THE ROOFIT TOOLKIT FOR DATA MODELING

W. VERKERKE
University of California Santa Barbara, Santa Barbara, CA 93106, USA

D. KIRKBY
University of California Irvine, Irvine CA 92697, USA

`RooFit` is a library of C++ classes that facilitate data modeling in the ROOT environment. Mathematical concepts such as variables, (probability density) functions and integrals are represented as C++ objects. The package provides a flexible framework for building complex fit models through classes that mimic math operators, and is straightforward to extend. For all constructed models `RooFit` provides a concise yet powerful interface for fitting (binned and unbinned likelihood, χ^2), plotting and toy Monte Carlo generation as well as sophisticated tools to manage large scale projects. `RooFit` has matured into an industrial strength tool capable of running the BABAR experiment's most complicated fits and is now available to all users on SourceForge[1].

1. Introduction

One of the central challenges in performing a physics analysis is to accurately model the distributions of observable quantities $\vec{x}$ in terms of the physical parameters of interest $\vec{p}$ as well as other parameters $\vec{q}$ needed to describe detector effects such as resolution and efficiency. The resulting model consists of a "probability density function" (PDF) $F(\vec{x}\,;\vec{p},\vec{q})$ that is normalized over the allowed range of the observables $\vec{x}$ with respect to the parameters $\vec{p}$ and $\vec{q}$.

Experience in the BaBar experiment has demonstrated that the development of a suitable model, together with the tools needed to exploit it, is a frequent bottleneck of a physics analysis. For example, some analyses initially used binned fits to small samples to avoid the cost of developing an unbinned fit from scratch. To address this problem, a general-purpose toolkit for physics analysis modeling was started in 1999. This project fills a gap in the particle physicists' tool kit that had not previously been addressed.

A common observation is that once physicists are freed from the constraints of developing their model from scratch, they often use many observables simultaneously and introduce large numbers of parameters in order to optimally use the available data and control samples.

2. Overview

The final stages of most particle physics analysis are performed in an interactive data analysis framework such as PAW[2] or ROOT[3]. These applications provide an interactive environment that is programmable via interpreted macros and have access to a graphical toolkit designed for visualization of particle physics data. The `RooFit` toolkit extends the ROOT analysis environment by providing, in addition to basics visualization and data processing tools, a language to describe data models. The core features of `RooFit` are:

- A *natural and self-documenting vocabulary* to build a model in terms of its building blocks (e.g., exponential decay, Argus function, Gaussian resolution) and how they are assembled (e.g., addition, composition, convolution). A template is provided for users to add new PDFs specific to their problem domain.
- A *data description language* to specify the observable quantities being modeled using descriptive titles, units, and any cut ranges. Various data types are supported including real valued and discrete valued (e.g. decay mode). Data can be read from ASCII files or ROOT ntuples.
- *Generic support for fitting* any model to a dataset using a (weighted) unbinned or binned maximum likelihood, or χ^2 approach
- *Tools for plotting data with correctly calculated errors*, Poisson or binomial, and superimposing correctly normalized projections of a multidimensional model, or its components.
- *Tools for creating a event samples from any model with Monte Carlo techniques*, with some variables possibly taken from a prototype dataset, e.g. to more accurately model the statistical fluctuations in a particular sample.

• *Computational efficiency.* Models coded in RooFit should be as fast or faster than hand coded models. An array of automated optimization techniques is applied to any model without explicit need for user support.
• *Bookkeeping tools for configuration management*, automated PDF creation and automation of routine tasks such as goodness-of-fit studies.

3. Object-Oriented Mathematics

To keep the distance between a physicists' mathematical description of a data model and its implementation as small as possible, the RooFit interface is styled after the language of mathematics. The object-oriented ROOT environment is ideally suited for this approach: each mathematical object is represented by a C++ software object. Table 1 illustrates the correspondence between some basic mathematical concepts and RooFit classes.

Concept	Math Symbol	RooFit class name
Variable	x, p	RooRealVar
Function	$f(\vec{x})$	RooAbsReal*
PDF	$F(\vec{x};\vec{p},\vec{q})$	RooAbsPdf*
Space point	$\vec{x}$	RooArgSet
Integral	$\int_{\vec{x}_{min}}^{\vec{x}_{max}} f(\vec{x})d\vec{x}$	RooRealIntegal
List of points	$\vec{x}_k$	RooAbsData*

* Abstract base classes

Composite objects are built by creating all their components first. For example, a Gaussian probability density function with its variables is created as follows:

```
RooRealVar x("x","x",-10,10) ;
RooRealVar m("m","mean",0) ;
RooRealVar s("s","sigma",3) ;
RooGaussian g("g","gauss(x,m,s)",x,m,s) ;
```

Each object has a name, the first argument, and a title, the second argument. The name serves as unique identifier of each object, the title can hold a more elaborate description of each object and only serves documentation purposes.

Function objects are linked to their ingredients: the function object `g` *always* reflects the values of its input variables `x`,`m`, and `s`. The absence of any explicit invocation of calculation methods allows for true symbolic manipulation in mathematical style.

RooFit implements its data models in terms of probability density functions. The normalization of probability density functions, traditionally one of the most difficult aspects to implement, is handled internally by RooFit: all PDF objects are automatically normalized to unity. If a specific PDF class doesn't provide its normalization internally, a variety of numerical techniques are used to calculate the normalization.

Composition of complex models from elementary PDFs is straightforward: a sum of two PDFs is a PDF, the product of two PDFs is a PDF. The RooFit toolkit provides as set of 'operator' PDF classes that represent the sum of any number of PDFs, the product of any number of PDFs and the convolution of two PDFs.

Existing PDF building blocks can be tailored using standard mathematical techniques by substituting a variable with a formula expression. Free-form interpreted C++ function and PDF objects are available to glue together larger building blocks. The universally applicable composition operators and free-style interpreted functions make it possible to write probability density functions of arbitrary complexity in a straightforward mathematical form.

4. Composing and Using Data Models

We illustrate the process of building a model and its various uses with a simple one-dimensional yield fit example.

The RooFit models library provides more than 20 basic probability density functions that are commonly used in high energy physics applications, including basics PDFs such Gaussian, exponential and polynomial shapes, physics inspired PDFs, e.g. decay functions, Breit-Wigner, Voigtian, Argus shape, Crystal Ball shape, and non-parametric PDFs (histogram and KEYS[4]).

In the example below we use two such PDFs: a Gaussian and an ARGUS background function:

```
// Observable
RooRealVar mes("mes","mass_ES",-10,10) ;

// Signal model and parameters
RooRealVar mB("mB","m(B0)",0) ;
RooRealVar w("w","Width of m(B0)",3) ;
RooGaussian G("G","G(meas,mB,width)",mes,mB,w) ;
```

```
// Background model and parameters
RooRealVar m0("m0","Beam energy / 2",-10,10) ;
RooRealVar k("k","ARGUS slope parameter",3) ;
RooArgusBG A("A","A(mes,m0,k)",mes,m0,k) ;

// Composite model and parameter
RooRealVar f("f","signal fraction",0,1) ;
RooAddPdf M("M","G+A",RooArgList(G,A),f) ;
```

The `RooAddPdf` operator class `M` combines the signal and background component PDFs with two parameters each into a composite PDF with five parameters:

$$M(m_{ES};m_B,w,m_0,k,f) = f\cdot G(m_{ES};w,g) + (1-f)\cdot A(m_{ES};m_0,k).$$

Once the model `M` is constructed, a maximum likelihood fit can be performed with a single function call:

```
M.fitTo(*data) ;
```

Fits performed this way can be unbinned, binned and/or weighted, depending on the type of dataset provided. The result of the fit, the new parameter values and their errors, are immediately reflected in the `RooRealVar` objects that represent the parameters of the PDF, `mB`,`w`,`m0`,`k` and `f`. Parameters can be fixed in a fit or bounded by modifying attributes of the parameter objects prior to the fit:

```
m0.setConstant(kTRUE) ;
f.setRange(0.5,0.9) ;
```

Visualization of the fit result is equally straightforward:

```
RooPlot* frame = mes.frame() ;
data->plotOn(frame) ;
M.plotOn(frame) ;
M.plotOn(frame,Components("A"),
                LineStyle(kDashed)) ;
frame->Draw()
```

A `RooPlot` object represents a one-dimensional view of a given observable. Attributes of the `RooRealVar` object `mes` provide default values for the properties of this view (range, binning, axis labels). Figure 1 shows the result of the `frame->Draw()` operation in the above code fragment.

The default error bars drawn for a dataset are asymmetric and correspond to a Poisson confidence interval equivalent to 1σ for each bin content. The curve of the PDF is automatically normalized to the number of events of the dataset last plotted in the same frame. The points of the curve are chosen by an adaptive resolution-based technique: the deviation between the function value and the curve will not exceed a given tolerance regardless of the binning of the plotted dataset.

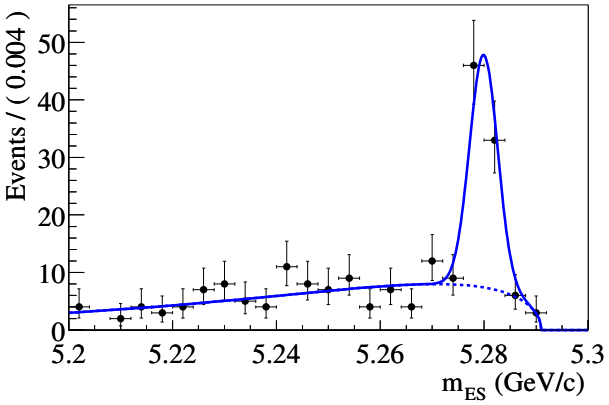

Fig. 1. One dimensional plot with histogram of a dataset, overlaid by a projection of the PDF `M`. The histogram error are asymmetric, reflecting the Poisson confidence interval corresponding to a 1σ deviation. The PDF projection curve is automatically scaled to the size of the plotted dataset.

The `plotOn()` methods of datasets and functions accept optional arguments that modify the style and contents of what is drawn. The second `M.plotOn()` call in the preceding example illustrates some of the possibilities for functions: only the `A` component of the composite model `M` is drawn and the line style is changed to a dashed style. Similarly, the presentation of datasets can be changed, for example a sum-of-weights error ($\sqrt{\Sigma_i w_i^2}$) can optionally be selected for use with weighted datasets.

5. Efficiency and Optimal Function Calculation

As the complexity of fits increases, efficient use of computing resources becomes increasingly important. To speed up the evaluation of probability density functions, optimization techniques such as value caching and factorized calculations can be used.

Traditionally such optimizations require a substantial programming effort due to the large amount of bookkeeping involved, and often result in incomplete use of available optimization techniques due to lack of time or expertise. Ultimately such optimizations represent a compromise between development cost, speed and flexibility.

RooFit radically changes this equation as the object-oriented structure of its PDFs allows centrally provided algorithms to analyze any PDFs structure and to apply generic optimization techniques to it. Examples of the various optimization techniques are:
• *Precalculation of constant terms.* In a fit, parts of a PDF may depend exclusively on constant parameters. These components can be precalculated once and used throughout the fit session.
• *Caching and lazy evaluation.* Functions are only recalculated if any of their input has changed. The actual calculation is deferred to the moment that the function value is requested.
• *Factorization.* Objects representing a sum, product or convolution of other PDFs, can often be factorized from a single N-dimensional problem to a product of N easier-to-solve 1-dimensional problems.
• *Parallelization.* Calculation of likelihoods and other goodness-of-fit quantities can, due to their repetitive nature, easily be partitioned in to set of partial results that can be combined a posteriori. RooFit automates this process and can calculate partial results in separate processes, exploiting all available CPU power on multi-CPU hosts.

Optimizations are performed automatically and tailored to each potentially CPU intensive operation. This realizes the maximum available optimization potential for every operation at no cost for the user.

6. Data and Project Management Tools

As analysis projects grow in complexity, users are often confronted with an increasing number of logistical issues and bookkeeping tasks that may ultimately limit the complexity of their analysis. RooFit provides a variety of tools to ease the creation and management of large numbers of datasets and probability density functions such as:
• *Discrete variables.* A discrete variable in RooFit is a variable with a finite set of named states. The naming of states, instead of enumerating them, facilitates symbolic notation and manipulation.
• *Automated PDF building.* A common analysis technique is to classify the events of a dataset D into subsets D_i, and simultaneously fit a set of PDFs $P_i(\vec{x}, \vec{p}_i)$ to these subsets D_i. In cases where individually adjusted PDFs $P_i(\vec{x}, \vec{p}_i)$ can describe the data better than a single global PDF $P(\vec{x}, \vec{p})$, a better statistical sensitivity can be obtained in the fit. Often, such PDFs do not differ in structure, just in the value of their parameters. RooFit offers a utility class to automate the creation the the PDFs $P_i(\vec{x}, \vec{p}_i)$: given a prototype PDF $P(\vec{x}, \vec{p})$ and a set of rules that explain how the prototype should be altered for use in each subset (e.g. "Each subset should have its own copy of parameter `foo`") this utility builds entire set of PDFs $P_i(\vec{x}, \vec{p}_i)$.
• *Project configuration management.* Advanced data analysis projects often need to store and retrieve the projection configuration, such as initial parameters values, names of input files and other parameter that control the flow of execution. RooFit provides tools to store such information in a standardized way in easy-to-read ASCII files. The use of standardized project management tools promotes structural similarity between analyses and increases a users' ability to understand other RooFit projects and to exchange ideas and code.

7. Development Status

RooFit was initially released as RooFitTools in 1999 in the BaBar collaboration and has over the years been adopted by virtually all BaBar physics analyses. Analysis topics include searches for rare B decays, measurements of B branching fractions and CP-violating rate asymmetries, time-dependent analyses of B and D decays to measure lifetime, mixing, and symmetry properties, and Dalitz analyses of B decays to determine form factors. Since October 2002 RooFit is available to the entire HEP community: the code and documentation repository has been moved from BaBar to SourceForge, an OpenSource development platform, which provides easy and equal access to all HEP users. (`http://roofit.sourceforge.net`). Since July 2005 RooFit is also bundled with ROOT releases, starting with ROOT version 5.02-00.

References

1. `http://roofit.sourceforge.net`
2. R. Brun et al., *Physics Analysis Workstation*, CERN Long Writeup Q121
3. R. Brun and F Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also `http://root.cern.ch`
4. K. Cranmer, *Kernel Estimation in High-Energy Physics*, Comp. Phys. Comm **136**, 198-207 (2001).