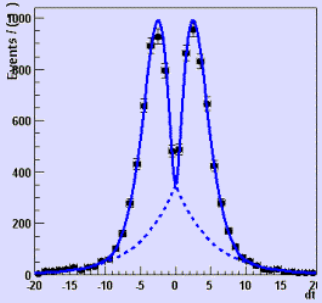
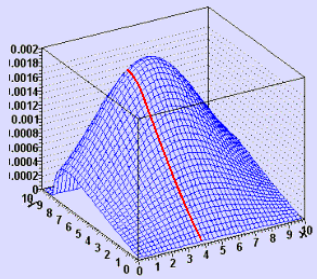
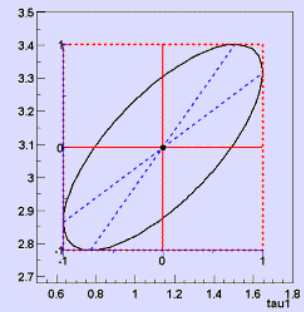






RooFit

A general purpose tool kit for data modeling

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

RooFit purpose - Data Modeling & Analysis

Distribution of observables $\vec{x}$

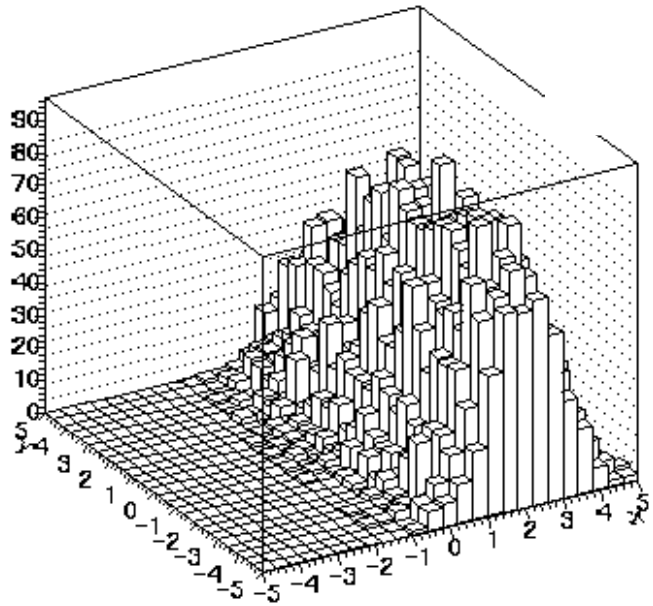
Define data model

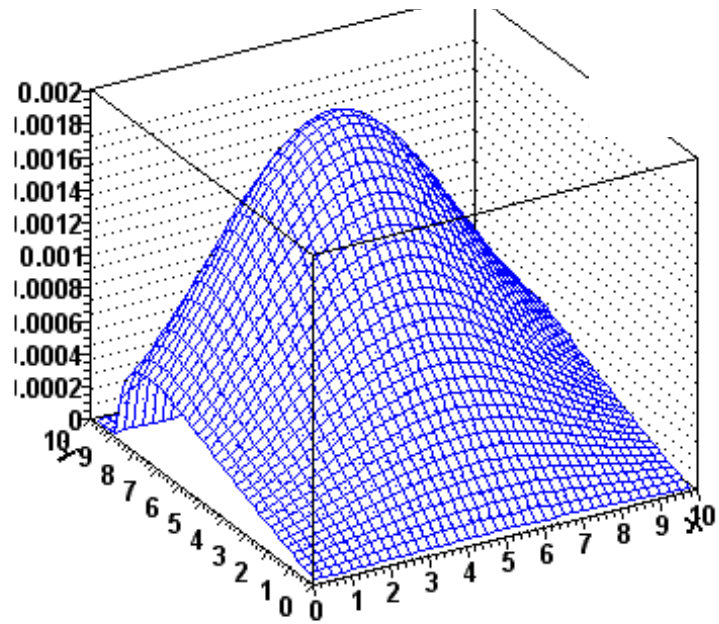
Probability density function $\mathbf{F(\vec{x}; \vec{p}, \vec{q})}$

- 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
  - C++ command line interface & macros
  - Data management and histogramming
  - Graphics interface
  - I/O support

- RooFit is a collection of classes that augment the ROOT environment
  - *Focus on data modeling*
  - Implemented as 2 shared libraries

Data modeling – Probability Density Functions

- RooFit models are Probability Density Functions (PDF)

observables: variables in the data

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

parameters: all other variables

- Properties of PDFs:

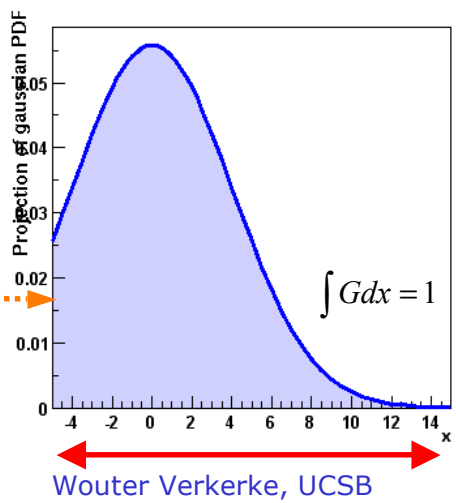
$$\int_{\vec{x}_{\min}}^{\vec{x}_{\max}} F(\vec{x}, \vec{p}) d\vec{x} \equiv 1$$

Normalized to unity over all observables

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

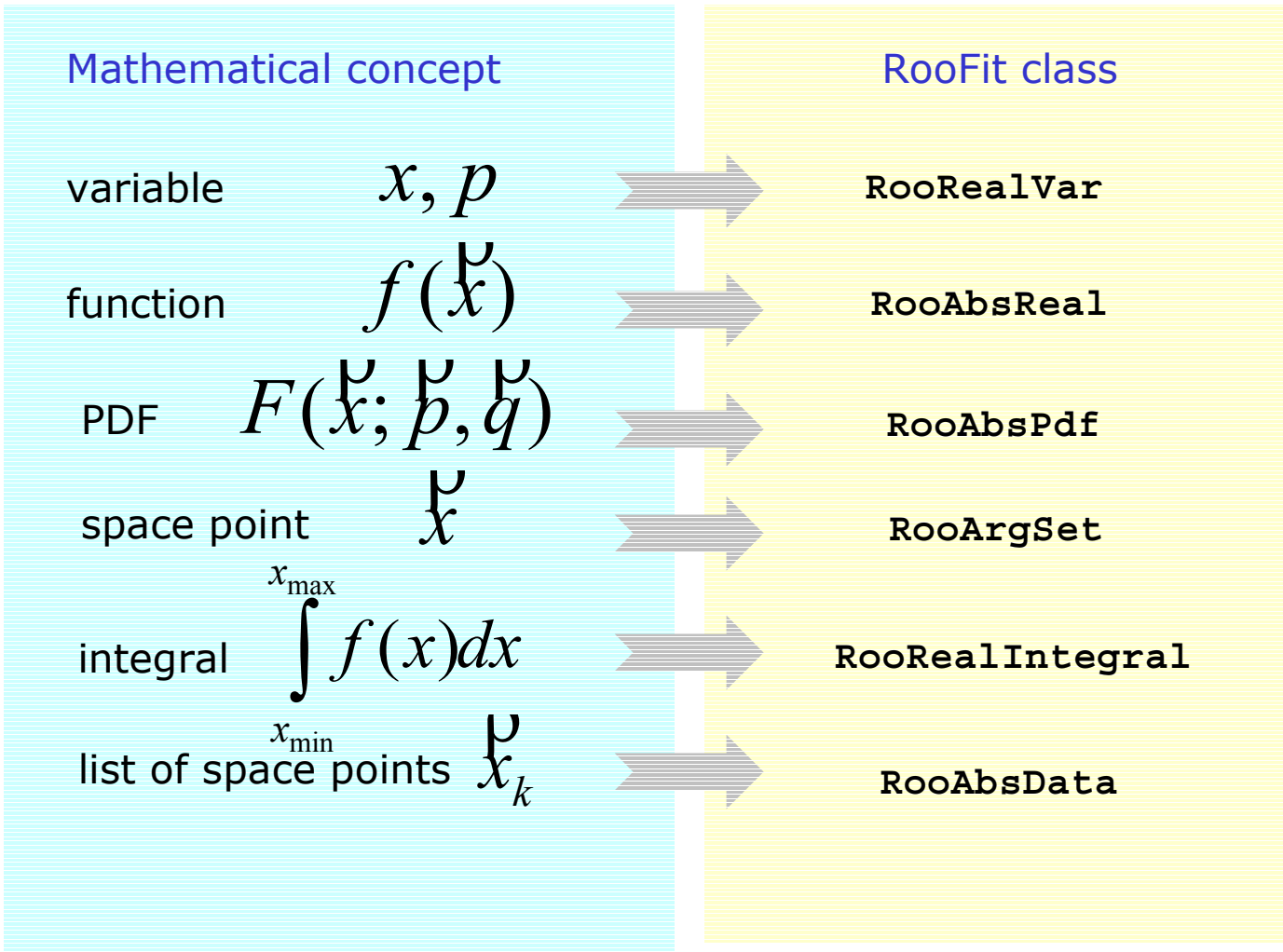
Positive definite

This essential property is traditionally the most difficult part of any implementation.



RooFit key features – OO Data/Model language

- Mathematical objects are represented as C++ objects



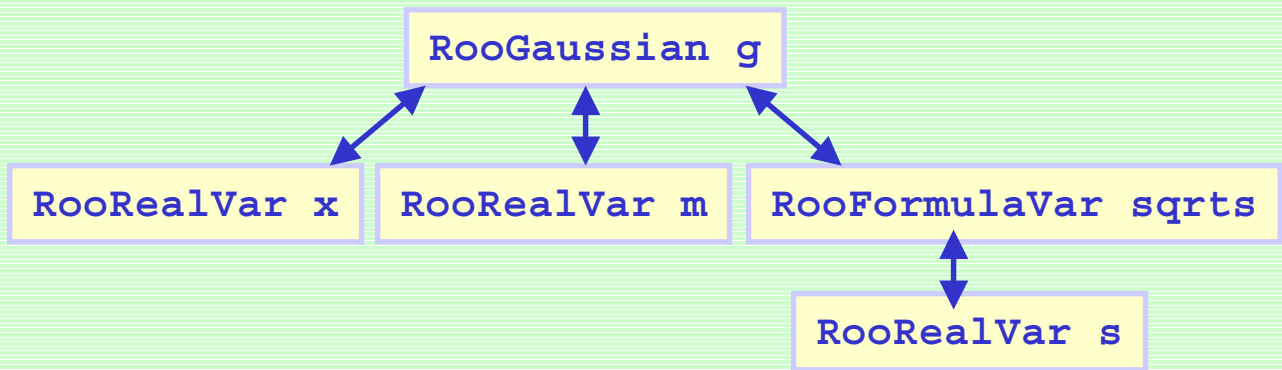
RooFit key features – OO Data/Model language

- Straightforward correlation between mathematical representation of formula and RooFit code

Math

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

RooFit diagram



RooFit code

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

RooFit key features – OO Data/Model language

- All objects are *self documenting*

Name – Unique identifier of object

Associated data
Limits, units, plot labels,… stored in object

Title – More elaborate description of object

Initial range

Objects representing a variable

```
RooRealVar mass("mass","Invariant mass",5.20,5.30) ;
RooRealVar width("width","B0 mass width",0.00027,"GeV");
RooRealVar mb0("mb0","B0 mass",5.2794,"GeV") ;
```

Initial value Optional unit

Function object

```
RooGaussian b0sig("b0sig","B0 sig PDF",mass,mb0,width);
```

References to variables

RooFit key features – Integration & PDF Normalization

- Integrals are also represented as objects
  - Easy to create:

```
// Create object representing Int(g)dx
RooRealIntegral* gi = g.createIntegral(x) ;
```

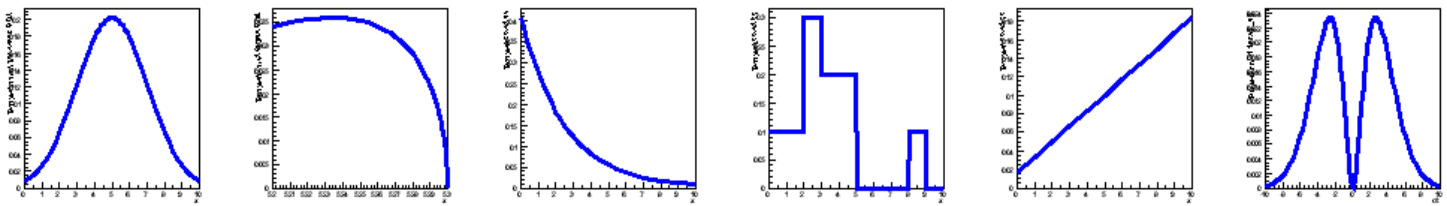
`gi` changes as parameters of `g`, or limits on `x` change.

- *Every* function can be integrated
  - By default numeric techniques are used:
    Adaptive trapezoid for 1-D, Monte Carlo (VEGAS) for >1-D
  - Functions and PDFs that have known (partial) analytical integrals can advertise these (but it is not required)
  - Final integral can be a combination of analytic and numeric methods.
    Optimal combination automatically selected
  - Integral values are automatically cached and recalculated only when necessary

- Integral objects take care of PDF normalization
  - Raw PDF value always divided by value of integral object
  - Proper normalization guaranteed while support in PDF optional

RooFit key features - Modularity

- RooFit provides a collection of standard PDF classes, e.g.
  - Gaussian, Exponential, Polynomial *(basic)*
  - Argus, Crystal Ball, *B*-Decay, Breit-Wigner, Voigtian *(physics inspired)*
  - Histogram, KEYS *(non-parametric)*



- Generic expression based PDF also available:
  - Based on **`Tformula`**
  - Expression normalized via numeric integration

```
RooGenericPdf gp("gp","Generic PDF","exp(x*y+a)-b*x",
                                    RooArgSet(x,y,a,b));
```

- Easy for users to write/contribute new PDF/function classes

Writing new PDF & function classes

- Base classes **`RooAbsPdf,RooAbsReal,…`** hides all complex issues

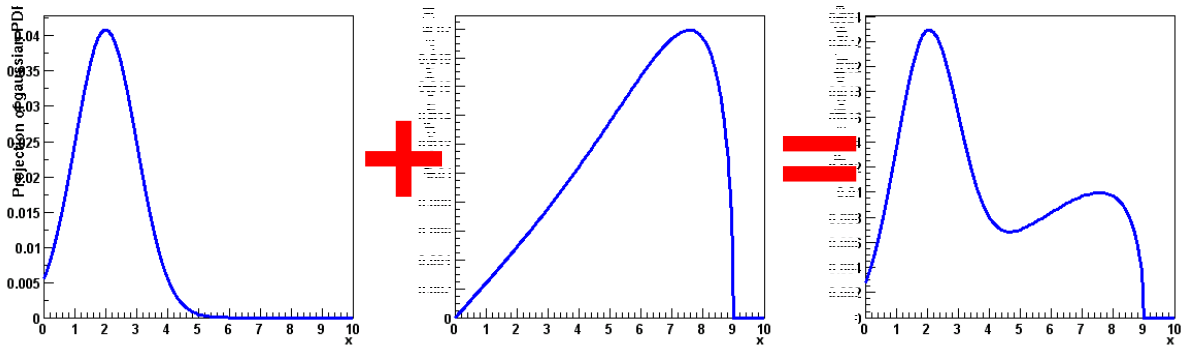
```
class RooGaussian : public RooAbsPdf {
public:
  RooGaussian(const char *name, const char *title,
              RooAbsReal& x, RooAbsReal& mean, RooAbsReal& sigma);
  RooGaussian(const RooGaussian& other, const char* name=0) ;
  virtual TObject* clone(const char* newname) const
  inline virtual ~RooGaussian() { }


protected:
  Double_t evaluate() const ;

  RooRealProxy _x ;
  RooRealProxy _mean ;
  RooRealProxy _sigma ;

private:
  ClassDef(RooGaussian,0)          ssian PDF
};
```

Minimal implementation:
`ctor,cctor,dtor,clone()` and **`evaluate()`**

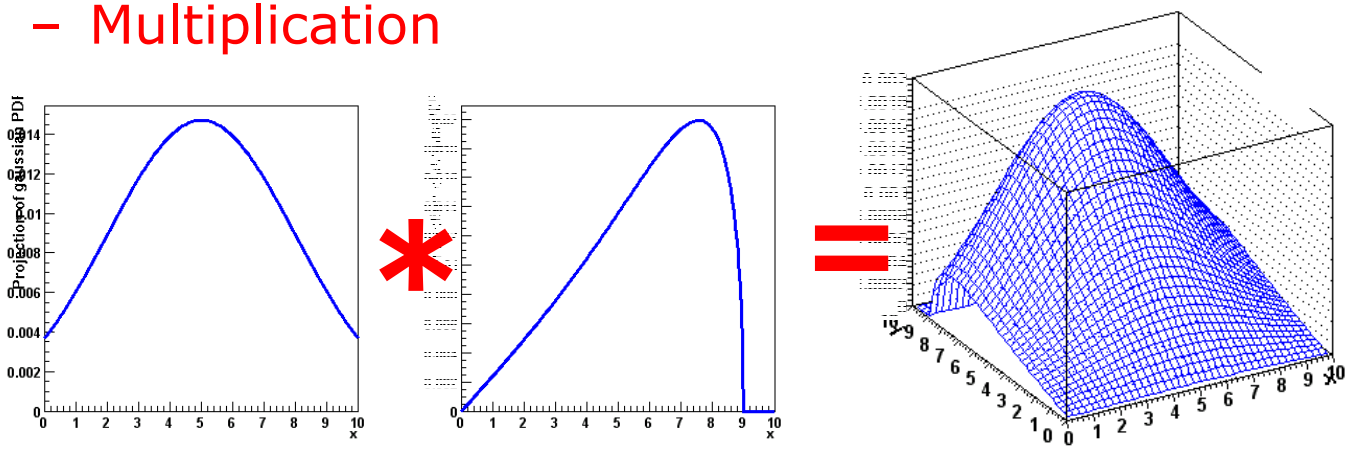
Proxy classes store references to value servers.
- All client/server link management taken care of
- **Easy to use**: behaves like **`Double_t`**

RooFit key features - Modularity

- Complex PDFs be can be trivially composed from basic PDFs
  - Addition

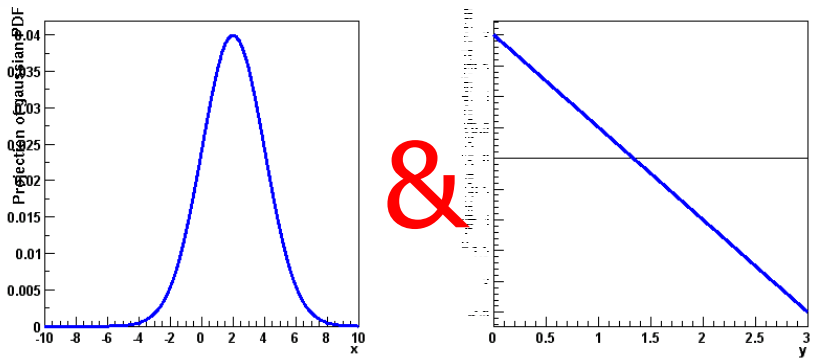


  - Multiplication

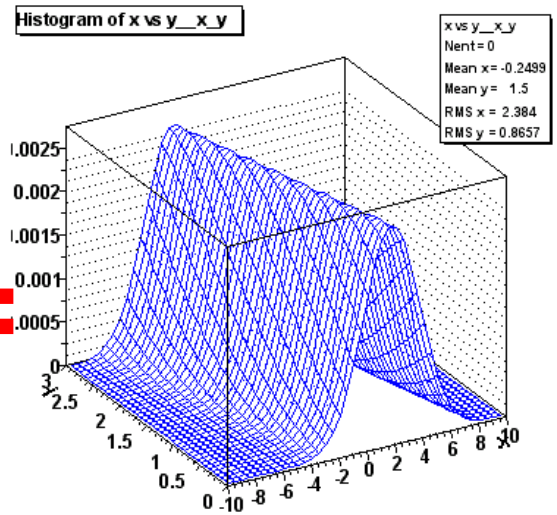


RooFit key features - modularity

– Composition ('plug & play')



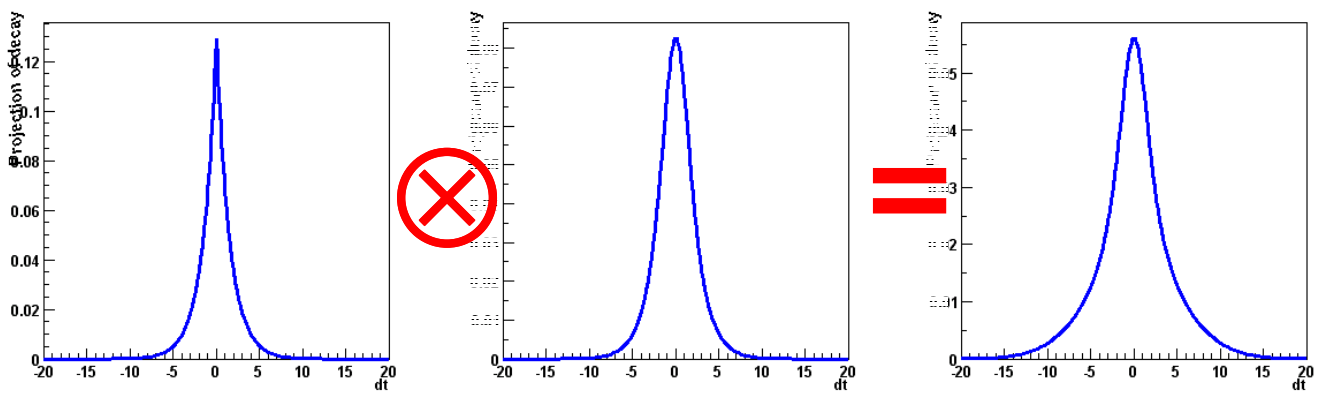
g(x;m,s) & m(y;a_0,a_1) =



g(x,y;a0,a1,s)

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

– Convolution



Datasets & Data types

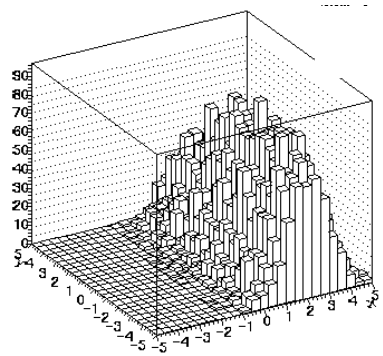
Unbinned

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

Weighted unbinned

x	y	z	*wgt*
1	3	5	*0.11*
2	4	6	*0.75*
1	3	5	*0.30*
2	4	6	*0.92*

Binned



RooDataSet

RooDataHist

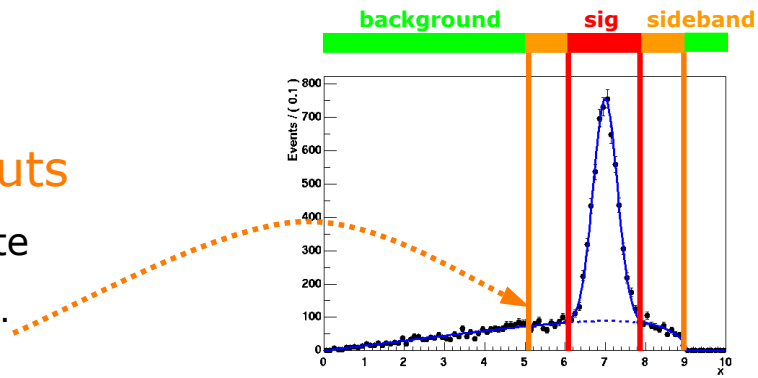
RooAbsData

Many RooFit methods accept this abstract data type → Use data set types interchangeably

- Implemented with **`TTrees`**
- Current row represented by set of RooFit value objects
- Unlimited number of dimensions (also binned)
- Dimensions may be mix of real,discrete,or string values
- Import data from existing ROOT types:
  - **`TTree, ASCII file → RooDataSet`**
  - **`THx,RooDataSet → RooDataHist`**

Discrete variables

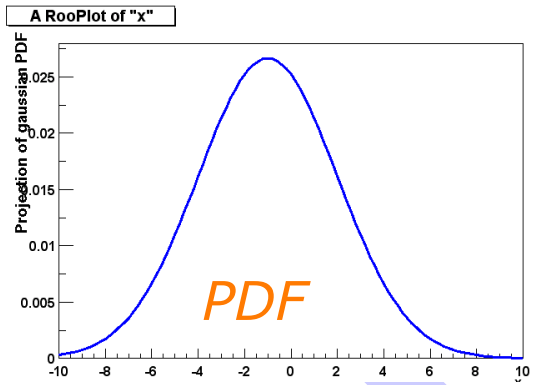
- Many physics problems involve discrete variables
  - Decay mode, particle ID, flavor, charge, reconstruction technique
  - Traditionally represented by integers,
    but name→index mapping can be cumbersome
- RooFit has discrete variable base type with named states
  - Named states facilitate symbolic manipulation of data
  - Index→name mapping optional
- Discrete functions implement cuts
  - Real-to-discrete, Discrete-to-discrete
  - Multiplication,Mapping,Thresholds,…
- Distribution of discrete variables in data can be tabulated

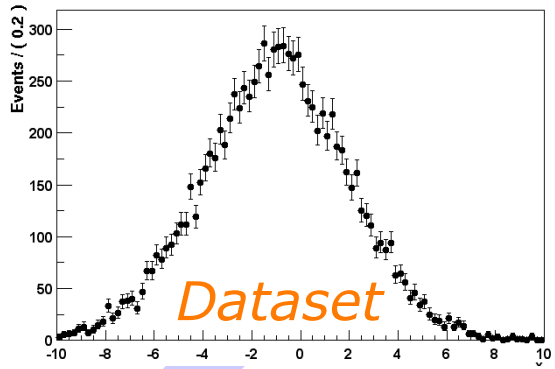



```
data->table(tagCat,"x>8.23")

  Table tagCat : aData(x>8.23)
  +-------------+-----+
  |      Lepton | 668 |
  |        Kaon | 717 |
  | NetTagger-1 | 632 |
  | NetTagger-2 | 616 |
  +-------------+-----+
```

RooFit key features – Fitting

*Given **any** PDF, fitting is a 1-line operations*



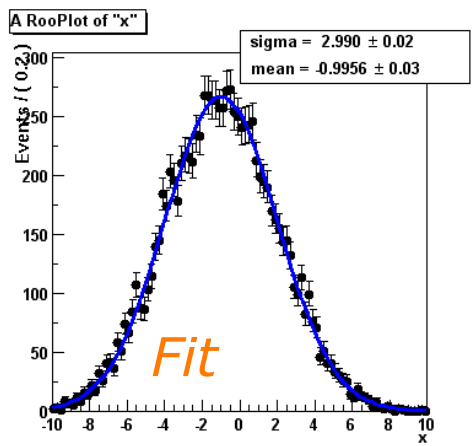


`fitResult` *stores snapshot of parameters, correlation matrix etc…*

```
fitResult = gauss.fitTo(data)
```

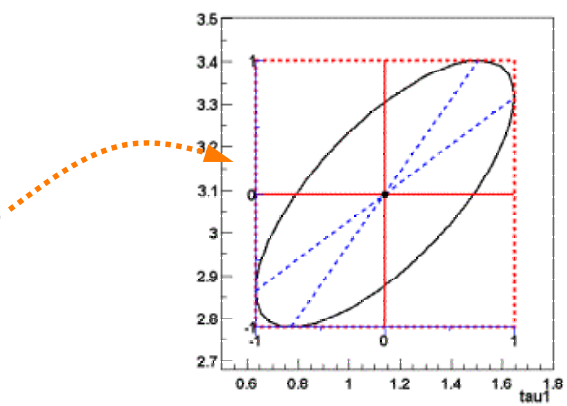
Fit result reflected in parameters of **`gauss`**

```
 COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=25054.9 FROM HESSE     STATUS=OK             10 CALLS          69 TOTAL
                    EDM=3.65627e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   INTERNAL      INTERNAL
 NO.   NAME      VALUE            ERROR          STEP SIZE       VALUE
  1  mean         -9.95558e-01   3.01321e-02   6.59595e-04  -9.95558e-01
  2  sigma         2.99001e+00   2.20203e-02   9.66748e-05   2.99001e+00
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  2    ERR DEF=0.5
 9.079e-04 -1.787e-05
-1.787e-05  4.849e-04
```



RooFit key features – Fitting

- RooFit provides out-of-the box support for
  - **Unbinned maximum likelihood fit** [PDF & unbinned dataset]
  - Unbinned **weighted** maximum likelihood fit [PDF & weighted unbinned dataset]
  - **Binned** maximum likelihood fit [PDF & binned dataset]
  - **Extended** maximum likelihood fit (binned, unbinned, unbinned weighted)
  - Binned **chi-square** fit [PDF & binned dataset]
  - Support for other user-defined goodness-of-fit quantities
- Easy to add additional terms to fit function
  - Penalty terms, Lagrange multiplier etc.
- Seamless integration of MINUIT with RooFit objects
  - Interactive MINUIT sessions possible from ROOT command line (in C++)
  - Full fit output, including correlation matrix, savable for easy access later
  - Graphical analysis of correlation matrix elements



RooFit key features – Fitting

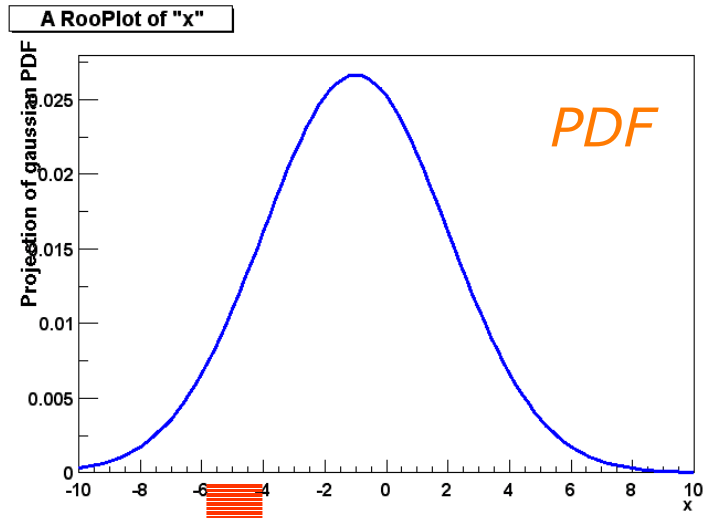
- Automatic pre-fit PDF optimization
  - Prior to each fit, the PDF is analyzed for possible optimizations
  - Optimization algorithms:
    - Detection and *precalculation of constant terms* in any PDF expression
    - Function *caching* and lazy evaluation
    - *Factorization* of multi-dimensional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

- **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

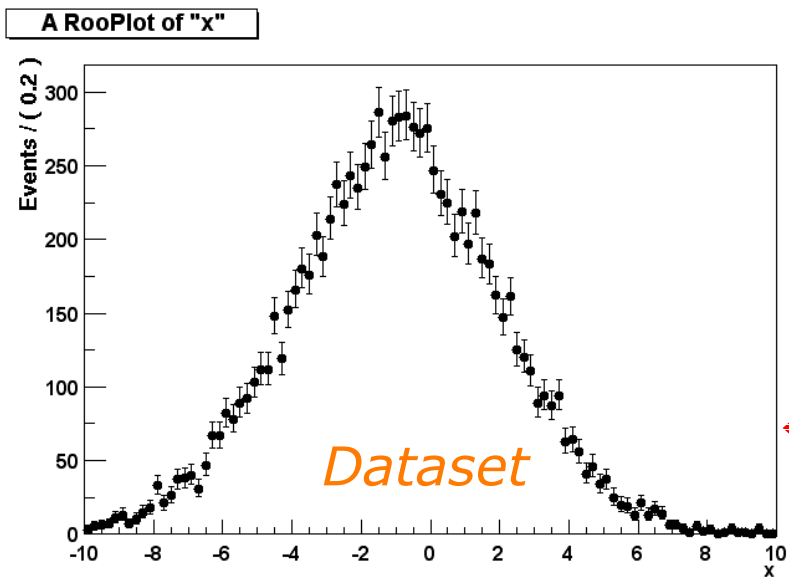
- Fit parallelization on multi-CPU hosts
  - Option for automatic parallelization of fit function on multi-CPU hosts (no explicit or implicit support from user PDFs needed)

RooFit key features – Event Generation

*Given **any** PDF, Toy Monte Carlo generation Is a 1-line operations*




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



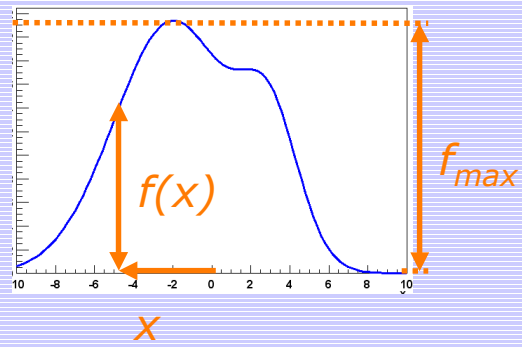
RooFit key features – Event generation

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

- 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***
  - Easy: just supply prototype dataset

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

Accept/reject sampling



0) Estimate maximum PDF value f_{max} by repeated sampling at random x

1) Throw a uniform random value for x

2)Throw uniform random number (ran) between 0 and f_{max}.

3) If ran*f_{max} < f(x), accept x as generated event

Repeat 1)-3) as necessary

RooFit key features – Plotting support

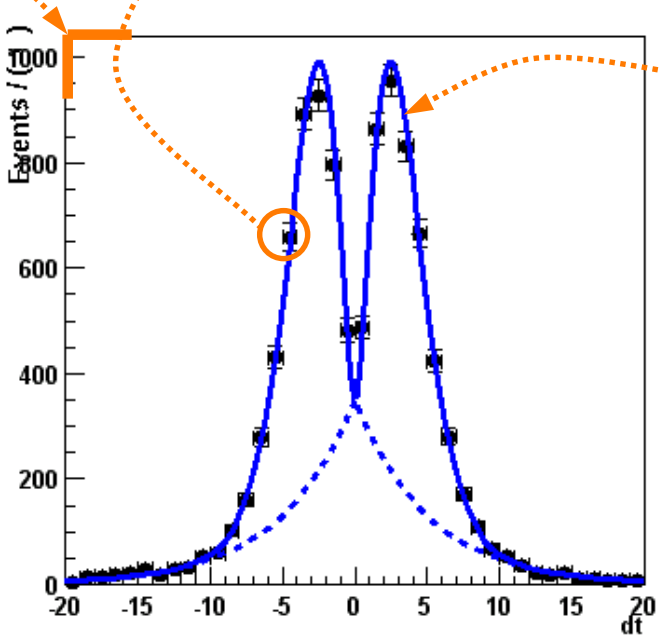
- Plot container class holds all elements of 1D plot together
  - Histograms of datasets, PDF projections, text boxes, arrow,…

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

Created from variable object.
Remembers variable on x-axis

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

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

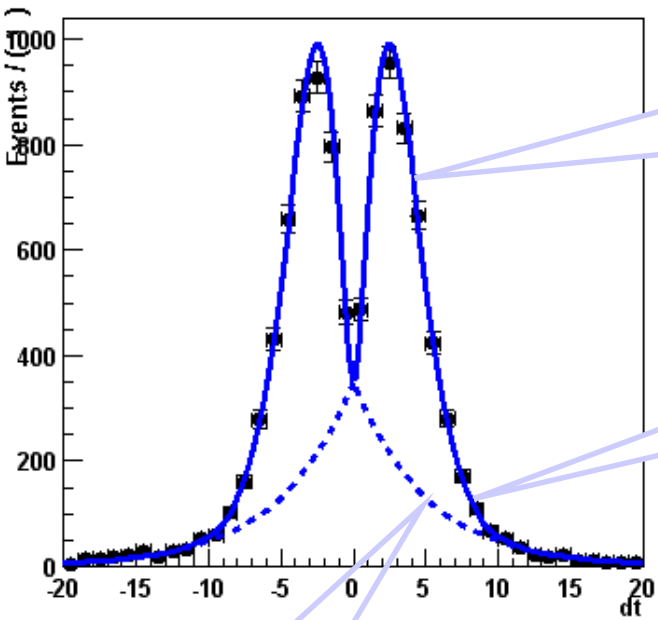



```
pdf->plotOn(frame) ;
```

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

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

RooFit key features – Plotting support

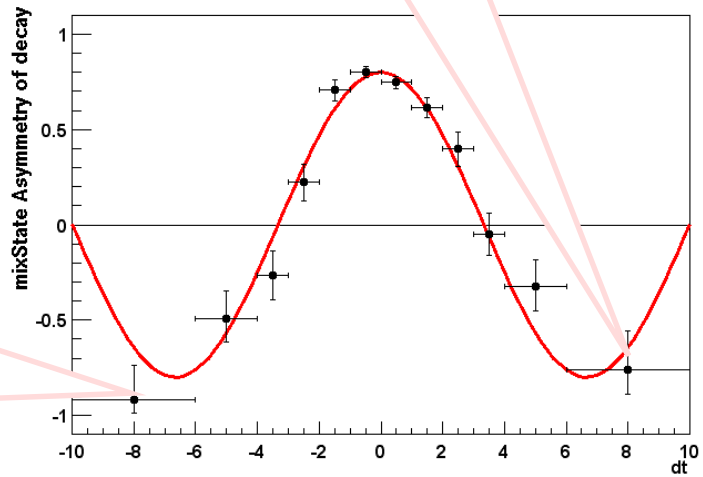


Adaptive spacing of curve points to achieve 1‰ precision

Poisson errors on histogram

Easy plotting of summed PDF components

Binomial errors on asymmetry histograms



Variable bin sizes OK (contents will be adjusted, if necessary)

RooFit key features – Plotting support

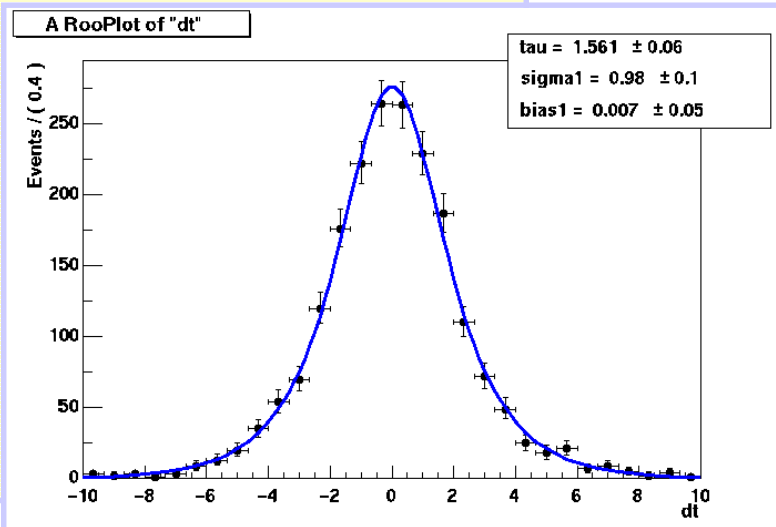
- Additional methods available to
  - Plot/project slices or arbitrarily shaped regions of PDFs
  - Plot PDF projections averaged over observables provided in a dataset
  - Plot generic asymmetries (A-B)/(A+B)

- 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")) ;
```

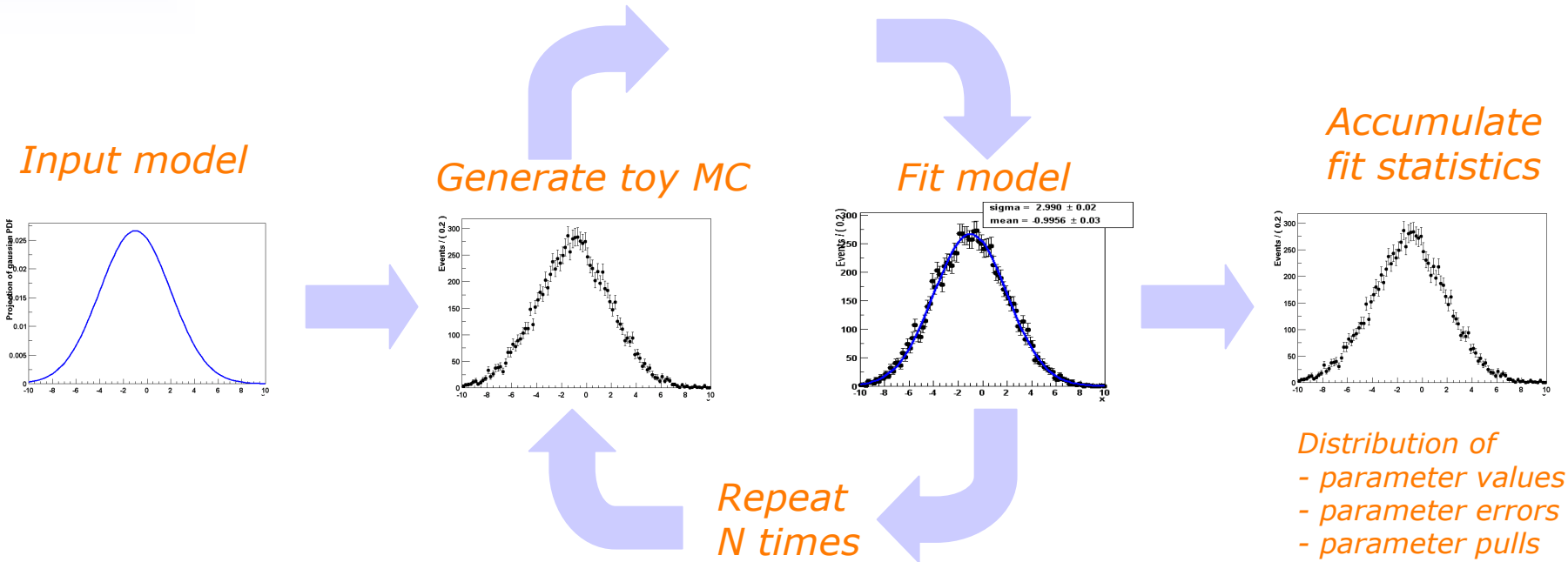
Complete example: generating and fitting a decay PDF

```
// Decay PDF variable
RooRealVar dt("dt","dt",-20,20) ;
RooRealVar tau("tau","tau",1.548,-10,10) ;

// Build a gaussian resolution model
RooRealVar bias("bias","bias",0,-5,5) ;
RooRealVar sigma("sigma","sigma",1,0.1,2.0) ;
RooGaussModel gm("gm","gauss model",dt,bias,sigma) ;

// Construct a decay (X) gm
RooDecay decay("decay","decay",dt,tau,gm,RooDecay::DoubleSided) ;

// Generate toy monte carlo dataset from the decay PDF
RooDataSet *data = decay.generate(dt,2000) ;

// Fit the generated data to the model
RooFitResult* r = decay.fitTo(*data) ;
r->correlation(sigma,tau) ;
-0.818443

// Make a plot of the data and PDF
RooPlot* dtframe = dt.frame(-10,10,30) ;
data->plotOn(dtframe) ;
pdf.plotOn(dtframe) ;
pdf.paramOn(dtframe) ;
dtframe->Draw() ;
```



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()
```

Advanced features – Data sub-classing

Probability density function $\mathbf{F(\vec{x}; \vec{p}, \vec{q})}$

- Physical parameters of interest $\vec{p}$
- Other parameters $\vec{q}$ to describe detector effect (resolution,efficiency,…)

- **Goal**: Exploit discrete information available in data sample to enhance statistical sensitivity of measurement of **p**
  - For example: reconstruction technique, particle decay mode,…
- **Method**: Use separate PDFs, with tailored set of parameters **q**, for each data subset. Fit all sub-samples simultaneously.
- **Challenge**: Build PDFs $\mathbf{F_A(x;p,q_A)}$, $\mathbf{F_B(x,p,q_B)}$,…
  - Code replication and bookkeeping can be cumbersome
- **Solution**: Use RooFit replication & customization manager
  - User provides single 'prototype' **F(x,p,q)**
  - Automatic building of $\mathbf{F_A(x;p,q_A)}$,… from prototype and customization prescription

Advanced features – Automated PDF building

- RooFit PDF replication & customization manager
  - Takes 'prototype' PDF and dataset with discrete subset index variable
  - Makes customized copies following a customization prescription

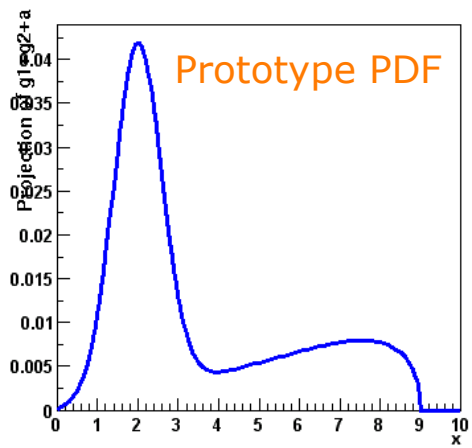
Replicated PDFs will have individually named copies of parameters f,s

`Dataset(x,subsetID)`

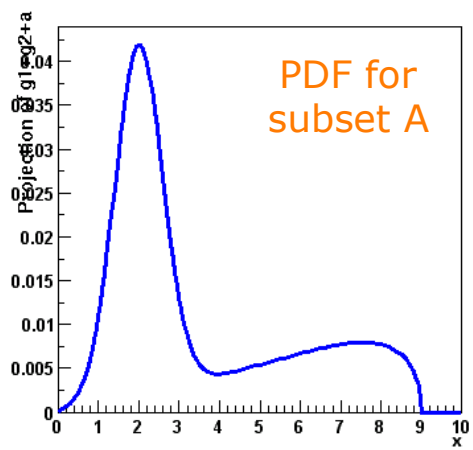
Customization prescription

```
f → f[subsetID]
s → s[subsetID]
```

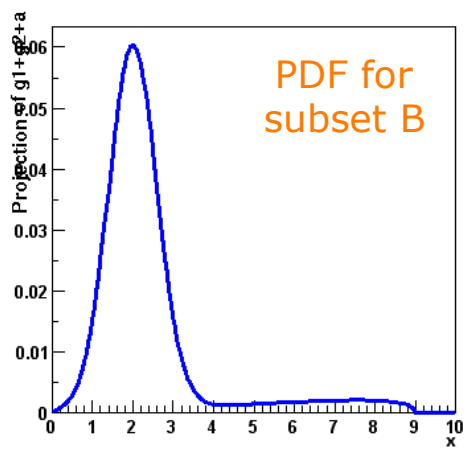
`f *G(x;m,s)+ (1-f)*A(x;a,c)`



`fA *G(x;m,sA)+ (1-fA)*A(x;a,c)`



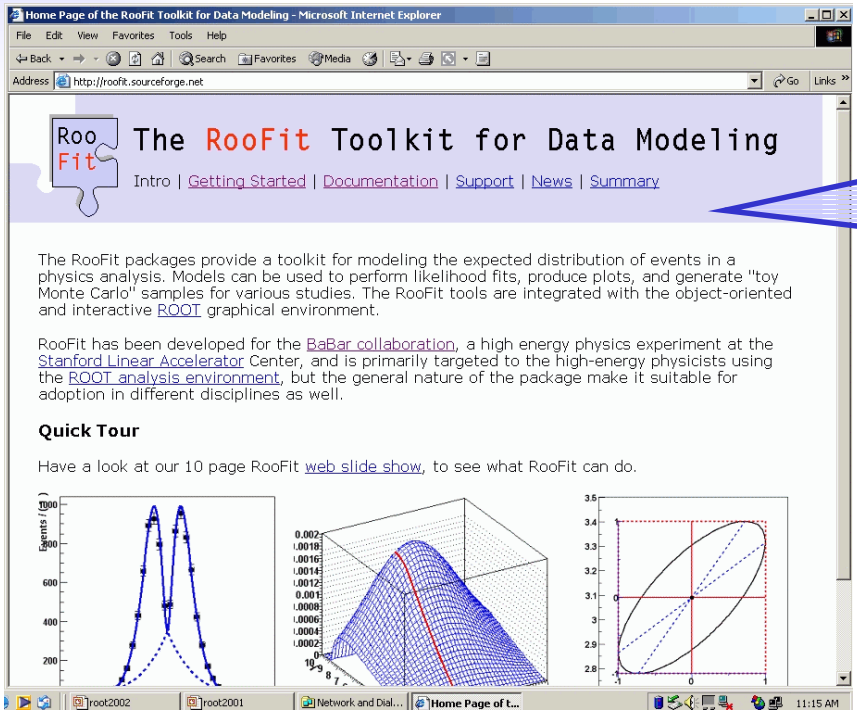
`fB *G(x;m,sA)+ (1-fB)*A(x;a,c)`



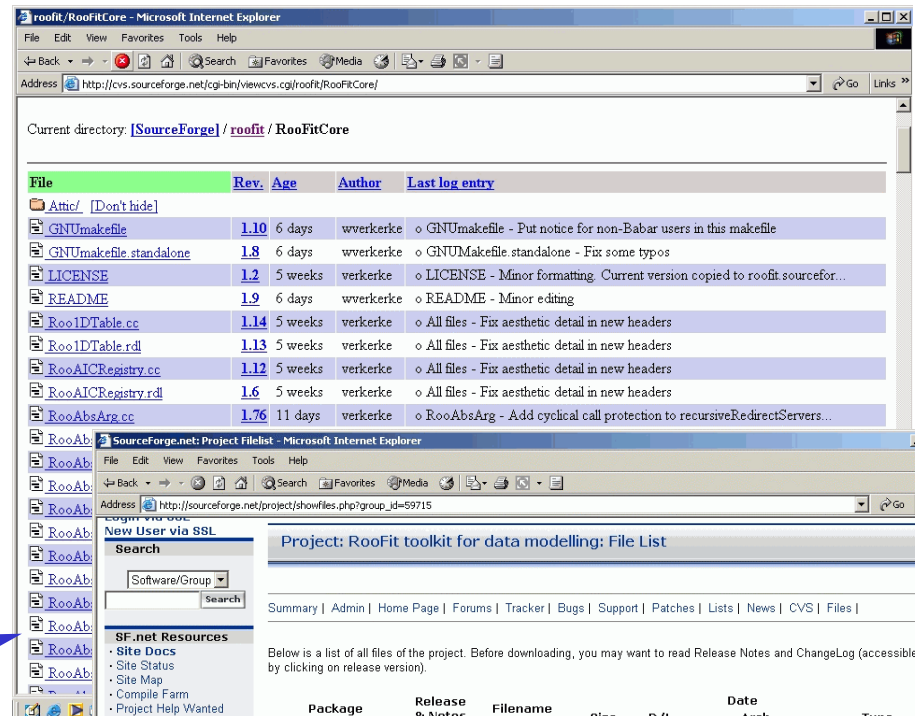
Development and Use of RooFit in BABAR

- 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, $D^{*}\lambda\nu$ 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*

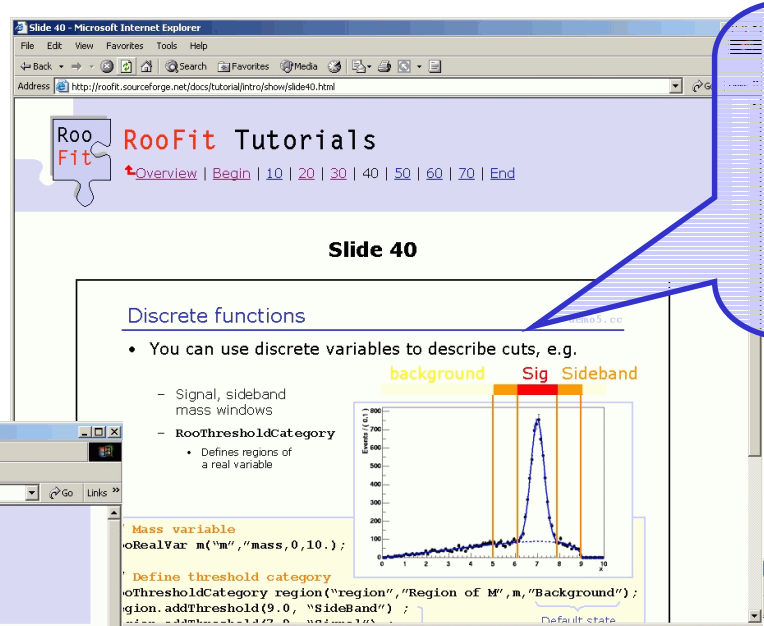
RooFit at SourceForge - Documentation

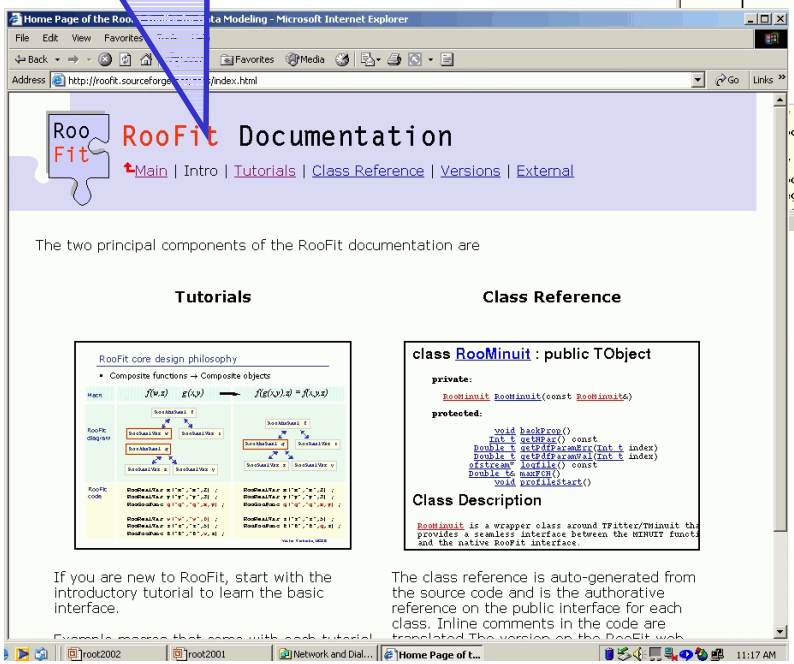
Documentation

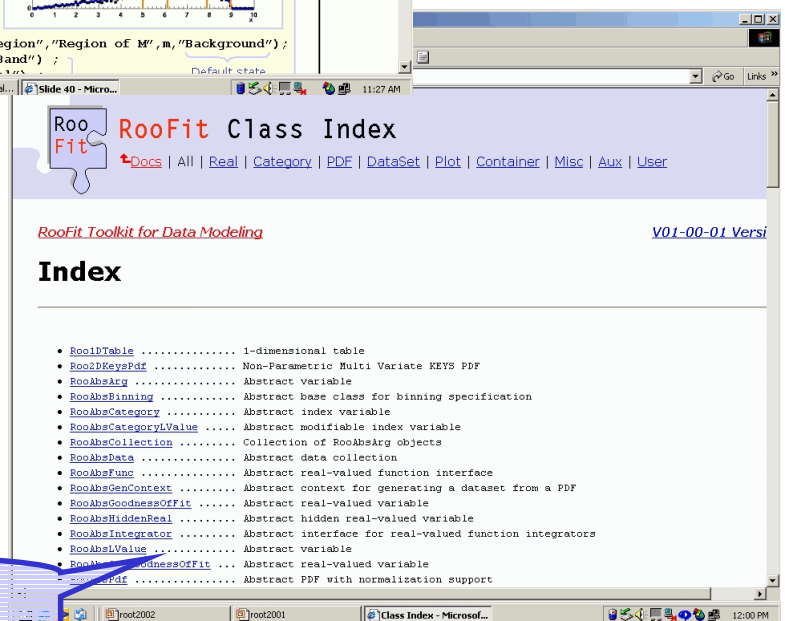
Comprehensive set of tutorials (PPT slide show + example macros)

Five separate tutorials

More than 250 slides and 20 macros in total







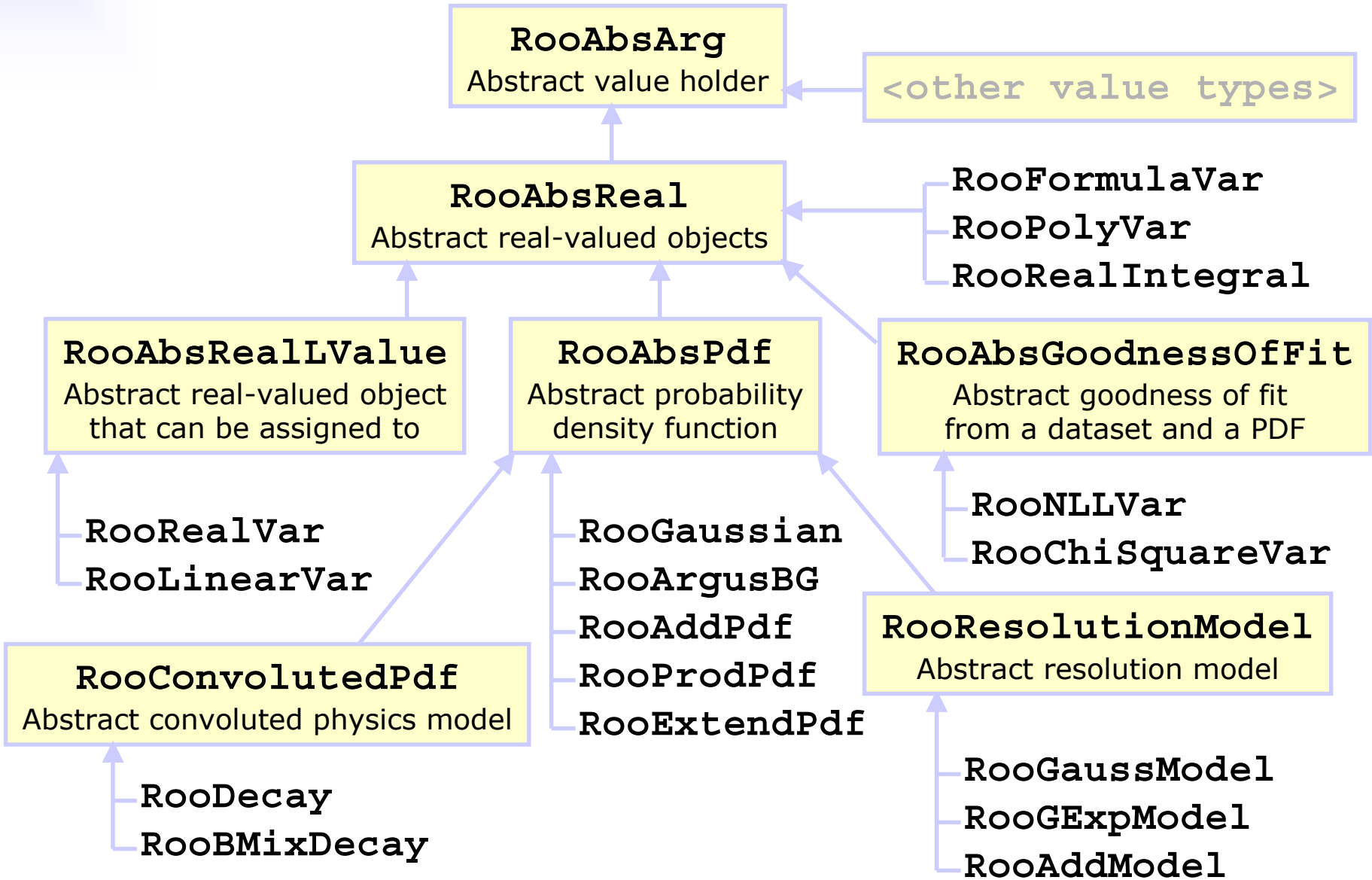
Class reference in `THtml` style

Summary / ROOT issues

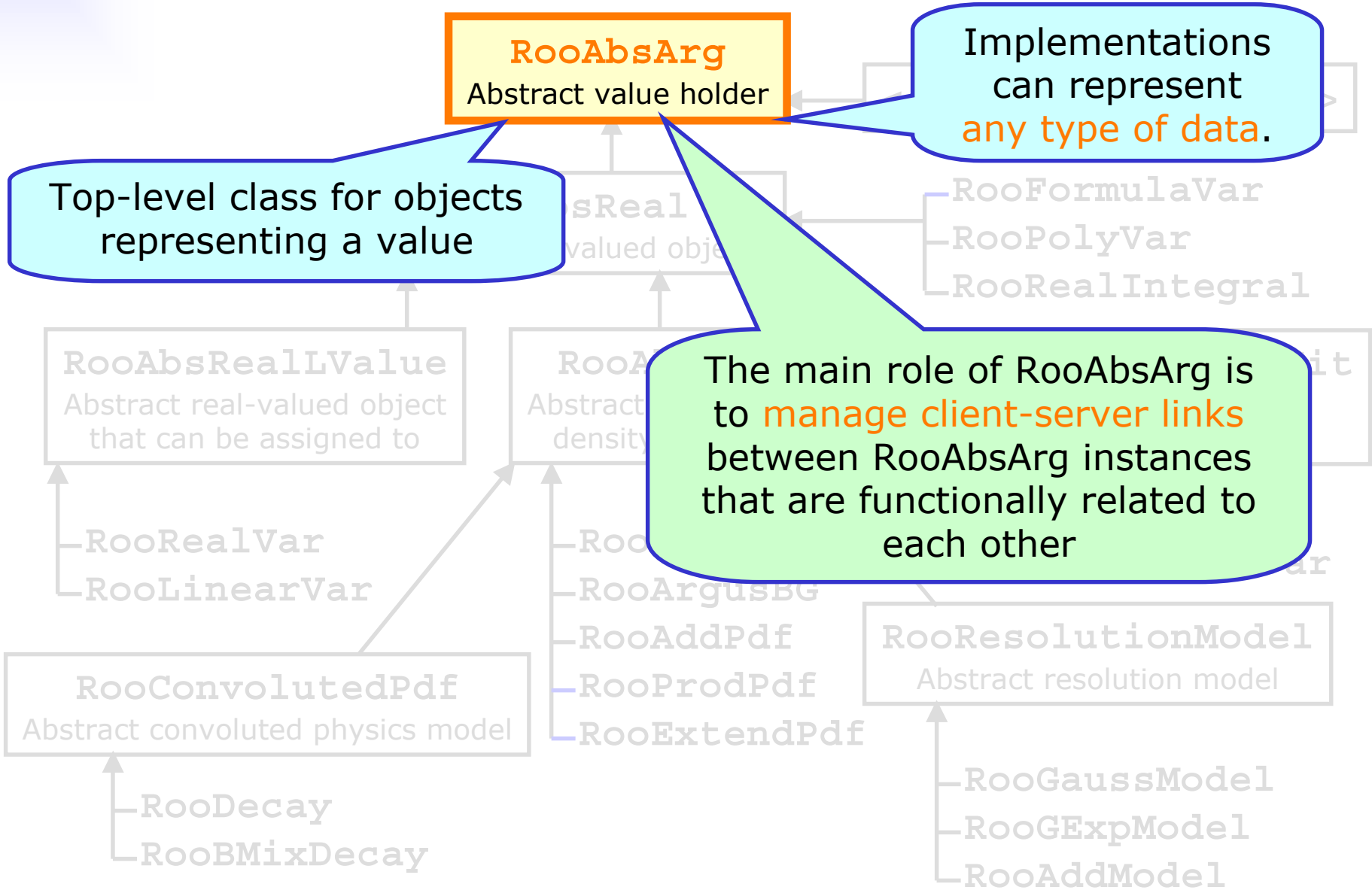
- RooFit has been developed into a stable product in the course of $2^1/_2$ years
- Now available to all (HEP) users: **`roofit.sourceforge.net`**
  - Documentation in good shape

- ROOT issues
  - In general, ROOT works great
  - RooFit only uses a small set of available ROOT features
    - Mostly trees and graphics primitives, MINUIT
  - **`TFitter/TMinuit`** has no good interface to extract correlation matrix
    - Must dig into MINUIT memory space to retrieve info, not very robust
  - CINT is steadily improving, but still running into limitations once in a while.

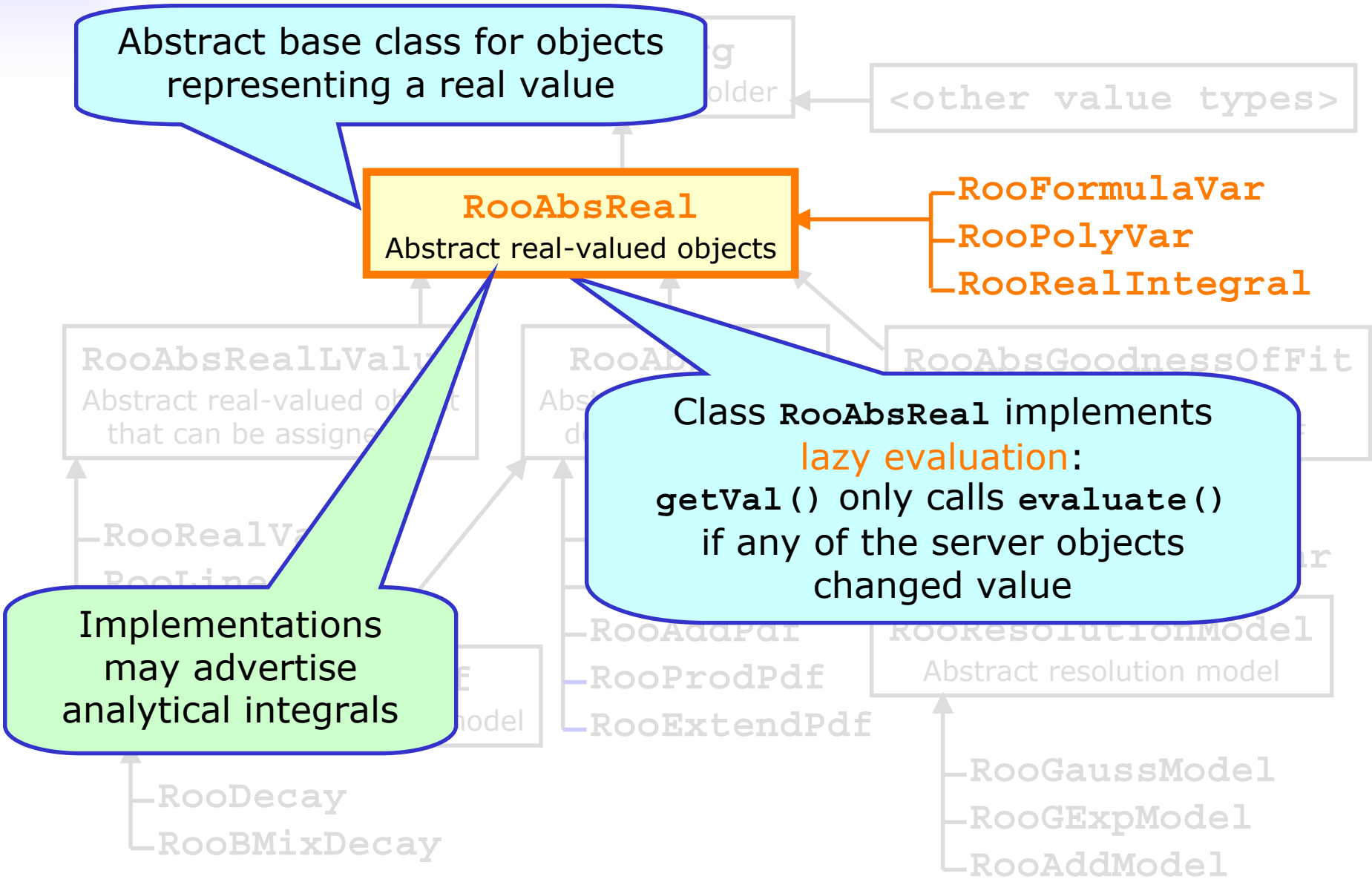
Hierarchy of classes representing a value or function



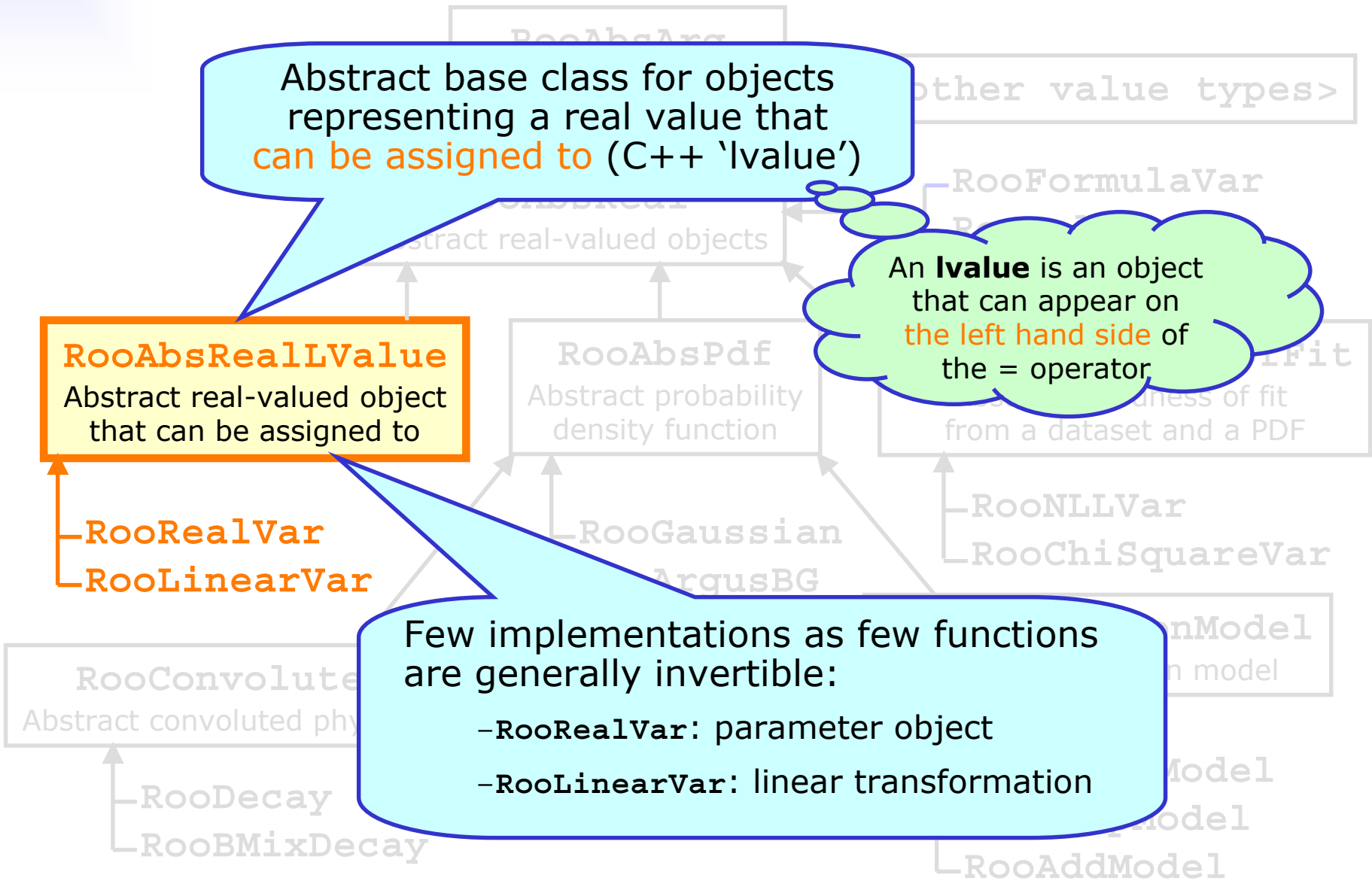
Class RooAbsArg



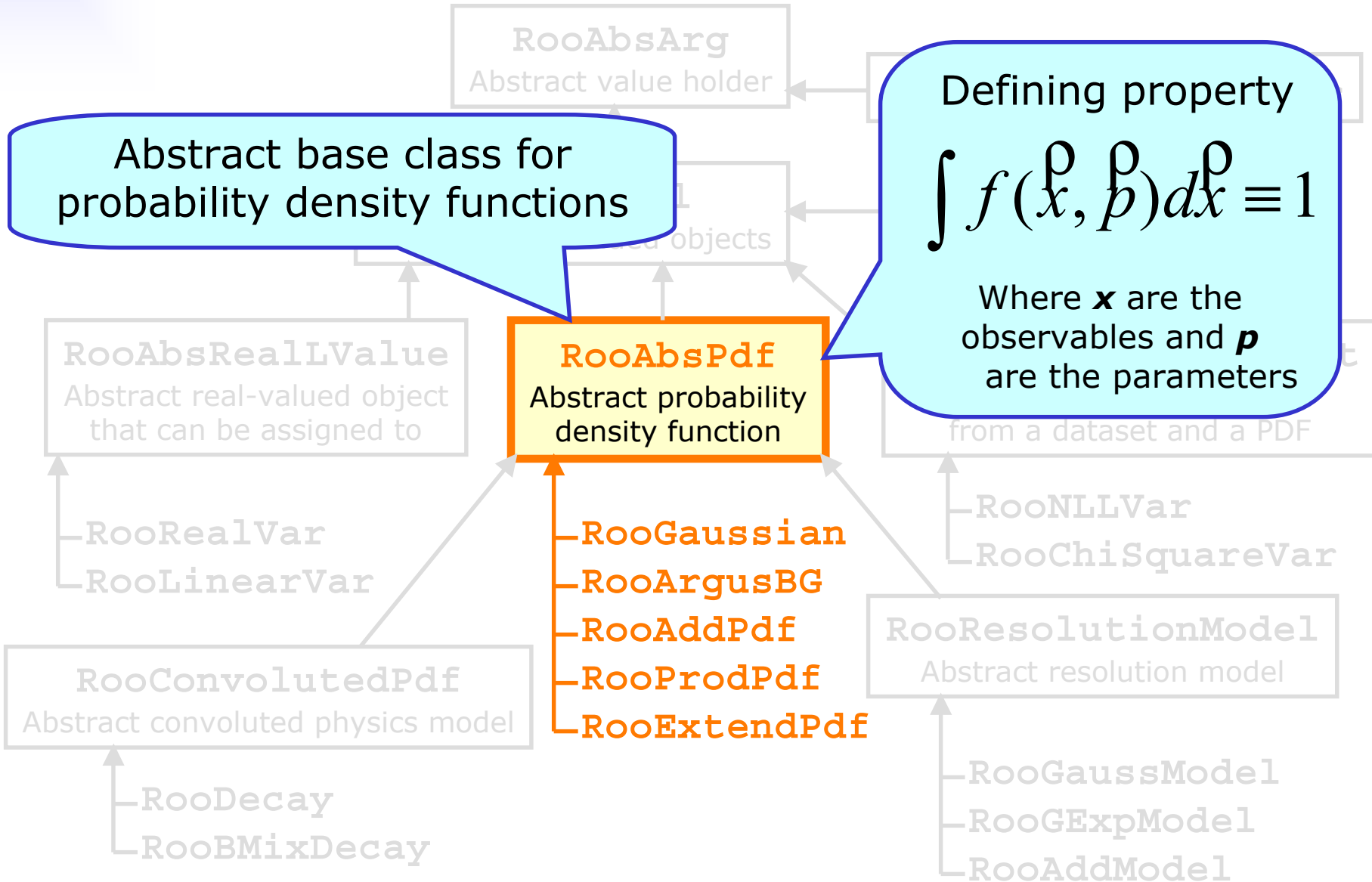
Class RooAbsReal



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

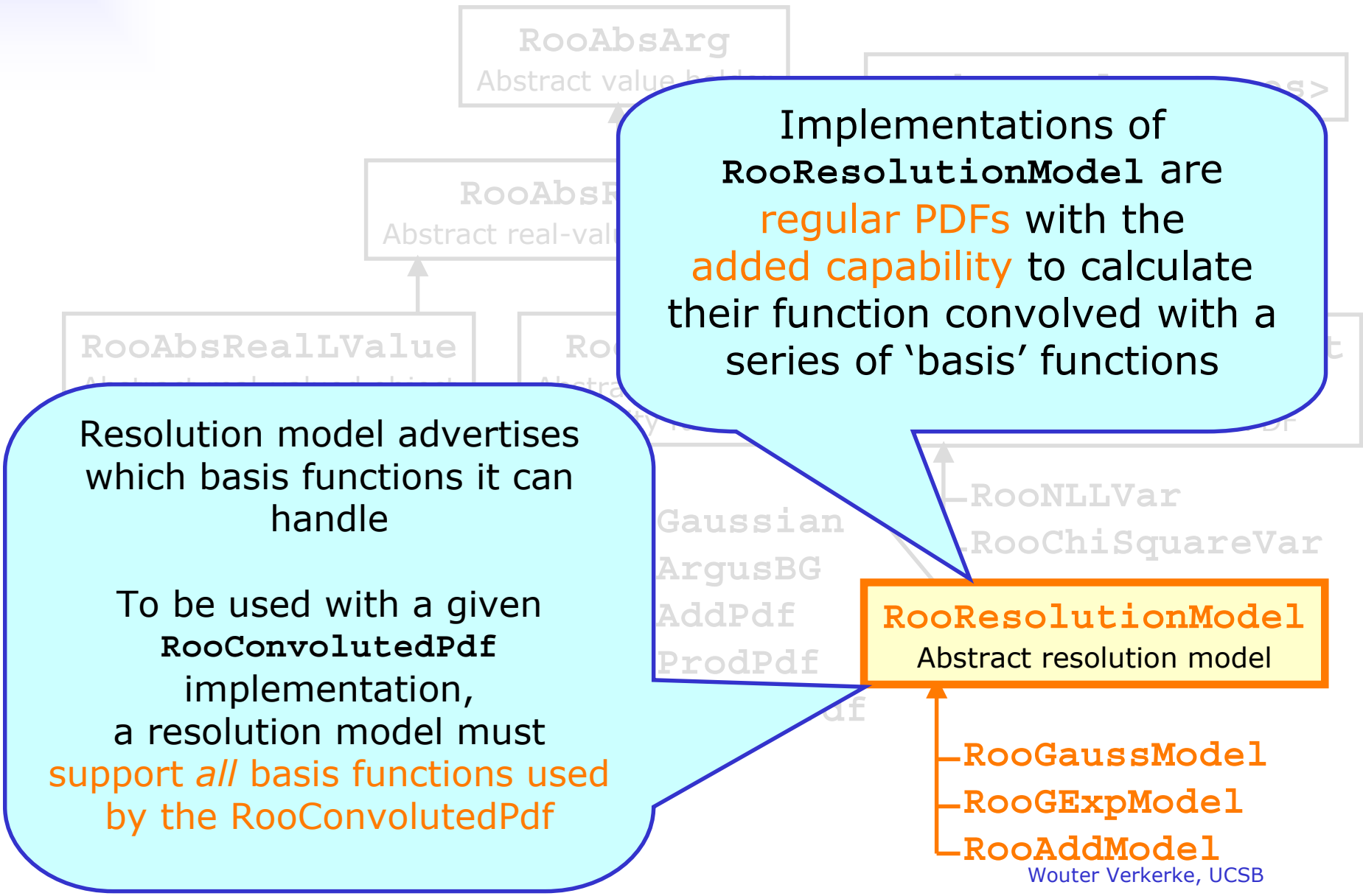
Abstract base class for PDFs that can be convoluted with a resolution model

RooArgusBG
RooAddPdf
RooProdPdf
RooExtendPdf

RooResolutionModel
Abstract resolution model

RooChiSquareVar

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