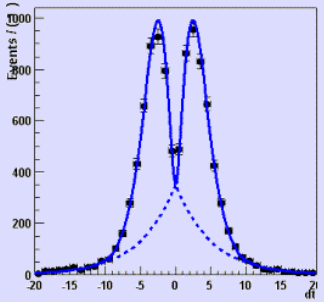
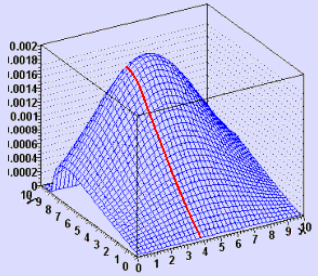
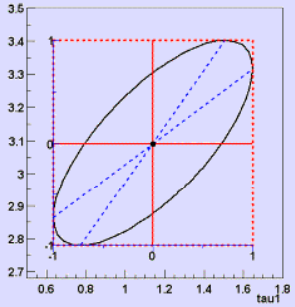






RooFit

A tool kit for data modeling in ROOT

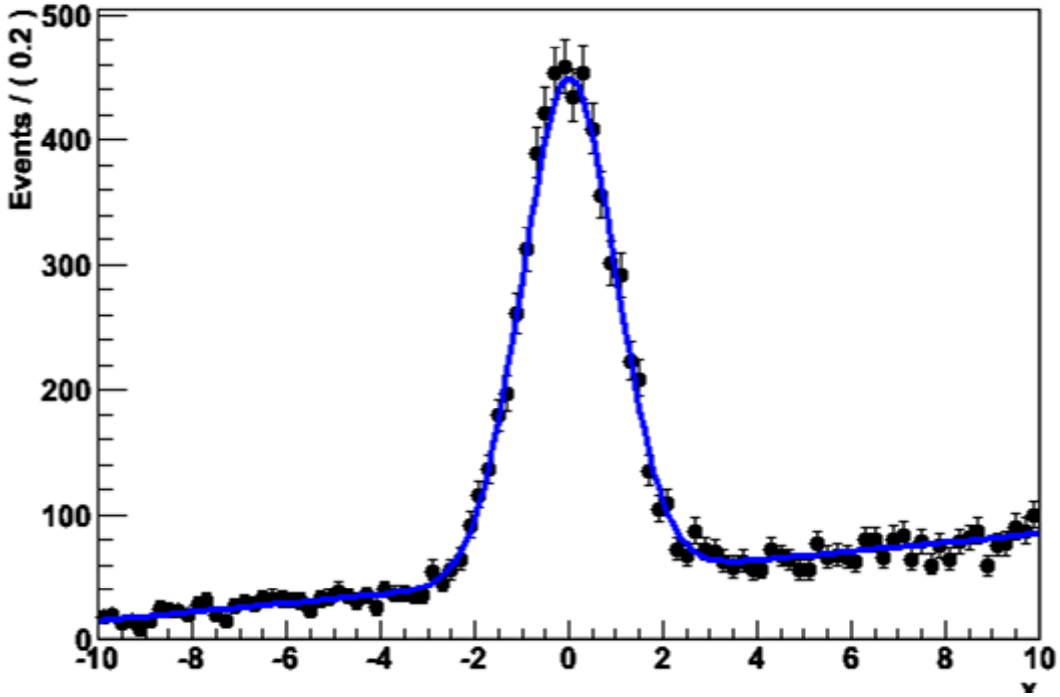
Wouter Verkerke (NIKHEF)
David Kirkby (UC Irvine)

1st half of talk – RooFit overview

2nd half of talk – Plans for 2007

Focus: coding a probability density function

- Focus on one practical aspect of many data analysis in HEP: How do you formulate your p.d.f. in ROOT
  - For 'simple' problems (gauss, polynomial), ROOT built-in models well sufficient



  - But if you want to do unbinned ML fits, use non-trivial functions, or work with multidimensional functions you are quickly running into trouble

Why RooFit was developed

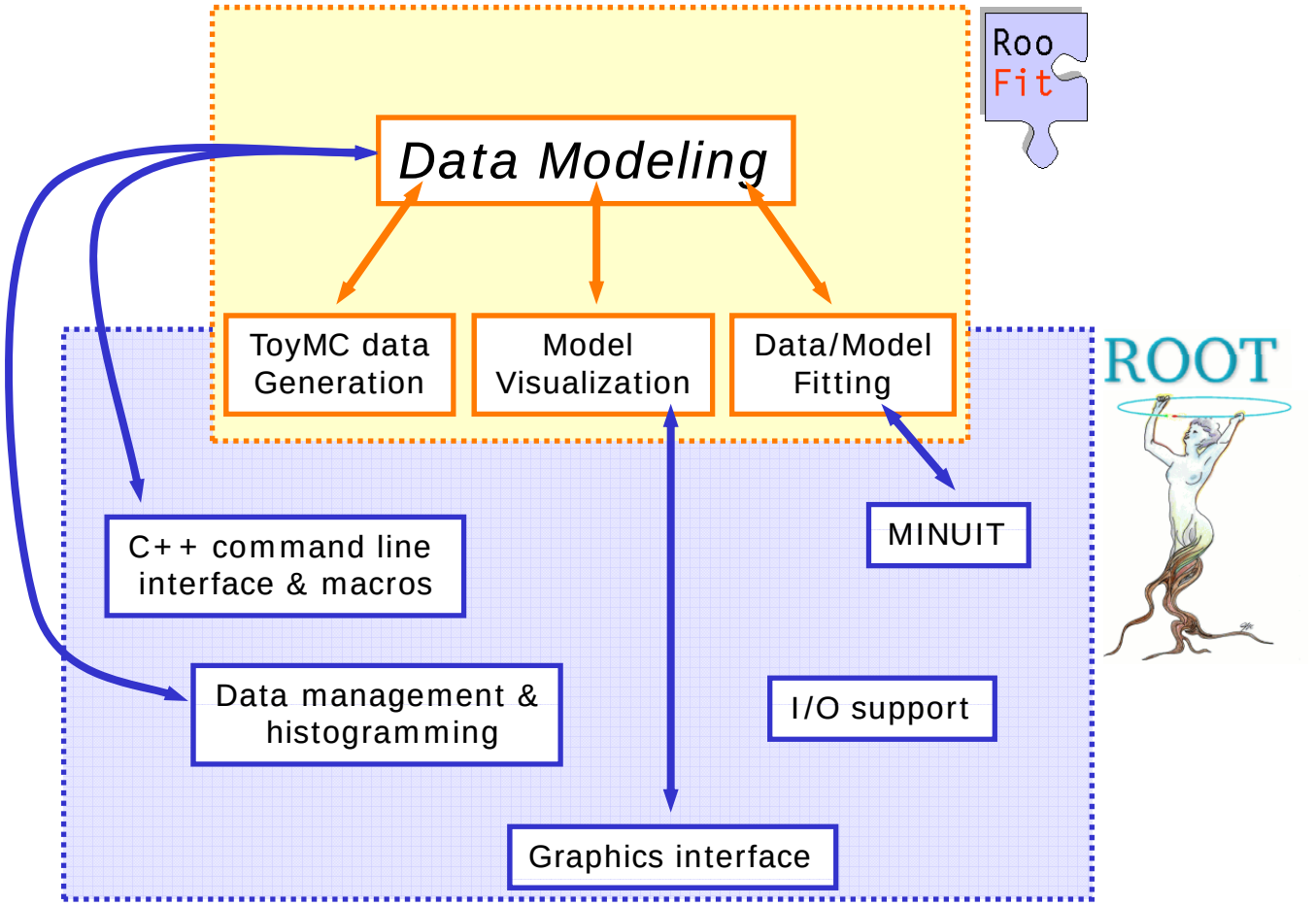
- BaBar experiment at SLAC: Extract sin(2β) from time_ dependent CP violation of B decay: e^+e^- → Y(4s) → $B\bar{B}$
  - Reconstruct both Bs, measure decay time difference
  - Physics of interest is in decay time dependent oscillation

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

- Many issues arise
  - Standard ROOT function framework clearly insufficient to handle such complicated functions → must develop new framework
  - Normalization of p.d.f. not always trivial to calculate → may need numeric integration techniques
  - Unbinned fit, >2 dimensions, many events → computation performance important → must try optimize code for acceptable performance
  - Simultaneous fit to control samples to account for detector performance

RooFit – a data modeling language for ROOT

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



Data modeling - Desired functionality

Analysis work 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 – OO representation

- Idea: represent math symbols as C++ objects

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

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

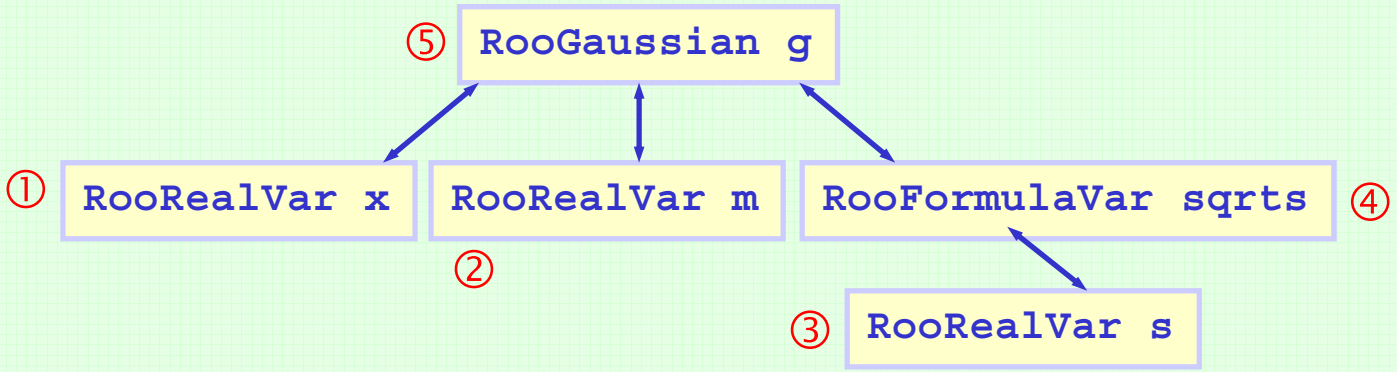
Data modeling – Constructing composite objects

- Straightforward correlation between mathematical representation of formula and RooFit code

Math

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

RooFit diagram

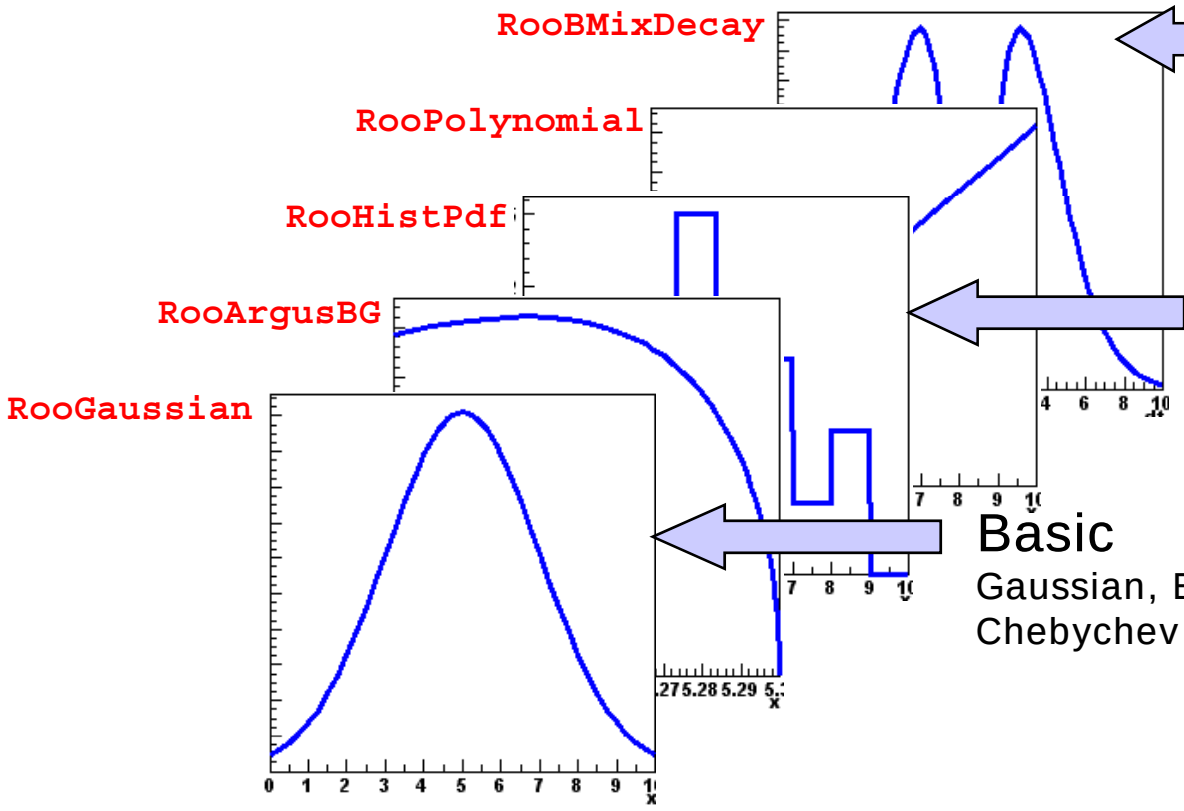


RooFit code

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

Model building – (Re)using standard components

- RooFit provides a collection of compiled standard PDF classes



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

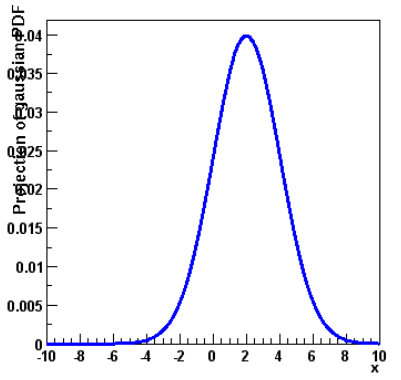
Non-parametric
Histogram, **KEYS**

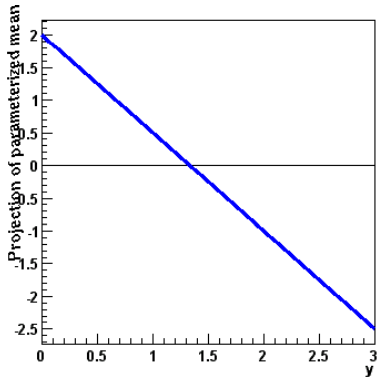
Basic
Gaussian, Exponential, Polynomial,...
Chebychev polynomial

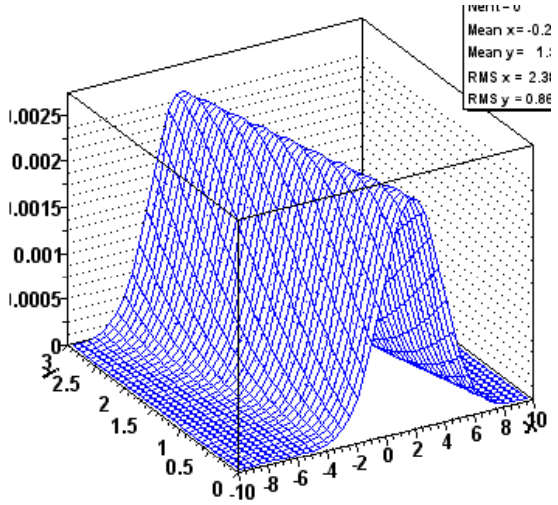
Easy to extend the library: each p.d.f. is a separate C++ class

Model building – (Re)using standard components

- Library p.d.f.s can be adjusted on the fly.
  - Just plug in *any function expression* you like as input variable
  - Works universally, even for classes you write yourself







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

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

- Maximum flexibility of library shapes keeps library small

Handling of p.d.f normalization

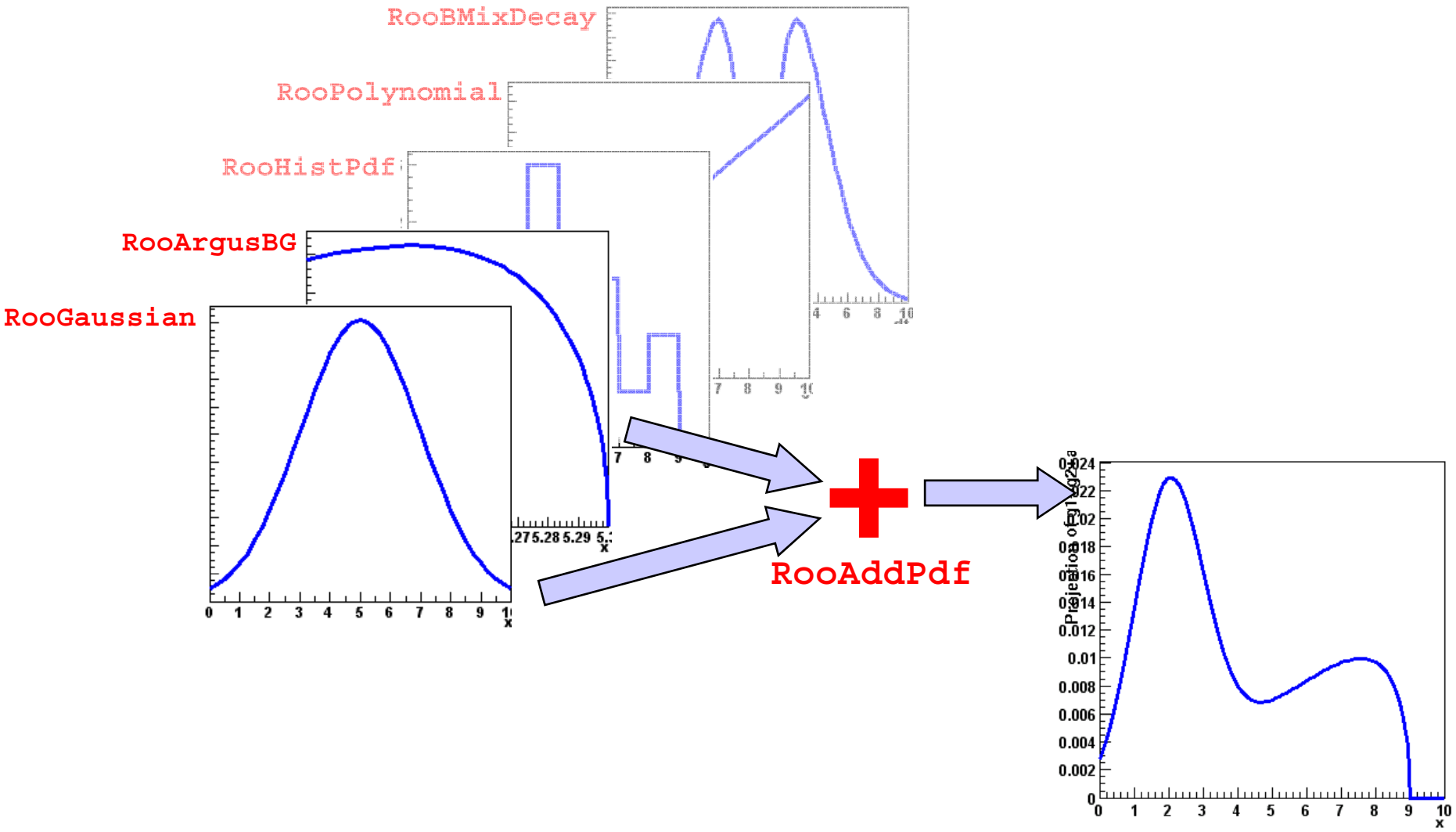
- Normalization of (component) p.d.f.s to unity is often a good part of the work of writing a p.d.f.

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

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

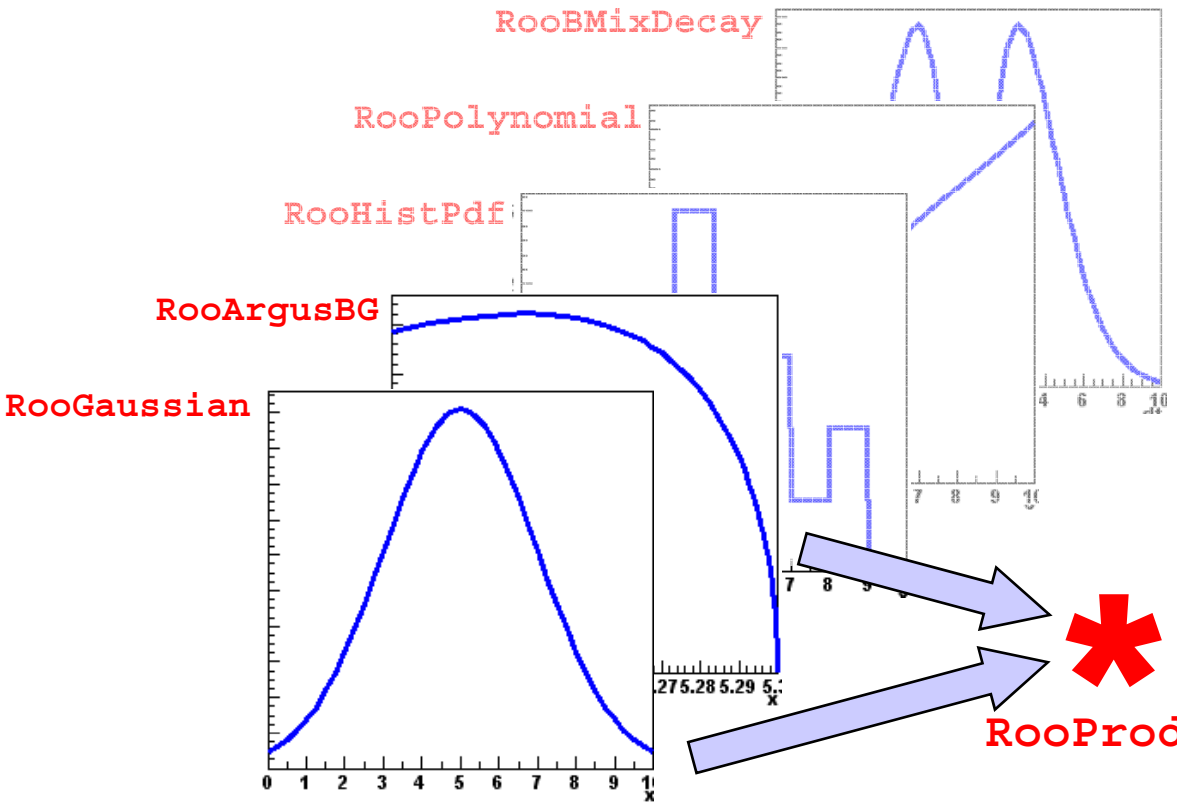
Model building – (Re)using standard components

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



Model building – (Re)using standard components

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

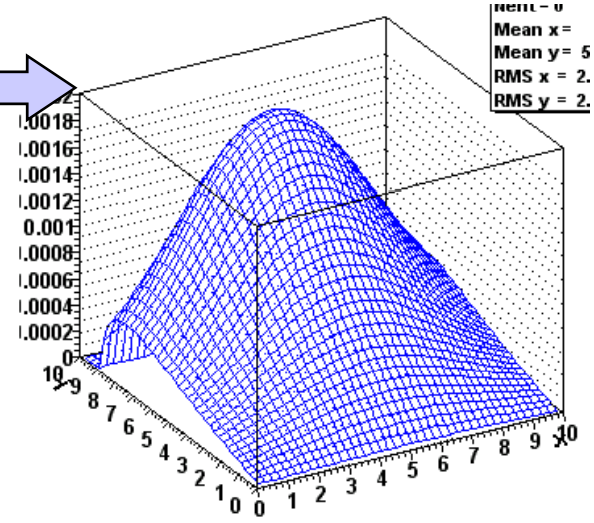



```
RooProdPdf h("h","h",
            RooArgSet(f,g))
```

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

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

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

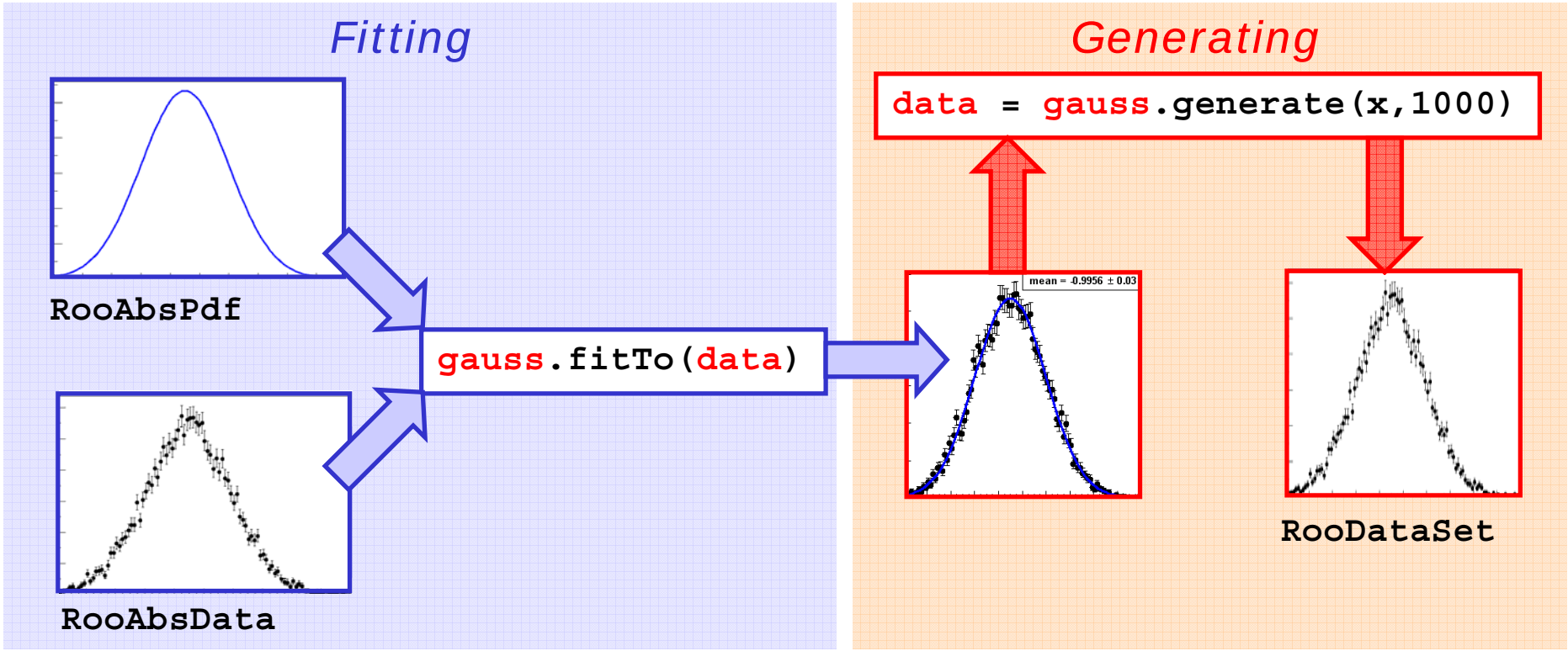


Convolution operation also supported

(upgrade of this capability in progress)

Using models - Overview

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



Fitting functionality

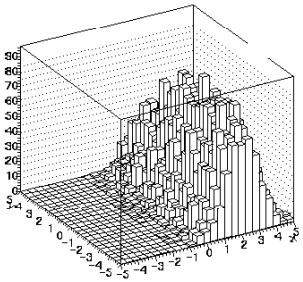
pdf.fitTo(data)

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

Unbinned

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

Binned



RooDataSet

RooDataHist

RooAbsData

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

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

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

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

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

Plans for RooFit in the next year (roughly)

- NB: I work for ATLAS now, no longer for BaBar
  - I will continue working on RooFit (part time – no change)
  - ATLAS Statistics workshop in January 2007 showed already quite some of use of RooFit in ATLAS → RooFit should be of interest to most LHC experiments
- CVS: Change of repository from SourceForge → ROOT.
  - Now: Have script to translate code from BaBar → ROOT organization
  - Ship tarballs to ROOT team for updates.
  - Cumbersome for small updates. No flexibility for ROOT team to make trivial changes to code (e.g. to follow a const declaration change)
  - Will provide reverse translation script for BaBar
- Aim to reduce package to 'core' business
  - Quite a few imported numeric algorithms for integration, function calculated over time.
  - Most are meanwhile also available in ROOT → Make adapter classes to interface ROOT code where necessary to RooFit code
  - Also applies to large extent to basic p.d.f classes that are being developed for MathCore/MathMore.
    - RooFit p.d.f. design conceptually very different due to concept of separate objects represening variable, so one cannot simply plug in ROOT functions into RooFit, but (perhaps universal) adapter class(es) can be written that make the binding between ROOTs 'float' variables and RooFits object variables.

Plans for RooFit in the next year (roughly)

- Rewrite generic 1D-convolution operation using FFT
  - Calculation of convolutions through Fast Fourier Transforms much more efficient that direct calculation of convolution integral
  - Will rely on FFTW interface in ROOT for actual calculation of fourier transforms

- Reorganize p.d.f. classes that cache pre-calculated values
  - Now many classes that require/benefit from caching of p.d.f values over entire observable domain:
    - FFT-based convolution: Calculates p.d.f values for all observable value in one operation
    - KEYS p.d.f: Calculation of p.d.f. value is very expensive and value never changes as p.d.f. has no parameters → Currently ad-hoc solution with caching array
    - Histogram p.d.f: The array *is* the p.d.f
    - Any other user-invented classes where p.d.f. calculation per event is very expensive, but over the entire domain not.
  - Create common base class with caching implementation.
    - Derived class decide if caching details should be exposed in user interface

Plans for RooFit: Add project management infrastructure

- Two-fold goal of infrastructure:
  1) Incorporating & recycling experience from BaBar
  2) Ground work to make RooStats effort initiated by Kyle to (re)organize statistics tools in ROOT work on top of RooFit.

- On the 1st goal: Experience with RooFit in BaBar
  - RooFit is common language of data models
    - It is straightforward to understand other peoples models/fits
    - It is easy to share code. Not just model classes, but also macros building up complex analysis models (example: shared control samples)
  - Almost everybody in BaBar uses it
    - No formal policy behind this, people just concluded it was the easiest thing to do
    - Used in variety of statistical techniques: 'plain' unbinned ML fits, Likelihood ratios, confidence intervals..
  - It still leaves room for desire to develop 'own' fitting package.
    - Many analysis groups write higher-level package to construct models, steer analysis etc…
    - RooFit provides tools & utilities to automate p.d.f. building to a great extent
    - But engine is always RooFit, which takes care of the difficult stuff (integration, normalization, projections, optimization etc…)

Design study by WV, Kyle Cranmer, Amir Farbin, Frank Wrinklmeier

Plans for RooFit: Add project management infrastructure

- RooFit in BaBar (continued)
  - Analysis of BaBar user experience shows that many of the 'project management' wheels have been reinvented many times.
    - Aim to provide tools to standardize this
    - Don't want a single tool, but rather a flexible toolkit to do this
      → People like something of their own in the project

'Typical' analysis

1) Analysis model implemented in RooFit
   - Pretty much all standard RooFit classes

2) Proprietary steering package to act as driver of the analysis
   - Container of all involved RooFit objects (provide standard tool for this)
   - High-level steering code to produce plots, toy MC (mostly analysis-specific)
     - Most complicated stuff (projections, toy mc etc) delegated to RooFit core code.
   - Script-driven production of 'chunks' of RooFit models (provide standard tools for this)

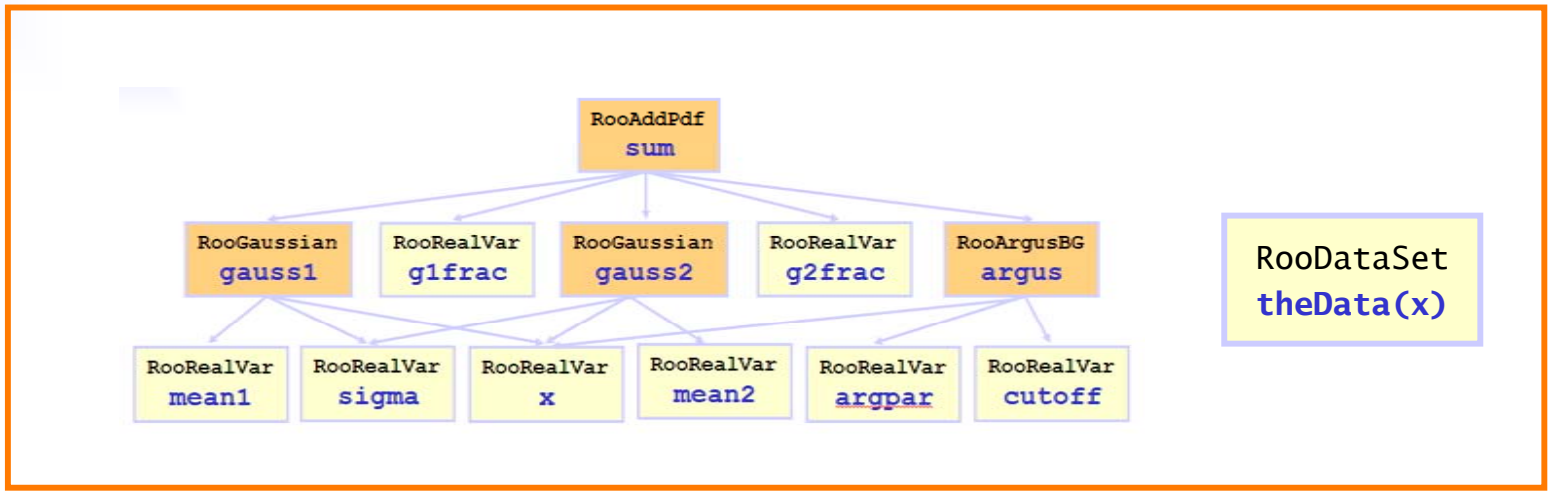
3) Statistics analysis of result
   - E.g. Frequentist confidence interval on result, perform profile likelihood analysis etc… (provide standard tools for this)

Plans for RooFit: Add project management infrastructure

- On the 2nd goal: RooStats on top of RooFit
  - Subject of next talk by Kyle Cranmer
  - Effort to organize set of standard statistical tools in ROOT for e.g. Bayesian analysis, Neyman construction, Profile Likelihood etc…
  - Tools are demanding on underlying model as many complex calculations like projections, integrals etc may need to be calculated
  - RooFit already implements nearly all the difficult tasks that are needed.
  - Provide missing pieces and 'glue' code to make things work.

- Many of the 'missing' functionality between these goals is common. Two new concepts introduced:
  - ***'Workspace management'*** – Storing and creating p.d.f.s
  - ***'Model interpretation management'*** – Formalizing specific interpretation of pdf's and write advanced statistical tools that can operate on these 'pdf+interpretation' objects

RooFit infrastructure plans – Workspace management



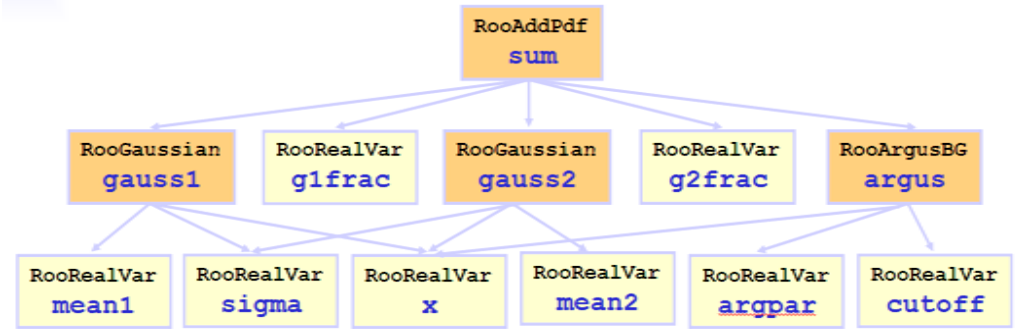
- Will provide a RooWorkspace class to hold all project components
  - Relieves users of burden of homegrown solutions
  - All p.d.f. and components, variables, datasets, plots can be stored in and owned by a RooWorkspace.
  - Can enforce compatibility inside workspace
    - e.g. datasets have same variables as models in workspace, to be checked or fixed upon insertion.
  - Vehicle for persistence of entire project.
    - **Ultimate portability of entire analysis projects** → A ROOT file can contain a completely functional analysis with code and data in a standard interface

RooFit infrastructure plans – Workspace *tools*

- Standard infrastructure to manage projects allows to build standard tools to manage projects
  - A `RooClassFactory` that facilitates any kind of script-driven filling of workspaces, e.g.

    ```
    factory.addPdf(wspace,"RooGaussian(x,mean2,sigma)")
    ```

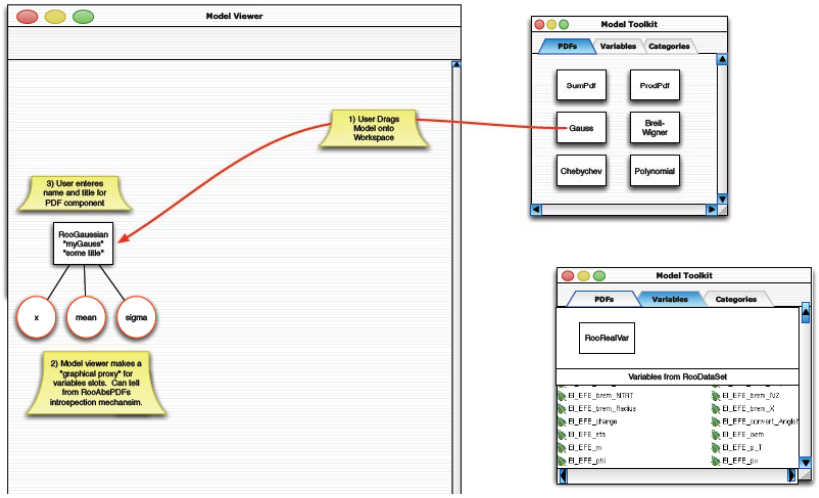
    where any objects existing in the workspace, such as `x`, `mean2` & `sigma` in this example can be referenced in the specification
  - Solves the 'hard' problem of automated p.d.f construction that many user implement in their driver code.



RooFit infrastructure plans – Workspace *tools*

- Standard infrastructure → standard tools
  - A RooWorkspaceGUI to visualize (in first instance) and to modify (in a next iteration) the contents of a workspace

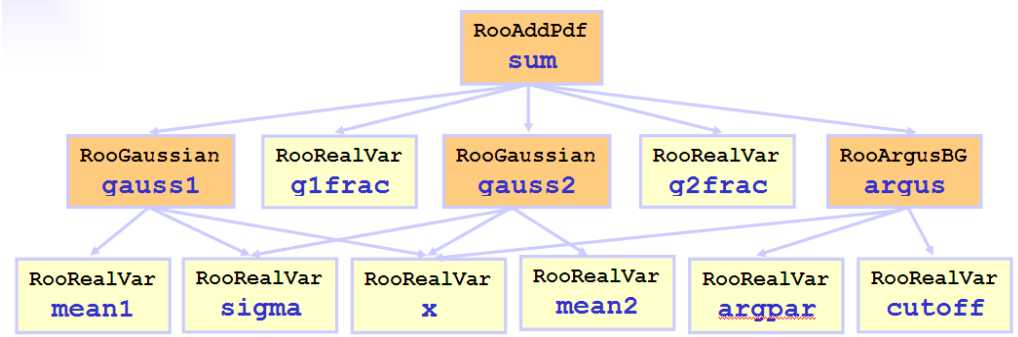
'Artists vision' by Kyle



  - RooSimPdfBuilder (existing tool).
    - A clone-and-modify tool to build p.d.f.s for simultaneous fits to many datasets with variations of one or more prototype p.d.f.s
  - RooScriptPdfBuilder (idea)
    - Similar to LMinuit concept in BaBar
    - Potential tool to populate entire workspace based off script.
    - Many physicist end up doing something like this. Benefits from standard approach but don't want to invent yet another language.

RooFit infrastructure plans – Model management

- Model management – A standard approach to specify interpretation of a p.d.f.
  - RooFit core classes have maximal flexibility: no need to say what is signal/background, observable/parameter: it will always work.
  - There is also no way to *fix* such conventions in RooFit
    - Complicate higher level analysis projects: user needs to do own bookkeeping of what is signal, background, parameter or observable in his interpretation
    - Private implementations are obstacle for common tools → need uniform language
  - Add RooModel concept to RooFit: a specific (frozen) interpretation or 'view' of a p.d.f.
    - A model is a p.d.f. with a fixed/chosen convention of what are parameters, observables, and what p.d.f components represent the various signal and background hypotheses.
    - A p.d.f. can have many associated models (views). The conventions are stored in the RooModel class, not in the p.d.f. class
    - Models (views) can be added to the workspace



RooFit infrastructure plans – Model-based tools

- Most high level analysis tools are model-oriented tools rather than p.d.f oriented tools
  - A `RooMCStudy` – Manager for goodness-of-fit study
    - Performs cycles of generating toy MC, fitting those and collecting the fit statistics.
    - Existing p.d.f-based tool, but works more naturally as model-based tool
  - A `RooSplot` variety – Splots are a special tool for visualizing data
    - If you don't know what it is I'm not going explain it here
    - A p.d.f. based version exists, but given its explicit signal/background hypothesis interpretation of signal an ideal candidate to be rewritten as model-based tool.
  - A `RooLLRatio` – Likelihood ratio analysis/plot
    - Select events, compute significances based on ratios of the signal and background likelihood
    - Common analysis technique in BaBar. Implementation can be standardized to great extent with model concept.

RooFit infrastructure plans – Model-based tools

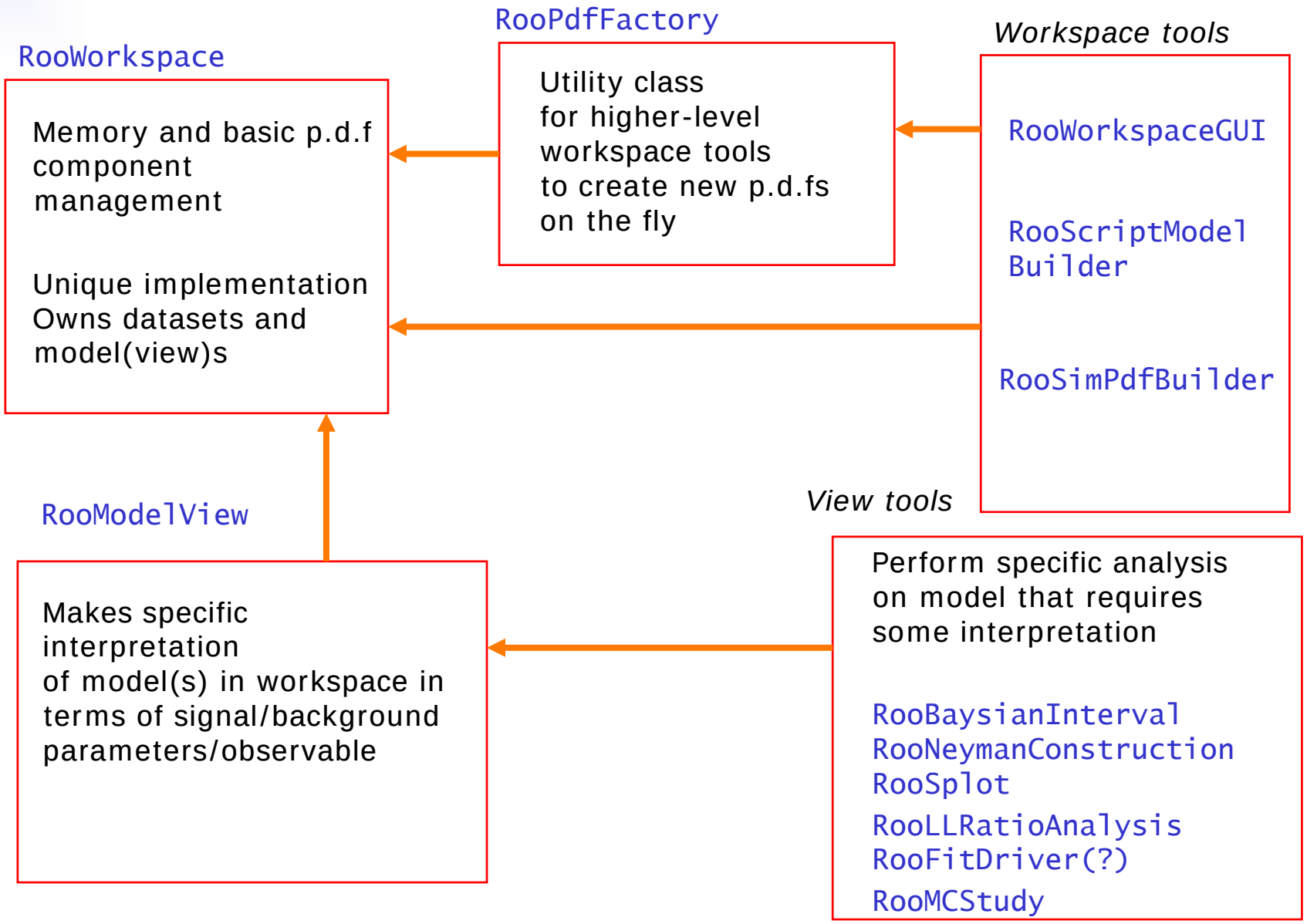
- All RooStats tools are also model-oriented tools
  - A `RooBayesianInterval`, `RooNeymanConstruction` `RooFeldmanCousins` – The RooStats suite of statistical tools
    - Subject of Kyle Cranmers presentation.
    - E.g. all of the non-Bayesian techniques need explicit determination of observables and parameters

- Many, if not all, of these tools would have a *very complex user interface* if the complete model interpretation would have to be specified to each of them.
  - Now one can design a tool with a very simple interface

```
RooFeldmanCousins fc(myModel) ;
RooAbsInterval* interval = fc.getInterval(myParam) ;
```

because definition of signal, background, observables and parameters is fully defined inside `myModel`

Overview of new management classes



Summary

- RooFit has been around for nearly 7 years now.
  - Has been and still used intensively BaBar
  - Quite a bit of use outside BaBar too (Belle, CMS, ATLAS, LHCb)
- Little development has happened in past 2 years
  - Mostly bug fixes
- Now planning a larger design effort to
  - Incorporate lessons from BaBar
  - Lay groundwork for RooStats statistical tools (see next talk)
  - Time scale for project ~3-6 months…