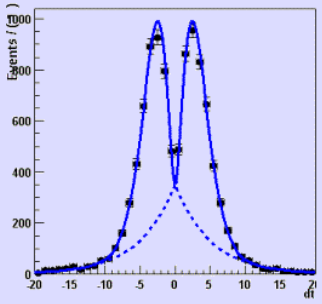
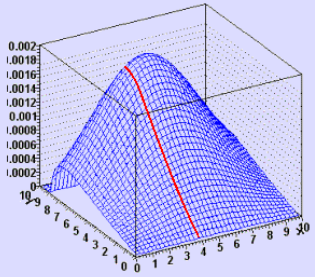
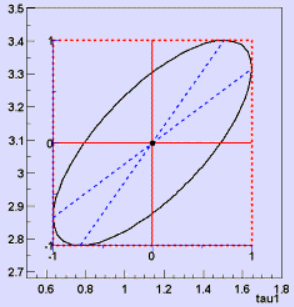






RooFit

A general purpose tool kit for data modeling

Wouter Verkerke (UC Santa Barbara)
David Kirkby (UC Irvine)

RooFit purpose - Data Modeling for Physics Analysis

Distribution of observables $\vec{x}$

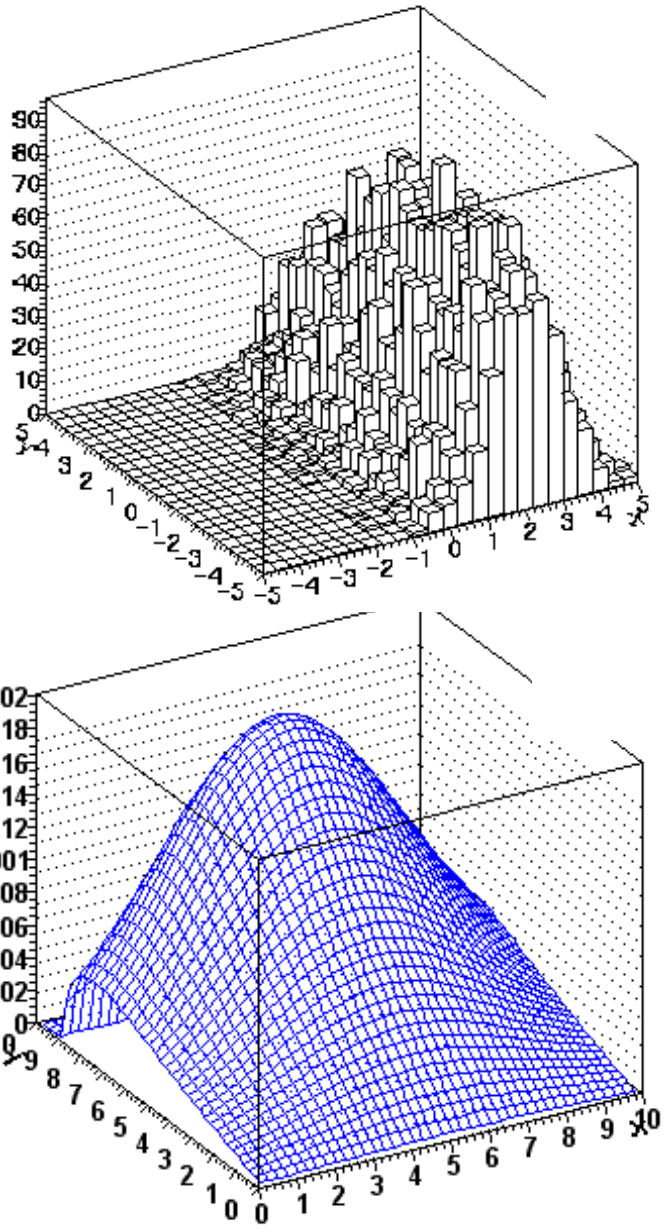
Define data model

Probability Density Function $F(\vec{x}; \vec{p}, \vec{q})$

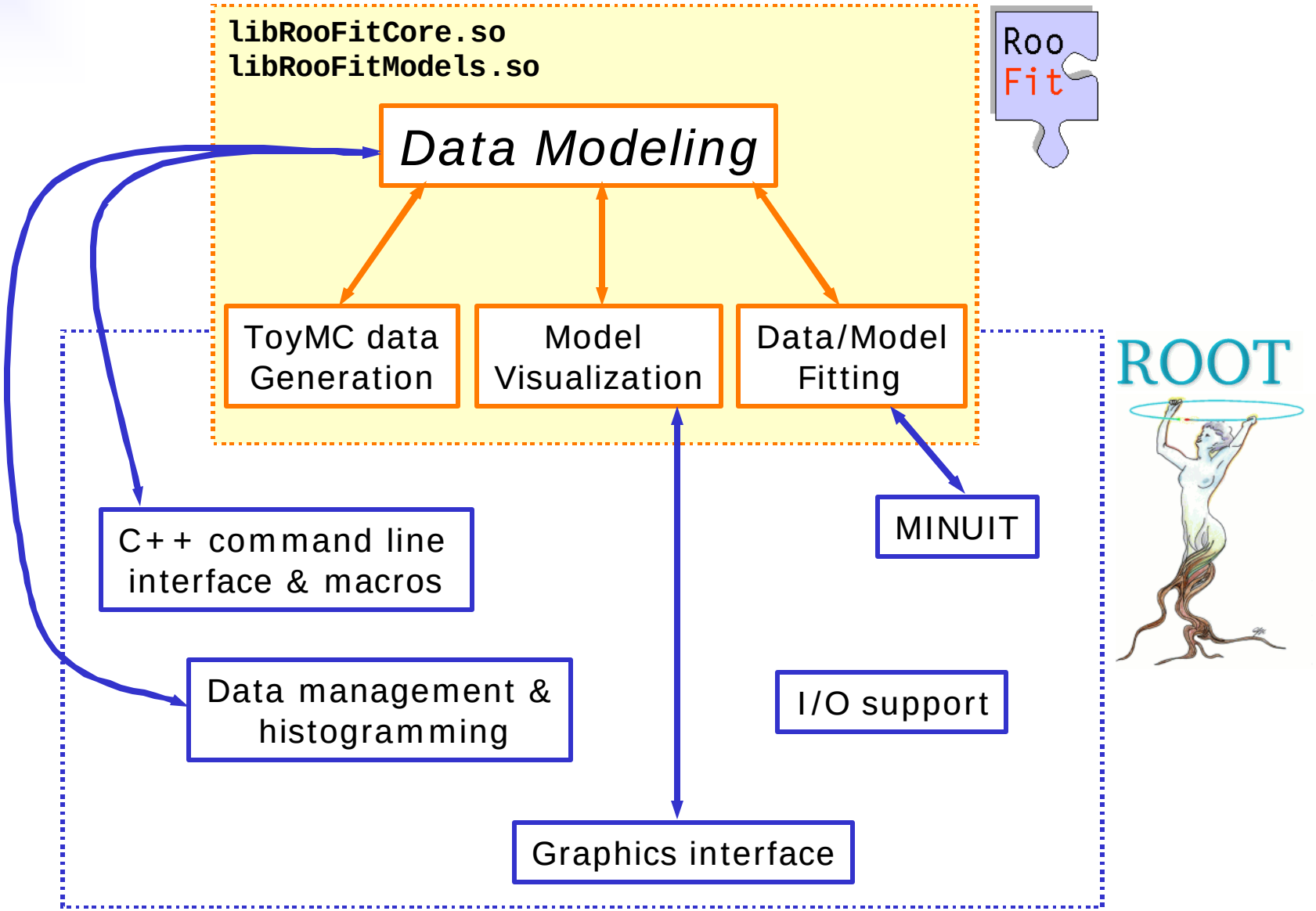
- Physical parameters of interest $\vec{p}$
- Other parameters $\vec{q}$ to describe detector effect (resolution,efficiency,...)
- Normalized over allowed range of the observables $\vec{x}$ w.r.t the parameters $\vec{p}$ and $\vec{q}$

Fit model to data

Determination of $\vec{p},\vec{q}$



Implementation – Add-on package to ROOT



Data modeling - Desired functionality

Analysis cycle

Building / Adjusting Models

- ✓ *Easy to write* basic PDFs (→ normalization)
- ✓ Easy to *compose complex models* (modular design)
- ✓ *Reuse* of existing functions
- ✓ *Flexibility* – No arbitrary implementation-related restrictions

Using Models

- ✓ Fitting : Binned/Unbinned (extended) MLL fits, Chi^2 fits
- ✓ Toy MC generation: Generate MC datasets from *any* model
- ✓ Visualization: Slice/project model & data in *any possible way*
- ✓ Speed – Should be *as fast or faster* than hand-coded model

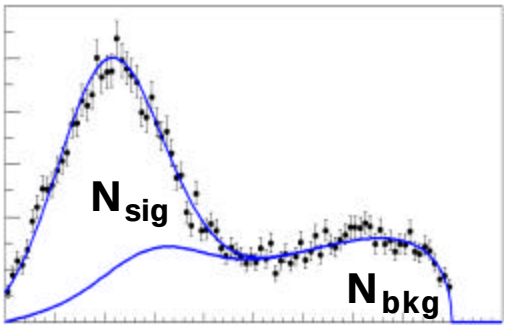
Data modeling – Mathematical formulation

- RooFit implements data models as *Probability Density Functions*

- Properties of PDFs
  - Unit normalized
  - Positive definite

$$\int_{\vec{x}_{\min}}^{\vec{x}_{\max}} F(\vec{x},\vec{p})\,d\vec{x} \equiv 1 \qquad F(\vec{x},\vec{p}) \geq 0$$

- Benefits
  - Easy interpretation of model parameters
    - $F_{sum}(x) = \mathbf{N_{sig}} * F_{sig}(x) + \mathbf{N_{bkg}} * F_{bkg}(x)$
  - Transparent modularity
    - Component properties context independent
    - Modular PDF structure scale easily to complex models
    - Sum of PDFs is a PDF, product of PDFs is PDF…
  - Universal Toy Monte Carlo event generation capabilities



Data modeling – OO representation

- Mathematical objects are represented 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**

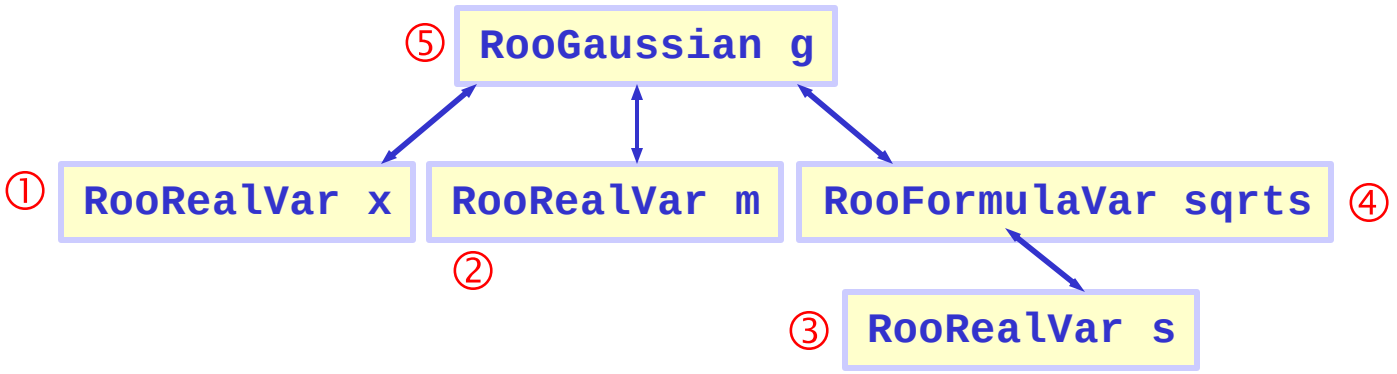
Data modeling – Constructing composite objects

- Straightforward correlation between mathematical representation of formula and RooFit code

Math

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

RooFit diagram



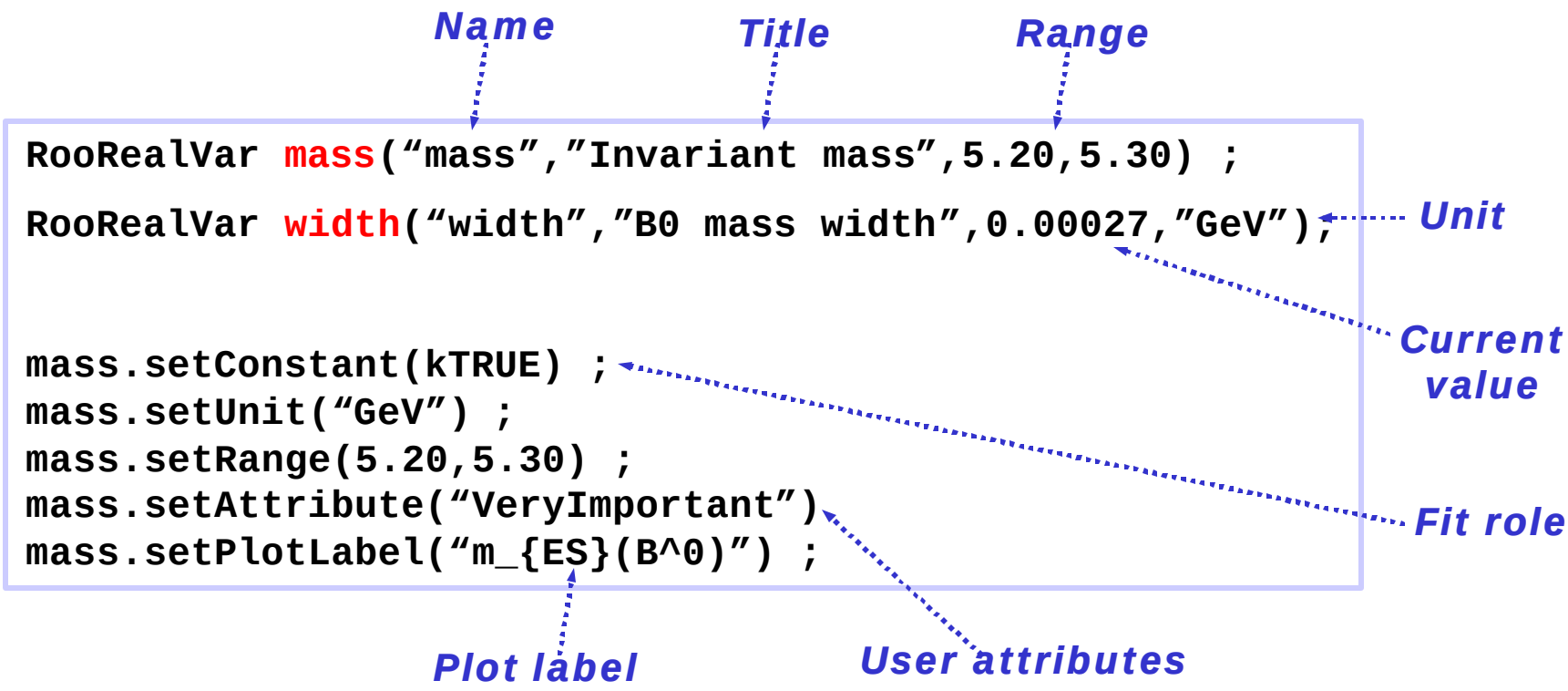
RooFit code

```
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) ;
```

Data modeling - Bookkeeping

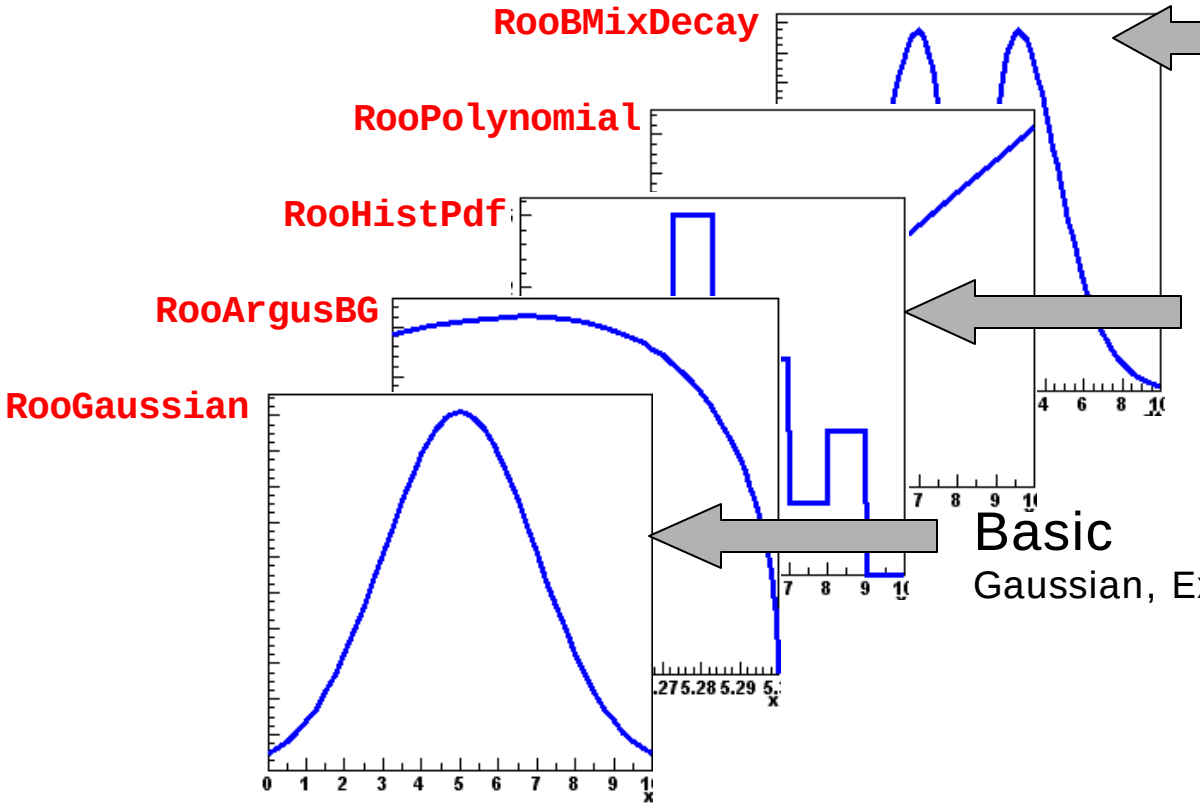
- All objects are *self documenting*

- Example: **RooRealVar** – representation of real-valued variable.
  - Associated properties are stored in objects



Model building – (Re)using standard components

- RooFit provides a collection of compiled standard PDF classes



Physics inspired
ARGUS,Crystal Ball, Breit-Wigner, Voigtian, B/D-Decay,....

Non-parametric
Histogram, KEYS

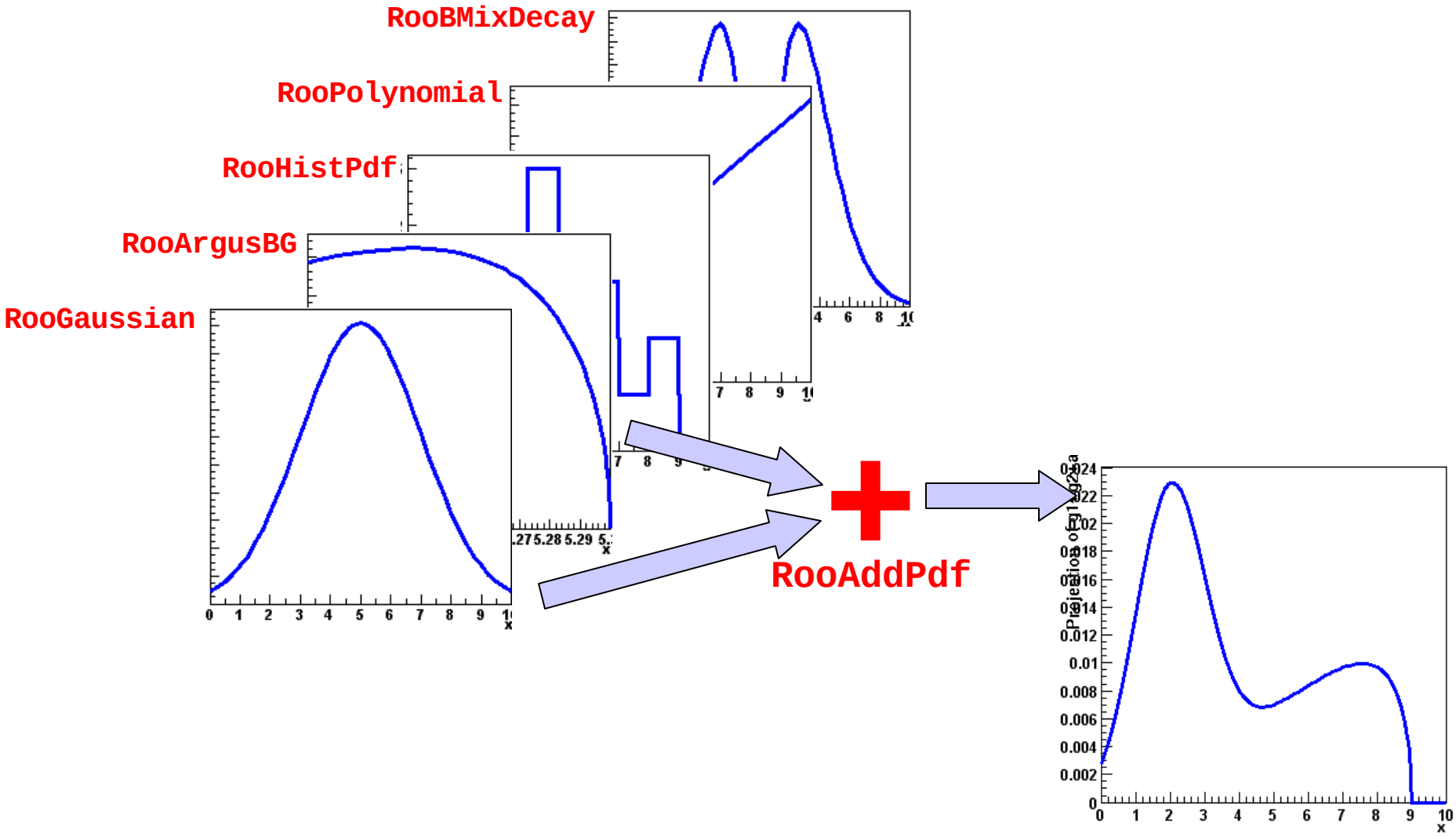
Basic
Gaussian, Exponential, Polynomial,...

PDF Normalization

- By default RooFit uses numeric integration to achieve normalization
- Classes can optionally provide (partial) analytical integrals
- Final normalization can be hybrid numeric/analytic form

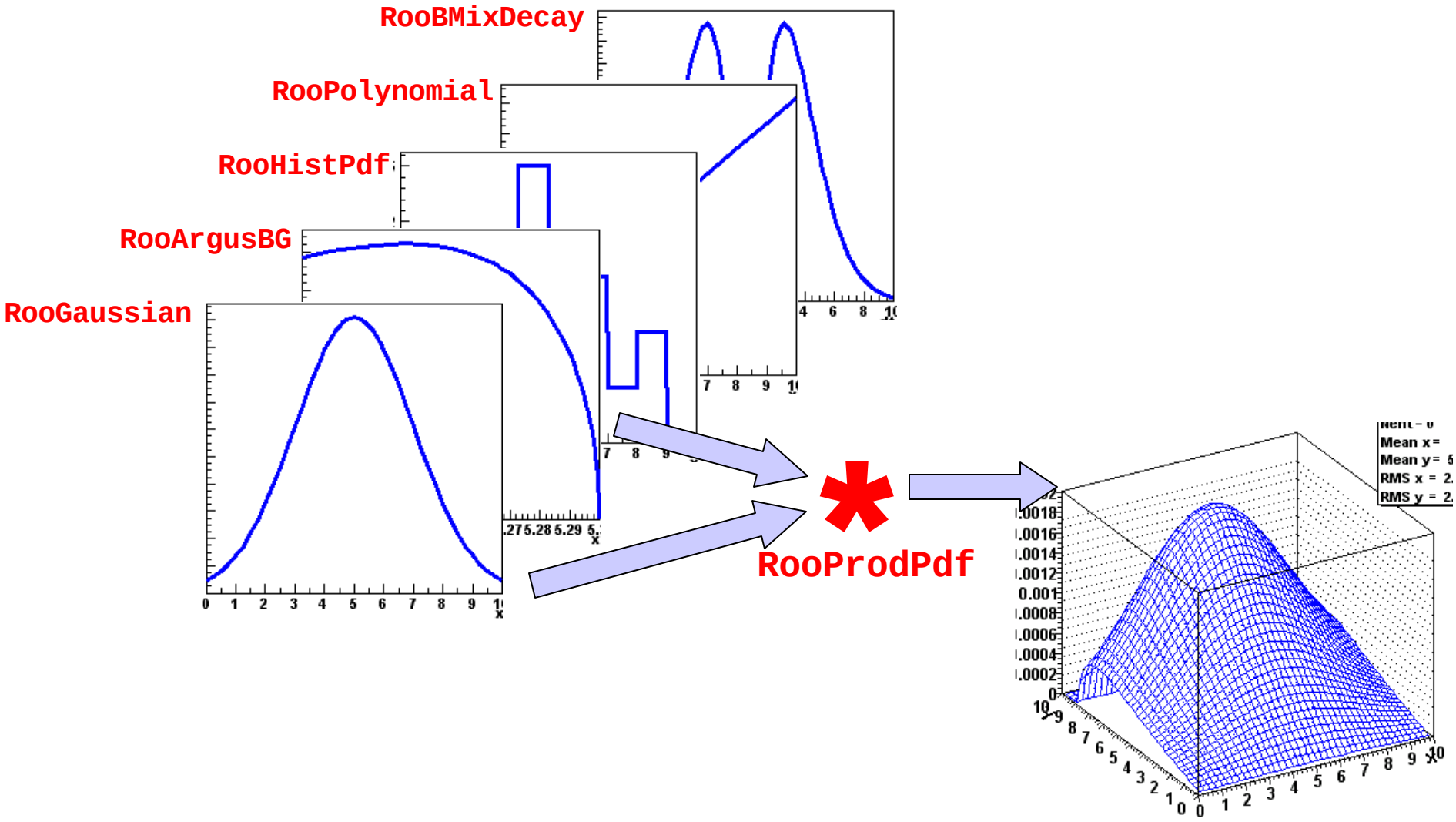
Model building – (Re)using standard components

- Most physics models can be composed from ‘basic’ shapes



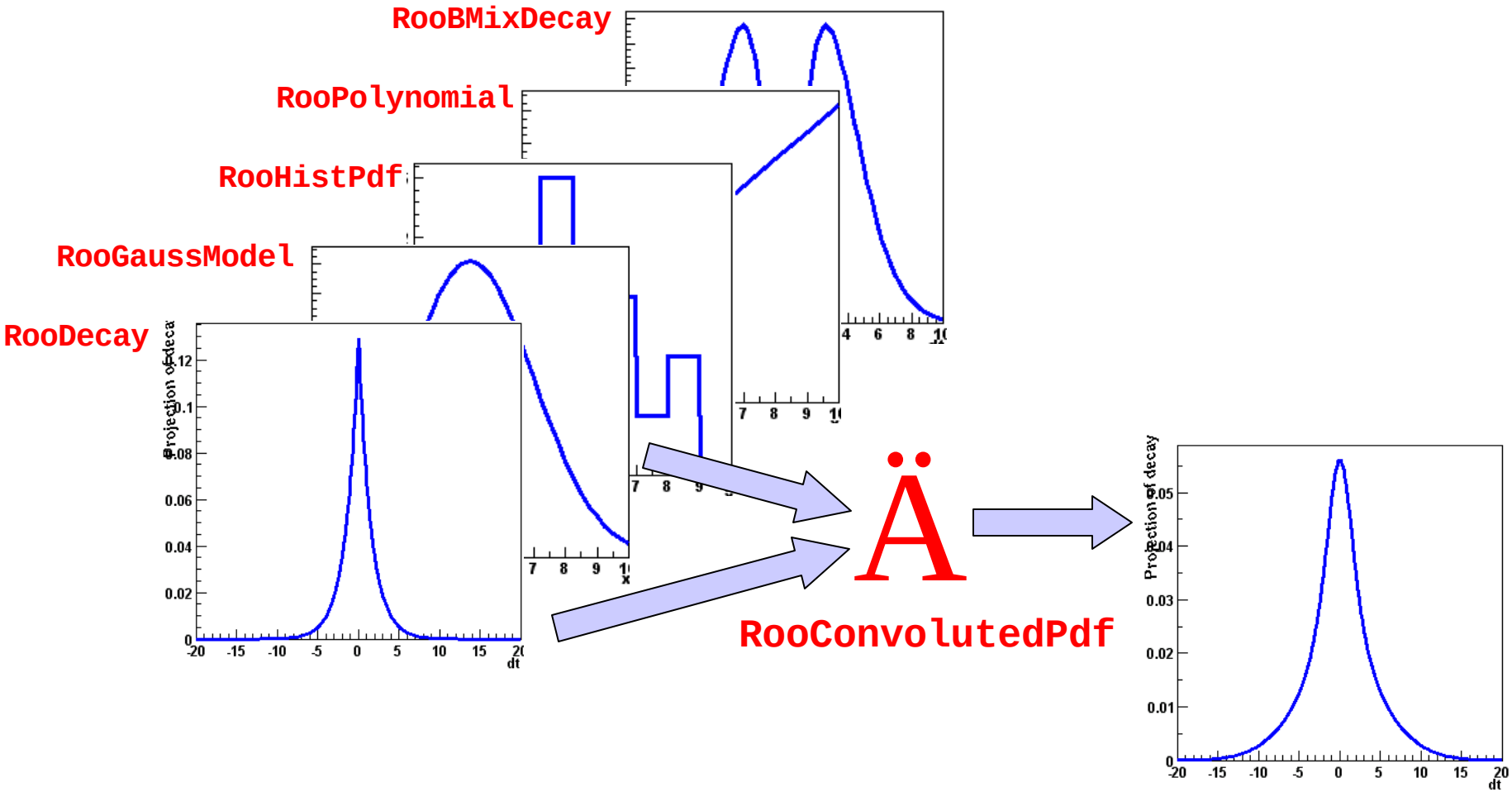
Model building – (Re)using standard components

- Most physics models can be composed from ‘basic’ shapes



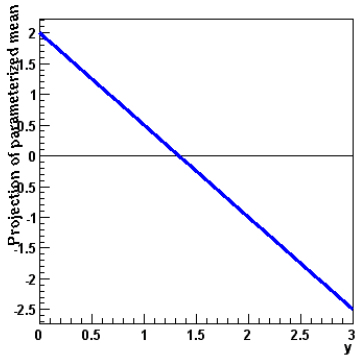
Model building – (Re)using standard components

- Most physics models can be composed from ‘basic’ shapes

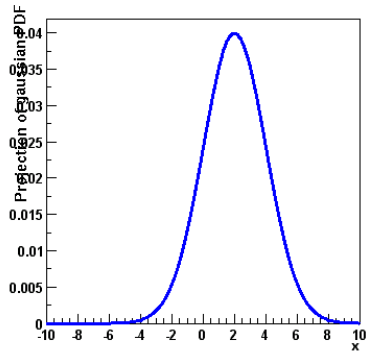


Model building – (Re)using standard components

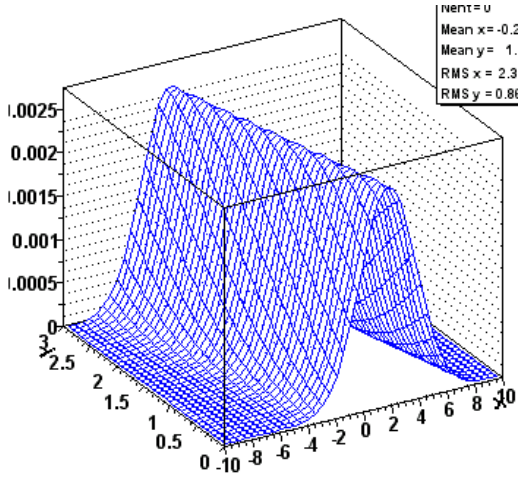
- Building blocks are *flexible*
  - Function *variables can be functions* themselves
  - Just plug in *anything* you like
  - Universally supported by core code
    (PDF classes don't need to implement special handling)



$m(y;a_0,a_1)$



$g(x;m,s)$

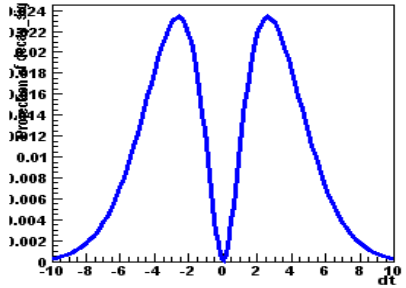


$g(x,y;a_0,a_1,s)$

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

Model building – Expression based components

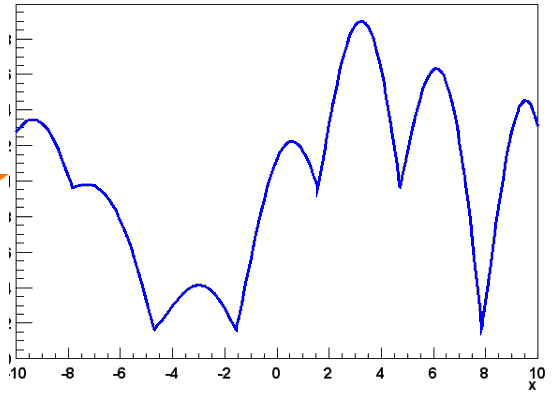
- **RooFormulaVar** – Interpreted real-valued function
  - Based on ROOT **TFormula** class
  - Ideal for modifying parameterization of existing compiled PDFs



RooBMixDecay(t,tau,w,...)

RooFormulaVar w("w","1-2*D",D) ;

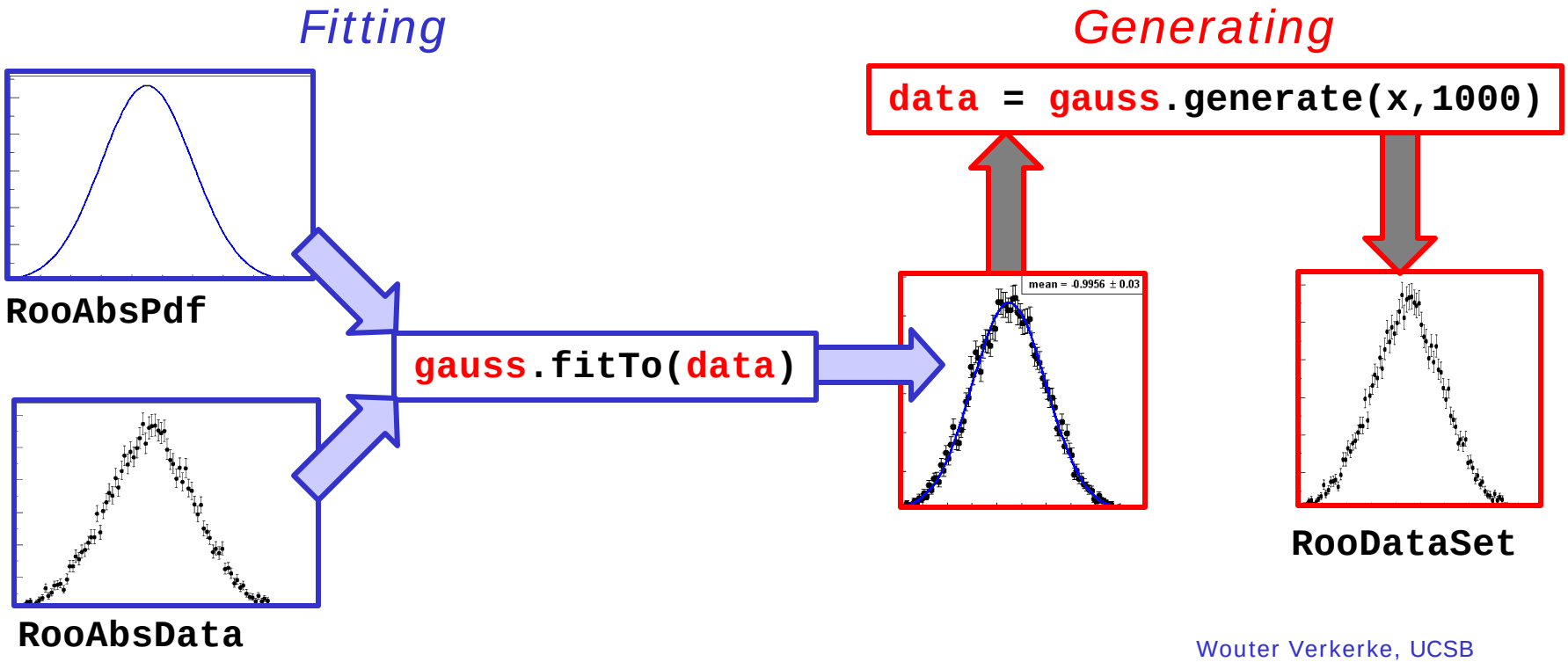
- **RooGenericPdf** – Interpreted PDF
  - Based on ROOT **TFormula** class
  - User expression doesn't need to be normalized
  - Maximum flexibility




```
RooGenericPdf f("f","1+sin(0.5*x)+abs(exp(0.1*x)*cos(-1*x))",x)
```

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
    - models with >16000 components, >1000 fixed parameters and>80 floating parameters have been used (published physics result)
  - Very easy to use – Most operations are one-liners



Using models – Fitting options

- Fitting interface is *flexible and powerful*, many options supported

Data type

✓Binned

✓Unbinned

✓Weighted unbinned

Goodness-of-fit measure

✓-log(Likelihood)

✓Extended –log(L)

✓Chi^2

✓User Defined

✓(add custom/penalty terms to any of these)

Interface

✓One-line: **`RooAbsPdf::fitTo(…)`**

✓Interactive: **`RooMinuit`** class

Sample interactive MINUIT session

```
RooNLLVar nll("nll","nll",pdf,data) ;
RooMinuit m(nll) ;

m.hesse() ;
x.setConstant() ;
y.setVal(5) ;
m.migrad() ;
m.minos()

RooFitResult* r = m.save() ;
```

Access any of MINUITs minimization methods

Change and fix param. values, using native RooFit interface during fit session

Output

✓Modifies parameter objects of PDF

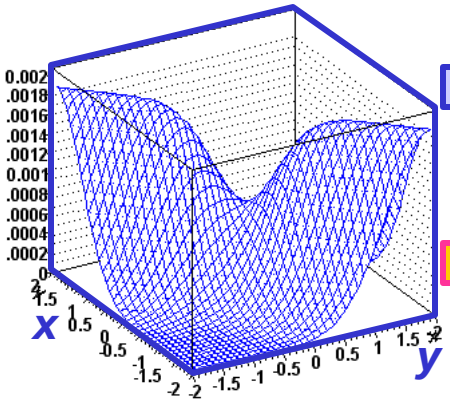
✓Save snapshot of initial/final parameters, correlation matrix, fit status etc…

Using models – Fitting speed & optimizations

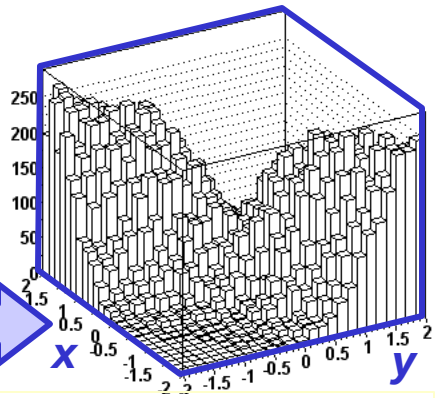
- *Benefit of function optimization traditionally a trade-off between*
  - *Execution speed (especially in fitting)*
  - *Flexibility/maintainability of analysis user code*
    - *Optimizations usually hard-code assumptions…*
- Evaluation of –log(L) in fits lends it well to optimizations
  - Constant fit parameters often lead to higher-level constant PDF components
  - PDF normalization integrals have identical value for all data points
  - Repetitive nature of calculation ideally suited for parallelization.
- RooFit automates analysis and implementation of optimization
  - Modular OO structure of PDF expressions facilitate automated introspection
    - Find and pre-calculate highest level constant terms in composite PDFs
    - Apply caching and lazy evaluation for PDF normalization integrals
    - Optional automatic parallelization of fit on multi-CPU hosts
  - **Optimization** concepts are applied **consistently and completely** to all PDFs
  - **Speedup of factor 3-10 typical** in realistic complex fits
- *RooFit delivers per-fit tailored optimization without user overhead!*

Using models – Toy MC Generation

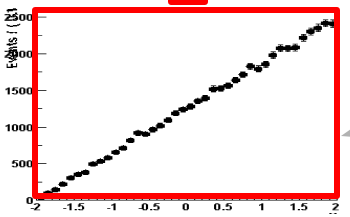
- Generate "Toy" Monte Carlo samples from *any* PDF
  - Sampling method used by default, but PDF components can advertise alternative (more efficient) generator methods
  - No limit to number of dimensions, discrete-valued dimensions also supported

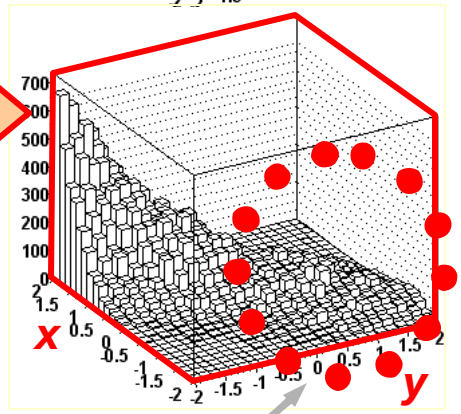



```
data=pdf.generate(x,y,1000)
```




```
data=pdf.generate(x,ydata)
```

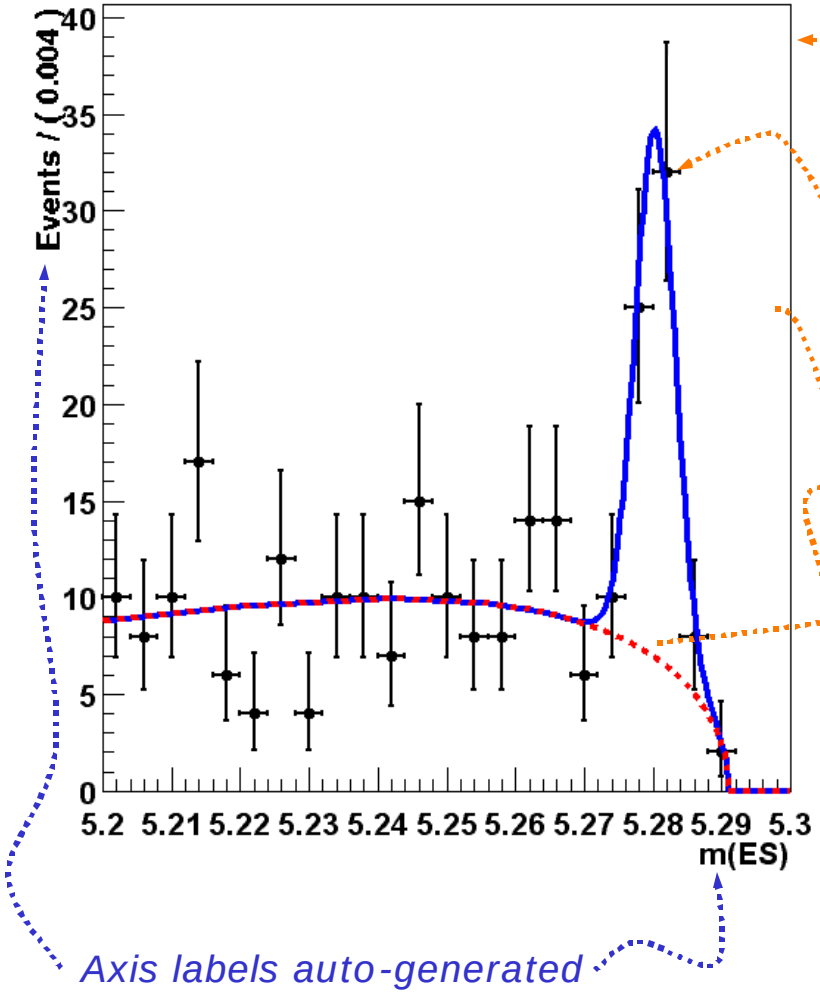




  - **Subset of variables can be taken from a prototype dataset**
    - E.g. to more accurately model the statistical fluctuations in a particular sample.
    - ***Correlations*** with prototype observables ***correctly taken into account***

Using models – Plotting

- **RooPlot** – View of 31 datasets/PDFs projected on the same dimension



Axis labels auto-generated

① ***Create the view on mes***

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

② ***Project the data on the mes view***

```
data->plotOn(frame) ;
```

③ ***Project the PDF on the mes view***

```
pdf->plotOn(frame) ;
```

④ ***Project the bkg. PDF component***

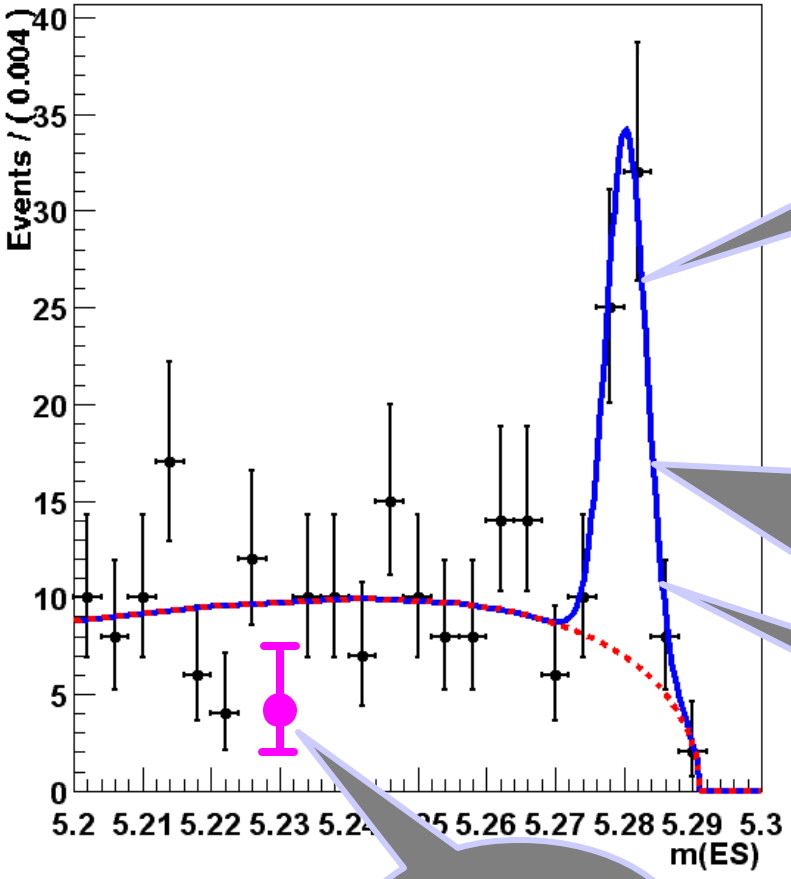
```
pdf->plotOn(frame,Components("bkg"))
```

⑤ ***Draw the view on a canvas***

```
frame->Draw() ;
```

Using models – Plotting

- **RooPlot** – View of 1 datasets/PDFs projected on the same dimension



Curve always normalized to last plotted dataset in **frame**

For multi-dimensional PDFs:
appropriate 1-dimensional projection is automatically created:

$$\text{Projection}[F](x) = N \cdot \frac{\int F(x,\vec{y})d\vec{y}}{\int F(x,\vec{y})dxd\vec{y}}$$

Adaptive spacing of curve points to achieve 1‰ precision, regardless of data binning

Poisson errors on histogram

Using models – Plotting

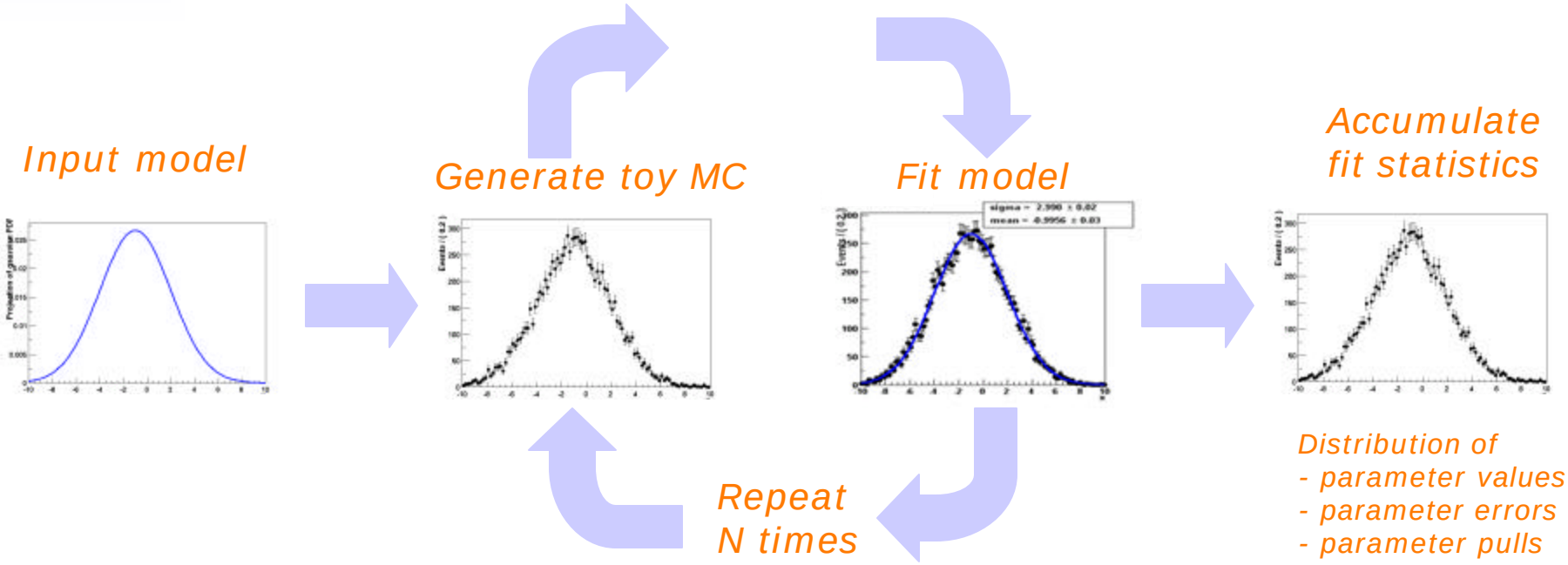
- Additional methods available to
  - Plot/project slices or arbitrarily shaped regions of PDFs
  - Plot PDF projections averaged over observables provided in a dataset
  - Plot generic asymmetries (A-B)/(A+B)

- Single method for *all plot varieties*: **plotOn()**
  - ***Named argument*** interface powerful yet easy to use

```
pdf->plotOn(frame) ;
pdf->plotOn(frame,Slice(x)) ;
pdf->plotOn(frame,Asymmetry(tag),LineColor(kRed)) ;
pdf->plotOn(frame,Components("Bkg*"),ProjWData(dterr)) ;
pdf->plotOn(frame,Normalization(0.5),DrawOption("F")) ;
```

Advanced features – Task automation

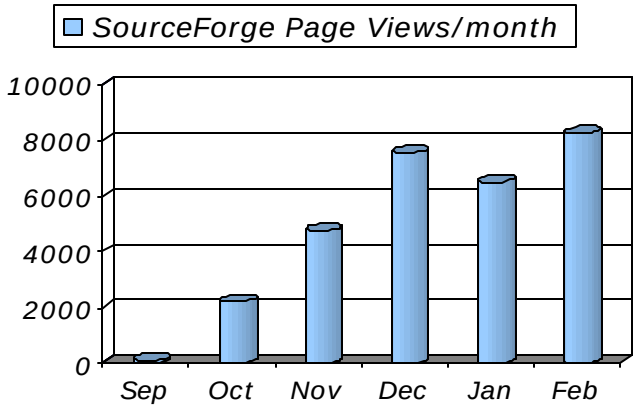
- Support for routine task automation, e.g. goodness-of-fit study



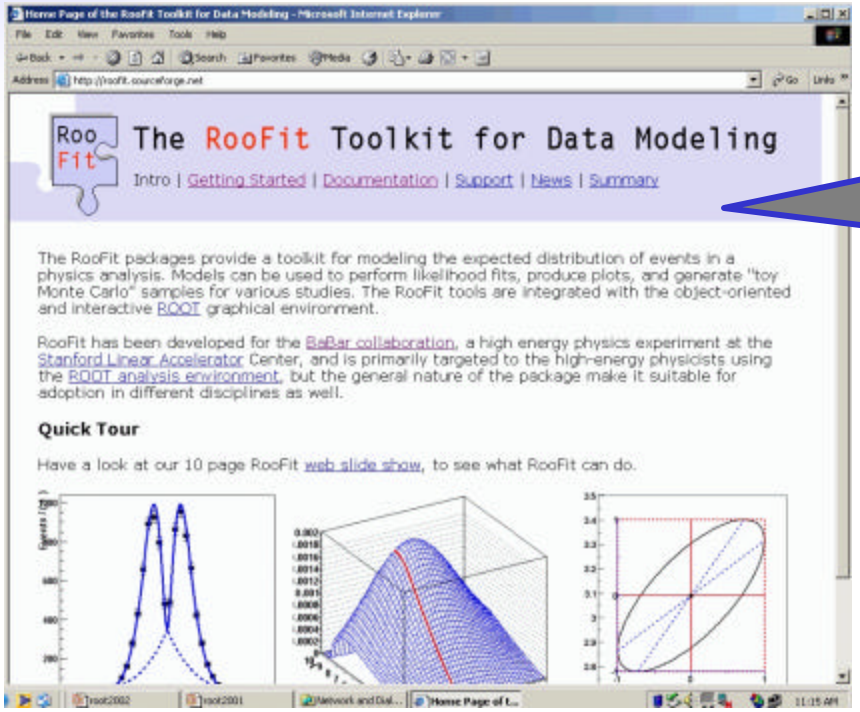

```
// Instantiate MC study manager
RooMCStudy mgr(inputModel) ;

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

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

Development and Use of RooFit in BABAR

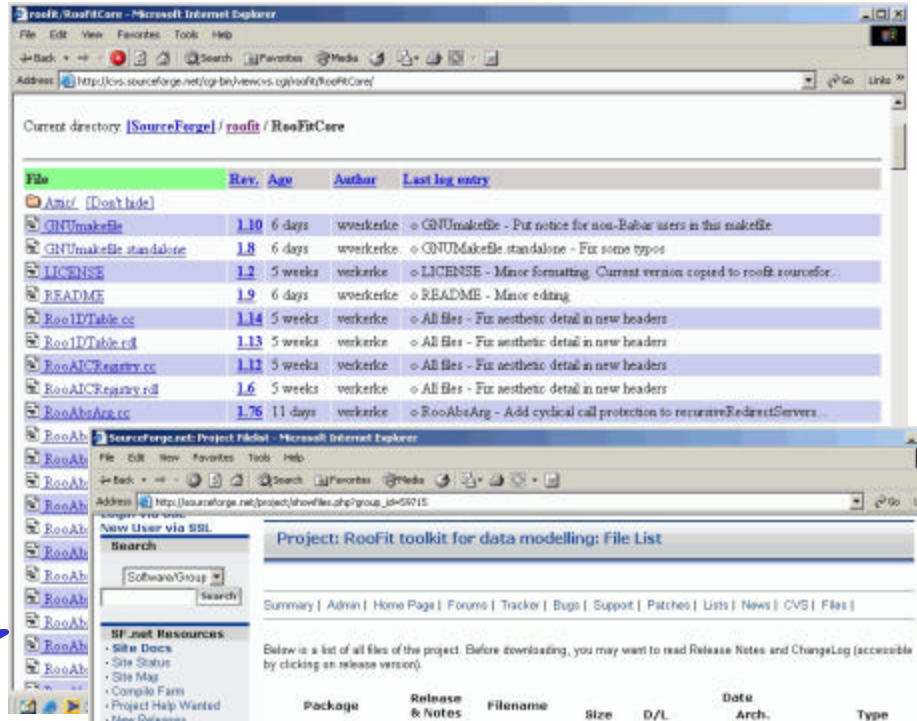
- Development
  - RooFit started as RooFitTools (presented at ROOT2001) in late 1999
    - Original design was rapidly stretched to its limits
  - Started comprehensive redesign early 2001
    - New design was released to BaBar users in Oct 2001 as RooFit
    - Extensive testing & tuning of user interface in the past year
  - RooFit released on SourceForge in Sep 2002 ·
- Current use
  - Almost all BaBar analysis requiring a non-trivial fit now use RooFit or are in the process of switching to RooFit, e.g.
    - CP violation and mixing in hadronic decays ('sin2β')
    - B-Mixing in di-lepton events, $D^{*}\ell\nu$ events
    - Measurement of $\sin 2\alpha_{(eff)}$ from $B \rightarrow \rho\pi$, $B \rightarrow \pi\pi$
    - Searches for rare decays ($B \rightarrow \phi K_S$, $\eta' K_S$, ...)
  - **Typical fit complexity**
    - **30 – 70 floating parameters**
    - **4-8 dimensions**
    - **PDF consists of 1000-10000 objects**
    - **Dataset of 500-100000 events**



RooFit at SourceForge - **roofit.sourceforge.net**



RooFit moved to SourceForge
to facilitate access and
communication
with non-BaBar users



Code access

- *–CVS repository via pserver*
- *–File distribution sets for production versions*

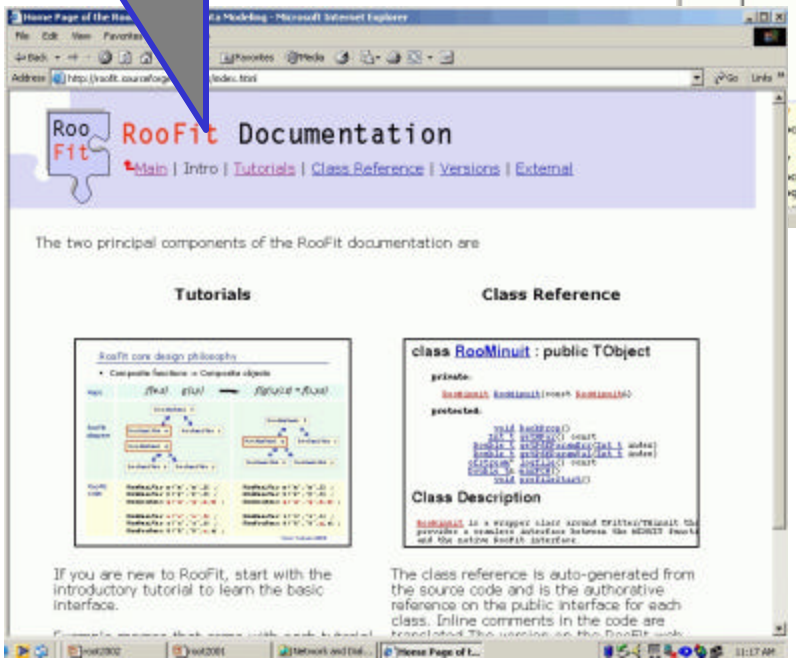
RooFit at SourceForge - Documentation

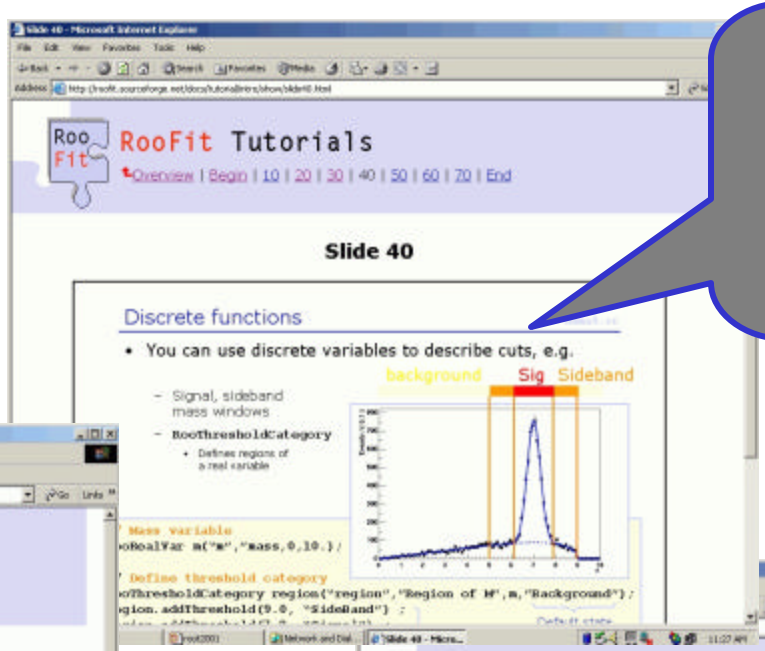
Documentation

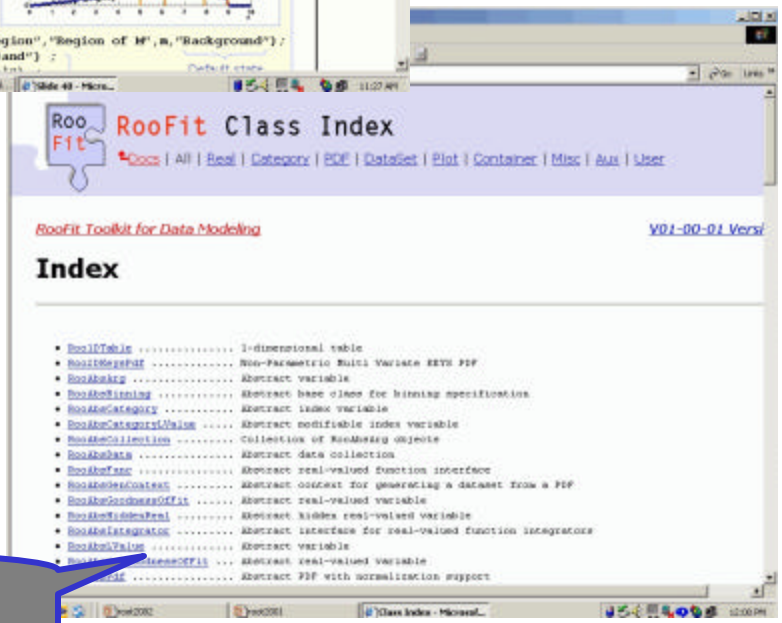
Comprehensive set of tutorials (PPT slide show + example macros)

Five separate tutorials

More than 250 slides and 20 macros in total







Class reference in **THtml** style

Summary

- RooFit adds a ***powerful data modeling language*** to ROOT
  - Mathematical objects (variables, functions…) represented as C++ objects
  - Complex models are easily composed from library of standard components
    - New fundamental components can be written easily
  - Powerful tools for fitting, Toy MC generation and visualization are easy to use
    - Compact syntax
    - Universal functionality *(almost no arbitrary / implementation related restrictions)*
  - Automated function optimization analysis and implementation delivers industrial strength performance

- RooFit makes it really easy to do the right thing
  - Unbinned maximum likelihood fits
  - Poisson/binomial errors on histograms
  - Simultaneous fits with control samples
  - Model validation with Toy MC studies, …

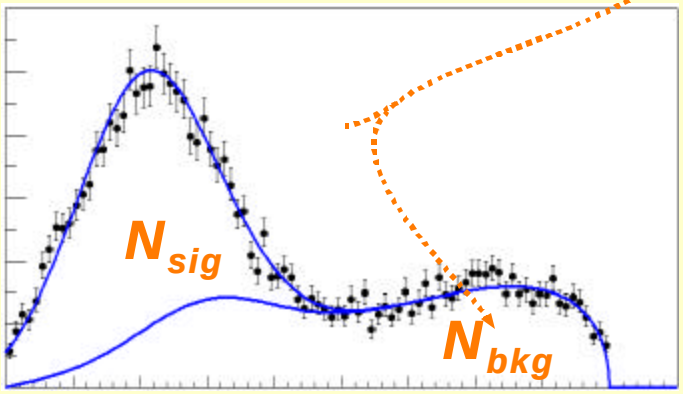
- RooFit enjoys a rapidly growing user community
  - The BaBar collaboration has enthusiastically embraced RooFit
    - Majority of physics analysis involving non-trivial fits now use RooFit
  - Publicly available on SourceForge since September 2002
  - Individual users in Belle,SLD,CDF,D0,CLEO,GLAST,LHCb,MiniBOONE,…

Data modeling – Mathematical formulation

- **Generic real-valued functions**

$$f(x) = A \cdot e^{-\left(\frac{x-m}{s}\right)^2} + B \cdot (a + b \cdot x)$$

  - **Usually we really want to know N_{sig}, N_{bkg}**
  - Relation between A,B and N_{sig}/N_{bkg} non-trivial
  - Doesn't scale easily to complex problems



- ***Probability Density Functions***

$$f(x) = N_{sig} \cdot \left[\frac{e^{-\left(\frac{x-m}{s}\right)^2}}{\int e^{-\left(\frac{x-m}{s}\right)^2}} \right] + N_{bkg} \cdot \left[\frac{(a + b \cdot x)}{\int (a + b \cdot x)} \right]$$

- **Benefits** of PDFs
  - *Straightforward interpretation of parameters*
  - *Enhanced modularity → better scaling to complex models*
  - *Mathematically rigorous definition*

Unit normalization

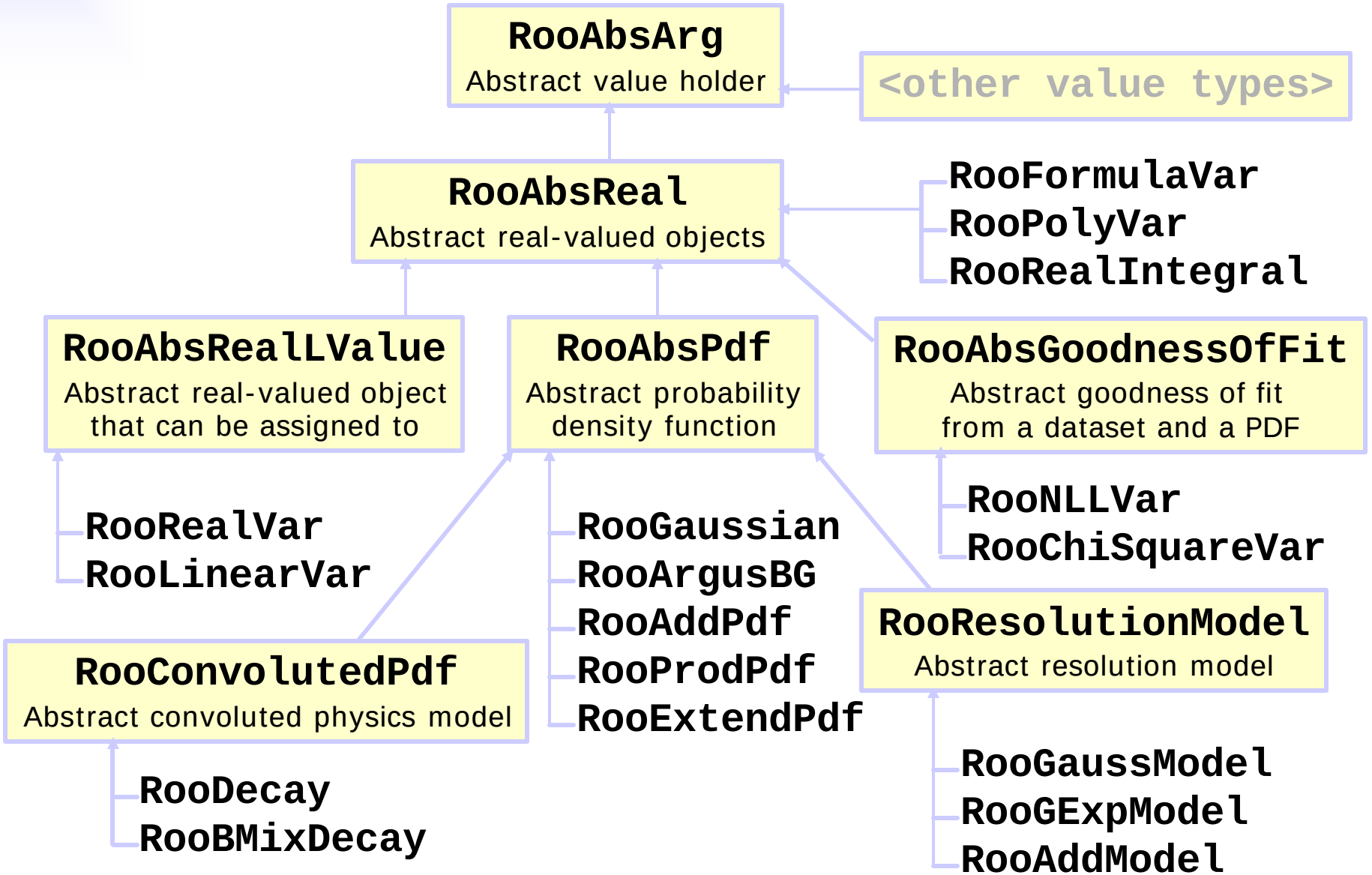
$$\int_{\vec{x}_{min}}^{\vec{x}_{max}} F(\vec{x}, \vec{p}) d\vec{x} \equiv 1$$

Positive definite

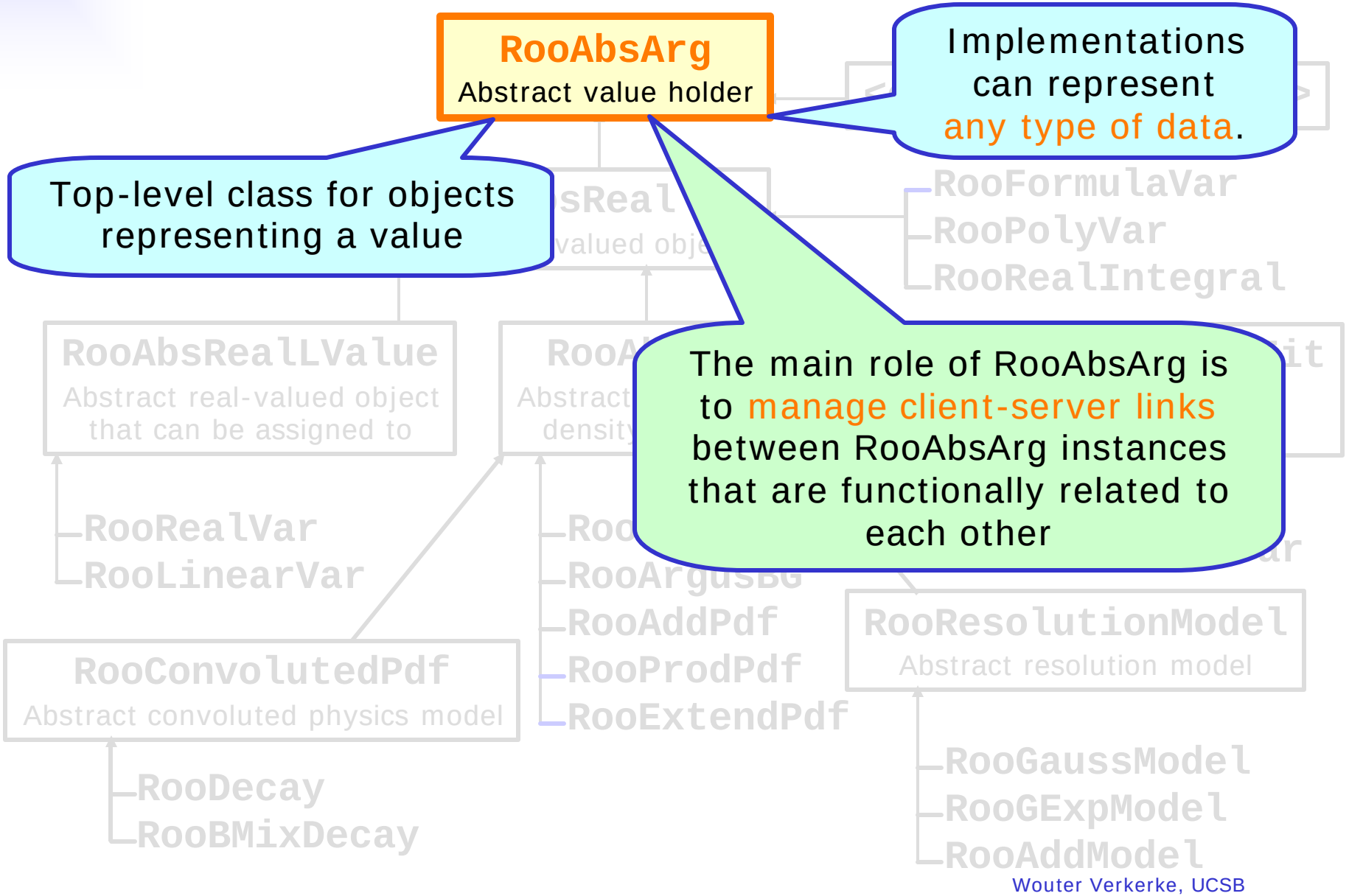
$$F(\vec{x}, \vec{p}) \geq 0$$

Achieving unit normalization is traditionally the most difficult aspect of implementation → Need to make this easy

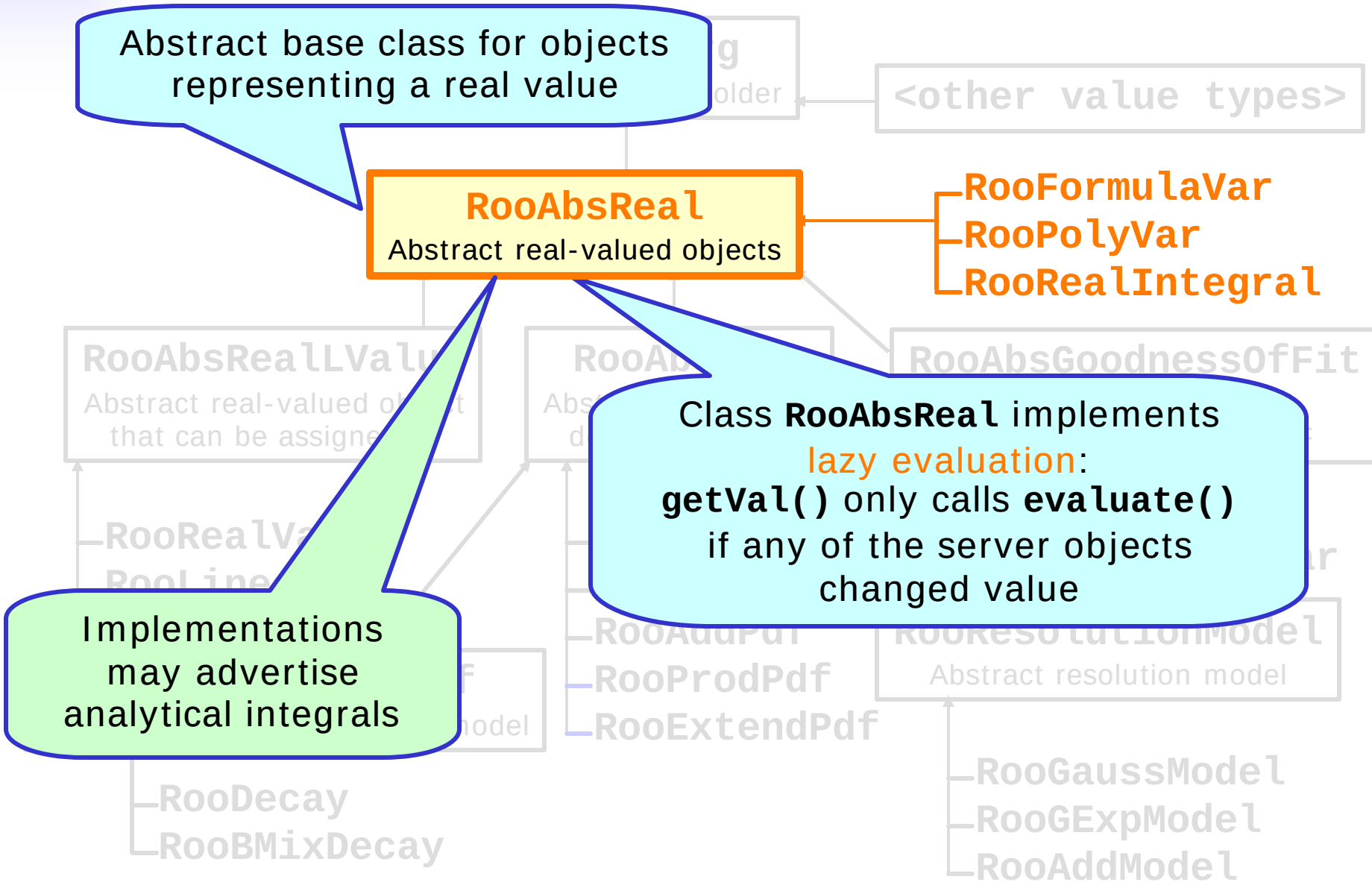
Hierarchy of classes representing a value or function



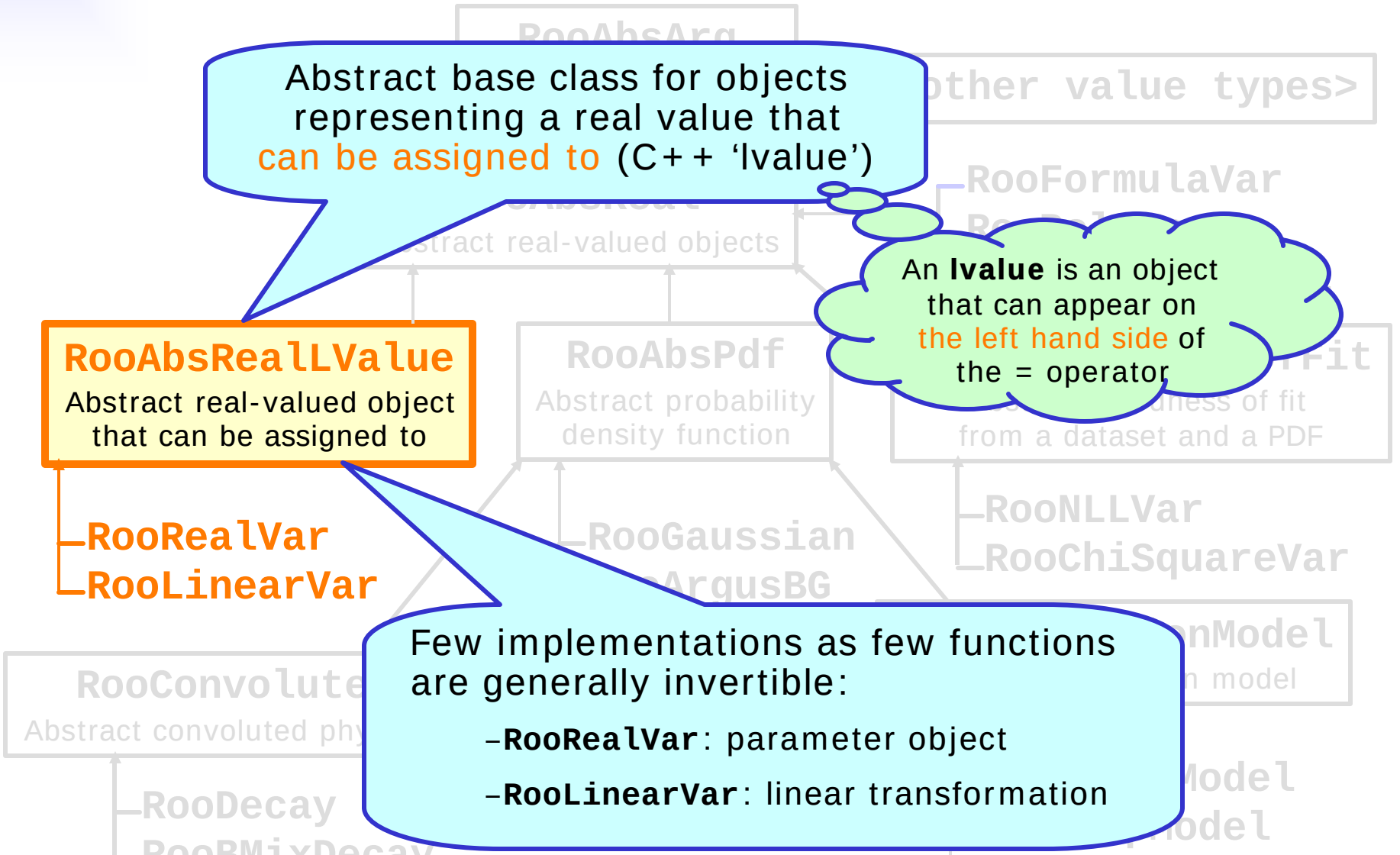
Class **RooAbsArg**



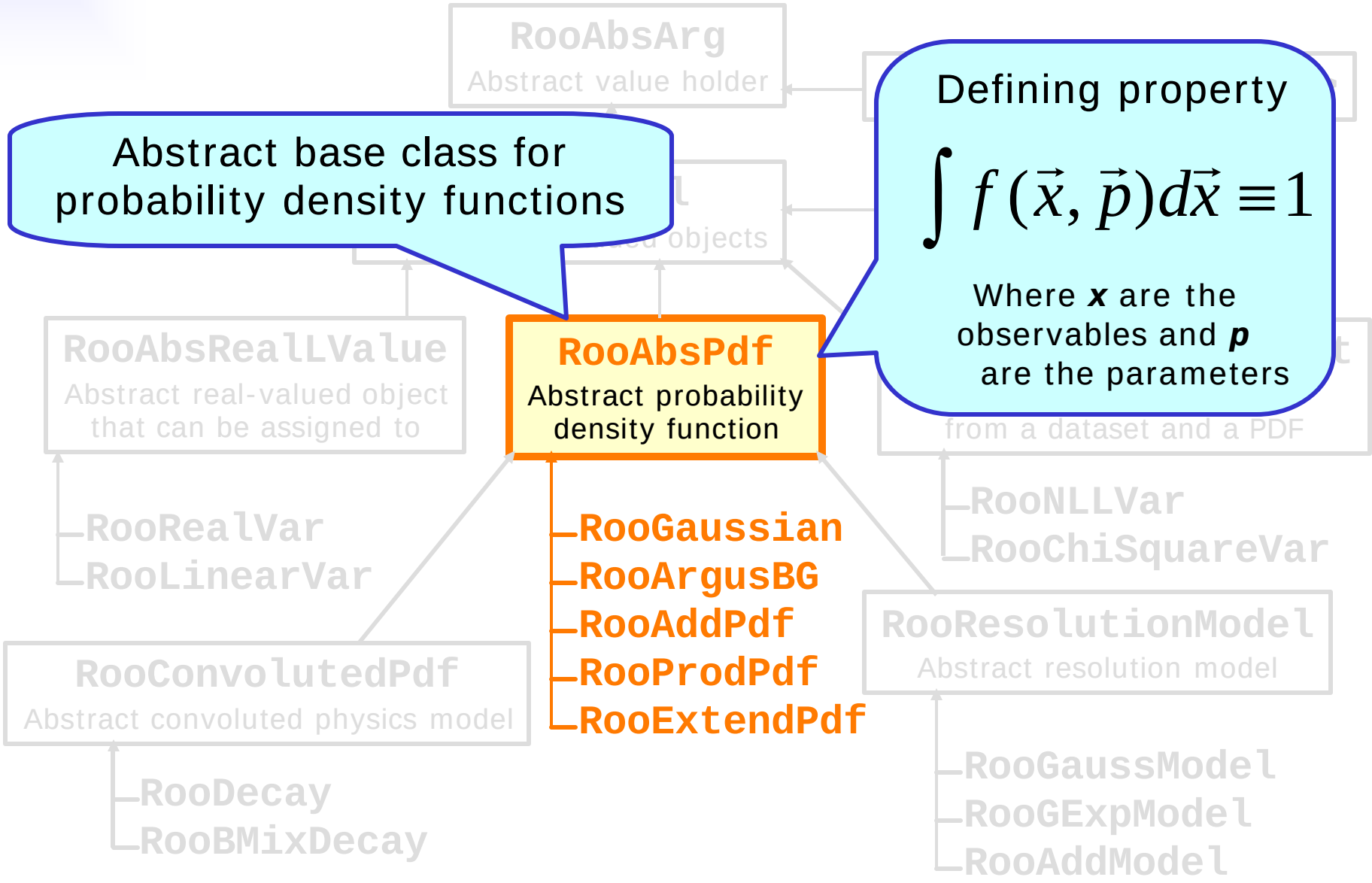
Class **RooAbsReal**



Class **RooAbsRealLValue**



Class **RooAbsPdf**



Class **RooConvolutedPdf**

Implements $f_i(dt,...) \otimes R(dt,...)$

RooResolutionModel

$$P(dt,...) = \sum_k c_k(...)\left(f_k(dt,...) \otimes R(dt,...)\right)$$

RooConvolutedPdf (physics model)

Implements c_k , declares list of f_k needed

No convolutions calculated in this class!

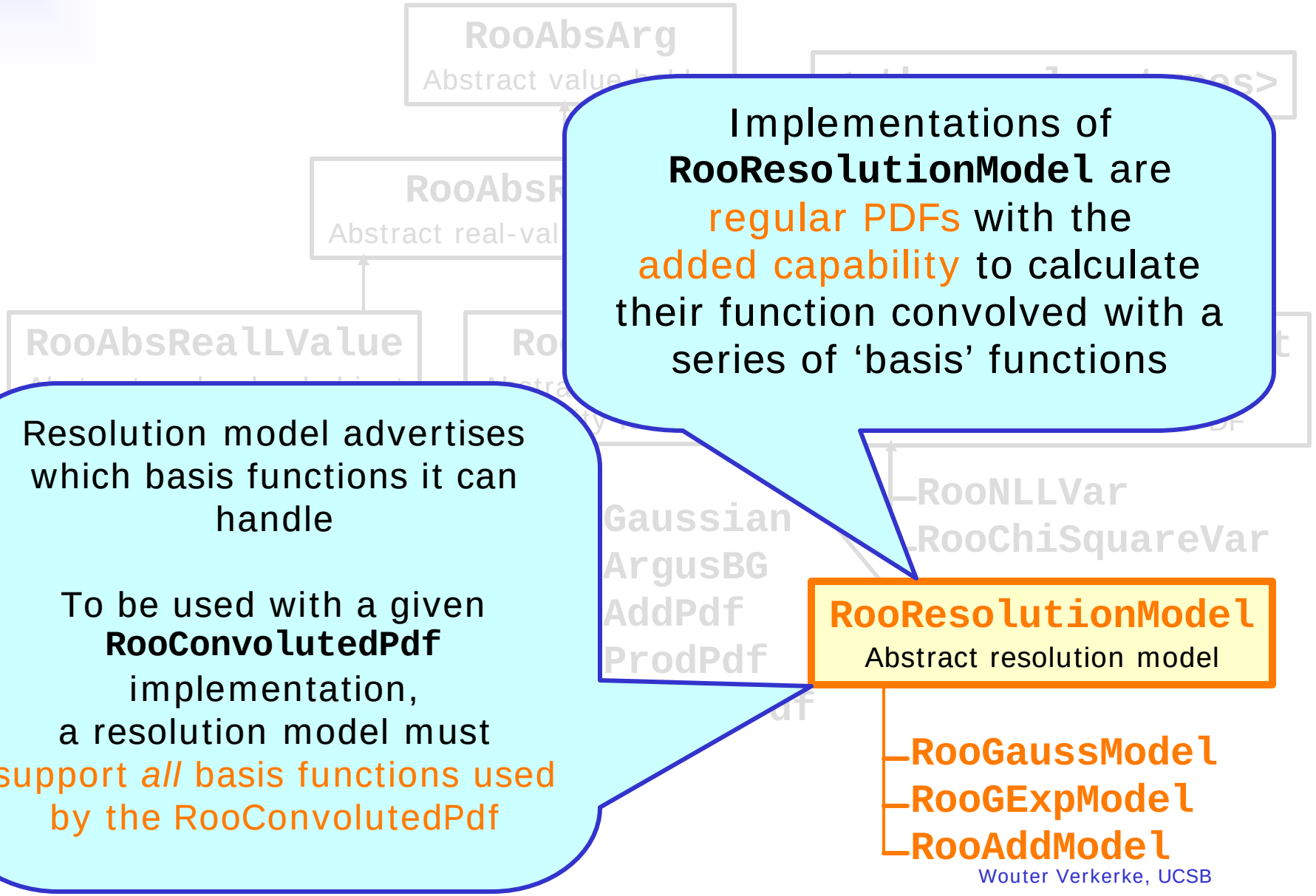
RooConvolutedPdf

Abstract convoluted physics model

- **RooDecay**
- **RooBMixDecay**

Abstract base class for PDFs that can be convoluted with a resolution model

Class **RooResolutionModel**



Implementations of **RooResolutionModel** are regular PDFs with the added capability to calculate their function convolved with a series of 'basis' functions

Resolution model advertises which basis functions it can handle

To be used with a given **RooConvolutedPdf** implementation, a resolution model must support *all* basis functions used by the RooConvolutedPdf

Class **RooAbsGoodnessOfFit**

Provides the framework for efficient calculation of goodness-of-fit quantities.

A goodness-of-fit quantity is a function that is calculated from

- A dataset
- the PDF value for each point in that dataset

RooAbsGoodnessOfFit

Abstract goodness of fit from a dataset and a PDF

- **RooNLLVar**
- **RooChiSquareVar**

Built-in support for

- **Automatic constant-term optimization** activated when used by RooMinimizer(MINUIT)
- **Parallel execution on multi-CPU hosts**
- Efficient **calculation of RooSimultaneous** PDFs

Class tree for discrete-valued objects