.d.f

$$f_y(x) = \int f(x,y)\,dy$$

```
RooAbsPdf* fy = f->createProjection(y) ;
```

  - RooFit consistently aims to find most efficient way to calculate value, e.g. repeated integration

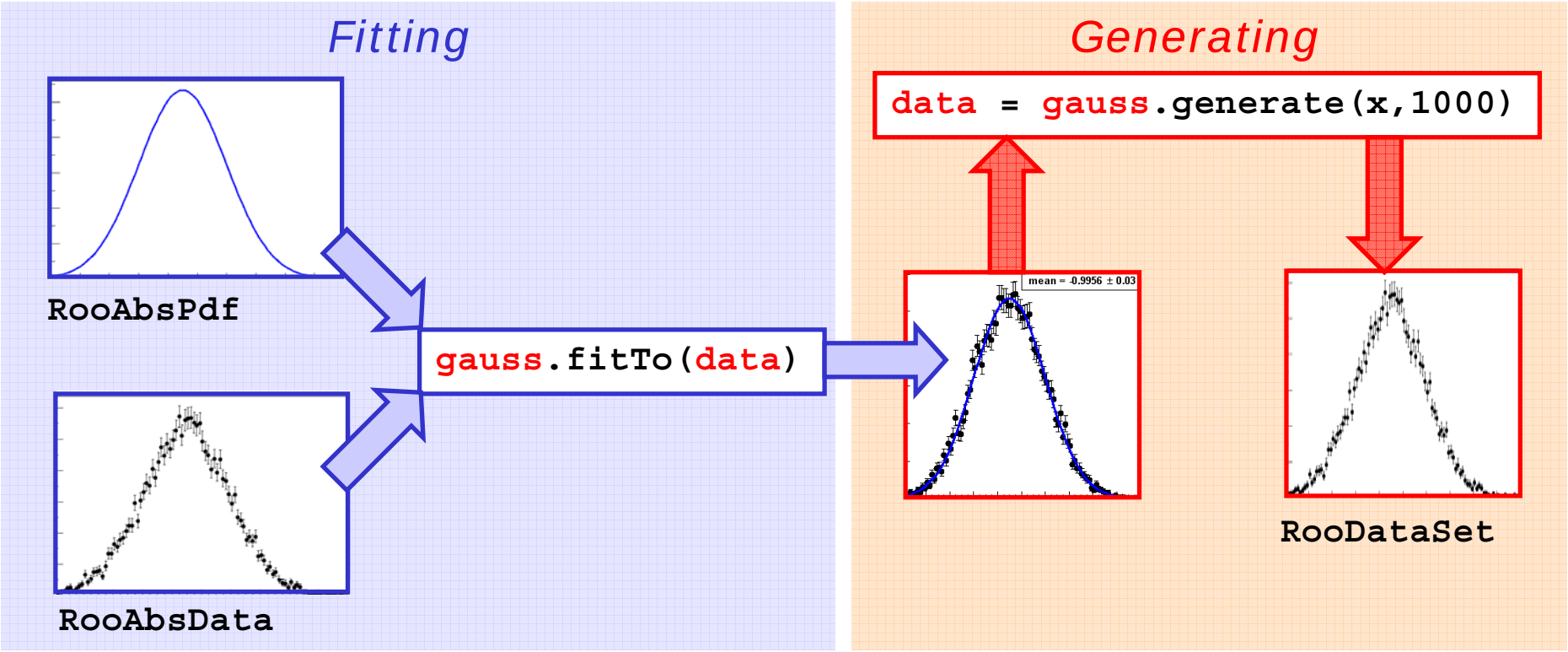
$$\int \left(\int f(x,y,z)\,dy \right) dx = \int\int f(x,y,z)\,dxdy$$

```
RooAbsPdf* fy = f->createProjection(y) ;
RooAbsPdf* fxy = fy->createProjection(x) ;
```

will be evaluated as one 2D integration and not a repeated 1D integration

Using models - Overview

- *All* RooFit models provide *universal and complete fitting* and Toy Monte Carlo *generating* functionality
  - Model complexity only limited by available memory and CPU power
  - Fitting/plotting a 5-D model as easy as using a 1-D model
  - Most operations are one-liners



Fitting functionality

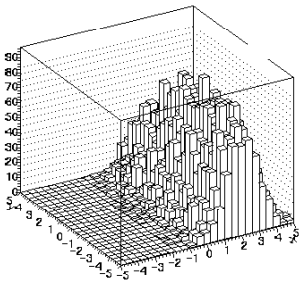
`pdf.fitTo(data)`

- Binned or unbinned ML fit?
  - For RooFit this is the essentially the same!
  - Nice example of C++ abstraction through inheritance

Unbinned

x	y	z
1	3	5
2	4	6
1	3	5
2	4	6

Binned



RooDataSet **RooDataHist**

RooAbsData

- Fitting interface takes abstract RooAbsData object
  - Supply unbinned data object (created from TTree) → unbinned fit
  - Supply histogram object (created from THx) → binned fit

Automated vs. hand-coded optimization of p.d.f.

- Automatic pre-fit PDF optimization
  - Prior to each fit, the PDF is analyzed for possible optimizations
  - Optimization algorithms:
    - Detection and *precalculation of constant terms* in any PDF expression
    - Function *caching* and lazy evaluation
    - *Factorization* of multi-dimensional problems where ever possible
  - Optimizations are always tailored to the specific use in each fit.
  - Possible because OO structure of p.d.f. allows automated analysis of structure

- **No need for users to hard-code optimizations**
  - Keeps your code understandable, maintainable and flexible without sacrificing performance
  - Optimization concepts implemented by RooFit are applied consistently and completely to all PDFs
  - Speedup of factor 3-10 reported in realistic complex fits

- Fit parallelization on multi-CPU hosts
  - Option for automatic parallelization of fit function on multi-CPU hosts (no explicit or implicit support from user PDFs needed)

MC Event generation

```
pdf.generate(vars,nevt)
```

- Sample "toy Monte Carlo" samples from *any* PDF
  - Accept/reject sampling method used by default

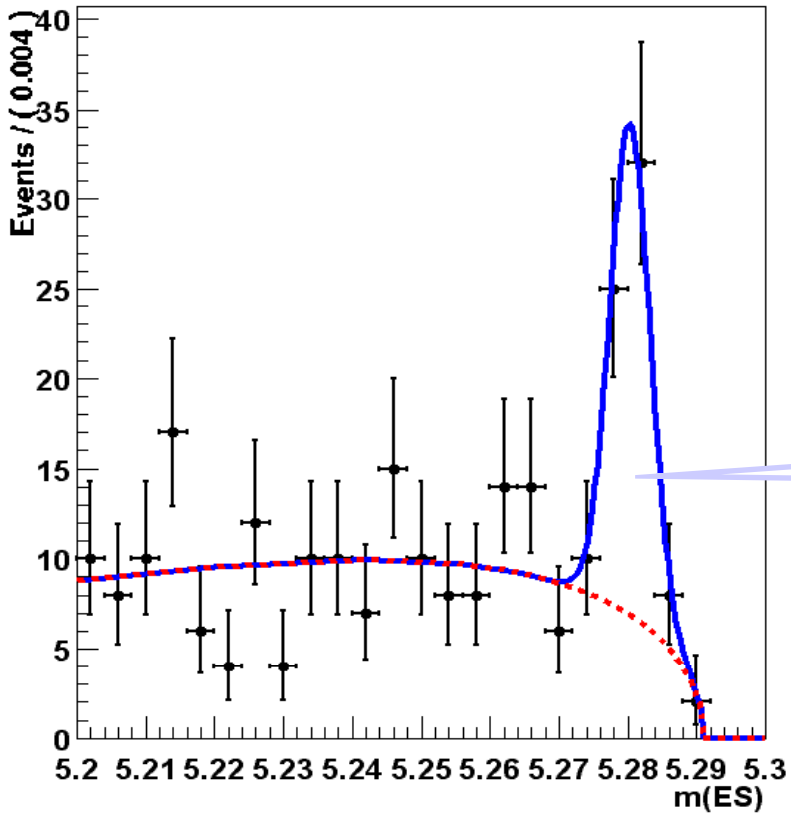
- MC generation method automatically optimized
  - PDF can advertise internal generator if more efficient methods exists (e.g. Gaussian)
  - Each generator request will use the most efficient accept-reject / internal generator combination available
  - Operator PDFs (sum,product,…) distribute generation over components whenever possible
  - Generation order for products with conditional PDFs is sorted out automatically
  - Possible because OO structure of p.d.f. allows automated analysis of structure

- Can also specify template data as input to generator

```
pdf.generate(vars,data)
```

Using models – Plotting

- Model visualization geared towards 'publication plots' not interactive browsing → emphasis on 1-dimensional plots
- Simplest case: plotting a 1-D model over data
- *Modular structure of composite p.d.f.s allows easy access to components for plotting*



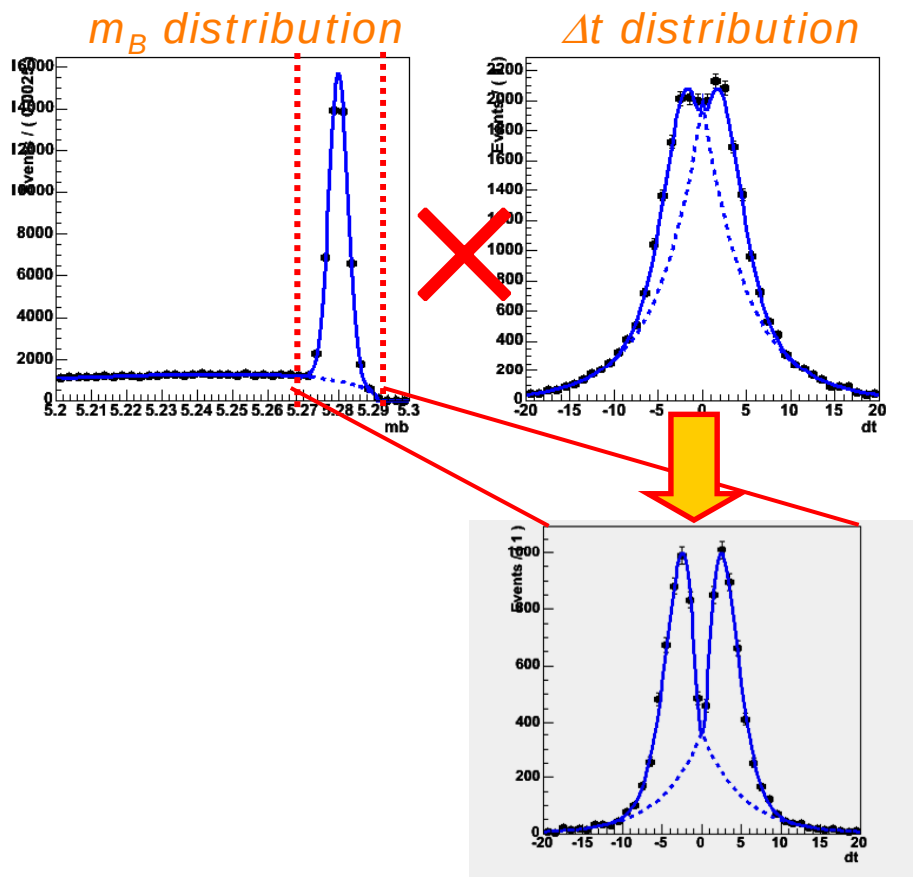

```
RooPlot* frame = mes.frame() ;

data->plotOn(frame) ;
pdf->plotOn(frame) ;
pdf->plotOn(frame,Components("bkg"))
```

Adaptive spacing of curve points to achieve 1‰ precision

Can store plot with data and all curves as single object

Using models – plotting multi-dimensional models

- *Excellent support for slicing/projecting of multi-dimensional models one 1-D plots*
  - **Often cumbersome extra math that is required handled transparantly**



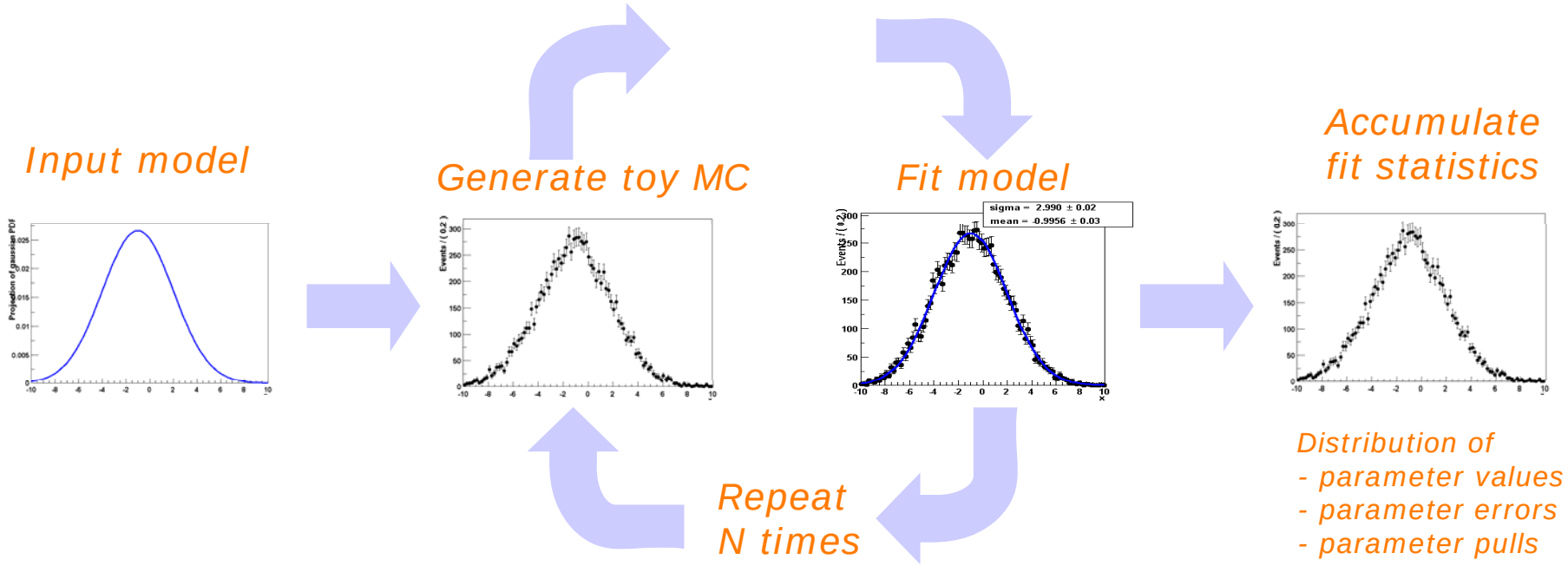

```
RooPlot* frame = dt.frame() ;
data.plotOn(frame) ;
model.plotOn(frame) ;
model.plotOn(frame,Components("bkg")) ;
```

```
RooPlot* frame = dt.frame() ;
dt.setRange("sel",5.27,5.30) ;
data.plotOn(frame,CutRange("sel")) ;
model.plotOn(frame,ProjectionRange("sel"));
model.plotOn(frame,ProjectionRange("sel"),
                    Components("bkg")) ;
```

$$c(t) = \int_{5.27}^{5.30} M(m,t)\,dm$$

Advanced features – Task automation

- Support for routine task automation, e.g fit validation




```
// Instantiate MC study manager
RooMCStudy mgr(inputModel) ;

// Generate and fit 100 samples of 1000 events
mgr.generateAndFit(100,1000) ;

// Plot distribution of sigma parameter
mgr.plotParam(sigma)->Draw()
```

RooFit – Plans for 2007

- Migrate code to ROOT CVS repository
  - Source code now maintained in ROOT CVS repository as of June 2007 (used to send tarballs) Easier to make small changes
- Improve MathMore/RooFit interoperability
  - E.g replace private implementations of GSL numeric integration classes and by interfaces to MathMore versions
  - Generic templated RooAbsReal/RooAbsPdf adapter function for MathMore functions and p.d.f.s
- Update and complete Users Manual

Tools for significance, limits, p-values goodness-of-fit

Wouter Verkerke, NIKHEF

Tools for significance, limits, confidence intervals

- Several tools available in ROOT
- Class TRolke (Conrad,Rolke)
  - Computes confidence intervals for the rate of a Poisson in the presence of background and efficiency
  - Fully frequentist treatment of the uncertainties in the efficiency and background estimate using the profile likelihood method.
  - The signal is always assumed to be Poisson.
  - Seven different models included.
    Background model options Poisson/Gaussian/known,
    Efficiency model options Binomial/Gaussian/known
  - Just #signal/bkg (does not include discriminating variables)

```
Root> TRolke g;
Root> g.SetCL(0.90);
Root> Double_t ul = g.CalculateInterval(x,y,z,bm,em,e,mid,sde,sdb,tau,b,m);
Root> Double_t ll = g.GetLowerLimit();
```

Tools for significance, limits, confidence intervals

- Class TFeldmanCousins (Bevan)
  - Fully frequentist construction as in PRD V57 #7, p3873-3889.
  - Not capable of treating uncertainties in nuisance parameters
  - For cases with no or negligible uncertainties .
  - Just #signal/bkg (does not include discriminating variables)

- Class TLimit (based on code by Tom Junk)
  - Algorithm to compute 95% C.L. limits using the Likelihood ratio semi-Bayesian method.
  - It takes signal, background and data histograms as input
  - Runs a set of Monte Carlo experiments in order to compute the limits.
  - If needed, inputs are fluctuated according to systematics.
  - Rewrite of original `mclimit` Fortan code by Tom Junk
  - Works with distributions in discriminating variables (histograms)

Tools for significance, limits, confidence intervals

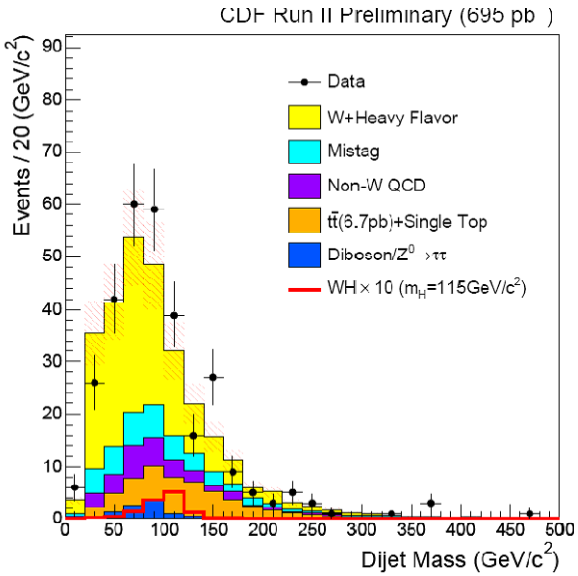
- Example use of class `TLimit`

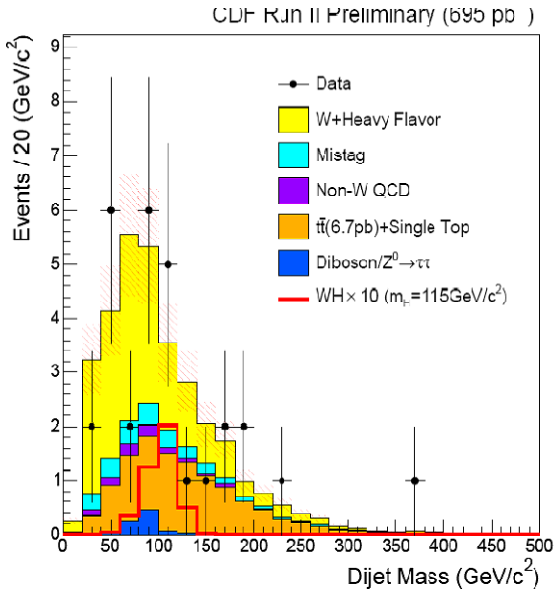
```
TLimitDataSource* mydatasource = new TLimitDataSource(hsig,hbkg,hdata);
TConfidenceLevel *myconfidence = TLimit::ComputeLimit(mydatasource,50000);
cout << " CLs     : " << myconfidence->CLs() << endl;
cout << " CLsb    : " << myconfidence->CLsb() << endl;
cout << " CLb     : " << myconfidence->CLb() << endl;
cout << "< CLs >  : " << myconfidence->GetExpectedCLs_b() << endl;
cout << "< CLsb > : " << myconfidence->GetExpectedCLsb_b() << endl;
cout << "< CLb >  : " << myconfidence->GetExpectedCLb_b() << endl;
```

- TLimit code, like original mclimit.f code does not take systematic on shape of inputs into account
  - New version of limit calculator written that can handle shape systematics as well `mclimit_cms.C`

csm_mclimit: Limit Calculator with shape systematics

- Tool to calculate limits for complex realistic analyses
  - Example Standard Model Higgs Boson Search with CDF





  - Two disjoint event samples, “single-tagged”and “double-tagged”
  - Nuisance parameters affect both in a correlated way: Luminosity, background rates, Jet energy scale and resolution, signal detection efficiency.
  - New physics prediction adds incoherently to the H0 prediction

csm_mclimit: Limit Calculator with shape systematics

- Problems addressed and what it does
  - Input data are binned in one or more histograms (1D or 2D).
  - Can have multiple signal and multiple background sources.
  - Model predictions are sums of template histograms from different sources.
  - Each source of signal and background can have rate and shape uncertainties from multiple sources of systematic uncertainty.
    - Systematic uncertainties are listed by name, and uncertainties with the same name are 100% correlated and uncertainties with different names are 0% correlated.
  - Shape uncertainties are handled by template morphing or by simple linear interpolation within each bin.
    - (Linear interpolation of histograms. Alexander L. Read (Oslo U.) . 1999.NIM.A425:357-360,1999.)
  - Uncertain parameters can be fit for, in the data and for each pseudo-experiment.
  - Finite MC statistical errors in each bin are included in the fits and the pseudo-experiments.

Limit Calculator with shape systematics

- Output
  - Given the data and model predictions p-values are computed to possibly discover or exclude.
  - CLs and CLs-based cross-section limits can be computed
  - Bayesian limits using a flat prior in cross section can be computed.

- Software and documentation provided at
  - http://www.hep.uiuc.edu/home/trj/cdfstats/mclimit_csm1/index.html
  - *"Interface is rather cumbersome due to the large number of possible systematic errors which need to be included"*
    - Tom is interested in further interfacing in ROOT / RooFit

LepStats4LHC – LEP-style Frequentist limit calculations

- Series of tools for frequentist limit calculations (K. Cranmer)
  - **Implement LEP-style calculation of significances (in C++)**
  - Uses FFTW package for convolution calculations
  - Interface is series of command line utilities
  - Code & manual available for download at `phystat.org`

- `PoissonSig`
  - Calculate the significance of a number counting analysis.
- `PoissonSig_syst`
  - Calculate the significance of a number counting analysis including systematic error on the background expectation.
- `Likelihood`
  - Calculate the combined significance of several search channels or to calculate the significance of a search channel with a discriminating variable.
- `Likelihood_syst`
  - Calculate the combined significance of several search channels including systematic errors associated with each channel.

MadTools – W. Quayle

- MadTools/StatTools package implement several algorithms to calculate limits, intervals
  - Code available at `www-wisconsin.cern.ch/physics/software.html`
  - Public version provides easy-to-use command line tools for calculation of limits, intervals

Goals and Status

- Aim to provide a unified interface to a variety of statistical treatments
  - Scripts that use statistical information can then be written in a way that is independent of the formulation
- Already implemented and public:
  - Toy MC computation similar to TLimit
  - FFT-based algorithms similar to CLFFT
- Currently under development:
  - Toy MC computation for fit-based searches (compatible with hand-coded direct calls to Minuit as well as RooFit)
- Not started yet, but under consideratic
  - Interface to "CL_s with fits" limit calculator f Tom Junk

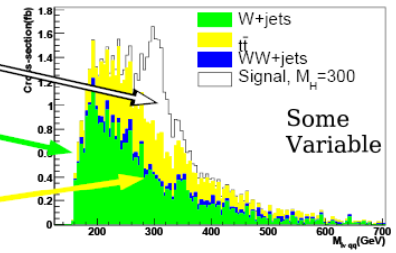
W. Quayle Page 3 Root 2007 Workshop, 28

Programming Interface (2)

Signal corresponds to one TDistribution

Background 1 corresponds to another TDistribution

Background 2 is another one



- A THypothesis represents the expectations for several processes in some region of phase space
  - Signal and background are labelled as such, so a THypothesis is used, e.g., to generate toy MC in S+B and BG-only hypothesis

W. Quayle Page 6 Root 2007 Workshop, 28 March 2007

HEF

Other statistics tools – sPlots

- sPlots – Presented at 2005 PhyStat
  - Tool for analysis of multidimensional likelihood models
  - Tool for background subtraction
  - Implemented in ROOT in class `TSplot`

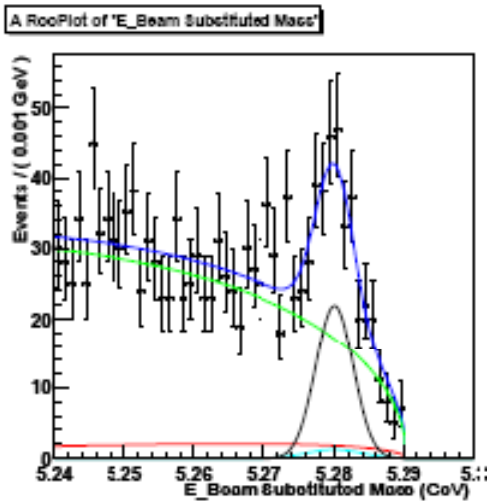
$$\ln \mathcal{L} = \sum_{e=1}^{N} \ln \left\{ \sum_{i=1}^{N_s} N_i \, PDF_i(y_e) \right\} - \sum_{i=1}^{N_s} N_i$$

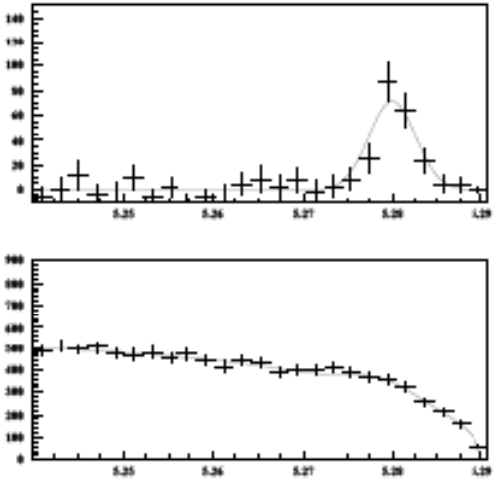
$$\mathcal{P}_n(y_e) = \frac{N_n PDF_n(y_e)}{\sum_{j=1}^{N_s} N_k PDF_j(y_e)} \quad \Rightarrow \quad {}_s\mathcal{P}_n(y_e) = \frac{\sum_{j=1}^{N_s} V_{nj} PDF_j(y_e)}{\sum_{j=1}^{N_s} N_k \, PDF_j(y_e)}$$

'regular' sig / bkg weights

sig / bkg sWeights including cov. matrix

Data with fit (in 3 D)





sPlot signal

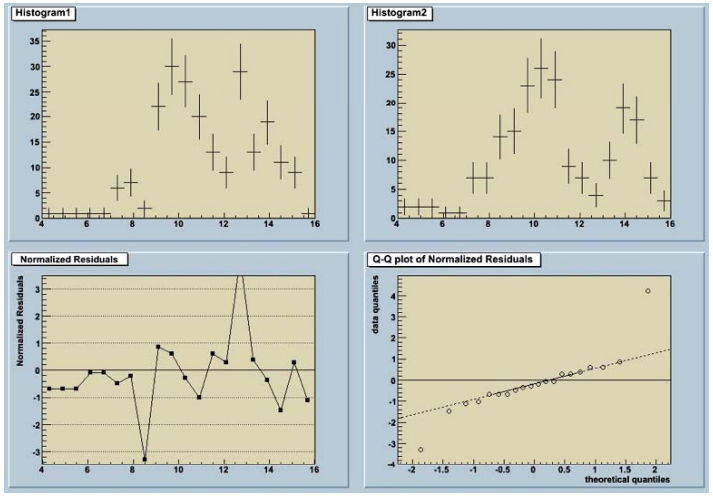
sPlot background

Tools for Goodness-of-Fit tests

- One-dimensional goodness-of-fit tests integrated in standard ROOT classes

- Chi2 comparison in `TH1::Chi2Test`
  - Algorithm from *N. Gagunashvili* as presented at PhyStat2005 and implemented in C++ by *D. Haertl*

- Kolmogorov tests in `TH1::KolmogorovTest`



- Also statistical toolkit with GOF tests
  - http://www.ge.infn.it/statisticaltoolkit/
  - Presented at PhyStat 2005

A Common Framework for statistical tools?

- Both LEP and Tevatron experiments have created tools that combine multiple channels and include systematic uncertainties, but
  - The tools generally implement a specific technique,
  - Combinations require significant manual intervention
- Would like a more versatile, generic solution
- 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

A common framework for statistical tools

- What techniques should such a framework implement?

- 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 (Kyle Cranmer)

- 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 λP method and CMS used Z_N, then they could "discover" with 56% less data!
- a bad situation for ATLAS
- a bad situation for HEP

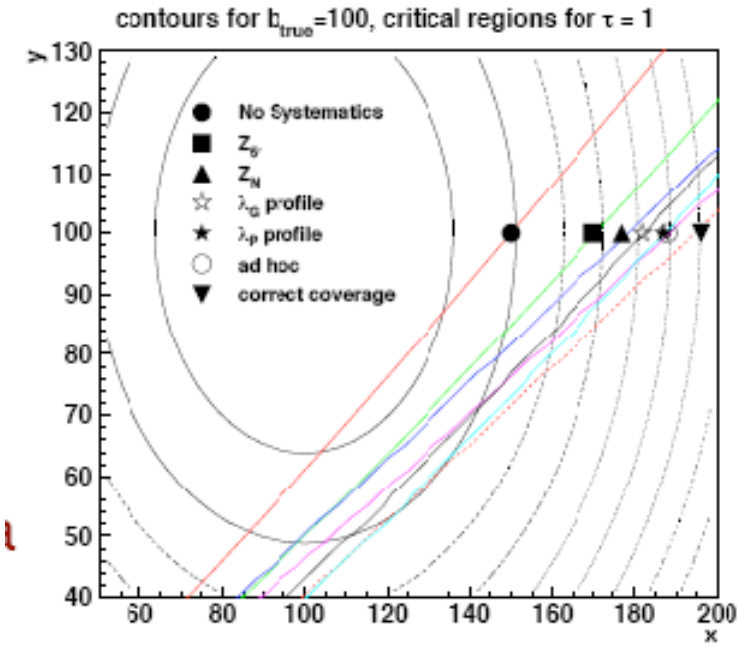
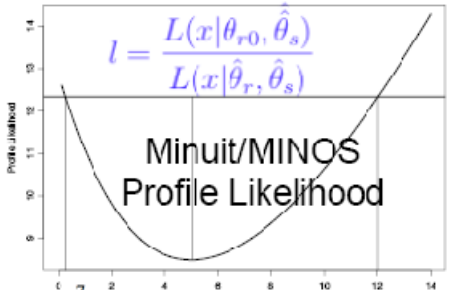


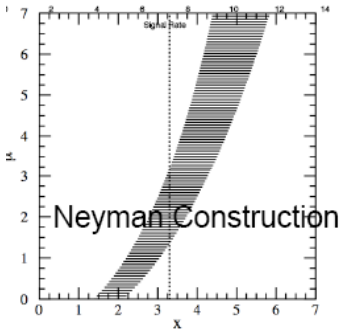
Figure 7. A comparison of the various methods critical boundary $x_{crit}(y)$ (see text). The concentric ovals represent contours of L_G from Eq. 15.

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

A common framework – Common tools

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

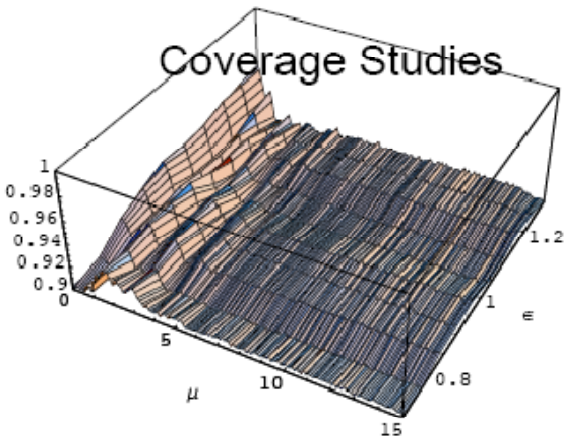
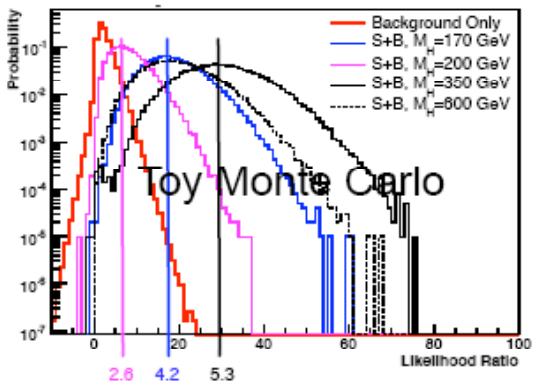


Fig. 2. Coverage plot for Unified limits, Gaussian uncertainty, $b = 3, \sigma = 0.1$.



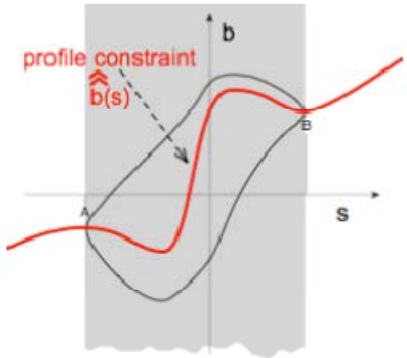


Figure 7.2: MINOS error confidence region for parameter 1

Common tools – Organization

- There appears to be interest in common statistical tools by ATLAS and CMS
- Initiative by Rene & Kyle to organize suite of common tools in ROOT
  - Propose to build tools on top RooFit following survey of existing software and user community
  - RooFit provides solutions for most of the hard problems → Would not like to reinvent wheel. It has been around of a while and has a fair share of users
  - No static notion of parameters vs observables in core code → RooFit models can be used for both Bayesian and Frequentists techniques
  - Idea to have few core developers maintaining the framework and have mechanism for users/collaborations to contribute concrete tools
- Working name: the RooStats project
- Kyle has done a couple of pilot studies to see how easy it is implement some example problems in RooFit
  - PhyStat 2005 example study $\boxed{L_P(x, y|\mu, b) = Pois(x|\mu + b) \cdot Pois(y|\tau b).}$
  - Higgs combined channel sensitivity

RooStats Pilot Study – The prototype problem in RooFit

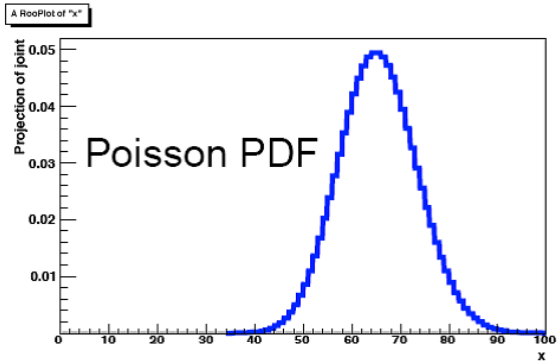
- Consider this prototype problem for new physics searches:

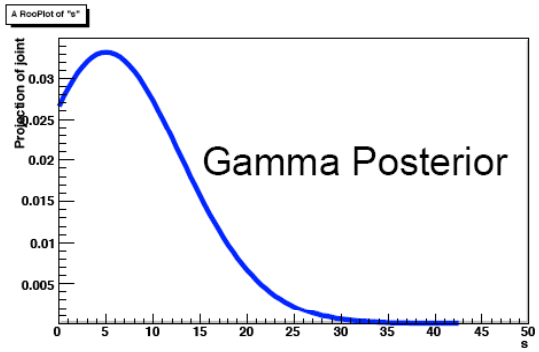
$$L_P(x, y|\mu, b) = Pois(x|\mu + b) \cdot Pois(y|\tau b).$$

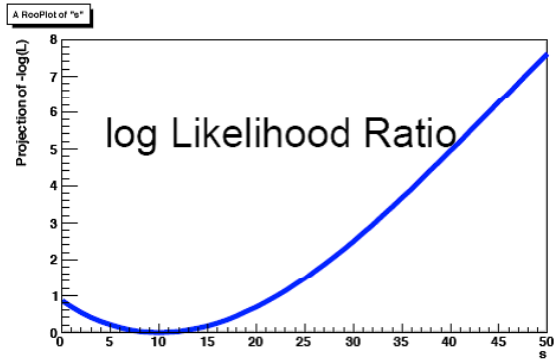
- Easy to code up in RooFit

```
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:

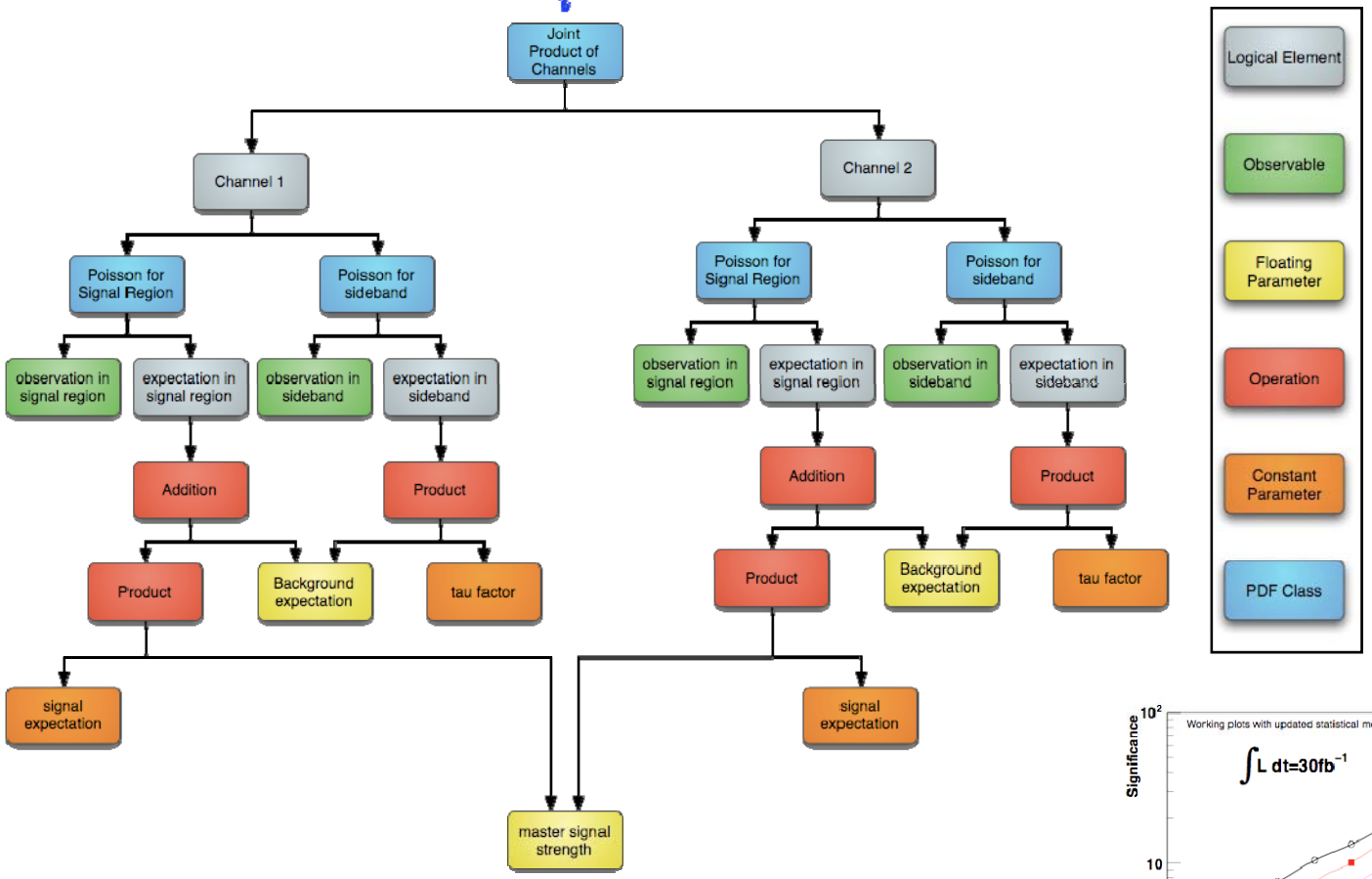


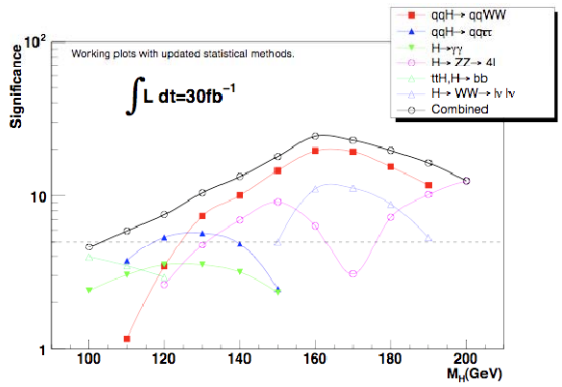




RooStats Pilot Study – Higgs Multi-channel sensitivity study

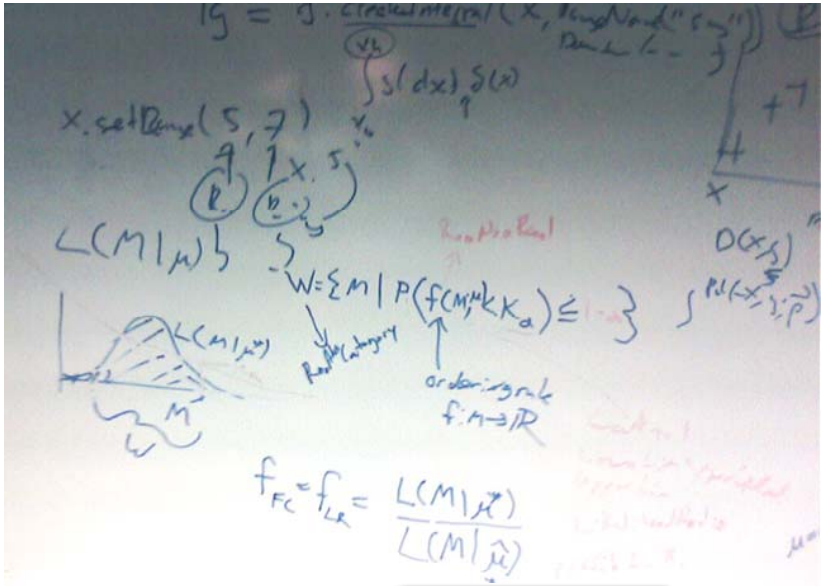
$$L(\mathbf{x},\mathbf{y}|\mu,\mathbf{s},\mathbf{b}) = \prod_i Pois(x_i|\mu s_i + b_i)\, Pois(y_i|\tau b_i)$$

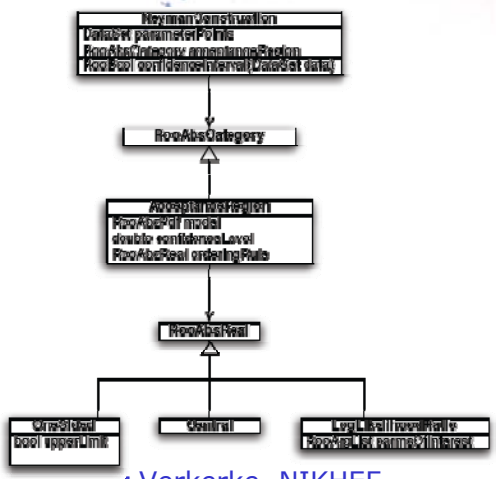




Designing the framework

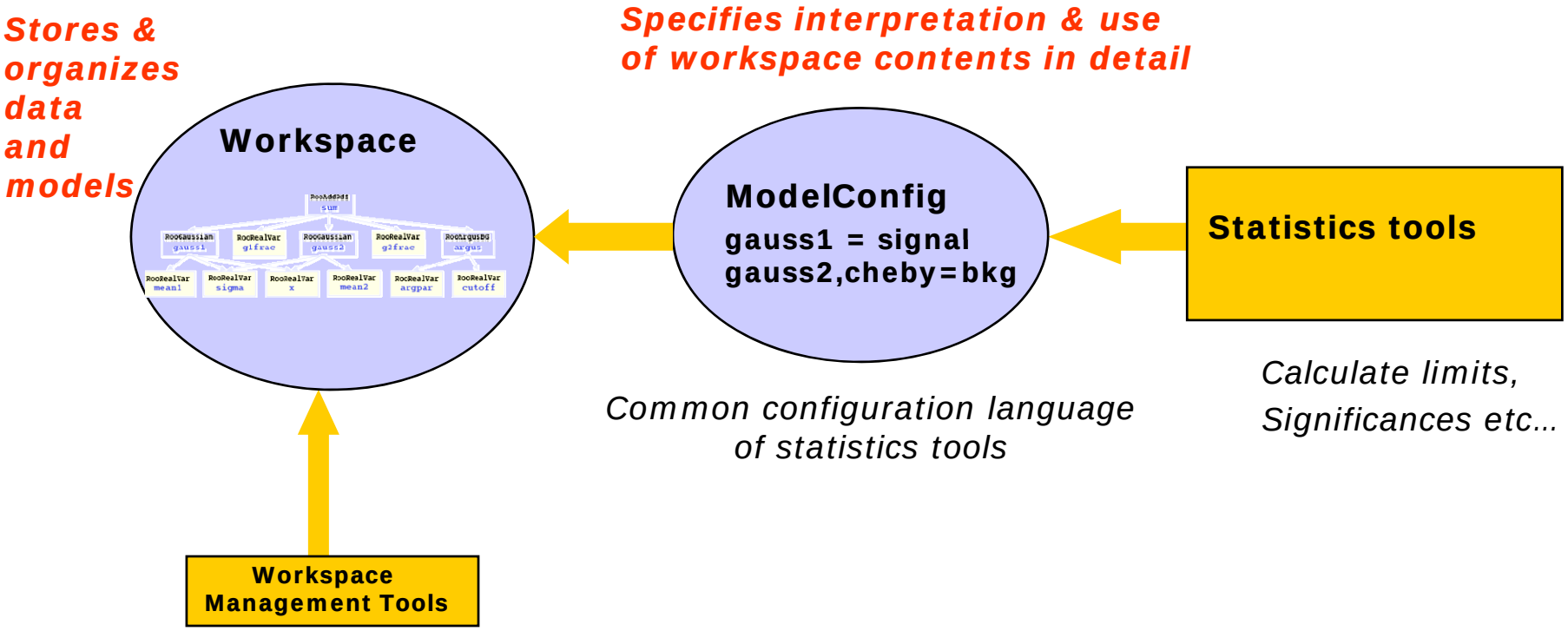
- Kyle & I met a couple months ago to discuss how to implement a few statistical concepts on top of RooFit
  - want class structure to maps onto statistical concepts
  - Successfully worked out a few of the methods
- The first examples were
  - Bayesian Posterior
  - Profile likelihood ratio
  - Acceptance Regio
  - Ordering Rule
  - Neyman Construction
  - Confidence Interval
- Many concepts already have an appropriate class in RooFit



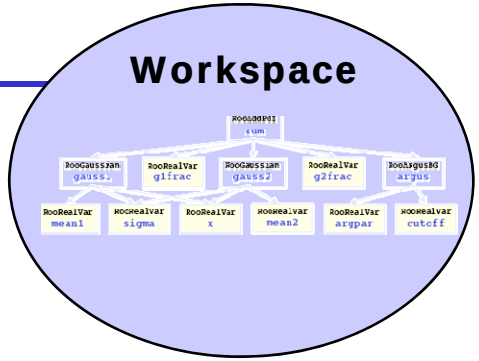


Framework design & RooFit adaptations

- Have had more meetings last 3 months to review RooFit lessons from BaBar
  - Kyle, Amir Farbin (ex-Babar), Frank Wrinklmeyer (ex-Babar), WV
  - Design for *WorkSpace* and *ModelConfig* concept in RooFit to interface with statistics tools



The Workspace as publication



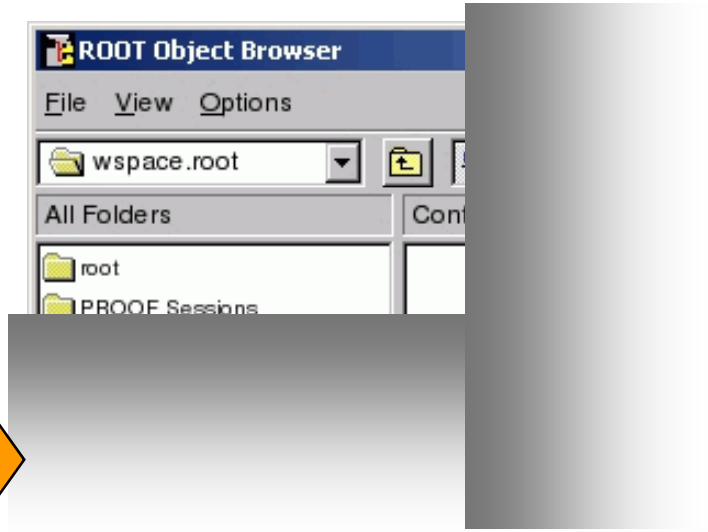
- Now have functional `RooWorkspace` class that can contain
  - Probability density functions and its components
  - (Multiple) Datasets
  - Supporting interpretation information (`RooModelConfig`)
  - Can be stored in file with regular ROOT persistence

- **Ultimate publication of analysis…**
  - Full likelihood available for Bayesian analysis
  - Probability density function available for Frequentist analysis
  - Information can be easily extracted, combined etc…
  - Common format for sharing, combining of various physics results

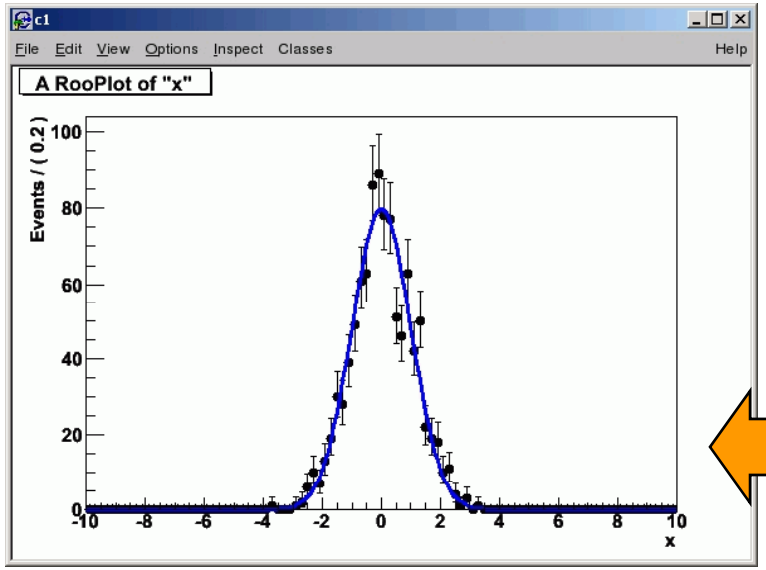
The Workspace in practice

- Creating & reading a workspace

```
// Create ws, import p.d.f and data
RooWorkspace ws("ws") ;
ws.addPdf(g) ;
ws.addData(d) ;

// Save workspace to file
TFile f("higgs_analysis.root","RECREATE") ;
ws.Write("AtlasAnalysis") ;
```



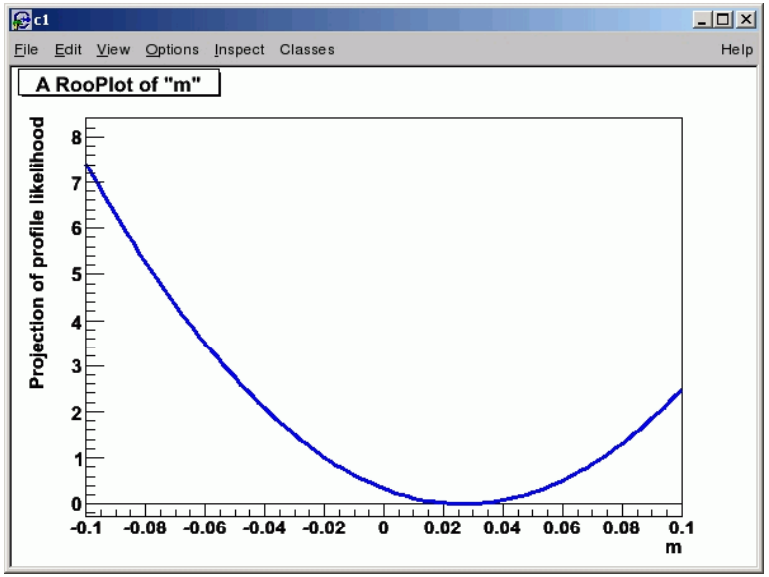

```
// Read workspace from file
TFile f("higgs_analysis.root") ;
RooWorkspace* ws = f.Get("AtlasAnalysis") ;

// Plot data and p.d.f.
RooPlot* frame = ws->var("x")->frame() ;
ws->data("d")->plotOn(frame) ;
ws->pdf("g")->plotOn(frame) ;
frame->Draw() ;
```



The Workspace in practice

- Profile likelihood from workspace

```
// Read workspace from file
TFile f("higgs_analysis.root") ;
RooWorkspace* ws = f.Get("AtlasAnalysis") ;

// Construct a profile likelihood object
RooNLLVar nll("nll","nll",*ws->pdf("g"),*ws >data("d")) ;
RooProfileLL pnll("pnll","pnll",nll,ws->var("s")) ;

// Plot profile likelihood
RooPlot* frame = ws->var("m")->frame() ;
pnll.plotOn(frame) ;
frame->Draw() ;
```



  - **NB: *Same* 7 lines of code works construct profiles for models of arbitrary complexity (e.g. Multichannel higgs)**

RooStats Framework – Status & plans

- Initial `RooWorkspace` class with persistence feature now ready, will available in next ROOT release

- Now working on design of `RooModelConfig` layer
  - Store details on specific use of model for a given problem, i.e. what is signal background etc…
  - Intended to solve 'cumbersome interface problem' of tools that require many input parameters (as in Tom Junks csm_mclimit.C)
  - Common language of various statistics tools

- Next step: add higher level tools to framework that calculate limits, confidence intervals from information in workspaces

  RooStats::BaysianInterval,
  RooStats::NeymanConstruction

RooStats Framework – Status & plans

- Design of `RooModelConfig` layer (cont'd)
  - Aim is for statistics tools to have single constructor argument
  - RooStats::BaysianInterval(modelConfig) ;
    RooStats::NeymanConstruction(modelConfig) ;
    RooStats::ProfileLikelihood(modelConfig) ;

    RooStats::SPlot(modelConfig) ;
  - Will facilitate TMVA-style comparison of methods

- Aiming for first functional release of RooStats in Fall 2007
  - Workspace & management tools
  - ModelConfig 'common' language for statistics tools
  - Initial sets of statistics tools

Summary

Summary

- Enormous growth in statistics software tools for/by HEP in the past years
- ROOT platform of choice for data analysis
  - Lots of basic infrastructure
  - Incorporates or interfaces many external software packages useful for statistical analysis
- Packages like TMVA and StatPatternRecognition bring many advanced Multivariate Analysis techniques to HEP
  - Can now get started with e.g. Boosted Decision Trees in 1 day
  - Will promote use and acceptance of MVA techniques in HEP
- Fitting, Minimization & Error analysis
  - MINUIT remains tool of choice for error analysis & minimization. ROOT Fitter interface being redesigned for enhanced modularity & flexibility
  - RooFit toolkit for data modeling exists since 7 years, used in >50 publications (Mostly B physics)

Summary

- Several LEP/Tevatron tools exist for calculation of limits, intervals in Bayesian or Frequentist methodology
  - Ongoing RooStats effort to organize limit calculation tools in common framework.
  - Offers persistence of p.d.f & data in Workspace as ultimate publishing tool. Aiming TMVA style comparison of methodologies

- Software landscape is evolving rapidly
  - 10 years ago most of us were working in Fortran and were writing our own tools
  - Meanwhile ROOT / OO programming has delivered enormous progress in software modularity, interoperability
  - 2008 – Common tools for statistical analysis. Publishing of likelihood in electronic form? Technically feasible…