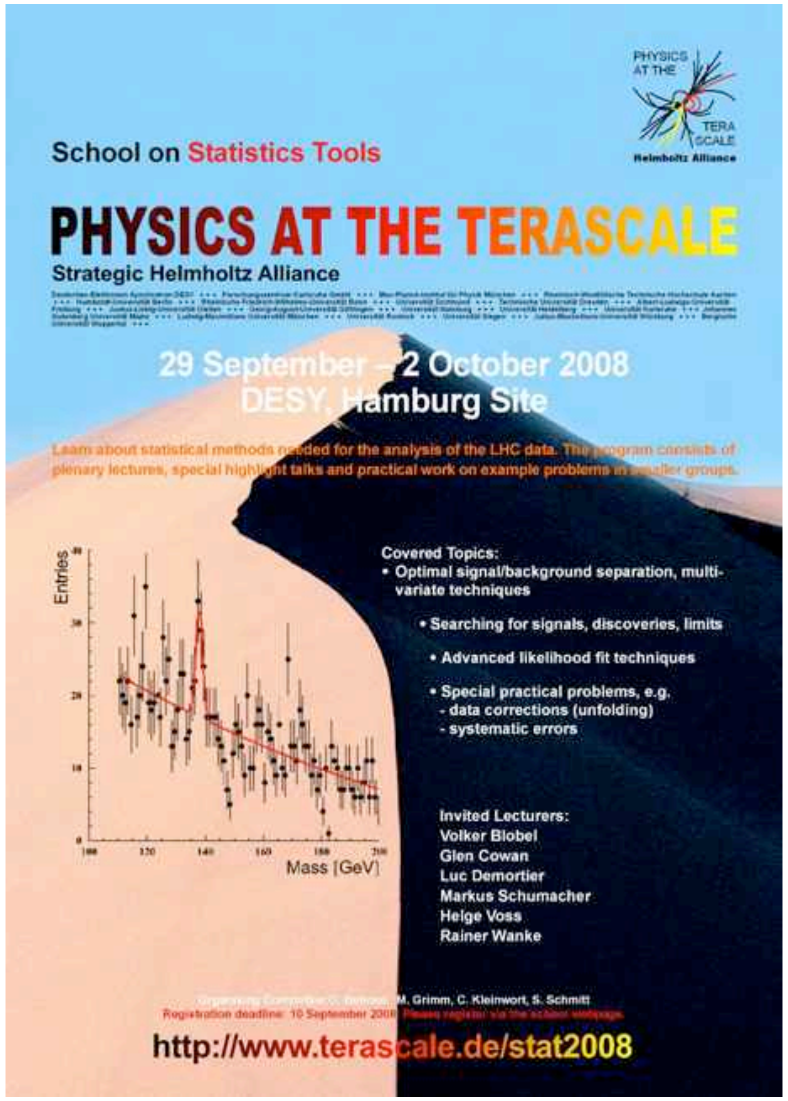
Recent Developments of ROOT Statistical Software

Lorenzo Moneta

(CERN, PH-SFT)



PHYSICS AT THE TERA SCALE Helmholtz Alliance

School on Statistics Tools

PHYSICS AT THE TERASCALE

Strategic Helmholtz Alliance

Deutsches Elektronen-Synchrotron DESY +++ Forschungszentrum Karlsruhe GmbH +++ Max-Planck-Institut für Physik München +++ Rheinisch-Westfälische Technische Hochschule Aachen +++ Humboldt-Universität Berlin +++ Rheinische Friedrich-Wilhelms-Universität Bonn +++ Universität Dortmund +++ Technische Universität Dresden +++ Albert-Ludwigs-Universität Freiburg +++ Justus-Liebig-Universität Gießen +++ Georg-August-Universität Göttingen +++ Universität Hamburg +++ Universität Heidelberg +++ Universität Karlsruhe +++ Johannes Gutenberg Universität Mainz +++ Ludwig-Maximilians-Universität München +++ Universität Rostock +++ Universität Siegen +++ Julius-Maximilians-Universität Würzburg +++ Bergische Universität Wuppertal +++

29 September – 2 October 2008
DESY, Hamburg Site

Learn about statistical methods needed for the analysis of the LHC data. The program consists of plenary lectures, special highlight talks and practical work on example problems in smaller groups.

Covered Topics:

- Optimal signal/background separation, multivariate techniques
- Searching for signals, discoveries, limits
- Advanced likelihood fit techniques
- Special practical problems, e.g.
  - data corrections (unfolding)
  - systematic errors

Invited Lecturers:
Volker Blobel
Glen Cowan
Luc Demortier
Markus Schumacher
Helge Voss
Rainer Wanke

Organizing Committee: O. Behnke, M. Grimm, C. Kleinwort, S. Schmitt
Registration deadline: 10 September 2008 Please register via the school webpage.

http://www.terascale.de/stat2008

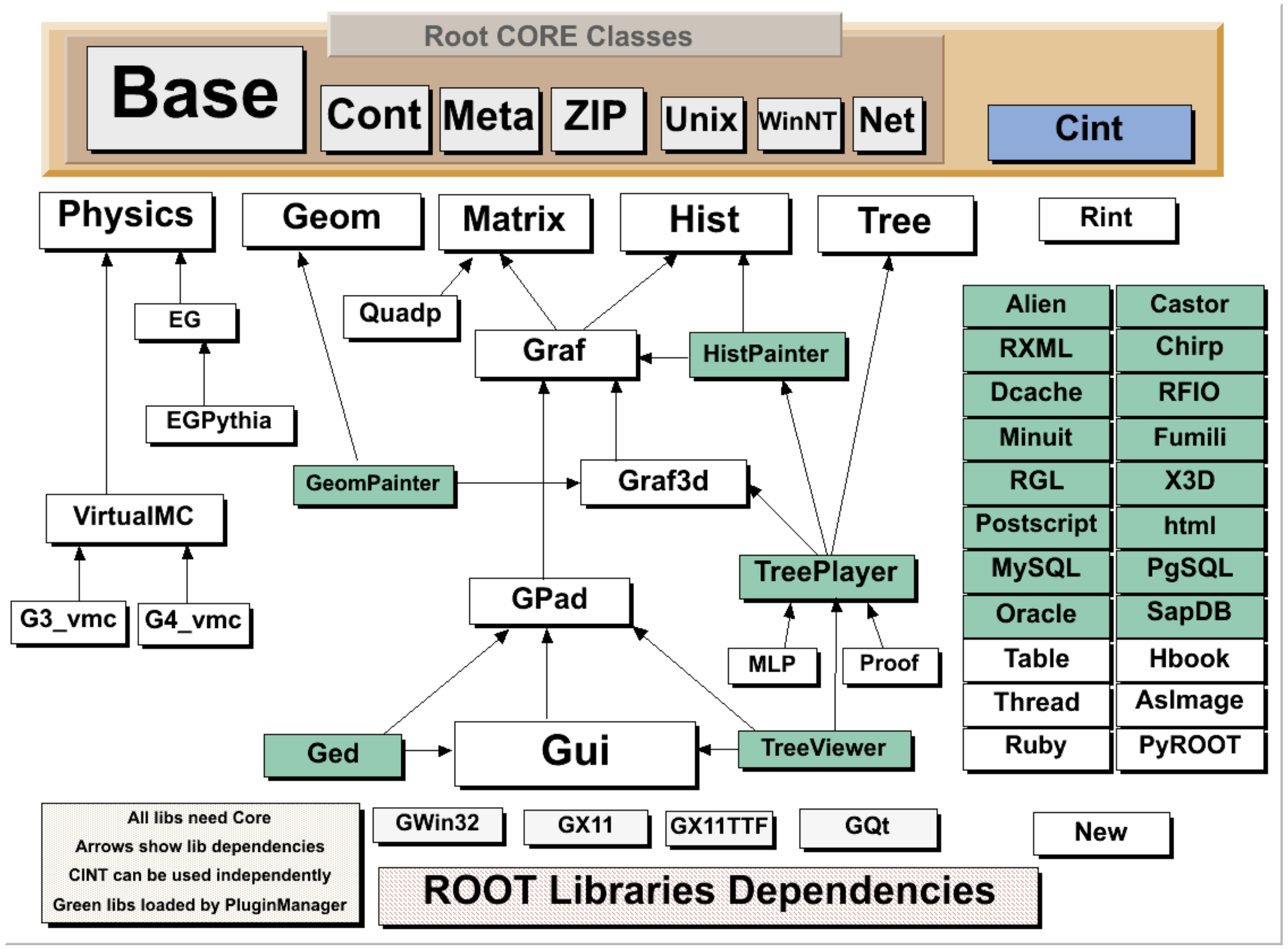
ROOT

An Object-Oriented
Data Analysis Framework

ROOT Math/Stat Work Package:
R. Brun, D. Gonzalez-Malin, A. Kreshuk, L.M., E. Offermann
W. Verkerke(RooFit), K. Cranmer (RooStat), TMVA team and many other contributors

ROOT

- *ROOT* is a large Object-Oriented data handling and analysis framework
  - Efficient object store scaling
  - C++ interpreter
  - Extended 2D+3D scientific data visualization capabilities
  - Extensive set of multi-dimensional histograms, data fitting, modeling and analysis methods
  - Complete set of GUI widgets
  - Classes for threading, shared memory, networking, etc.
  - Parallel version of analysis engine runs on clusters and multi-core
  - Fully cross platform, Unix/Linux, MacOS X and Windows



Root CORE Classes

Base, Cont, Meta, ZIP, Unix, WinNT, Net, Cint

Physics, Geom, Matrix, Hist, Tree, Rint

EG, EGPythia, Quadp, Graf, HistPainter, GeomPainter, Graf3d, VirtualMC, G3_vmc, G4_vmc, TreePlayer, GPad, MLP, Proof, Ged, Gui, TreeViewer

Alien, Castor, RXML, Chirp, Dcache, RFIO, Minuit, Fumili, RGL, X3D, Postscript, html, MySQL, PgSQL, Oracle, SapDB, Table, Hbook, Thread, AsImage, Ruby, PyROOT

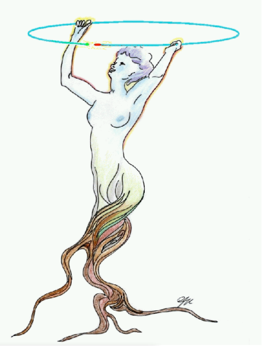
GWin32, GX11, GX11TTF, GQt, New

All libs need Core
Arrows show lib dependencies
CINT can be used independently
Green libs loaded by PluginManager

ROOT Libraries Dependencies

- 1,700,000 lines of code
- more than 100 shared libraries
- more than 500000 downloads (since 1997)

Outline

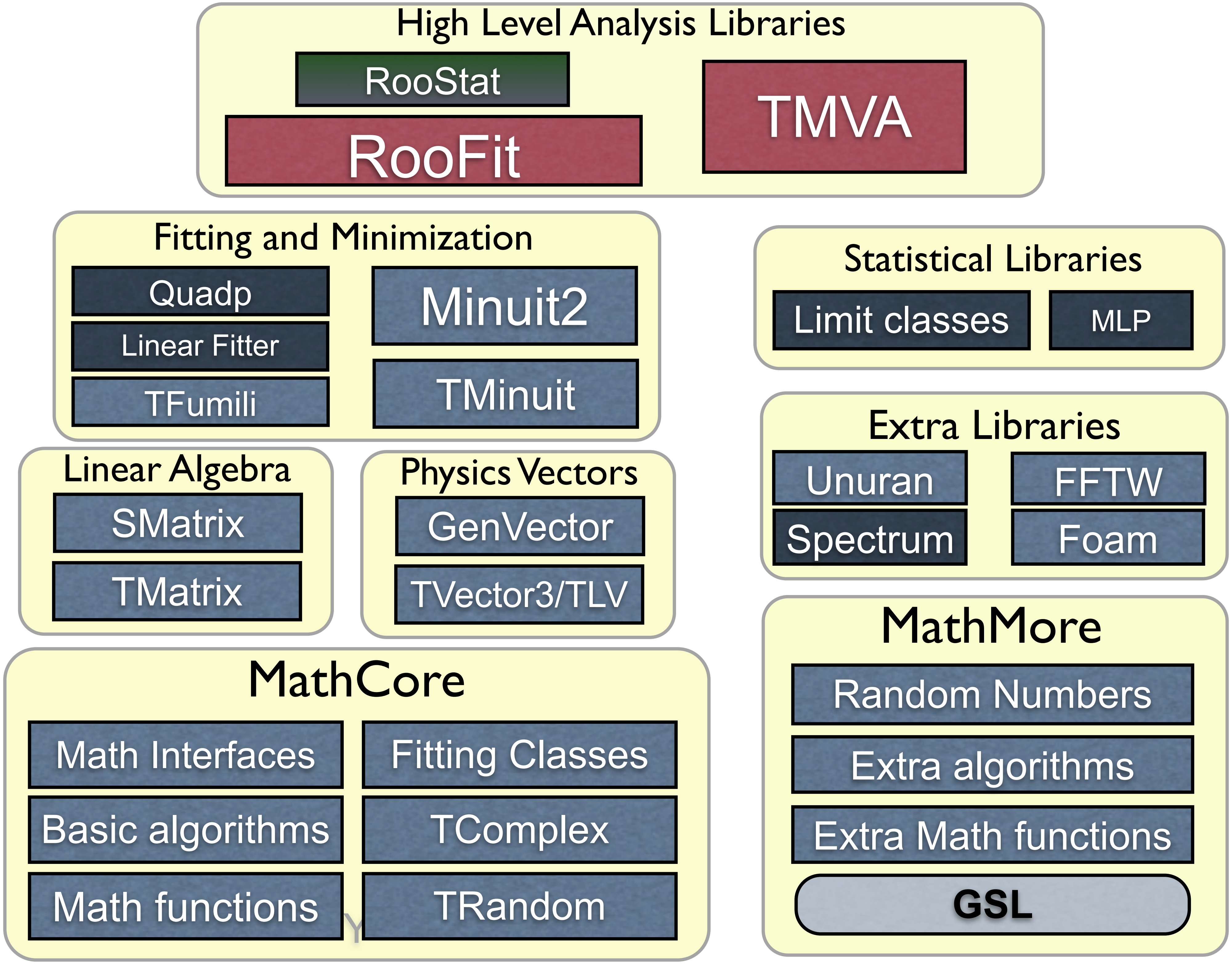


- ROOT Statistical classes
- Recent developments in
  - mathematical and statistical functions
  - random numbers
  - data analysis classes and their visualization
  - fitting
  - minimization
  - *RooFit*
  - smoothing (non-parametric regression)
  - multi-variate methods
  - confidence levels (limits settings)
- Conclusions
- Documentation

Library Organization

- Recent re-organization of mathematical and statistical libraries
    - more modular libraries
        - libraries as *MathCore* will provide the basic functionality
    - reduce dependency between libraries
    - make easier the integration of contributed software
    - easier maintainability in the long term
- Review and revise some of existing algorithms
    - remove duplications and correct and improve them
- Better documentation (more examples and tutorials)

Structure of ROOT Math/Stat Libraries



High Level Analysis Libraries

RooStat

RooFit

TMVA

Fitting and Minimization

Quadp

Linear Fitter

TFumili

Minuit2

TMinuit

Statistical Libraries

Limit classes

MLP

Extra Libraries

Unuran

FFTW

Spectrum

Foam

Linear Algebra

SMatrix

TMatrix

Physics Vectors

GenVector

TVector3/TLV

MathCore

Math Interfaces

Fitting Classes

Basic algorithms

TComplex

Math functions

TRandom

MathMore

Random Numbers

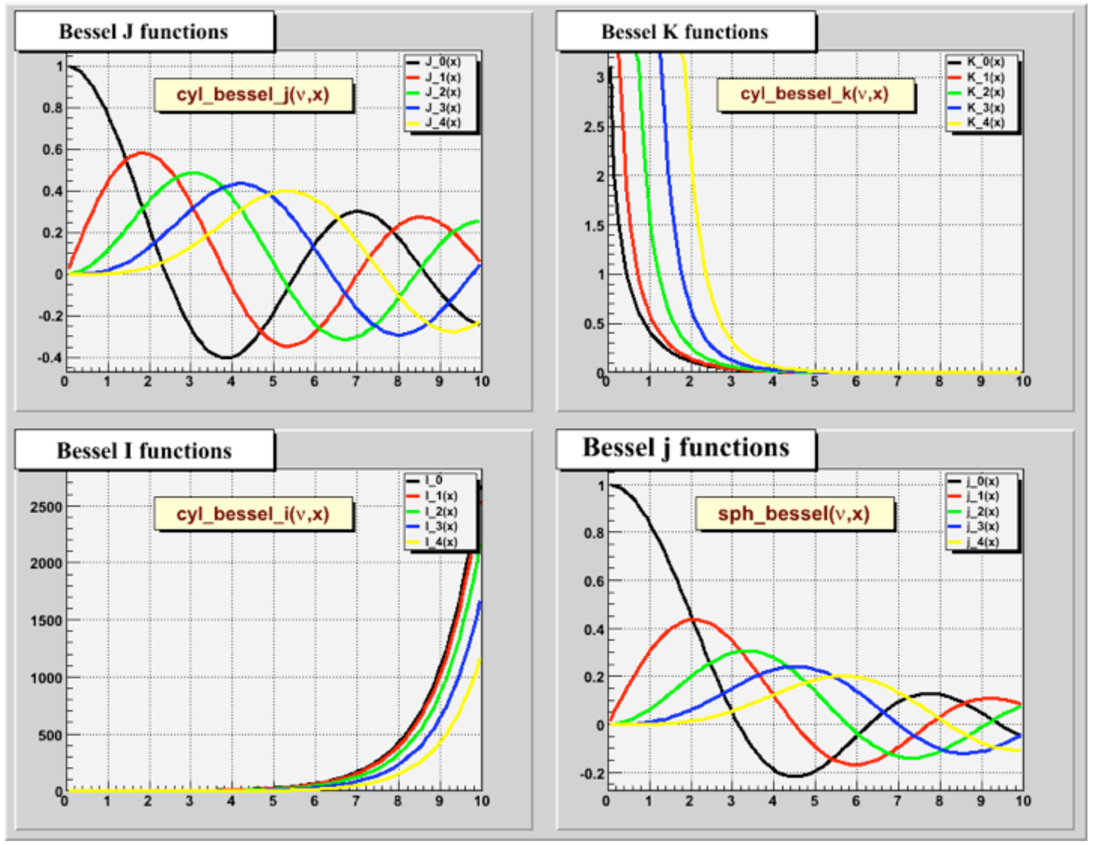
Extra algorithms

Extra Math functions

GSL

New Mathematical Functions

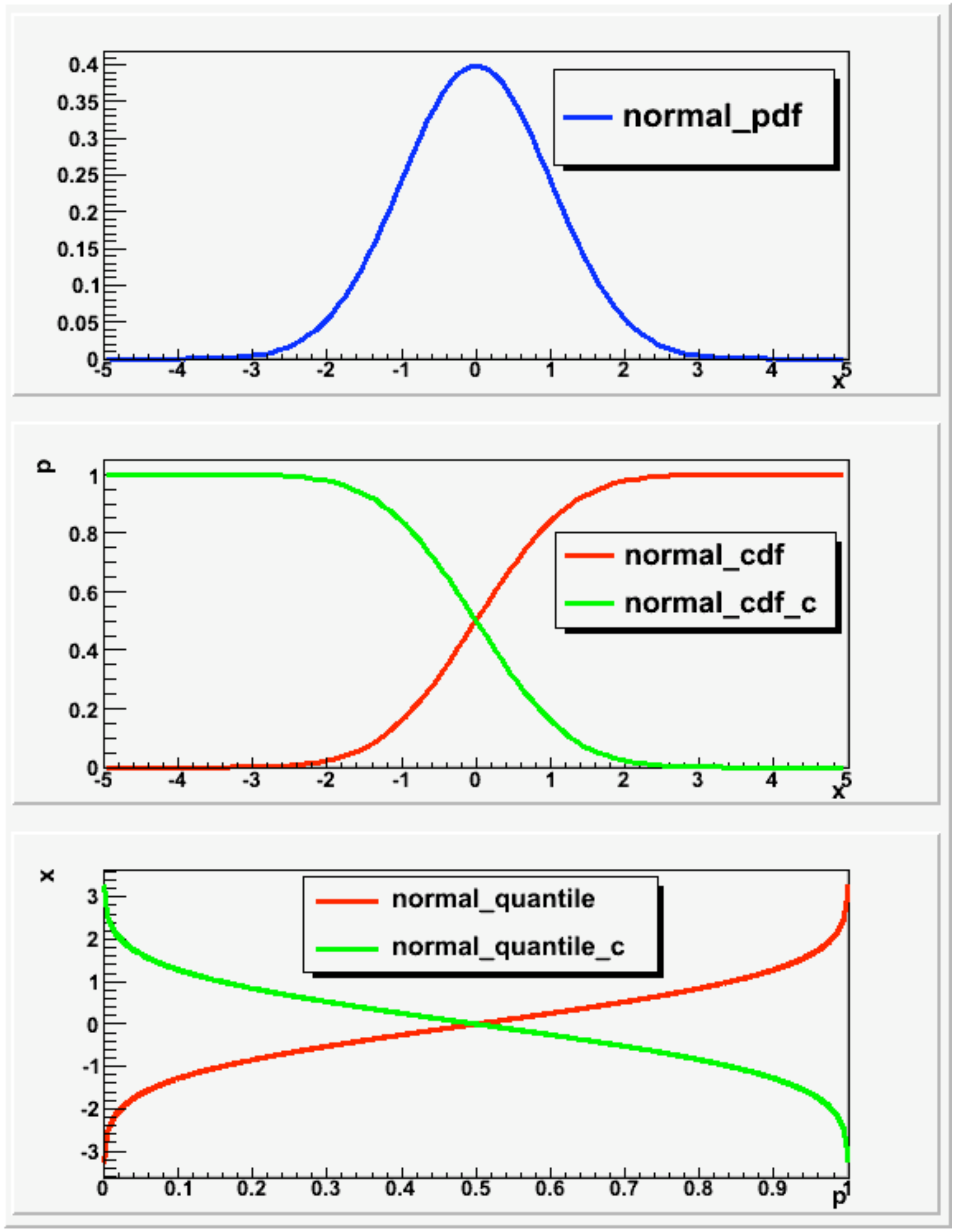
- New special functions in *MathCore and MathMore*
  - provided as free functions in `ROOT::Math` namespace
  - interface following C++ next standard proposal
  - new implementations with improved accuracy
    - code from Cephes library or using GSL (*MathMore* functions)
- TMath functions re-implemented using new code



- Available Functions
  - *error function, beta, gamma, log-gamma*
  - *incomplete gamma, incomplete beta*
    - implemented in *MathCore* library
    - using algorithms and code taken from Cephes library
  - *bessel, hypergeometric, Legendre, elliptic integrals, etc....*
    - implemented in *MathMore* using the GNU Scientific Library (GSL)

Statistical Functions

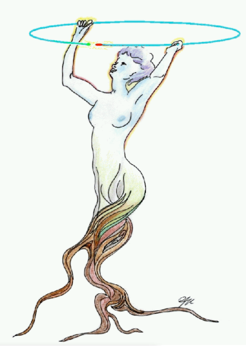
- Functions are provided in a coherent naming scheme
  - probability density functions (pdf)
    - i.e. normal distribution:
      - `normal_pdf( x, sigma, mu)`
  - cumulative distributions (cdf)
    - lower tail: `normal_cdf (x, sigma, mu)`
    - upper tail: `normal_cdf_c (x, sigma, mu)`
  - inverse of cumulative distributions (quantiles)
    - inverse of lower cumulative
      - `normal_quantile (z, sigma)`
    - inverse of lower cumulative
      - `normal_quantile_c (z, sigma)`
- Have all major statistical distributions
  - *normal, lognormal, Landau, Cauchy, χ^2, gamma, beta, F, t, poisson, binomial, etc..*
- Defined also as free functions in `ROOT::Math` namespace



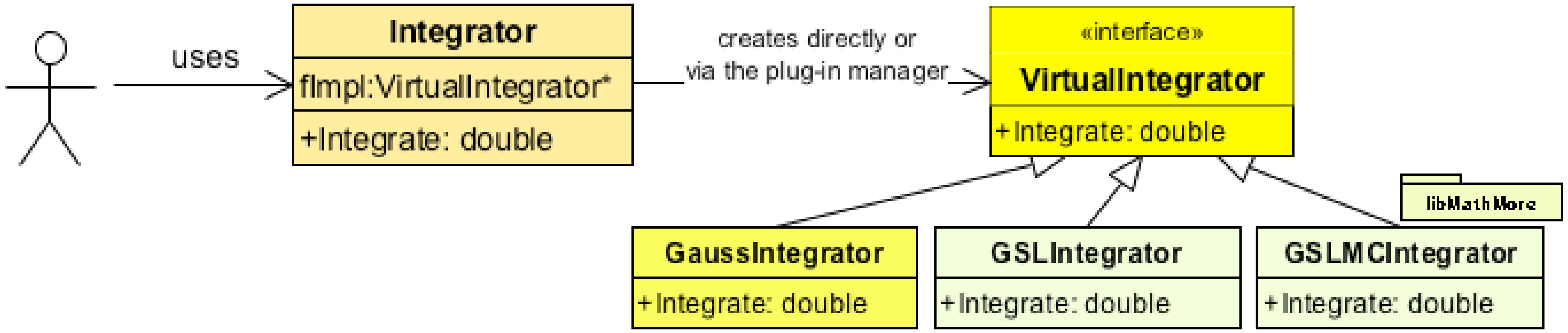
Numerical Algorithms

- Numerical algorithms in *MathMore* library implemented using GSL:
  - **Numerical Derivation**
    - central evaluation (5 points rule) and forward/backward
  - **Numerical Integration**
    - adaptive integration for finite and infinite intervals, singular functions and with Cauchy principal value.
    - multidimensional MC integration based on PLAIN, VEGAS and MISER
  - **Root Finders**
    - various bracketing and polishing algorithms using derivatives
  - **Minimization**
    - Golden section and Brent algorithm for 1D
    - conjugate gradient and BFGS algorithms for multi-dimensions
    - simulated annealing
    - solver for non linear least square solver with Levenberg-Marquardt algorithm
  - **Interpolation**
    - linear, polynomial, cubic and Akima spline
  - **Chebyshev polynomials** (for function approximation)
- Complement the various algorithms existing previously in ROOT (*TF1* class)
  - exported in *MathCore* classes (`TF1::Integral` uses class `GaussIntegrator`)

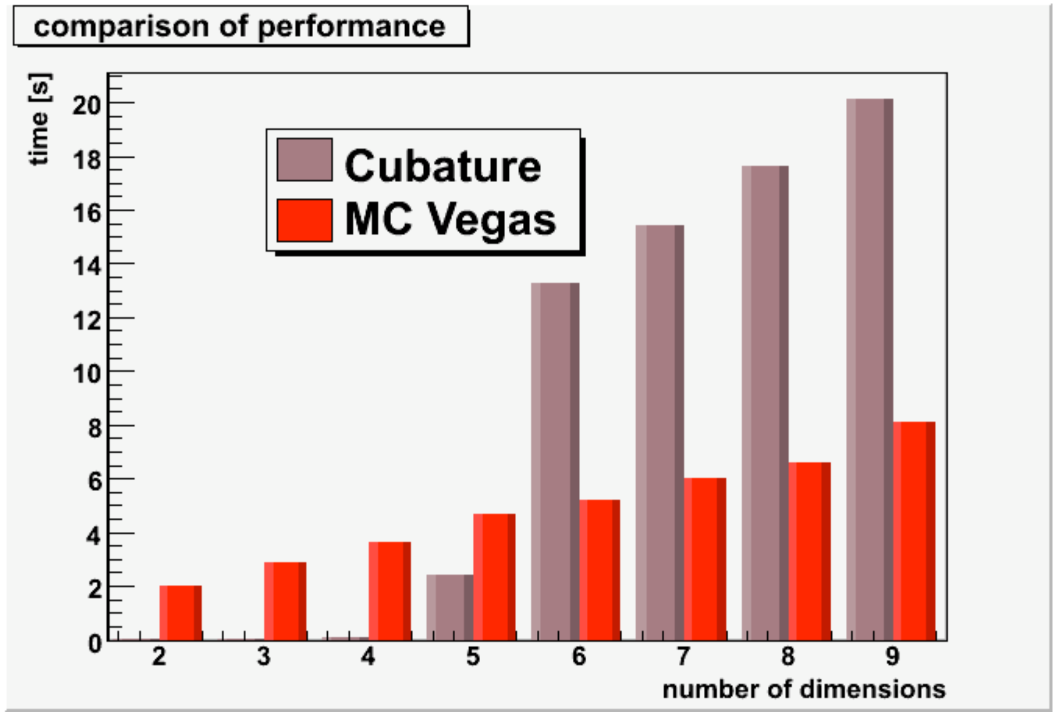
Numerical Algorithms



- Developed common interface for numerical algorithms
  - single entry point for multiple implementations
  - example: integrator class

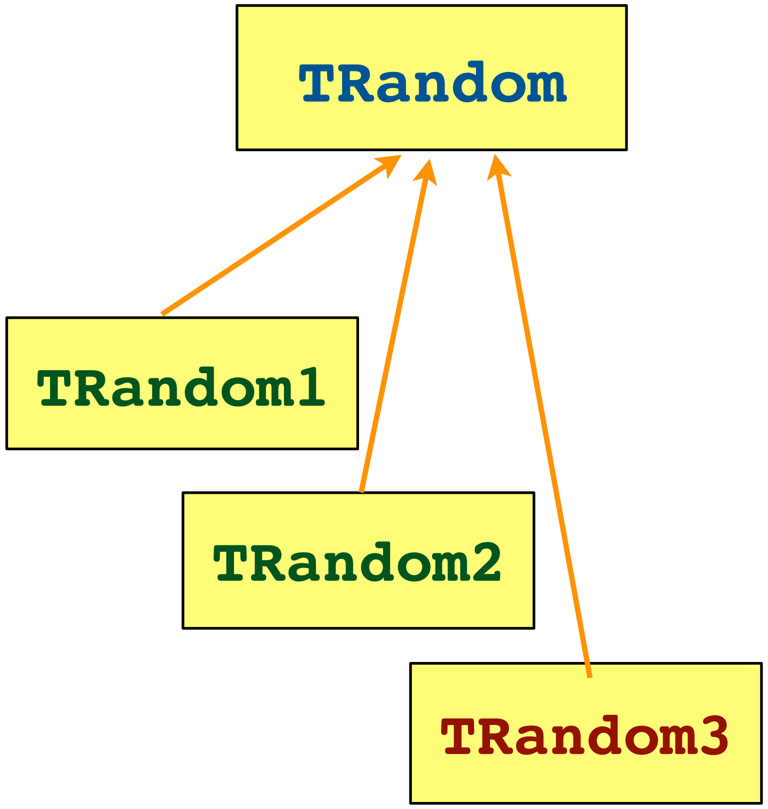


```
using namespace ROOT::Math;
//multidim integrand function
double func( const double* x, const double *p);
....
// Functor class to wrap user function in interface
Functor f(func,dimension);
// adaptive cubature method
IntegratorMultiDim ig(IntegrationMultiDim::kADAPTIVE);
double v1 = ig.Integral(f,xmin,xmax);

// MC method (VEGAS) loaded from MathMore library
IntegratorMultiDim ig(IntegrationMultiDim::kVEGAS);
double v2 = ig.Integral(f,xmin,xmax);
```



Random Number Generators

- *Mersenne-Twister* generator (`TRandom3`) default generator
  - fast and good pseudo-random quality
  - very long period, ~10^{6000} , large state (624 words)
- *RanLux* generator (`TRandom1`)
  - proven random quality, but slower
- *TausWorthe* generator from L'Ecuyer (`TRandom2`)
  - fast generator based only on 3 words (period ~ 10^{26})
- Linear congruential generator (`TRandom`)
  - maintained only for backward compatibility
  - have extremely short period (2^{31})
    - not good random quality, although improved recently
  - should never be used in statistical studies



TRandom

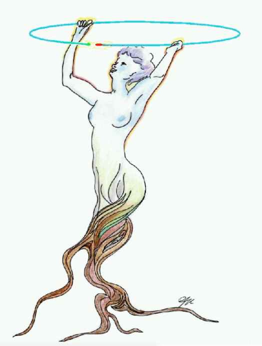
TRandom1

TRandom2

TRandom3

- `TRandom` is the base class with the common methods for all generators
- All generators can be seeded with an *UUID* (unique 128 bit number)
  - convenient when running parallel jobs on the Grid
  - obtained by using 0 as seed when constructing or in `TRandom::SetSeed`
    - `TRandom3 r(0); gRandom->SetSeed(0)`

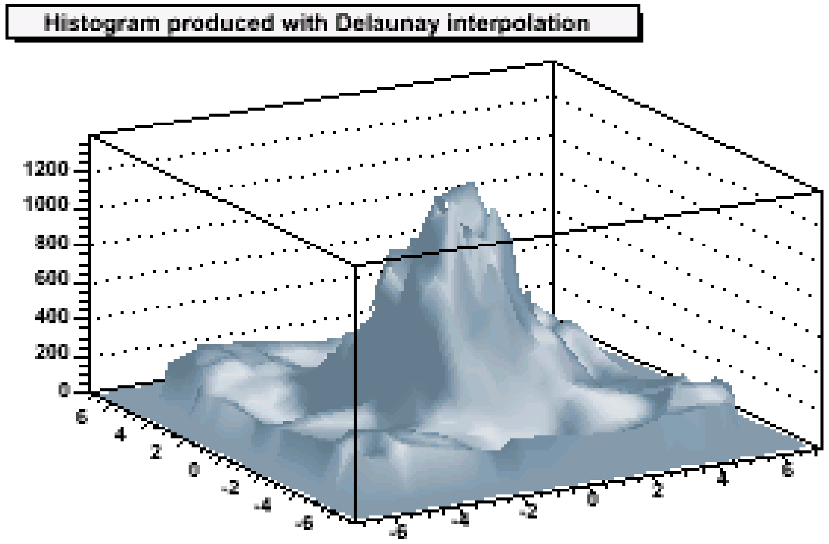
Random Number Distributions



- Methods available in the class `TRandom` for sampling according to some standard distributions
  - `gRandom->Gaus(mu,sig); gRandom->Poisson(mu)`
  - improved algorithms for generating Gaussian (more efficient) and Poisson random numbers (correct for all mu values)
  - approximate (but efficient) sampling for user functions via `TF1::GetRandom` (approximate function on a fixed grid)
    - possible also for TF2 and TF3 but not very efficient or too approximate
  - `TH1::GetRandom` for sampling using histograms
- Provide interface to *UNU.RAN*
  - package for generating non uniform random numbers (J. Leydold et al, Vienna TU)
  - various methods for generic 1D, multi-dim., discrete and empirical distributions (set of un-binned or binned data)
  - multi-dimensional methods based on Markov-Chain Monte Carlo
  - provides efficient and in general exact methods
- Interface to *FOAM* Monte Carlo generator (`TFoam`) (S. Jadach, Cracow)
  - multi-dimensional generator using self-adapting cellular grids

Data Analysis Classes

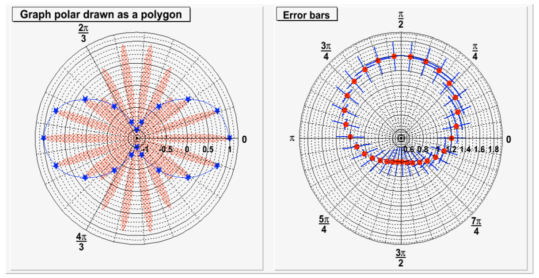
- TTree
  - for sets of un-binned data
  - optimized for dealing with large data volumes
- Histogram classes:
  - TH1,TH2,TH3 for binning data in 1,2,3 dimensions
    - THNSparse for sparse histograms in multi-dim
  - TProfile: profile histograms (1,2,3 dim.)
- TGraph classes:
  - TGraph,TGraphErrors, TGraphAsymmErrors, TGraphBentErrors
    - for sets of 2D (x,y) data
  - TGraph2D, TGraph2DErrors:
    - 3D (x,y,z) data
  - provide various interpolation functions
    - splines, Delaunay triangulation for 2D



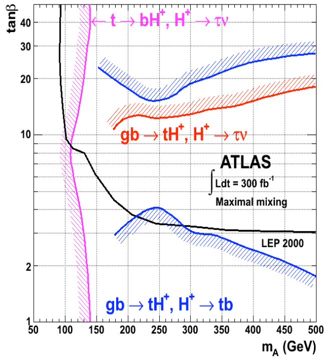
ROOT Visualization

- ✦ Possible to produce a very large variety of plots

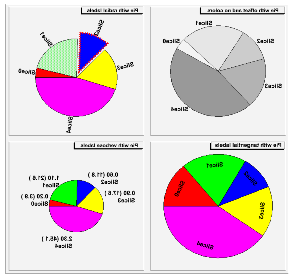
- ✦ Latest Developments in 2D graphics



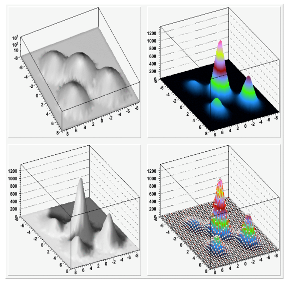
Polar plots



Exclusion plots



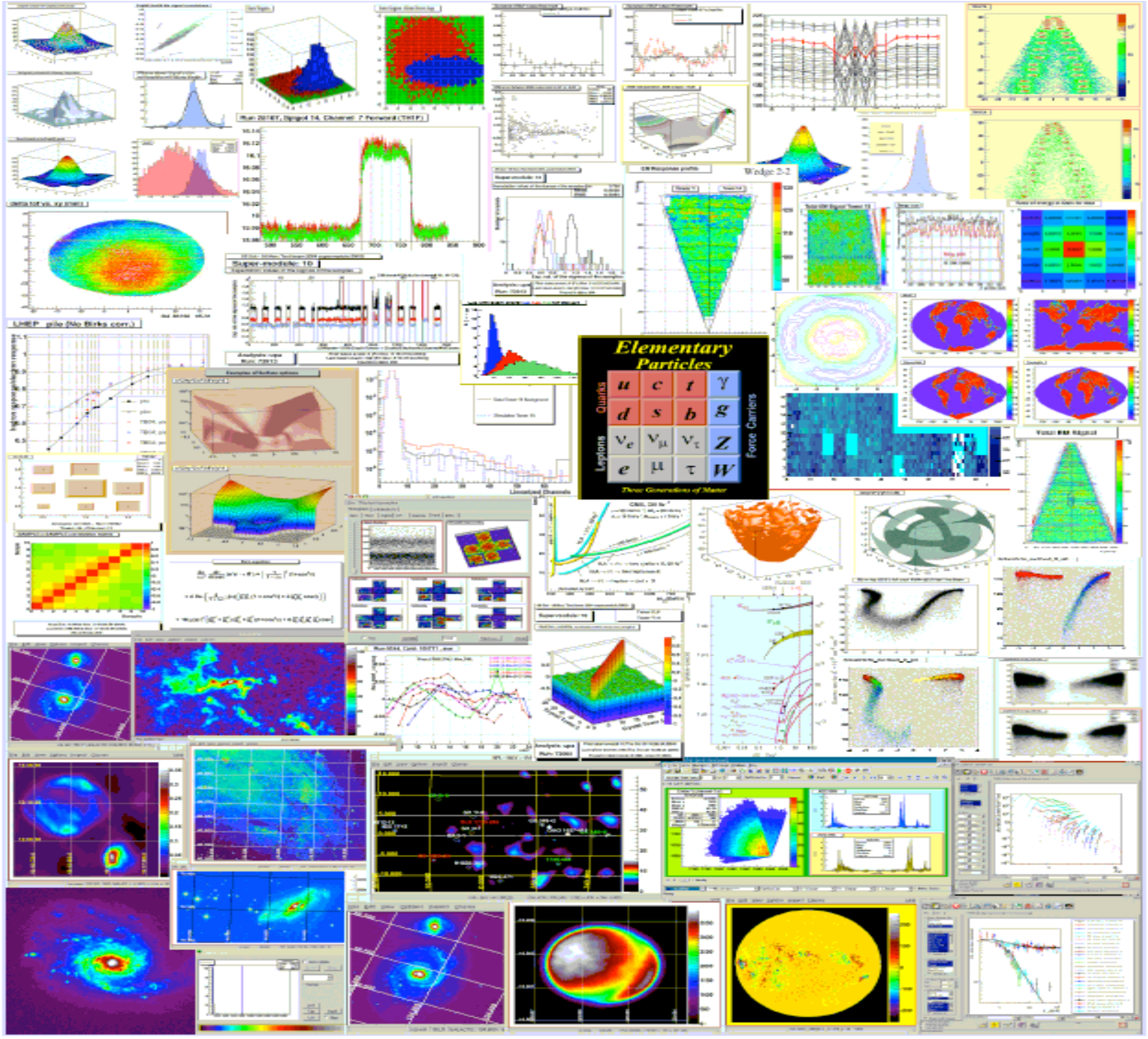
Pie charts



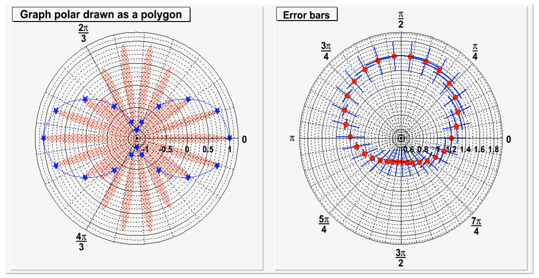
Spectrum plots

ROOT Visualization

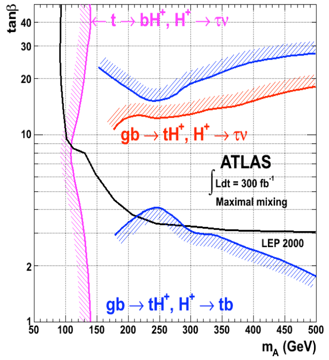
- Possible to produce a very large variety of plots



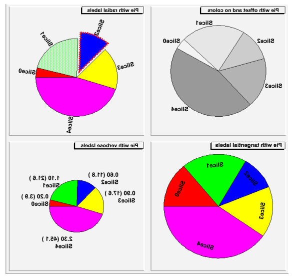
- Latest Developments in 2D graphics



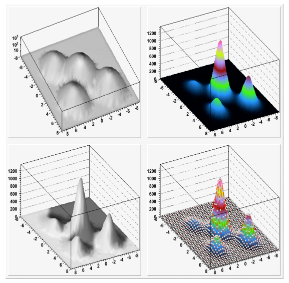
Polar plots



Exclusion plots

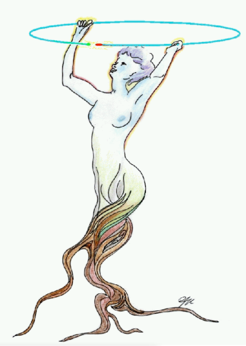


Pie charts

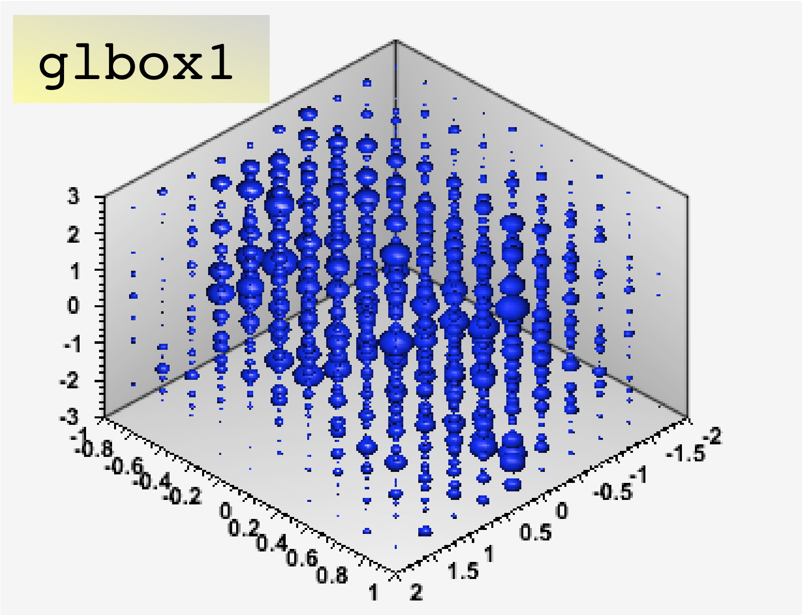


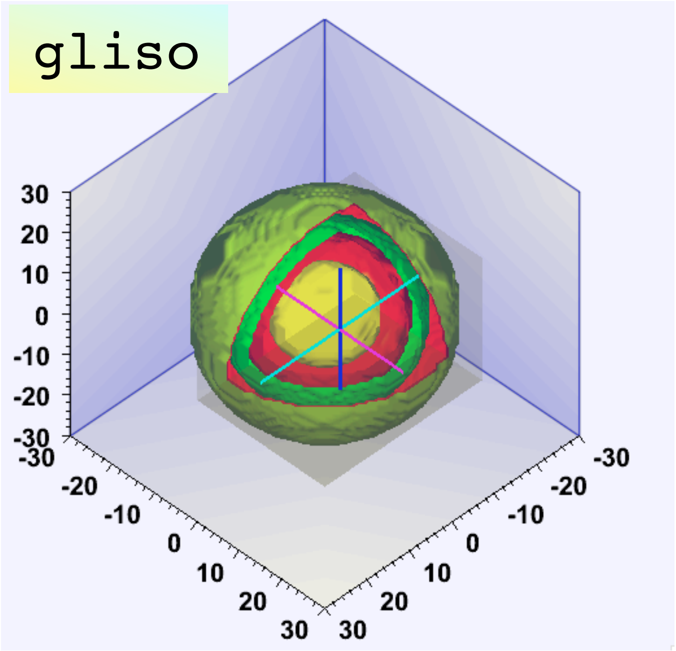
Spectrum plots

Visualization of Multi-Dimensional data

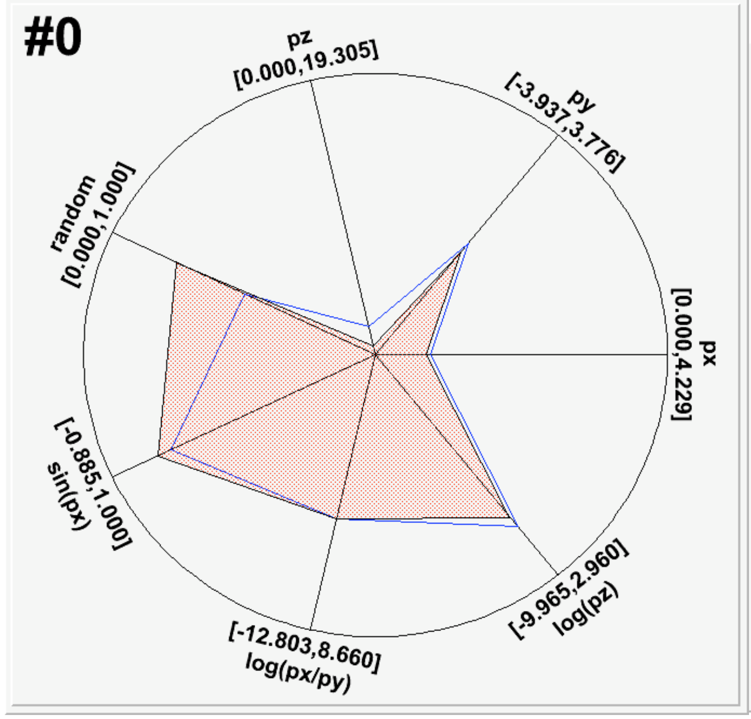


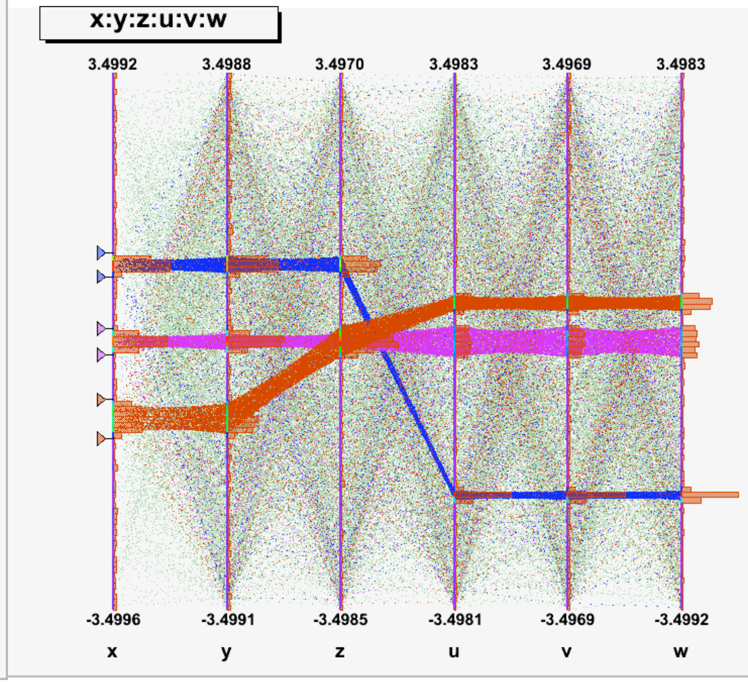
- Display of 3D histograms and functions (4D data) using OpenGL
  - GL plots with dynamic slicing (“`glsurf`” or “`glbox`”)
  - iso-surfaces (3D contour plot)





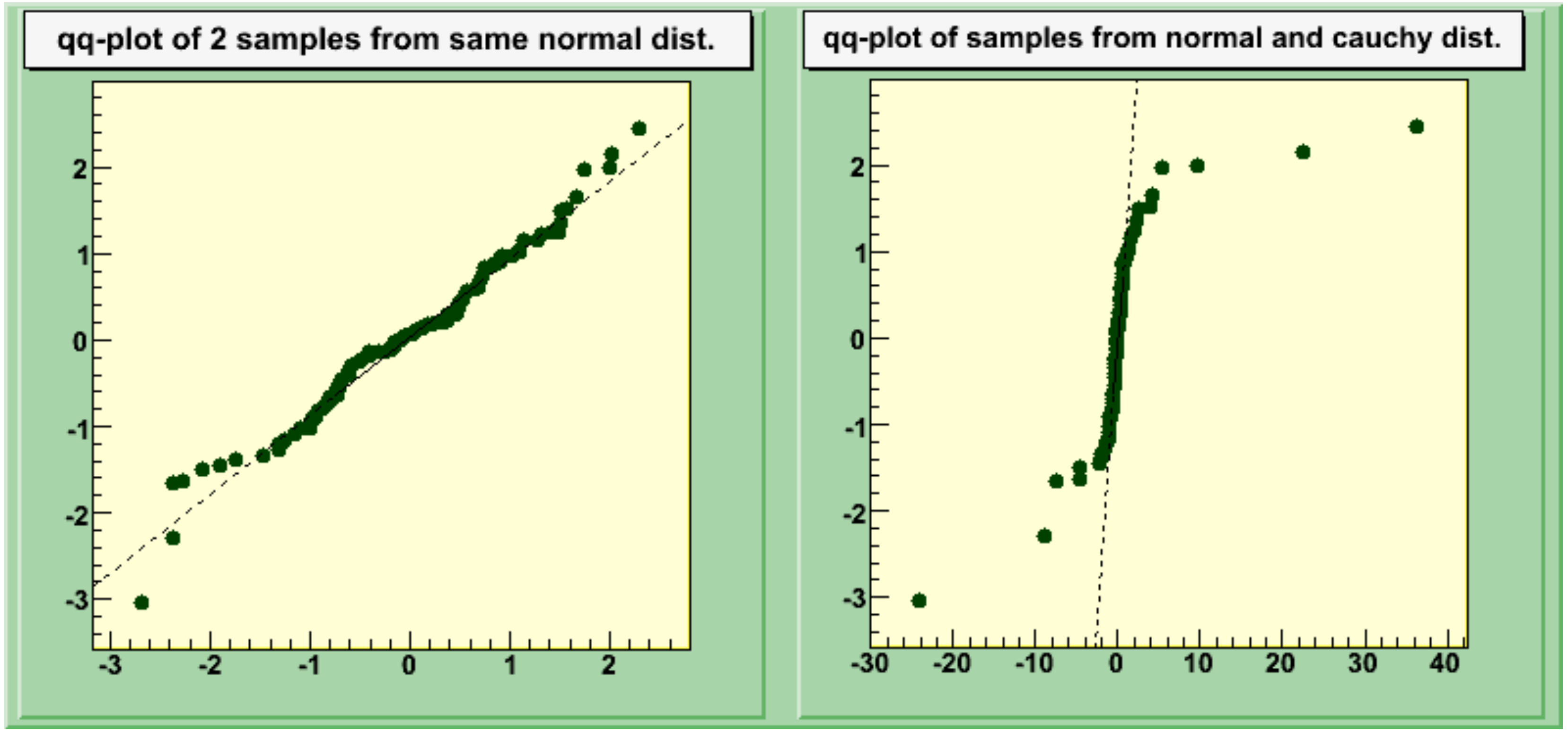
- Tools for multi-dimensional data sets
  - spider (radar) plots (`TSpider`)
    - for scan of a `TTree`
  - parallel coordinates (`TParallelCoord`)

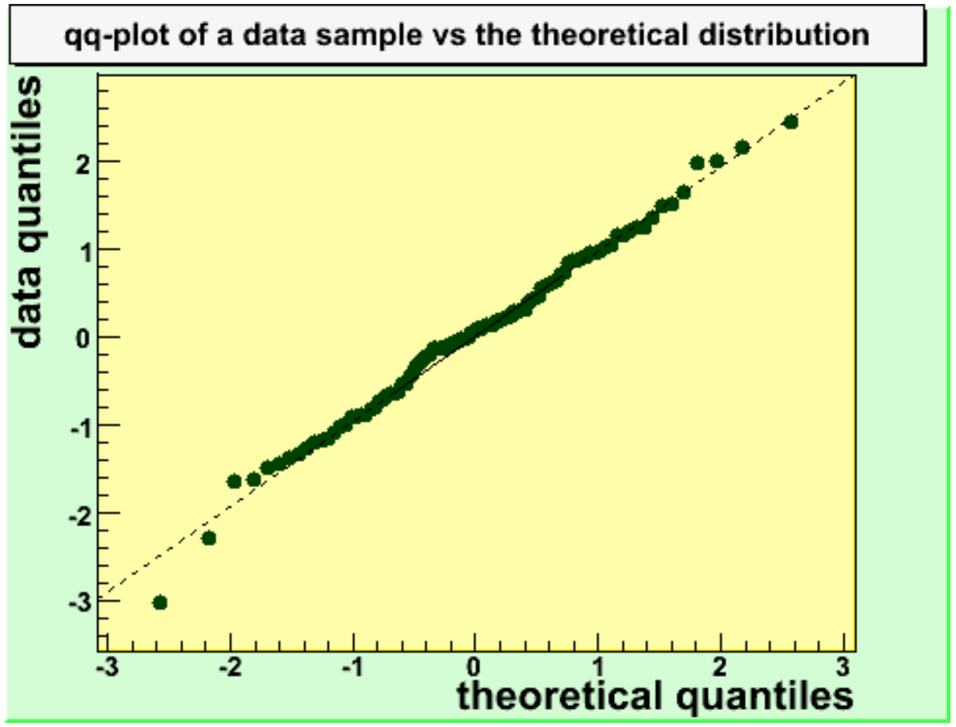




QQ Plot

- TGraphQQ
  - to draw quantiles of two data sets
  - to draw quantile of a data set vs a reference distribution



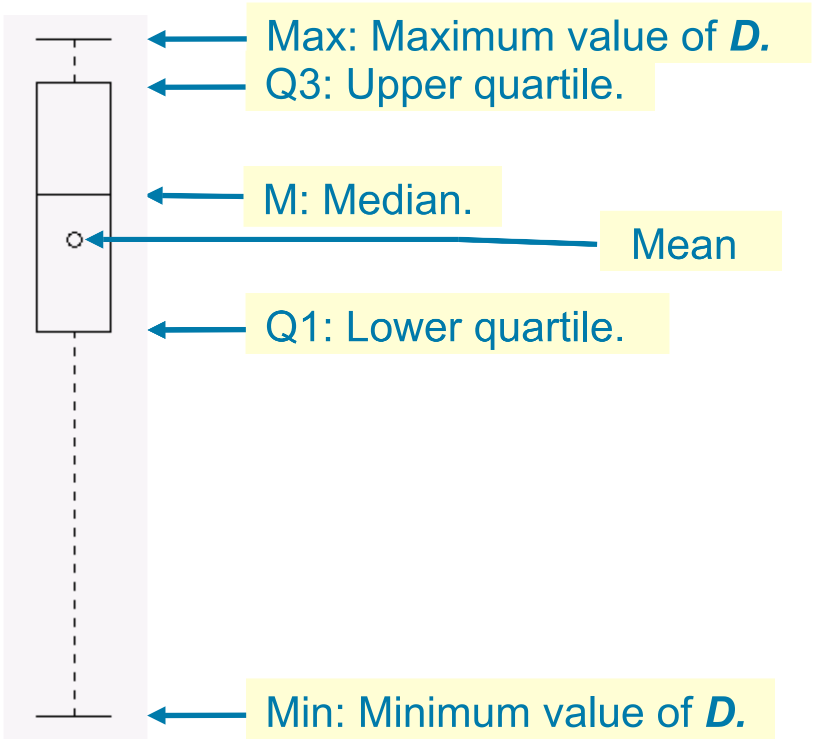


```
// x1[n1] first data set
// x2[n2] second data set
TGraphQQ(n,n1,x1,n2,x2);
```

```
// x[n]  data set
// TF1* func(th.function)
TGraphQQ(n,x,func);
```

Box Plots

- Convenient way to represent a data distribution with only 5 numbers



Max: Maximum value of ***D.***

Q3: Upper quartile.

M: Median.

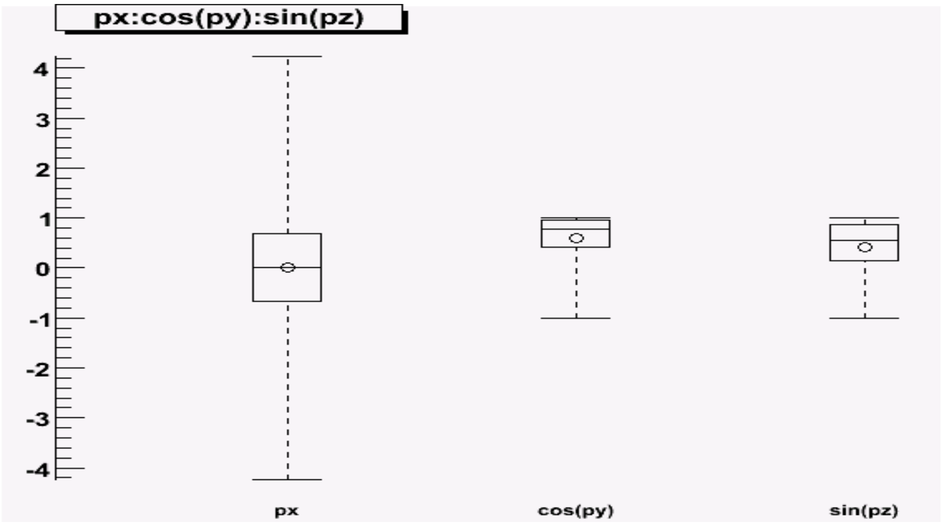
Mean

Q1: Lower quartile.

Min: Minimum value of ***D.***

- minimum value
- lower quantile Q1 (25% data lower than Q1)
- median (second quantile)
- third quantile Q3 (25% data larger than Q3)
- maximum value

In ROOT we also show the average as an open dot



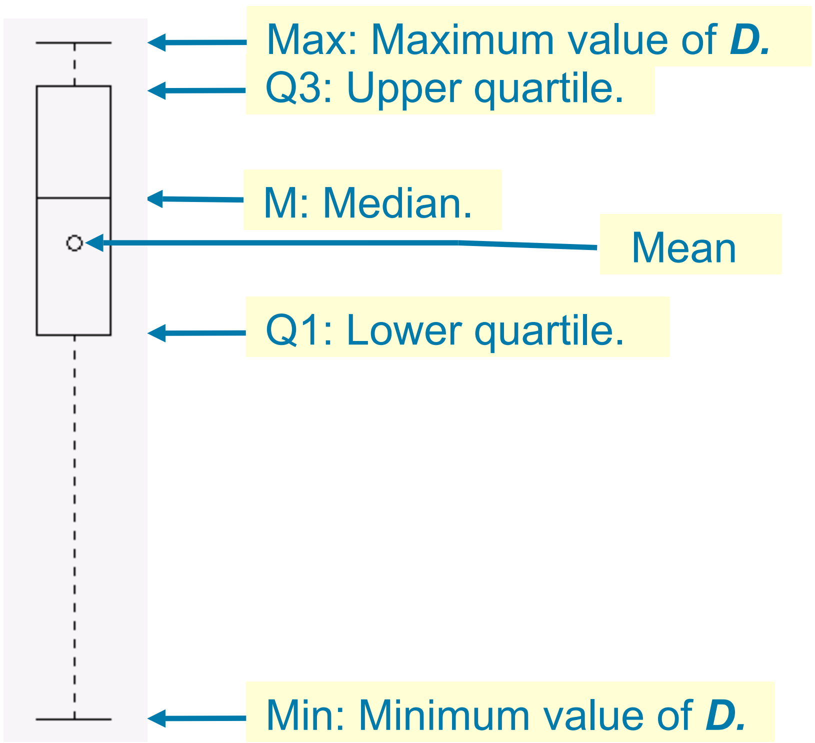
Possible to combine
box plots with
parallel coordinates

Option “candle” in TTree::Draw

```
tree->Draw(“px:cos(py):sin(pz)”,””,”candle”);
```

Box Plots

- Convenient way to represent a data distribution with only 5 numbers



Max: Maximum value of ***D.***

Q3: Upper quartile.

M: Median.

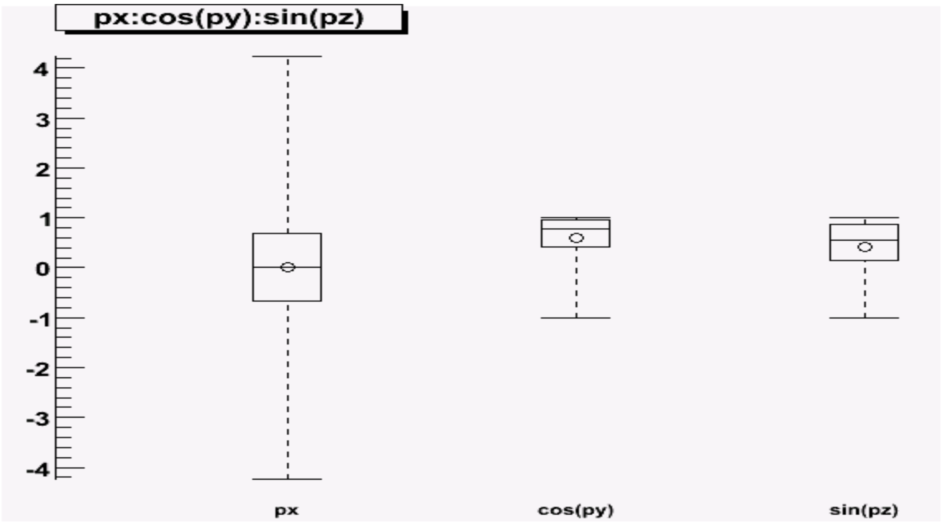
Mean

Q1: Lower quartile.

Min: Minimum value of ***D.***

- minimum value
- lower quantile Q1 (25% data lower than Q1)
- median (second quantile)
- third quantile Q3 (25% data larger than Q3)
- maximum value

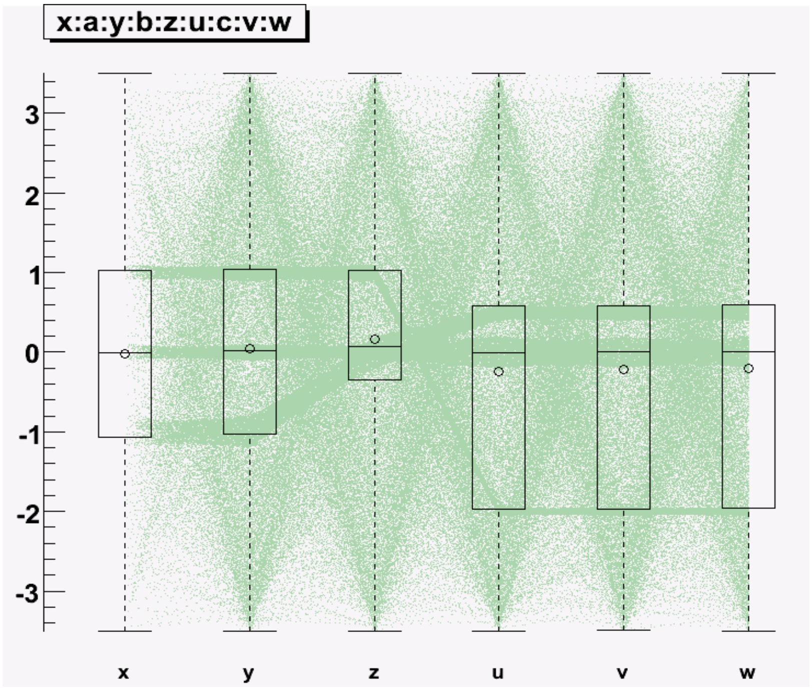
In ROOT we also show the average as an open dot



Option “candle” in TTree::Draw

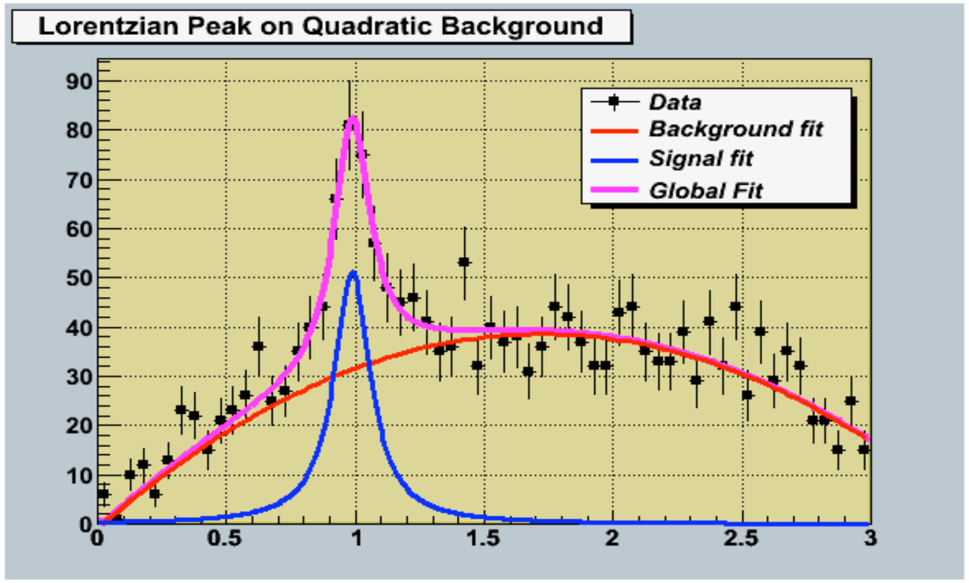
```
tree->Draw(“px:cos(py):sin(pz)”,””,”candle”);
```

Possible to combine box plots with parallel coordinates



Fitting in ROOT

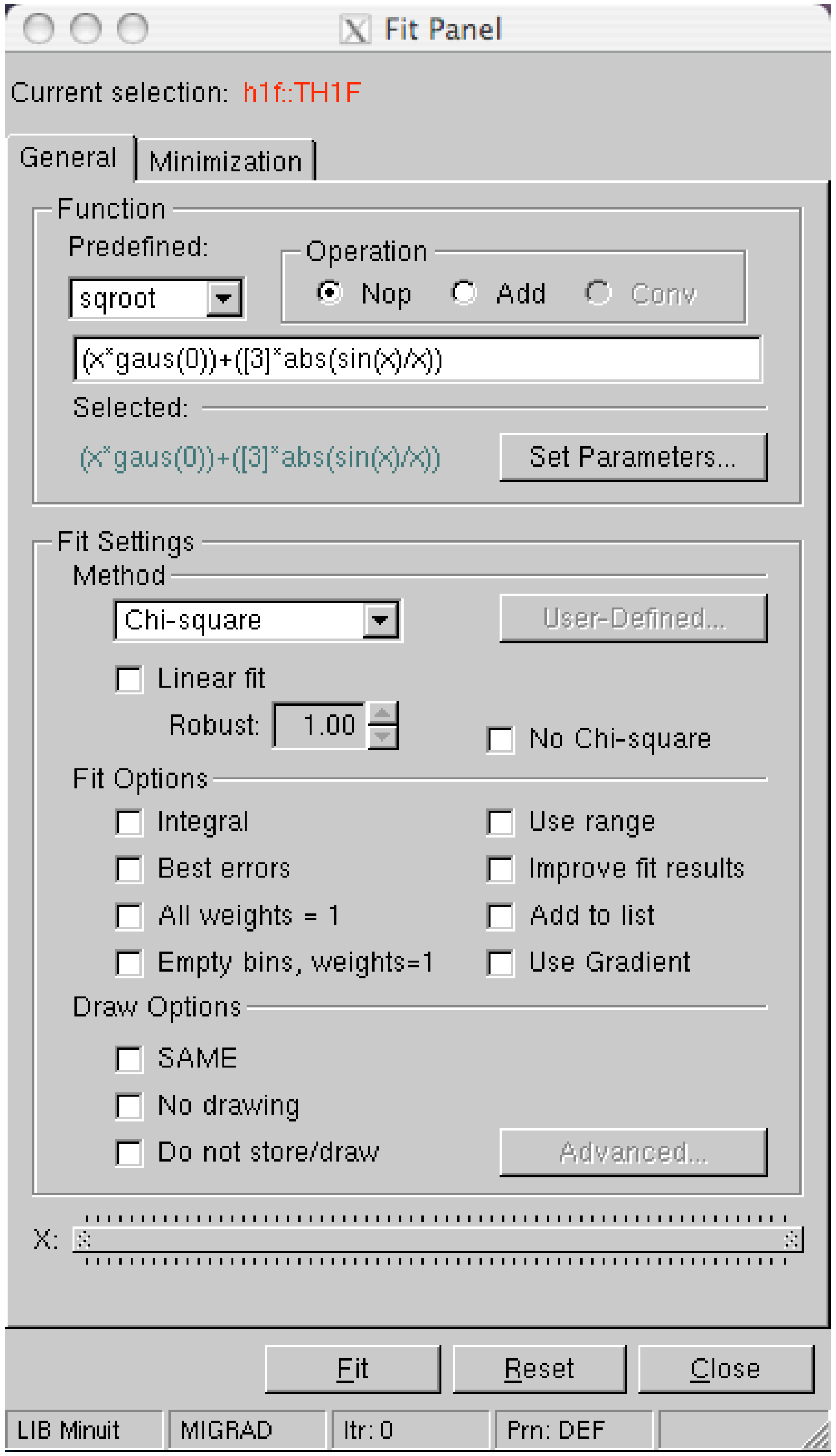
- Fit directly ROOT data classes (Histograms, Trees, Graphs)
- Various options available:
  - binned fits (`TH1::Fit`, `TGraph::Fit`)
    - Least Square fit (default)
    - Likelihood fits (`h1.Fit(func,"L")`)
  - unbinned likelihood fit (`TTree::UnbinnedFit`)
  - user defined model functions
  - user defined objective function
    - via option "U" and using `TVirtualFitter::SetFCN(myfcn)`
  - choice of minimization methods
    - *Minuit, Fumili, Minuit2, Fumili2, GSLMultiMin, GSLNLSFit*
  - automatic use of linear fits in case of linear functions (`"polN"`)
  - robust fit option (outlayer removal) in case of linear fits



- NEW: GUI (FitPanel) for driving the fit
  - available for all data objects (except the `TTree`)
- NEW: Improved ROOT fitting system and re-implemented the Fit methods for the data classes

Fit Panel

- GUI for fitting all ROOT data analysis objects
  - 1D and multi-dim histograms and graphs
    - Trees not yet available but planned soon
  - available via the context menu (right-click on the data object)
- Easy function selection, fit options and methods
- Allow use of different minimizers (via a separate panel)
- Function parameters settings and control



Fit Panel

Current selection: h1f::TH1F

General | Minimization

Function

Predefined: sqroot

Operation: ● Nop ○ Add ○ Conv

(x*gaus(0))+([3]*abs(sin(x)/x))

Selected:

(x*gaus(0))+([3]*abs(sin(x)/x))

Set Parameters...

Fit Settings

Method

Chi-square

User-Defined...

□ Linear fit

Robust: 1.00

□ No Chi-square

Fit Options

□ Integral
□ Use range
□ Best errors
□ Improve fit results
□ All weights = 1
□ Add to list
□ Empty bins, weights=1
□ Use Gradient

Draw Options

□ SAME
□ No drawing
□ Do not store/draw

Advanced...

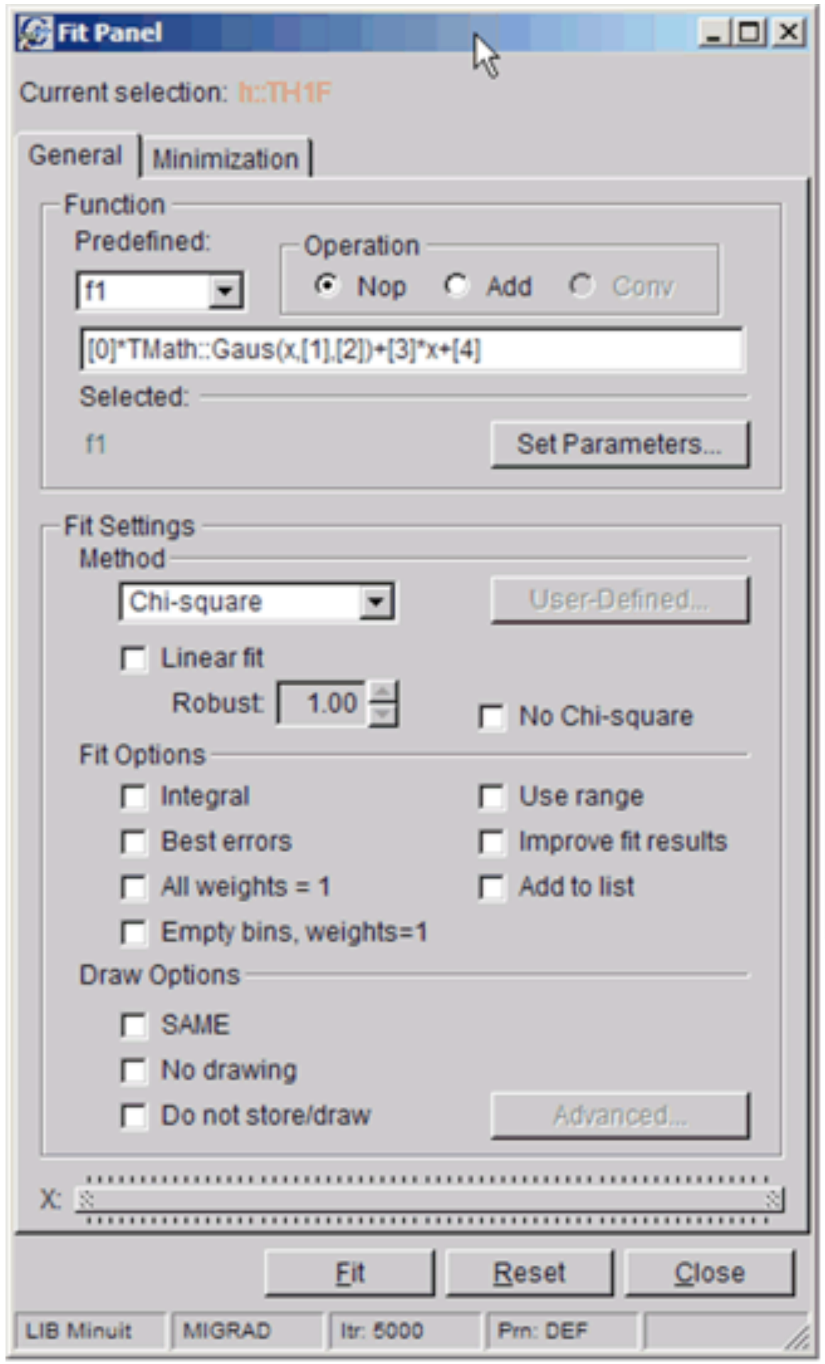
X:

Fit | Reset | Close

LIB Minuit | MIGRAD | Itr: 0 | Prn: DEF

Fit Parameters Control

Immediate preview of the changed function



Fit Panel

Current selection: h::TH1F

General | Minimization

Function

Predefined: f1

Operation: ● Nop ○ Add ○ Conv

[0]*TMath::Gaus(x,[1],[2])+[3]*x+[4]

Selected: f1

Fit Settings

Method: Chi-square

□ Linear fit

Robust: 1.00

□ No Chi-square

Fit Options

□ Integral □ Use range

□ Best errors □ Improve fit results

□ All weights = 1 □ Add to list

□ Empty bins, weights=1

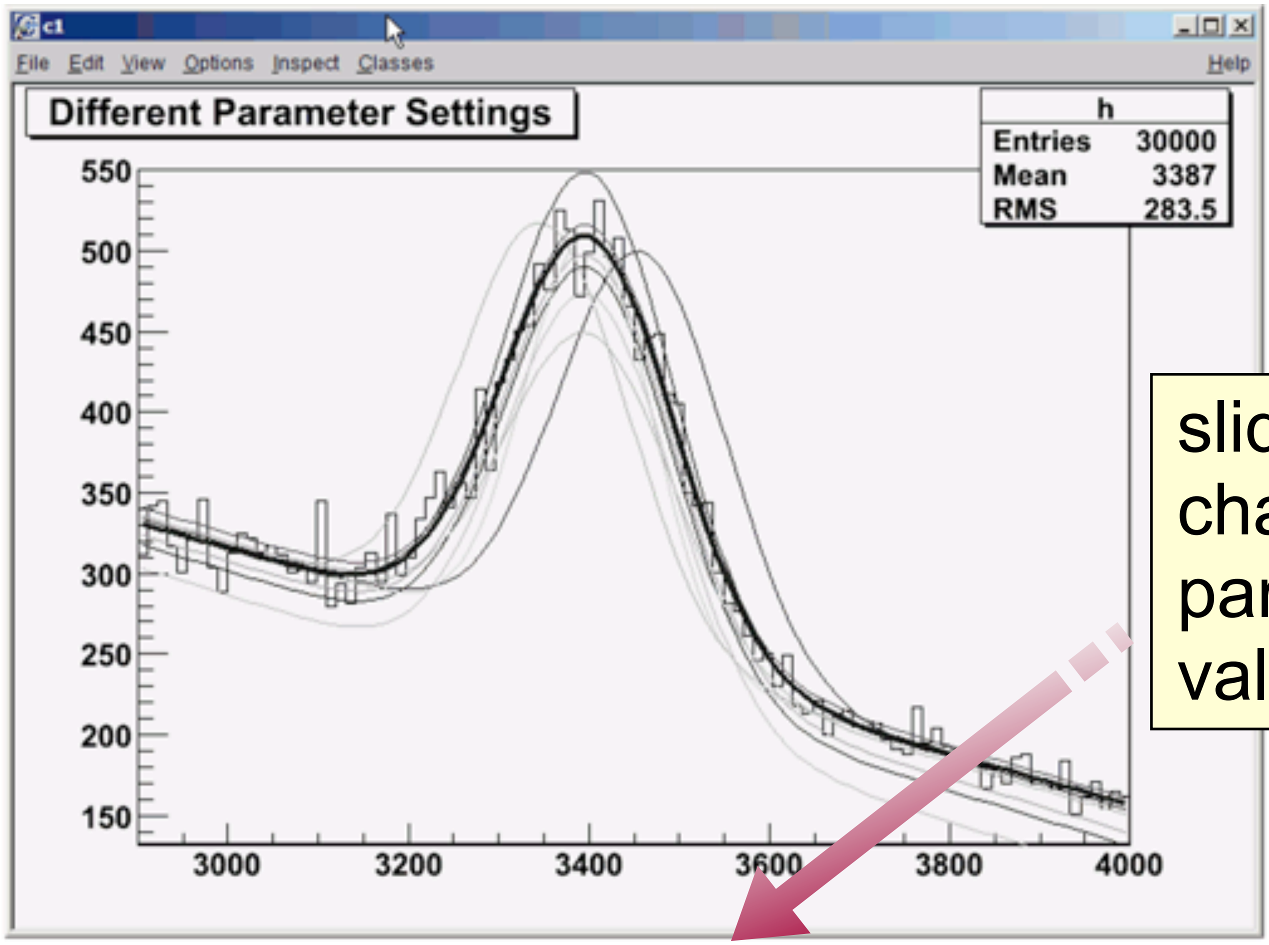
Draw Options

□ SAME

□ No drawing

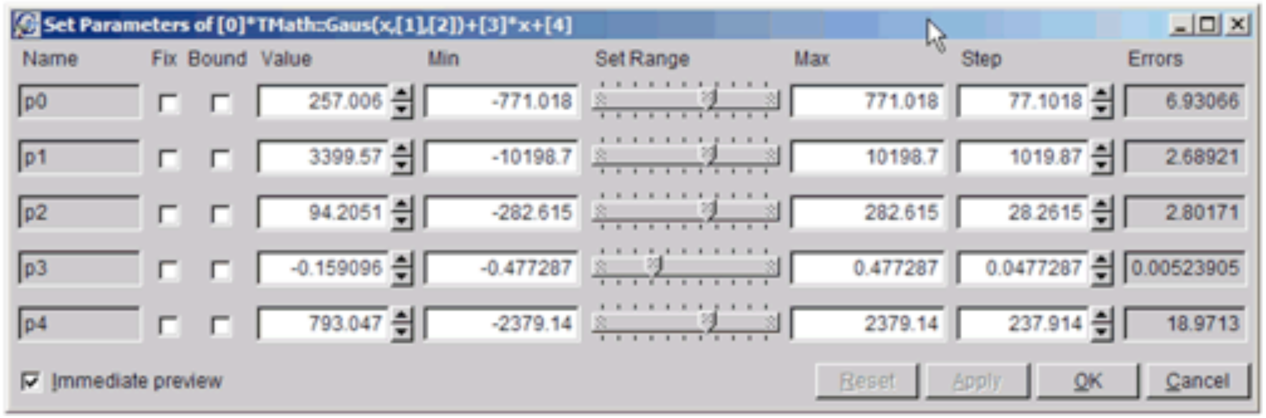
□ Do not store/draw

LIB Minuit | MIGRAD | Itr: 5000 | Prn: DEF



h	
Entries	30000
Mean	3387
RMS	283.5

slider for changing parameter values

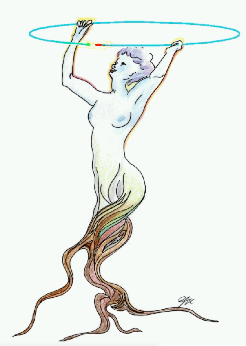


Set Parameters of [0]*TMath::Gaus(x,[1],[2])+[3]*x+[4]

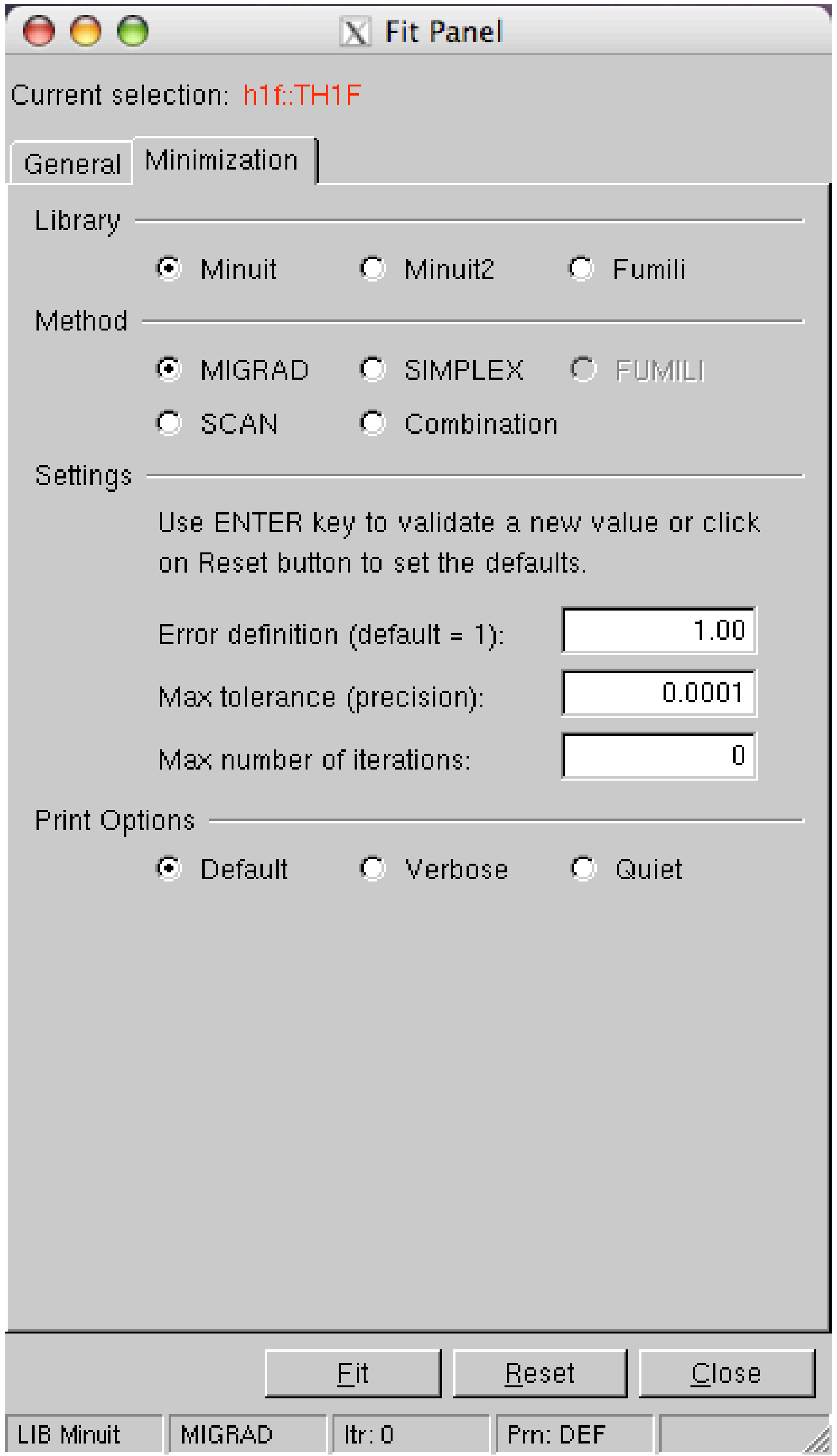
Name	Fix	Bound	Value	Min	Set Range	Max	Step	Errors
p0	□	□	257.006	-771.018		771.018	77.1018	6.93066
p1	□	□	3399.57	-10198.7		10198.7	1019.87	2.68921
p2	□	□	94.2051	-282.615		282.615	28.2615	2.80171
p3	□	□	-0.159096	-0.477287		0.477287	0.0477287	0.00523905
p4	□	□	793.047	-2379.14		2379.14	237.914	18.9713

☑ Immediate preview

Minimizer Control



- Interactive library selection
  - Minuit, Minuit2, Fumili, etc.
- Choice of minimization algorithm
  - Migrad, Simplex, etc.
- Control of minimizer parameters
  - Tolerance, Maximum number of iterations, etc..
- Print Options
  - Default, Verbose, Quite



Fit Panel

Current selection: h1f::TH1F

General | Minimization

Library

● Minuit ○ Minuit2 ○ Fumili

Method

● MIGRAD ○ SIMPLEX ○ FUMILI

○ SCAN ○ Combination

Settings

Use ENTER key to validate a new value or click on Reset button to set the defaults.

Error definition (default = 1):	1.00
Max tolerance (precision):	0.0001
Max number of iterations:	0

Print Options

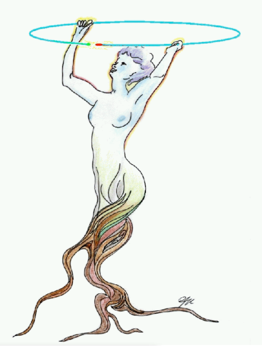
● Default ○ Verbose ○ Quiet

LIB Minuit | MIGRAD | Itr: 0 | Prn: DEF

Fitting Improvements

- Recently re-engineered fitting and minimization classes
  - maintained full backward compatibility
  - old interface (`TVirtualFitter`) implemented using new classes
- Feature of new fitting classes:
  - separate Fitter and Minimization interfaces
    - `Fitter` class and `Minimizer` interface (multiple implementations)
    - possible to use (and mix) the various minimization engines
  - decouple fitting from data source (`BinData` and `UnBinData` classes)
  - Fit results object which can be stored and retrieved (`FitResult`)
    - keep full result information (parameter, errors, covariance matrix, etc..)
  - Minimal and generic function interfaces for model functions (pdf) and objective (minimization) functions
    - easier for user to plug-in their functions (i.e. pdf classes from RooFit)
  - Support for parallel fits (usable in a multi-threads environment)

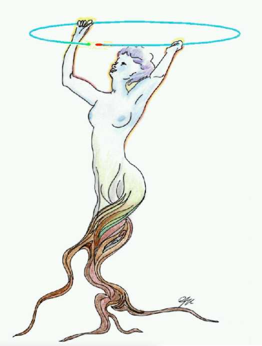
Example of Fitting



- Example of fitting via new classes
  - BinData class
    - fillable from ROOT objects (i.e. TH1) or simple data arrays
  - Fitter class configurable via the FitConfig class
  - Fit model functions in a defined interface (IParamFunction)
  - Have also interface for objective functions and used by the minimizer (IMultiGenFunction)
  - Produce FitResult class
    - keep all fit result information
    - provide methods for retrieving it
      - `fitresult.Parameters();`
`fitresult.Errors();`
`fitresult.CovMatrix(i,j);`

```
// fit inputs
TH1 * h1   = .....
TF1 * func = .....

ROOT::Fit::BinData d;
// fill the data set from the histogram
ROOT::Fit::FillData(d,h1);

// create wrapped parametric function for
// requested model function interface
ROOT::Math::WrappedTF1 f(*func);

// create fitter class
ROOT::Fit::Fitter

// set minimizer and configuration
fitter.Config().SetMinimizer(“Minuit2”);

//perform the fit using least square
bool ret = fitter.Fit(d,f);

//retrieve optionally the fit results
if (ret) fitter.Result().Print();

// fit using a user defined objective
// function implementing required interface
ROOT::Math::IGenFunction mySumSquare(d,f);

ret = fitter.FitFCN(mySumSquare);
```

Function Minimization



- Common interface class (`ROOT::Math::Minimizer`) for all ROOT minimizer implementations.
- Existing currently in ROOT:
  - Minuit (based on class `TMinuit`, direct translation from Fortran code)
  - Minuit2 (new C++ implementation with OO design)
  - Fumili (only for least-square or log-likelihood minimizations)
  - GSL minimizers
    - conjugate gradient algorithms (Fletcher-Reeves, BFGS)
    - Levenberg-Marquardt (only for minimizing least square functions)
  - Linear for least square functions (direct solution, non-iterative method)
- Easy to extend and plug-in new Minimizer engines
  - i.e. minimizer from NagC, or Opt++, etc..
- Possible to combine minimizers
  - working on a combination of Minuit and a genetic minimizer
- Easy usable outside fitting scope (general optimization problems)

Minuit2

- New Object-Oriented version of *Minuit* written in C++
  - written (by M. Winkler) under direct F. James supervision
  - same basic functionality as in old version
    - Migrad, Simplex and Scan
    - Minos algorithms and function contours for error analysis
  - extended functionality:
    - single side parameter limits
    - possibility to retrieve all information for each iteration
    - added Fumili method for least square and log-likelihood fits

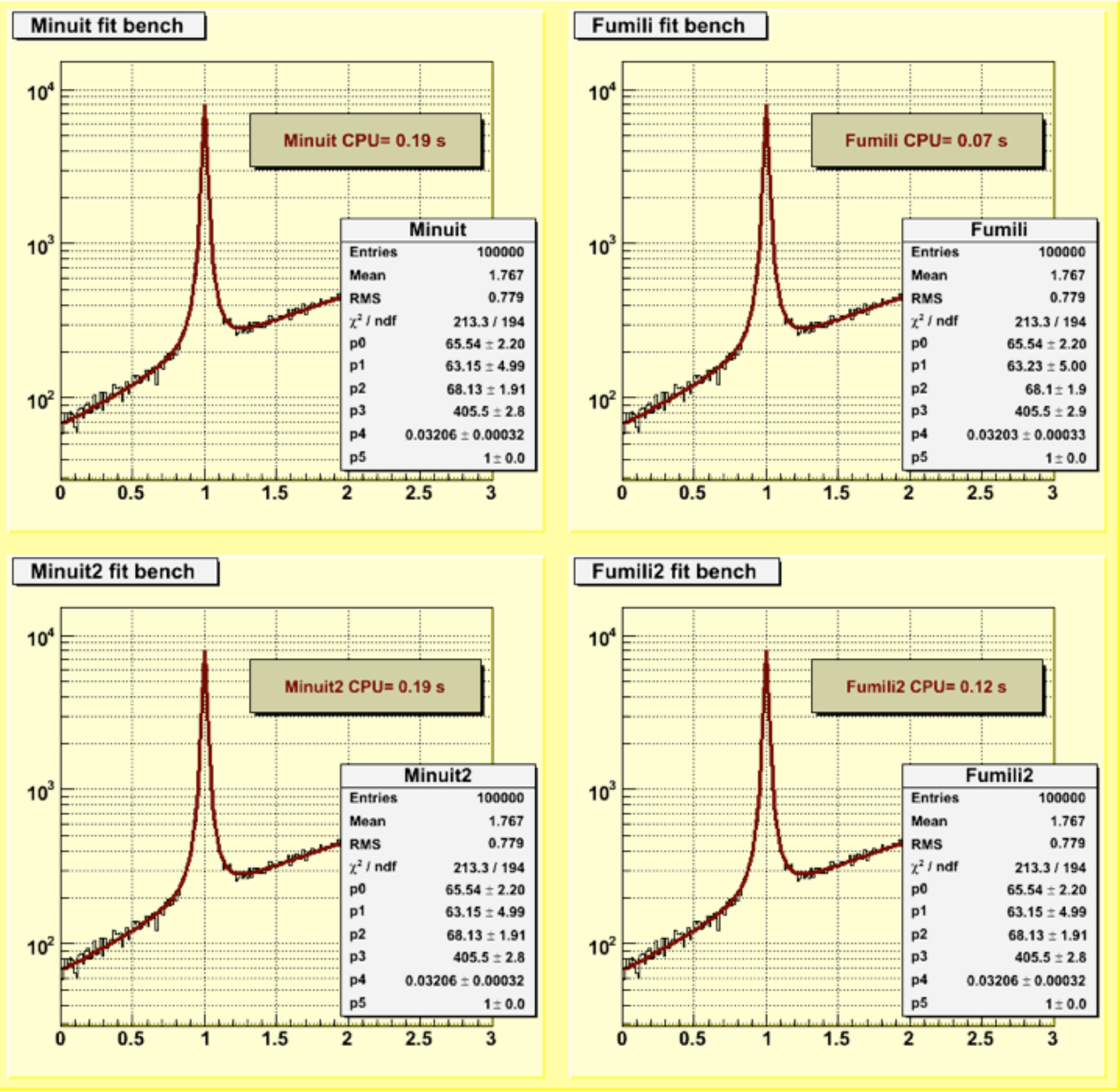
$$F(\mathbf{x}) = \sum_{k=1}^{K} f_k^2(\mathbf{x}) \qquad \frac{\partial^2 F}{\partial x_i \partial x_j} \approx \sum_k 2 \frac{\partial f_k}{\partial x_i} \frac{\partial f_k}{\partial x_j}.$$

neglect terms with
second derivatives in *f(x)*

$$F(\mathbf{x}) = -\sum_{k=1}^{K} \ln f_k(\mathbf{x}), \qquad \frac{\partial^2 F}{\partial x_i \partial x_j} \approx \sum_k \frac{1}{f_k^2} \frac{\partial f_k}{\partial x_i} \frac{\partial f_k}{\partial x_j}$$

Minuit2 Validation

- Validated with extensive testing
  - same results and number of function calls to find minimum
  - overhead observed only when minimizing functions easy to compute
    - more memory allocation since all information for each iteration is kept
- Interfaced and distributed in ROOT
  - can also be used as a standalone package
- Object-Oriented package for generic function minimization
  - easy to maintain and to extend with new algorithms
    - added Fumili method (Fumili2)
    - working on adding parallel support (multi-threads and multi-processes)



tutorials/fit/minuit2FitBench.C

Parallel Minimization

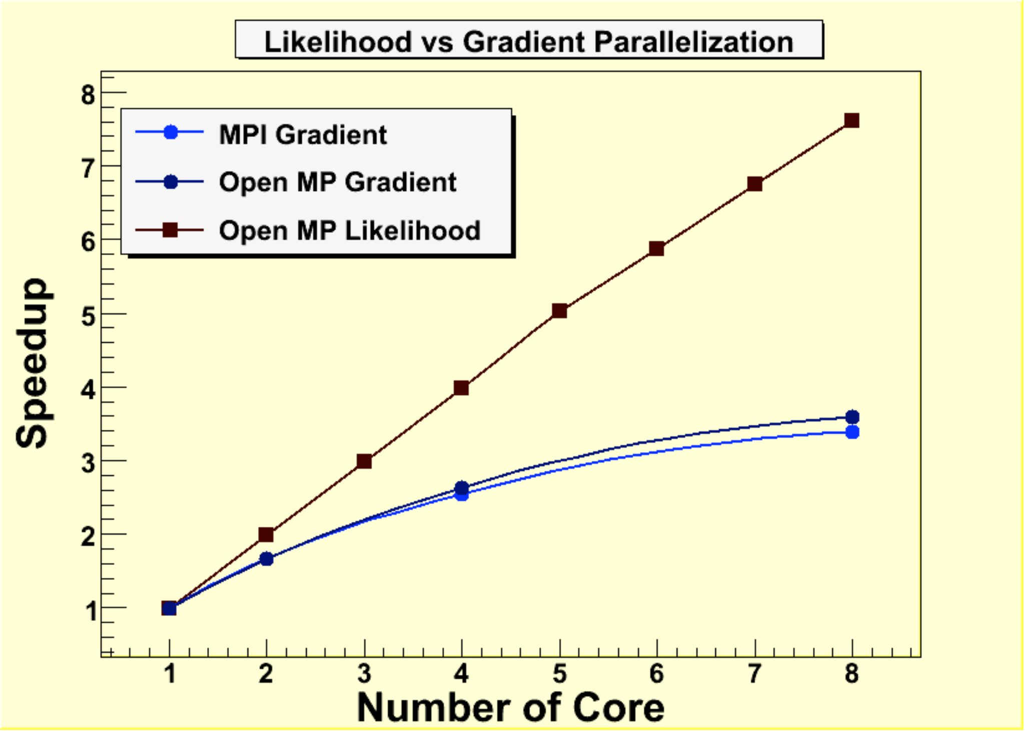
- Prototyped parallelization of MIGRAD algorithm
  - each Migrad iteration consists of:
    - computing function value and gradient
    - computing step by searching for minimum along the Newton direction
    - if satisfactory improve calculation of Hessian matrix, *H*
    - invert to get new matrix $V = H^{-1}$
    - repeat iteration until expected distance from minimum smaller than required tolerance
  - In case of many parameters (> 10) and complex function evaluation, gradient calculation dominates the process:

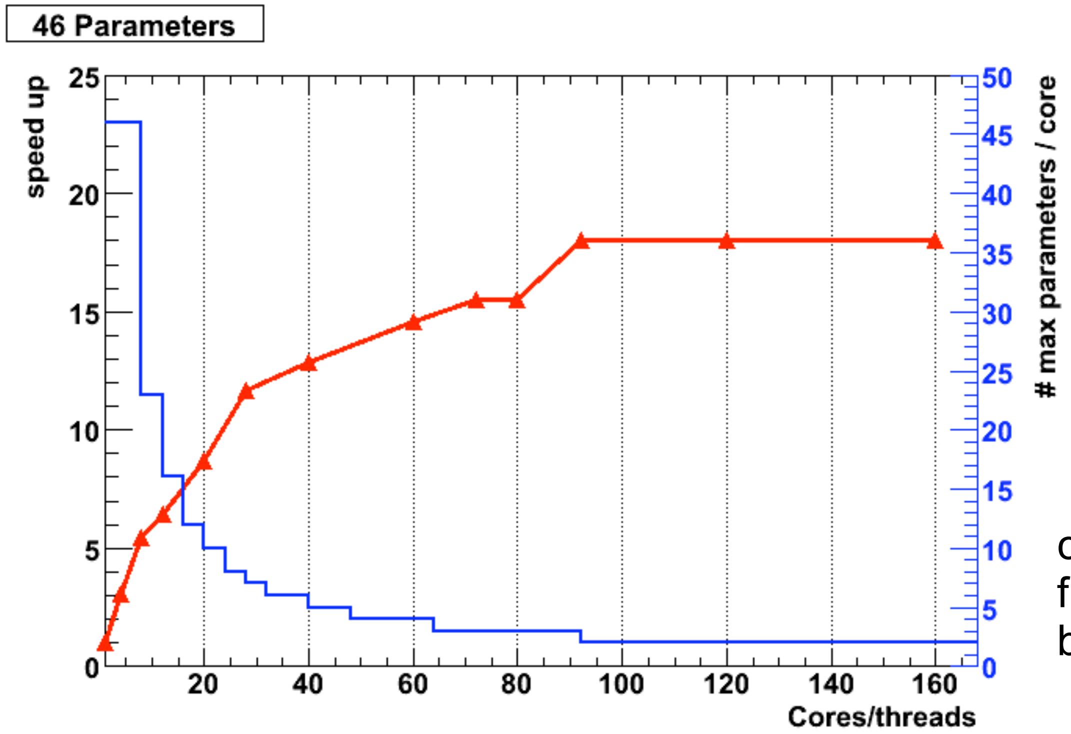
$$\nabla_i(x) = \frac{\partial f}{\partial x_i} \approx \frac{f(x_i + \delta x_i) - f(x_i - \delta x_i)}{2\delta x_i}$$

    - al least 2 * NDIM function evaluation are needed
  - Parallelize calculations by using a thread for each partial derivative
  - Studied using OpenMP (multi-thread) or MPI (multi-processes)
    - aim to distribute in one of the next ROOT release (as a config option)

Parallel Fitting and Minimization

- Advantage of parallelizing at Minuit level is independent of user code
- Log-likelihood parallelization (splitting the sum) more efficient
  - more demanding on thread safety of provided code



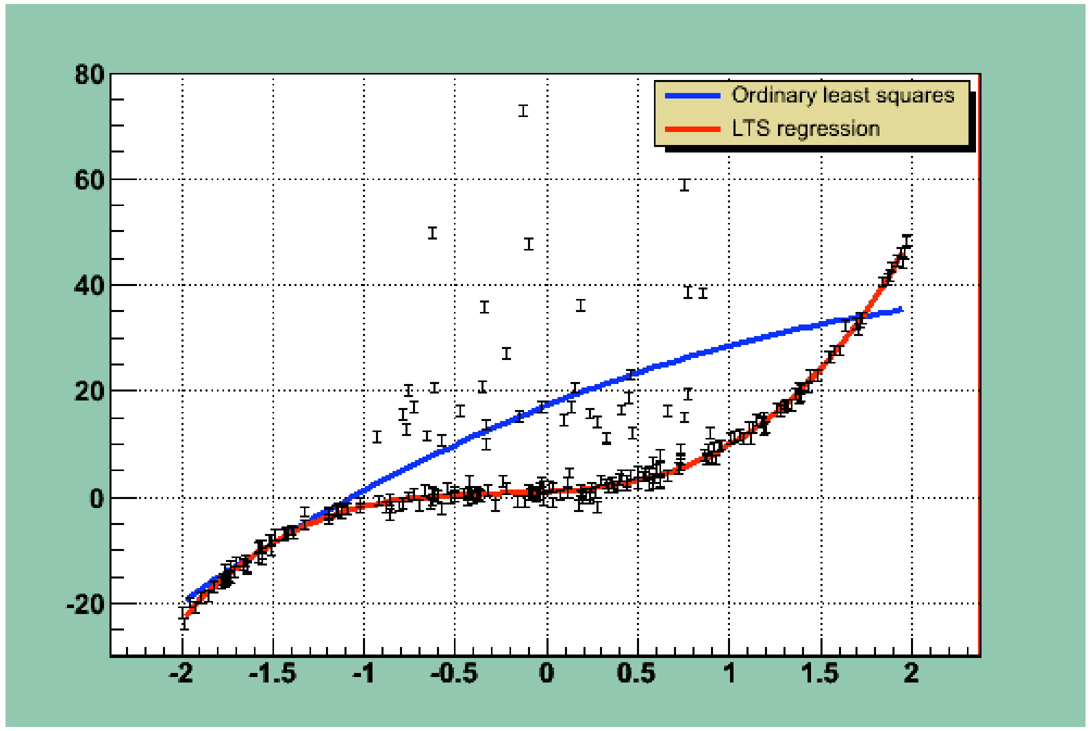


complex BaBar fitting provided by A. Lazzaro

- Can have combination on both
  - parallelization via multi-thread in a multi-core CPU
  - multiple process in a distributed computing environment

Linear Fitting

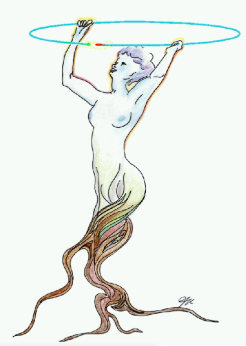
- `TLinearFitter` class to fit functions linear in the parameters (e.g Polynomial)
  - direct solution by solving a linear system
  - can be 10-15 times faster than Minuit
  - can also be used for a fast Gaussian or Exponential fit using a log scale



- Robust Fitting
  - outliers removal
  - use of Least Trimmed Square (LTS) regression

```
Graph.Fit("pol3", "rob=0.75", -2, 2);
```

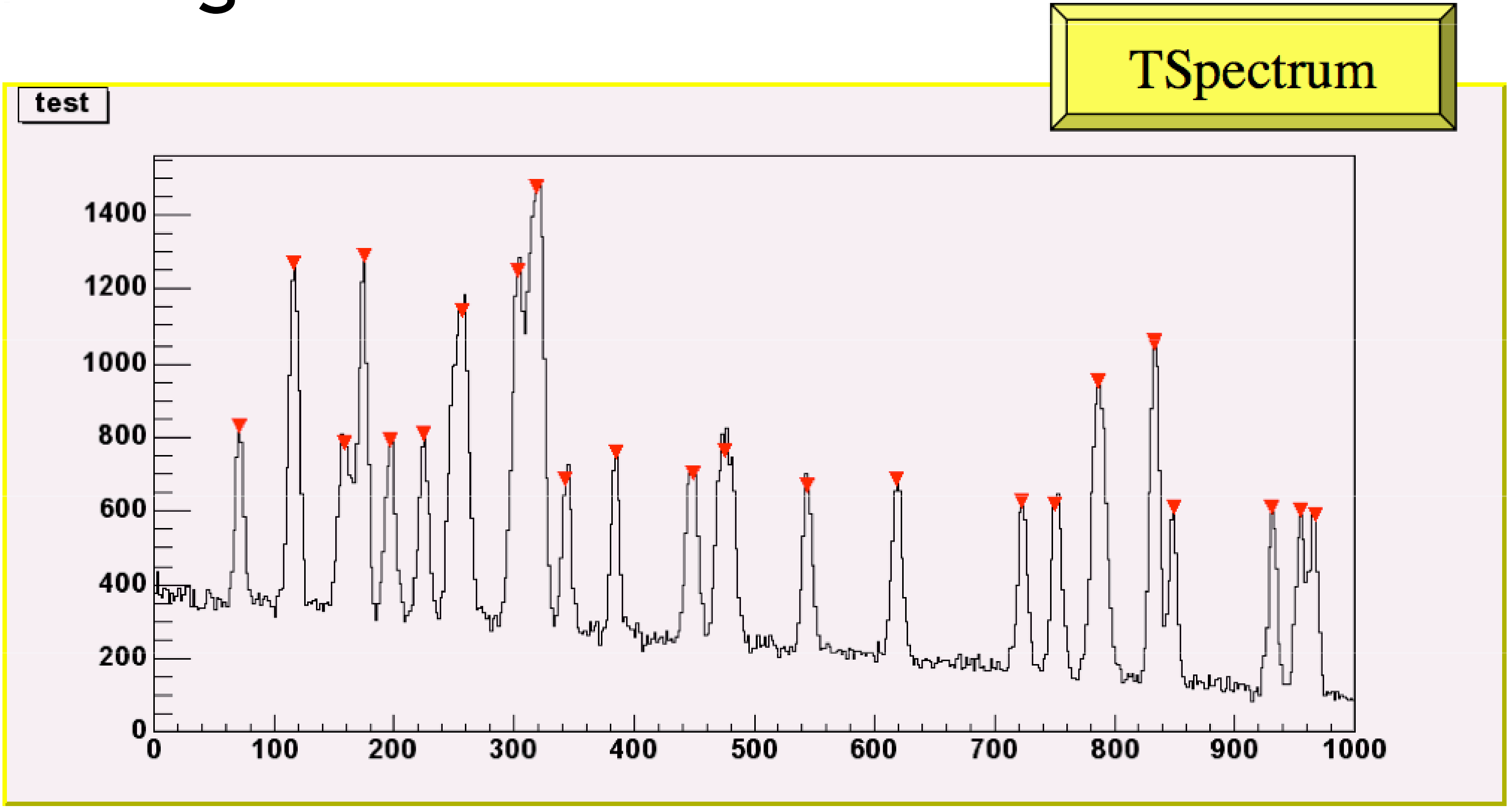
Classes for Specialized Fits



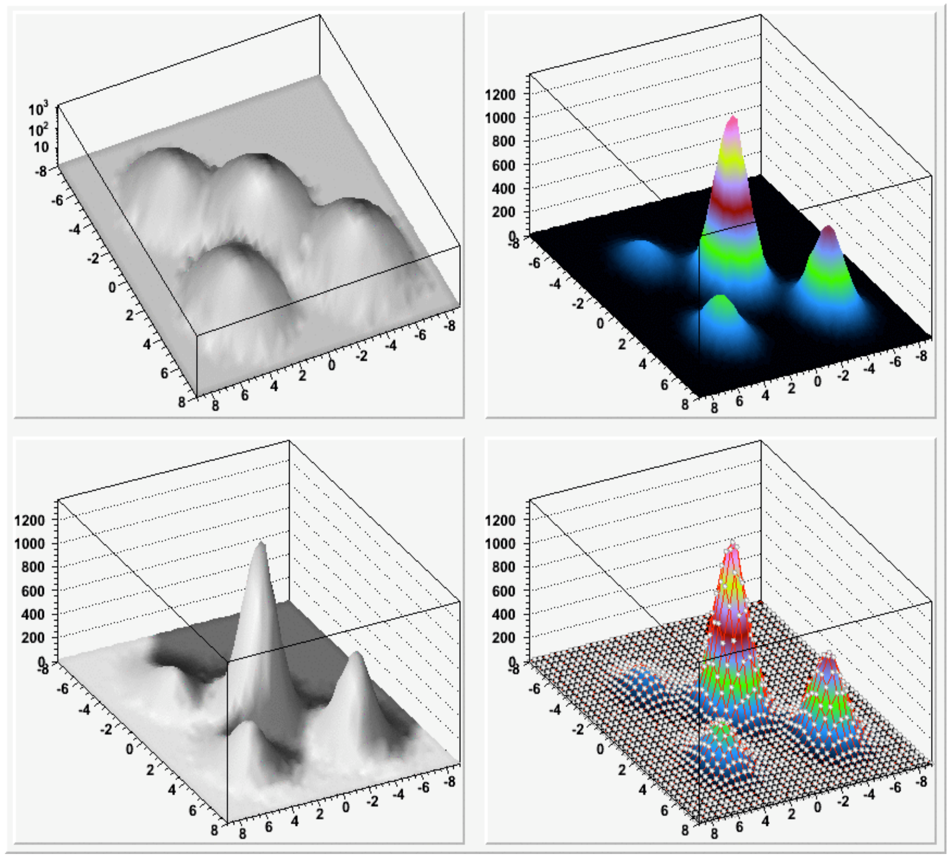
- *TBinomialEfficiencyFitter:*
  - likelihood fit for efficiencies (data with binomial errors)
    - obtained from division of two histograms
    - `TGraphAsymErrors::BayesDivide` for getting Bayesian errors from divisions of two histograms
- *TFractionFitter*:
  - likelihood fits to Data and MC predictions
    - method by *R. Barlow and C. Beeston, Comp. Phys. Comm. 77 (1993) 219-228*
    - implemented by F. Filthaut
- *TSplot*:
  - extended maximum likelihood fit to signal and background with a tool (*SPlot*) to access the validity of the fit (unbias distribution of control variables)

Spectrum

- `TSpectrum` class for peak finding and background subtraction



- `TSpectrum2` for 2-dimensional spectra

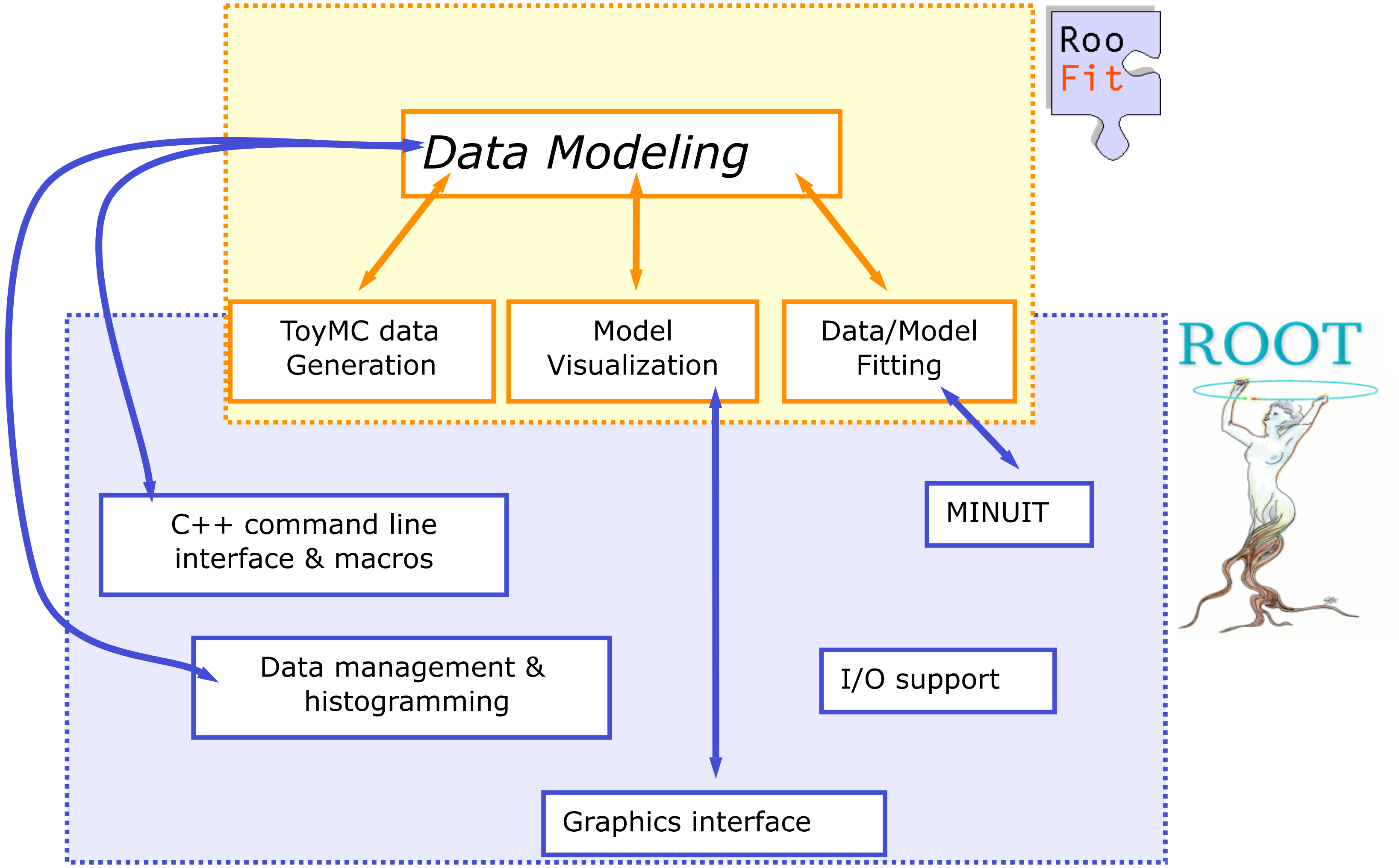


RooFit

- Toolkit for data modeling
  - Model distribution of observable x in terms of
    - parameter of interest p
    - other parameters q to describe detector effects (resolution , efficiency)
  - ➡ Probability density function (pdf) $F(x;p,q)$
    - normalized over range of observable x w.r.t. the parameters p and q
- *RooFit* provides the functionality for
  - building these probability density functions
    - scalable to complex models
  - maximum likelihood fitting (binned and unbinned)
  - visualization of the pdf
  - toy MC generator

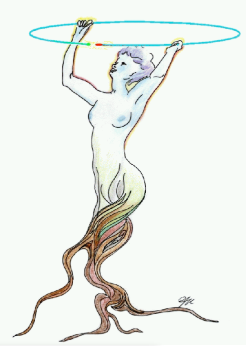
RooFit Functionality

- Package extending the ROOT functionality



Package developed, originally for BaBar analysis (by W. Verkerke and D. Kirkby)

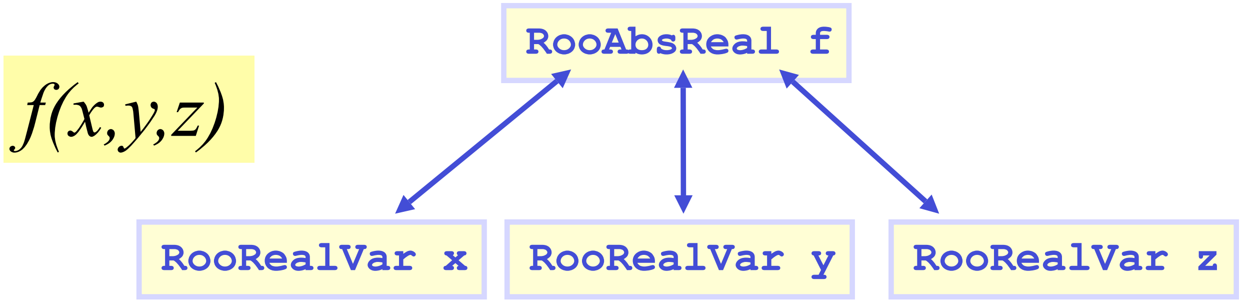
- actively maintained by W. Verkerke in view of LHC analysis
- Web site: http://roofit.sourceforge.net/
- many material begin shown taken from Wouter's presentations
  - see 200 slides presented at French statistics school (http://sos.in2p3.fr)



RooFit Design

- Mathematical concepts are represented as C++ objects

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



$f(x,y,z)$

RooAbsReal f

RooRealVar x

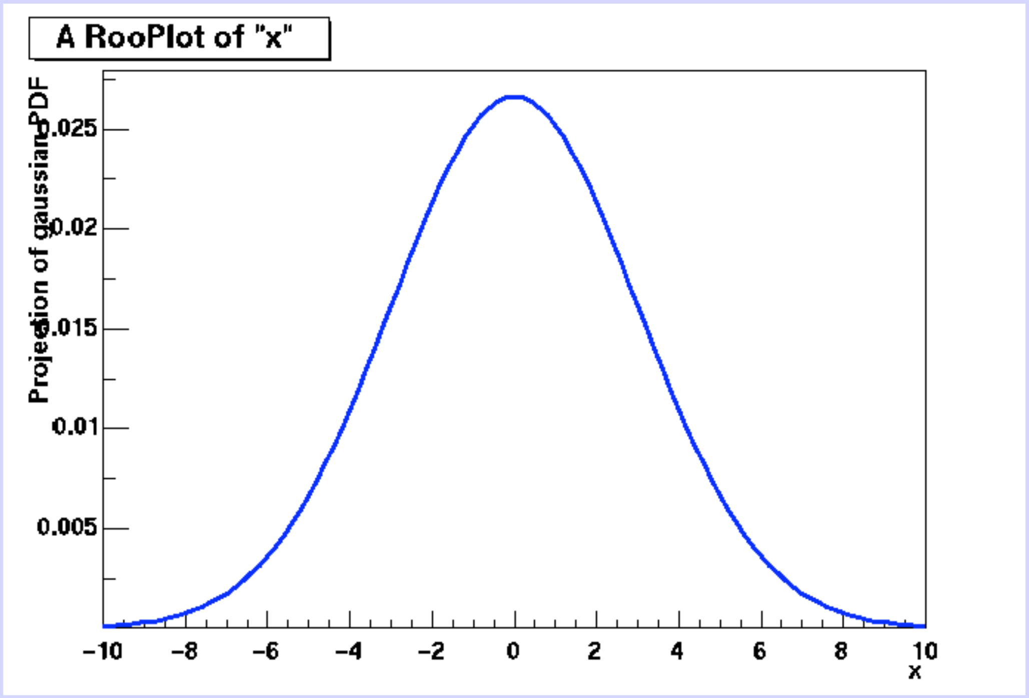
RooRealVar y

RooRealVar z

RooFit Example

- Gaussian pdf:

```
// Build Gaussian PDF
RooRealVar x("x","x",-10,10) ;
RooRealVar mean("mean","mean of gaussian",0,-10,10) ;
RooRealVar sigma("sigma","width of gaussian",3) ;

RooGaussian gauss("gauss","gaussian PDF",x,mean,sigma) ;

// Plot PDF
RooPlot* xframe = x.frame() ;
gauss.plotOn(xframe) ;
xframe->Draw() ;
```

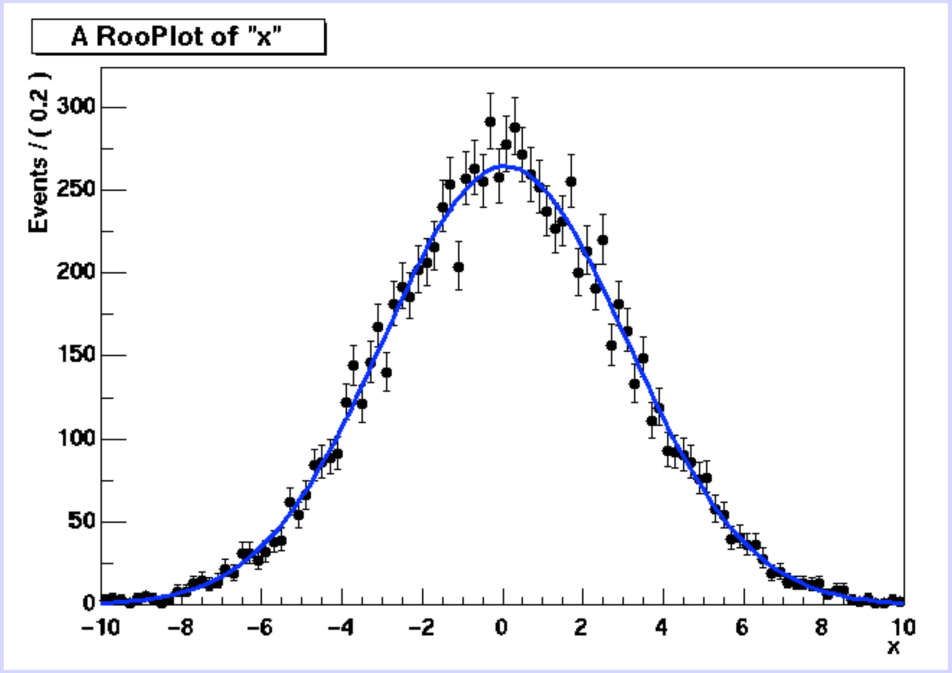


- MC data generation:

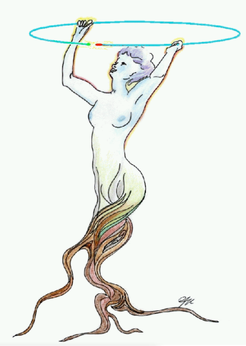
```
// Generate a toy MC set
RooDataSet* data = gauss.generate(x,10000) ;
```

- Maximum likelihood fit to the data

```
// ML fit of gauss to data
gauss.fitTo(*data) ;
// Plot fitted PDF and toy data overlaid
RooPlot* xframe2 = x.frame() ;
data->plotOn(xframe2) ;
gauss.plotOn(xframe2) ;
xframe2->Draw() ;
```



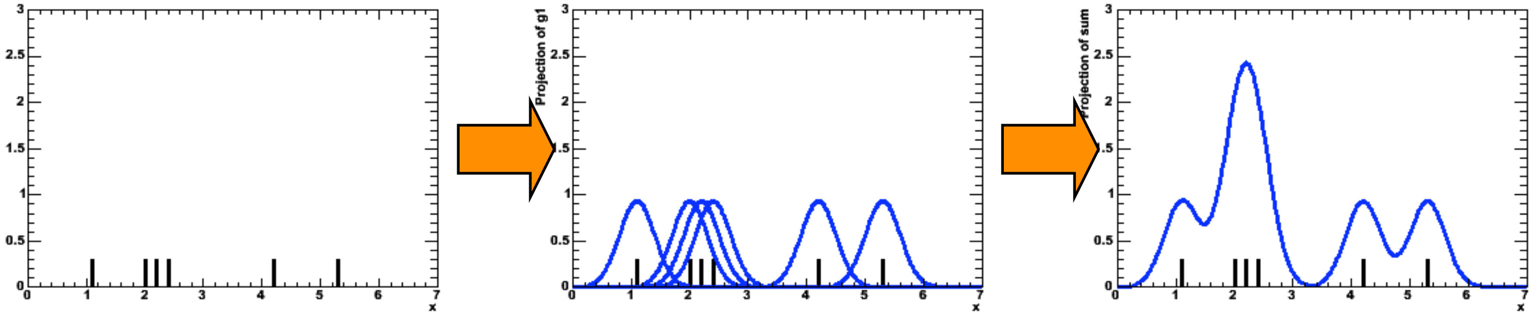
Model Building



- Complex model building for physics model
  - composite pdf by re-using standard components
    - addition (`RooAddPdf`) $S(x) = \alpha F(x) + (1-\alpha)G(x)$
    - product (`RooProdPdf`) $h(x,y) = f(x) \times g(y)$
    - composition $h(x,y) = h(x, f(y))$
    - convolution $h(x) = f(x) \otimes g(x) = \int f(x) g(x-t) dt$
  - each pdf type is a separate C++ classes
  - standard pdf classes
    - gaussian, exponential, polynomial, Chebychev polynomial, etc..
  - user provided pdf
    - based on simple expression (as `TFormula` class)
    - from a supplied C++ code (*MyPdf.cxx* and MyPdf.h)
  - non-parametric pdf (obtained from density estimators)
    - histogram PDF, kernel estimator (also in multi-dim)

Kernel Estimation

- Estimation of pdf directly from data:
  - from binned data (histogram estimation)
  - from un-binned data (kernel estimation)



Sample of events

Gaussian probability distributions for each event

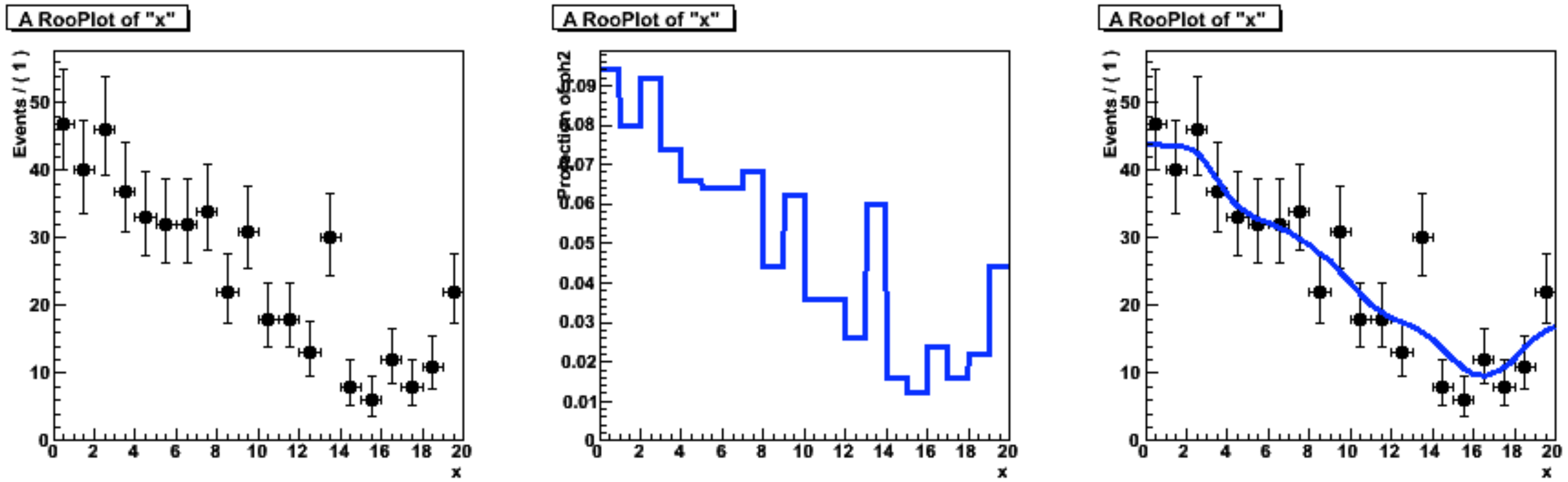
Summed probability distribution for all events in sample

- adaptive kernel estimation (gaussian width depending on density)

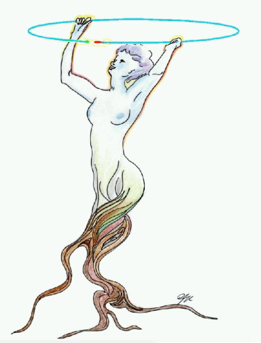
Data (N=500)

RooHistPdf(data)

RooKeysPdf(data)



Pdf Normalization



- Major effor in writing a pdf is to ensure that is normalized to 1
  - *RooFit* provides automatic normalization of pdf
- Normalization handled by a `RooRealIntegral` class
  - whenever possible can perform analytical calculations of integral expressions
  - support various numerical integration techniques
    - adaptive trapezoid, Gauss-Kronrod, MC (Vegas)
    - use of Fast Fourier Transforms for convolution integrals
      - much better numerical stability during minimization
- No intrinsic distinction between observable and parameters

- `gauss.getVal(x)`

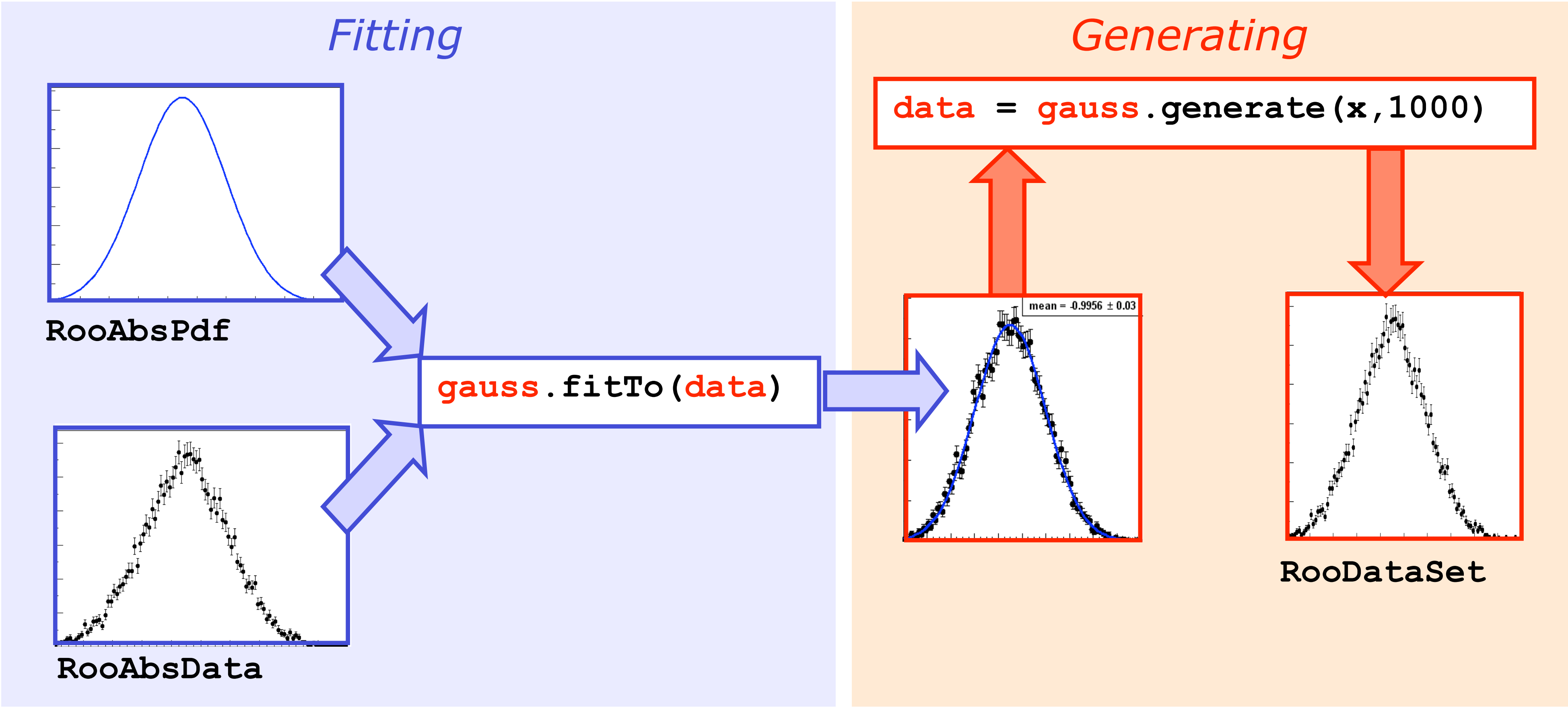
$$g(\mathbf{x};m,s)=\frac{g(x,m,s)}{\int_{x_{\min}}^{x_{\max}} g(x,m,s)dx}$$

- `gauss.getVal(s)`

$$g(\mathbf{s};m,x)=\frac{g(x,m,s)}{\int_{s_{\min}}^{s_{\max}} g(x,m,s)ds}$$

RooFit Models

- All *RooFit* model (pdf classes) provide functionality for fitting and generating toy Monte Carlo (sampling of the pdf)
  - complexity will be limited by memory and CPU power



Fitting

RooAbsPdf

RooAbsData

```
gauss.fitTo(data)
```

Generating

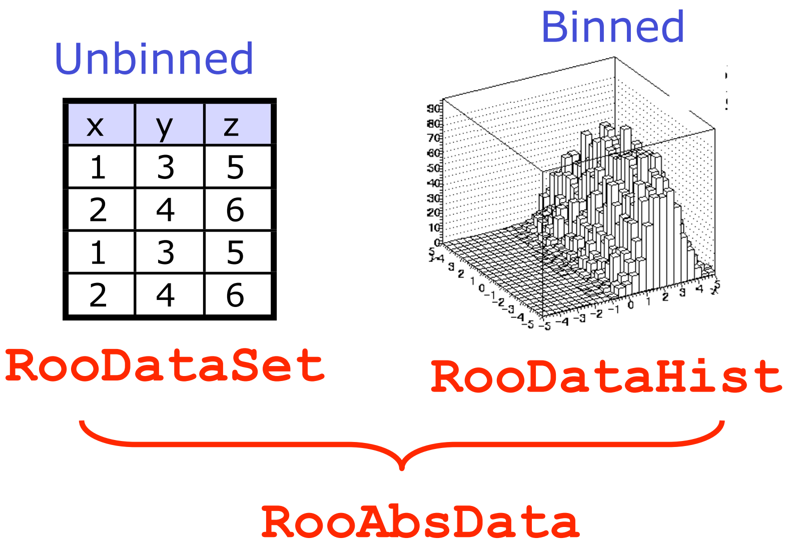
```
data = gauss.generate(x,1000)
```

RooDataSet

- Model Visualization
  - possibility to plot each selected component

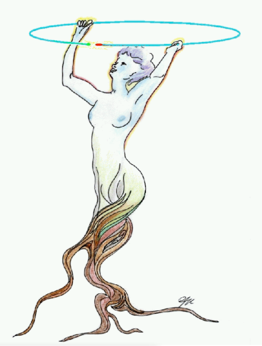
RooFit Fitting Functionality

- Support for both un-binned and binned fits
  - `TTree` (unbin data): unbinned likelihood fits
  - Histograms (bin data): binned fits
    - least square fits
    - likelihood fits with Poisson probabilities
- Support for extended maximum likelihood fits
  - fit also number of expected events to the ones observed
- Support for simultaneous fits (multiple data-sets)
- Manage also discrete variables (category)
- Use *MINUIT* for minimization of objective (likelihood or least square) function
  - class `RooMinuit` based on `TMinuit`



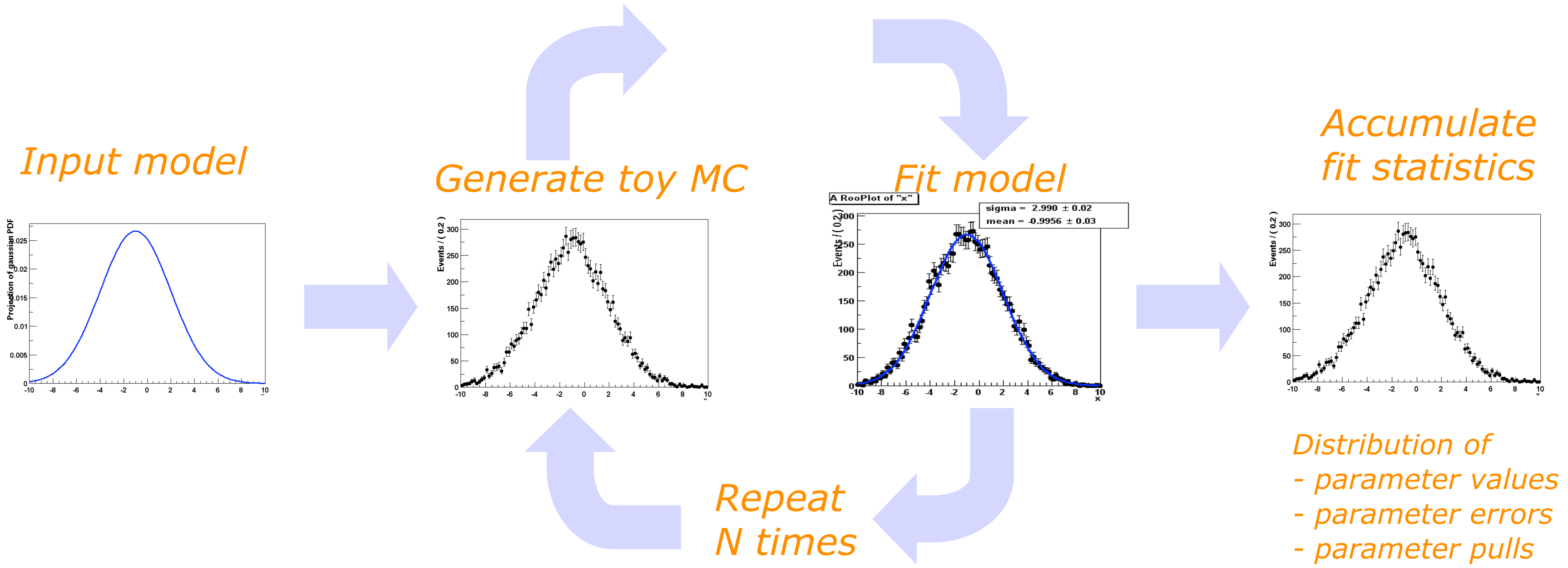
Unbinned

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



Other RooFit Features

- Automatic optimization of pdf evaluation
  - detection and pre-calculation of constant-terms
  - lazing evaluation and function caching
  - factorization of multi-dimensional problems (whenever possible)
- Parallelization of fit objective function on multiple hosts
  - evaluate likelihood components in separate processes or threads
- Support for task automation (fit validation)



Input model

Generate toy MC

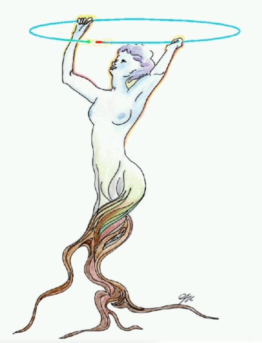
Fit model

Accumulate fit statistics

Repeat N times

Distribution of
- parameter values
- parameter errors
- parameter pulls

RooFit Tutorials

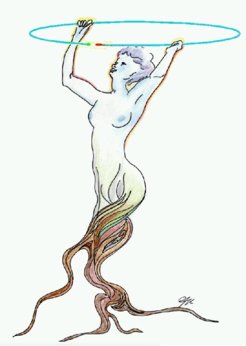


- A large collection of tutorials/examples available in ROOT svn:

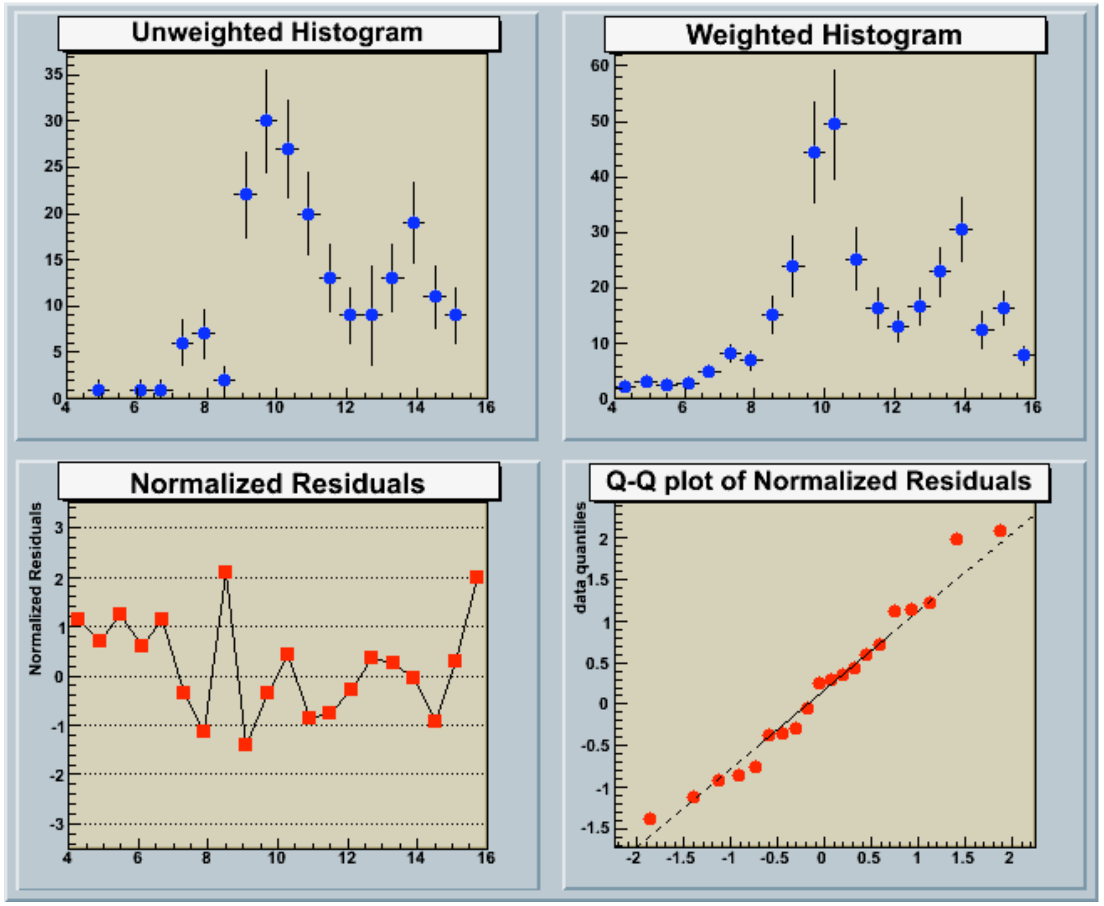
$ROOTSYS/tutorials/roofit

rf101_basics.C
rf102_dataimport.C
rf103_interprfuncs.C
rf104_classfactory.C
rf105_funcbinding.C
rf106_plotdecoration.C
rf107_plotstyles.C
rf108_plotbinning.C
rf109_chi2residpull.C
rf110_normintegration.C
rf111_numintconfig.C
rf201_composite.C
rf202_extendedmlfit.C
rf203_ranges.C
rf204_extrangefit.C
rf205_compplot.C
rf206_treevistools.C
rf207_comptools.C
rf208_convolution.C
rf209_anaconv.C
rf301_composition.C
rf302_utilfuncs.C
rf303_conditional.C
rf304_uncorrprod.C
rf305_condcorrprod.C
rf306_condpereventerrors.C
rf307_fullpereventerrors.C
rf308_normintegration2d.C
rf309_ndimplot.C
rf310_sliceplot.C
rf311_rangeplot.C
rf312_multirangefit.C
rf313_paramranges.C
rf314_paramfitrange.C
rf315_projectpdf.C
rf316_llratioplot.C
rf401_importttreethx.C
rf402_datahandling.C
rf403_weightedevts.C
rf404_categories.C
rf405_realtocatfuncs.C
rf406_cattocatfuncs.C
rf407_latextables.C
rf501_simultaneouspdf.C
rf502_wspacewrite.C
rf503_wspaceread.C
rf504_simwstool.C
rf505_asciicfg.C
rf505_asciicfg.txt
rf506_msgservice.C
rf507_debugtools.C
rf508_listsetmanip.C
rf601_intminuit.C
rf602_chi2fit.C
rf603_multicpu.C
rf604_constraints.C
rf605_profilell.C
rf606_nllerrorhandling.C
rf607_fitresult.C
rf701_efficiencyfit.C
rf702_efficiencyfit_2D.C
rf703_effpdfprod.C
rf704_amplitudefit.C
rf705_linearmorph.C
rf706_histpdf.C
rf707_kernelestimation.C
rf708_bphysics.C
rf801_mcstudy.C
rf802_mcstudy_addons.C
rf803_mcstudy_addons2.C
rf804_mcstudy_constr.C

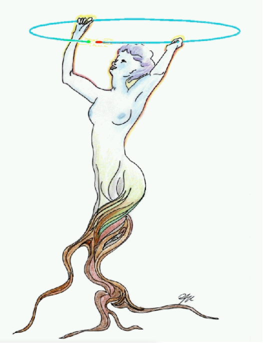
Goodness of Fit Tools



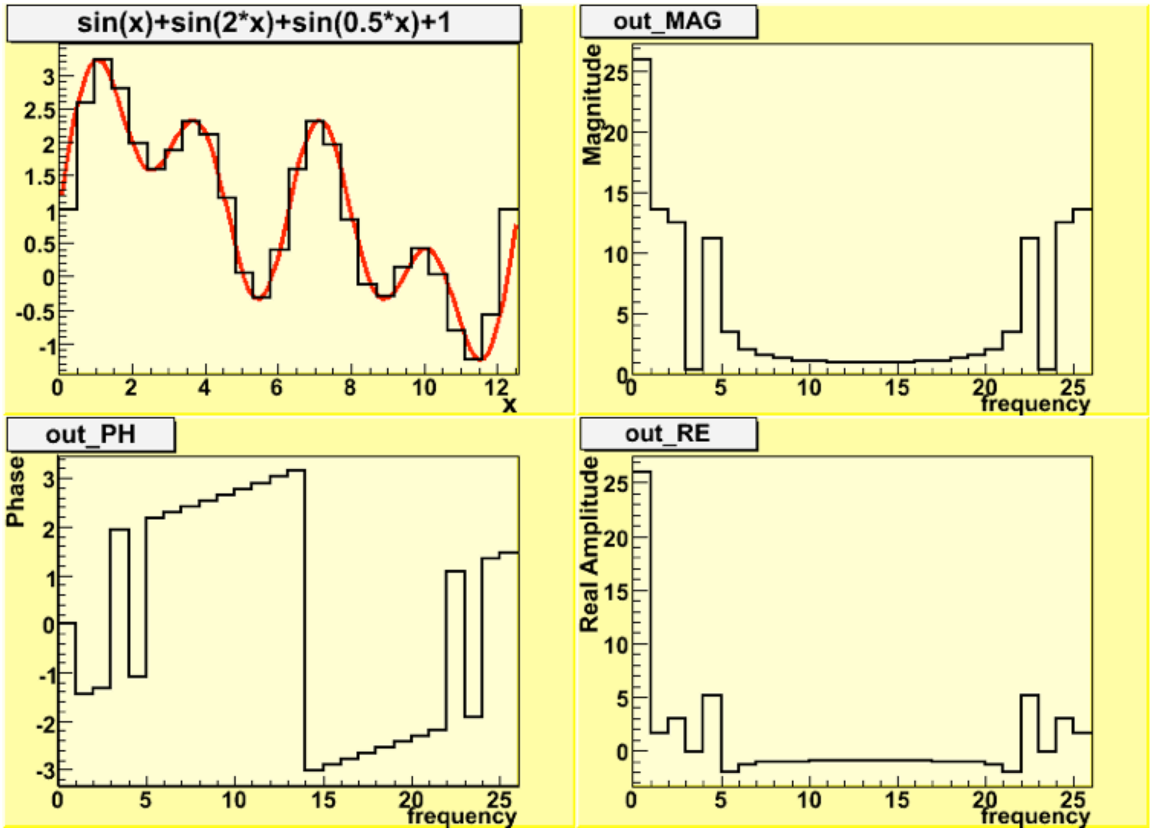
- Pearson Chi2 test for histogram comparison (`TH1::Chi2Test`)
  - updated with algorithm from *N. Gagunashvili*
    - weighted histograms comparisons
    - histograms with different scales
    - produce also normalized residuals
- Kolmogov-Smirnov test
  - for un-binned data
    - `TMath::KolmogorovSmirnov(`
  - have function also for bin data (`TH1::KolmogorovSmirnov`)
    - to be used with care
      - biased results towards larger probability
      - reduced if bin size is small enough



Fast Fourier Transform



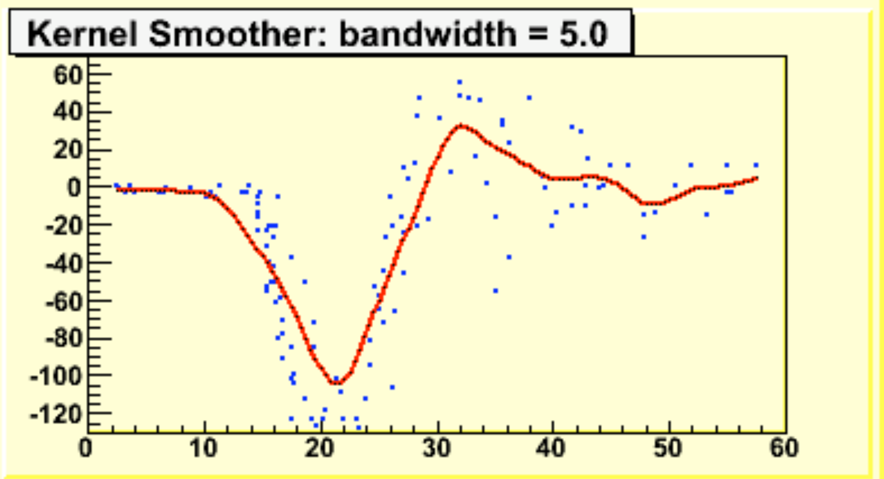
- Included in ROOT a common base class (`TVirtualFFT`)
  - add a functions to use it from `TH1` (`TH1::FFT`)
- Implemented an interface to the popular *FFTW* package (see www.fftw.org)
  - support for one and multi-dimensional transforms
  - support for complex and real transformations

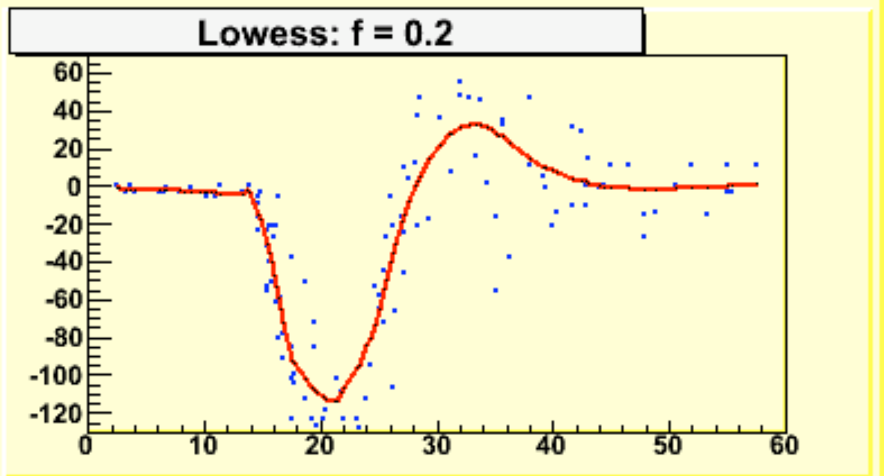


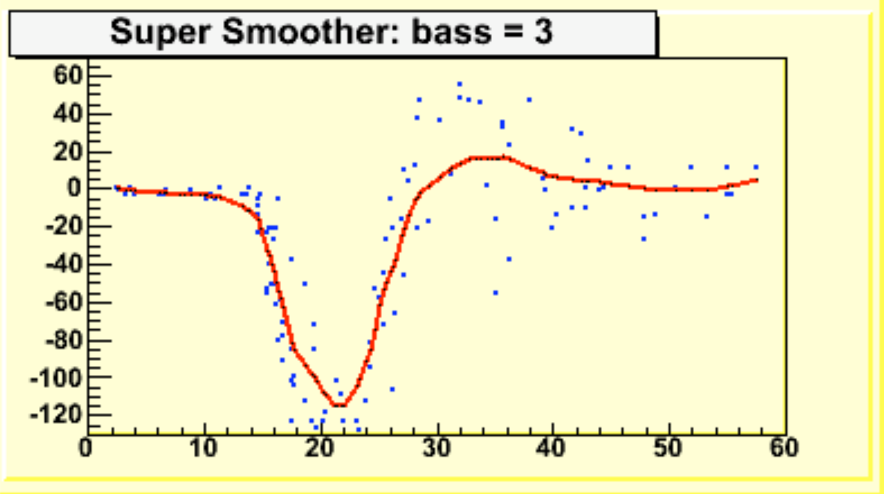
- `TFFTComplex` for complex input/ complex output transforms
- `TFFTRealComplex` for real input/ complex output
- `TFFTComplexReal` for complex input/real output
- `TFFTReal` for real input/output

Graphs Smoothing

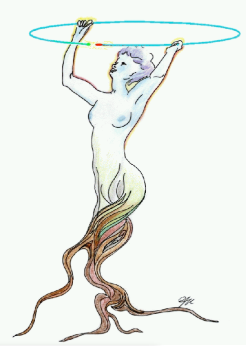
- Cubic and Quintic splines via `TSpline3,5` classes
- Smoothers of (x,y) data via the class `TGraphSmooth`
  - find regression function y(x)
  - algorithms from *R* software package
    - Kernel Smoother
    - Lowess Smoother
    - Super smoother (from Friedman)
- Plan to extend it for multi-dimensional data
  - for iso-surfaces $z(x_1,\ldots x_{n-1})$
- Add smoothing for 1D unbin data (kernel density estimator)







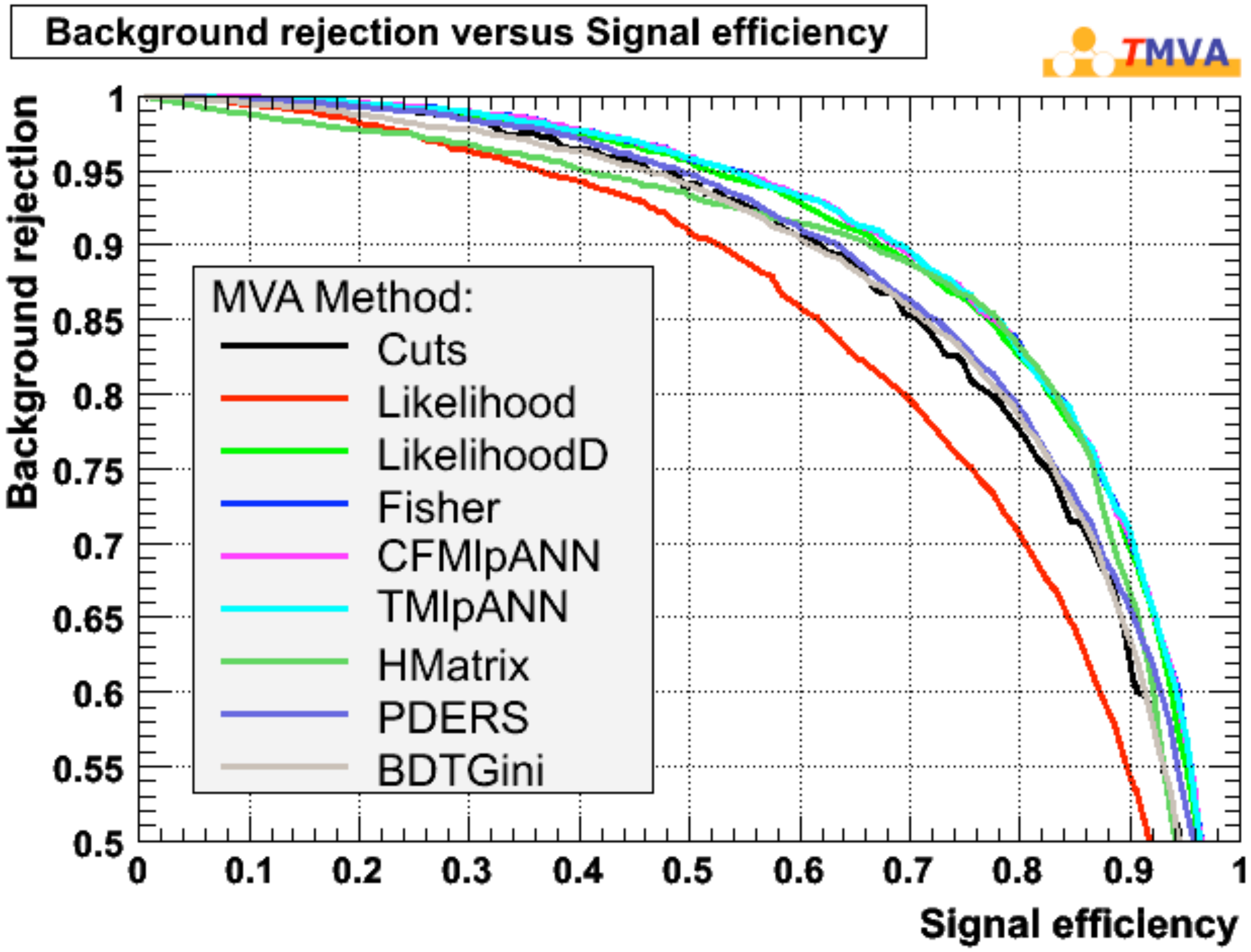
MultiVariate Methods



- Neural networks via the `TMultiLayerPerceptron` class
  - can be used for classification or for regression analysis
- `TMultiDimFit` for function approximation
  - find parametrization of multidimensional data using polynomials (or Chebyshev or Legendre)
    - example: LHCb magnetic field map
- `TPrincipal` : principal component analysis
  - linear transformation of variables
- *TMVA* : toolkit for multivariate analysis
  - various tools for multi-variate signal/background discrimination
  - implementing also possibility for performing regression

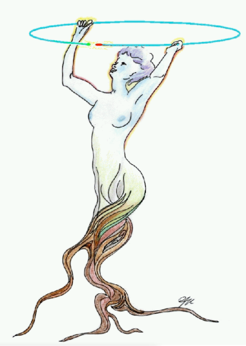
TMVA

- Package for multivariate analysis distributed within ROOT (from *A. Höcker, P. Speckmayer, J. Stelzer, F. Tegenfeldt, H. Voss, K. Voss, et al.)*
- Provides various methods for signal/background discrimination:



- Rectangular cut optimization
- Projective likelihood estimator
- Multi-dimensional probability density estimators
- Multi-dimensional kNN classifiers
- Linear discriminant analysis (Fisher and H-matrix)
- Function discriminant analysis
- Artificial Neural network (3 different implementations)
- Boosted/Bagged decision trees
- Support Vector Machines (SVN)
- RuleFit

Confidence Intervals



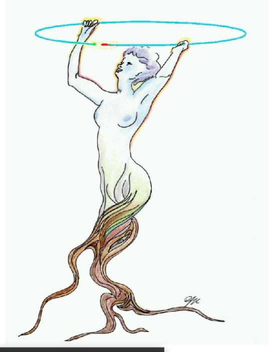
- Classes for confidence level estimation:
  - `TFeldmanCousins`
    - Feldman-Cousins confidence intervals for a Poisson process (use likelihood-ratio ordering rule)
      - without uncertainties in signal or background
  - `TRolke`
    - profile likelihood for Poisson process
      - with uncertainty in background and/or signal

$$PL(p) = \frac{L(p,\hat{q})}{L(\hat{p},\hat{q})}$$

  - `TLimit`
    - $CL_s = CL_{s+b}/CL_b$ method used at LEP
      - apply to histograms of data and MC (signal + background)
      - can incorporate systematic uncertainties

RooStat

- New statistical framework with tools aimed for discoveries
  - significance, limit, confidence levels
  - common framework for calculations using various tools and techniques
    - convenient for comparing different methods
  - provide possibility of combination of results of multiple measurements
  - support for both frequentists and bayesian statistical tools
    - also within these classes several techniques exist
    - useful for comparing differing methods
      - study coverage, incorporation of systematic errors, etc..
  - all methods basically start with probability density function (pdf) and/or the likelihood function
- Contributions from both ATLAS and CMS

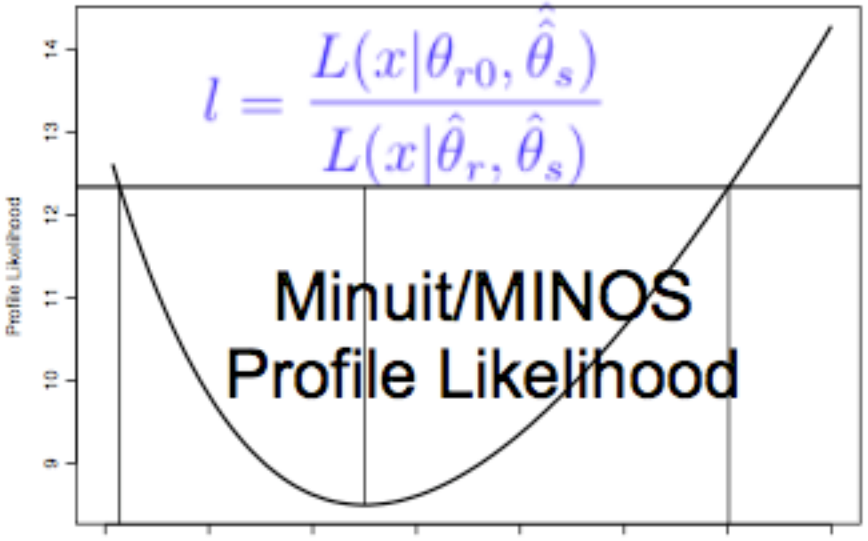


RooStat Methods

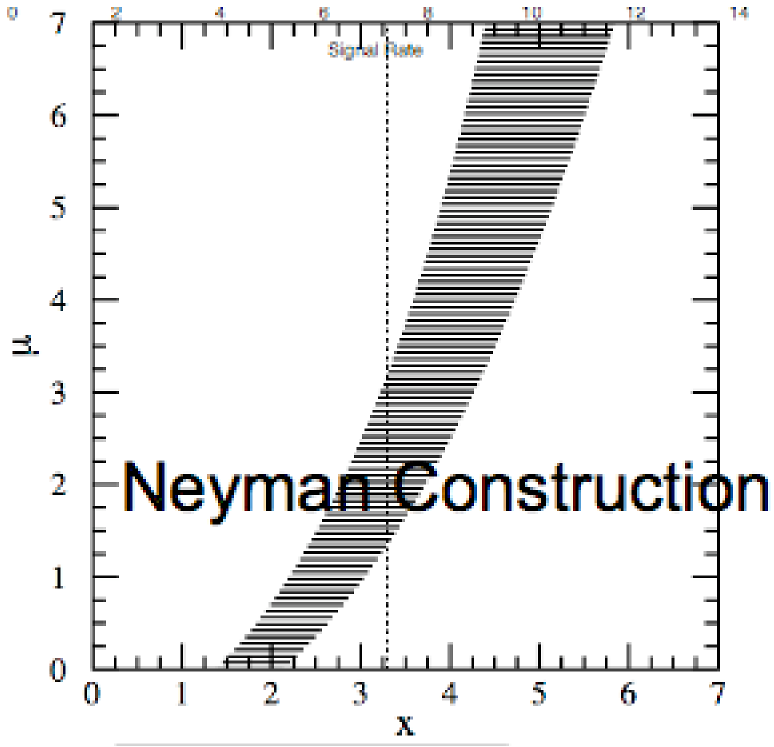
- Natural to build on top of *RooFit* toolkit
  - *RooFit* modeling of is very convenient
    - no static notion of parameter and observable
    - can work naturally with both frequentist and bayesian methods
- Major statistical techniques aimed to be in *RooStat* for limit and confidence interval calculations:
  - Profile likelihood ratio
  - Neyman constructions
  - Bayesian posterior

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

Bayes Theorem



Minuit/MINOS Profile Likelihood



Neyman Construction

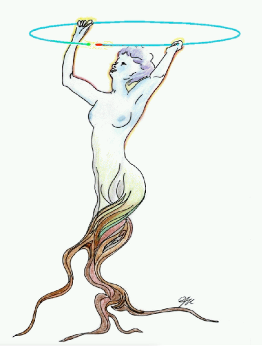
Workspace Concept

- Maintain a complete description of all the model
  - with possiblity to save entire model in a ROOT file
- WorkSpace class (`RooWorkSpace`)
  - probability density functions
  - (multiple) datasets
  - model configuration
- All information will be available for further analysis and combination of results
  - full likelihood for bayesian analysis
  - probability density functions for frequentists analysis
- Common format for combining and sharing physics results

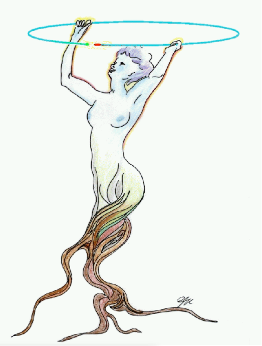
Conclusions

- Large collection of math and statistical tools currently available in ROOT
  - working recently on improving the overall quality and better usability
  - improving modularity and trying to remove duplications
- Considerable efforts from external contributors in developing tools for physics analysis
  - complex fitting (*RooFit*)
  - multivariate analysis (*TMVA*)
  - new statistical framework for discovery, confidence levels and result combination (*RooStat*)
- Important to ensure the correctness of tools we are using
  - large effort on improving validation and test suites
  - good documentation for reference and maintainability
- Need continuously the feedback from users

Documentation



- Online reference documentation
  - class description with *THtml* and *Doxygen*
    - see http://root.cern.ch/root/html520/MATH_Index.html
- ROOT User guides (from http://root.cern.ch/root/doc/RootDoc.html)
  - New Math chapter in the ROOT User Guide (chapter 13)
  - *RooFit* User Guide (being updated)
  - *TMVA* User Guide
- *Minuit2* User Guide
  - see http://www.cern.ch/mathlibs/documents/minuit/mnusersguide.pdf
- Maintain inventory of all Math functions and algorithms with links to ROOT online doc and CERNLIB
  - http://root.cern.ch/twiki/bin/view/ROOT/MathTable



References

- *ROOT* User Guide: http://root.cern.ch/root/doc/RootDoc.html
- *ROOT* reference guide: http://root.cern.ch/root/htmldoc/ClassIndex.html
- *MathCore* online doc: http://www.cern.ch/mathlibs/sw/MathCore/html/index.html
- *MathMore* online doc: http://www.cern.ch/mathlibs/sw/MathMore/html/index.html
- *Minuit2* online doc: http://www.cern.ch/mathlibs/sw/Minuit2/html/index.html
- *RooFit* homepage: http://roofit.sourceforge.net/
- *TMVA* homepage: http://tmva.sourceforge.net/
- Histogram comparison paper: http://arxiv.org/abs/physics/0605123
- *SPlot* paper: http://arxiv.org/abs/physics/0402083
- *UNURAN* homepage: http://statmath.wu-wien.ac.at/unuran/
- ROOT Talk Forum (for support, requests and discussions)
  - a thread is available for only Math and Statistical topics
- ROOT Savannah for reporting bugs