just an illustration of what can be done with ROOT/RooFit (not a lecture) for those who never tried it

we are just Belle members trying to use these packages...
ROOT package

Definition: An object oriented HEP analysis framework
Authors: developed by CERN people (René Brun, Fond Rademakers), Masaharu Goto for CINT
Documentation (far better than PAW):
  - Official web page (http://root.cern.ch/): manual, HTML pages to describe the classes...
  - ROOT educational resources at Fermilab (http://www-pat.fnal.gov/root/): slides, CDRoms, streaming video of the ROOT workshop...
Installation: tested with different OS (Linux (Redhat), Unix, Windows...)
ROOT is...

✦ Cint is not aimed to be a 100% ANSI/ISO compliant C/C++ language processor. It rather is a portable script language environment which is close enough to the standard C++.

✦ Command Line Typing: The interpreter knows all the classes, functions, variables and use defined types.
   Tab feature to help the user complete the command line (also command .class for the details of a given class)

✦ GUI (Graphics User Interface): windows, buttons, menus, Browsers, Panels, Tree Viewer...

✦ 3d interfaces with OpenGL and X3D

✦ Multi-windows

⇒ 2 examples:

✦ example from tutorials directory (see demos.C in the tutorials directory of root)

✦ a simple fit (how to improve the presentation of your plots in few clicks)
Start ROOT

✧ Convert a HBOOK/PAW histograms or ntuples files into ROOT files.
   Use the shell script command: `h2root <hbookfile> <rootfile>
(by the way, ROOT files significantly smaller than HBOOK files....)

✧ start a root session: `root` (also important quit: `.q`)

✧ run the examples under `$ROOTSYS/tutorials` (see demos.C for example)

✧ from PAW to ROOT, easy parallel:
  ◆ h/file 1 test.hbk is TFile *f = new TFile("test.root");
  ◆ h/list is f.ls();
  ◆ histogram are THist
  ◆ ntuple is called TNuple (which is a TTree limited to floating point numbers)
  ◆ ...

BAM - Jan 23 2003 Analysis with ROOT/Rooft packages Karim/Yang Heng
GUI (Graphical User Interface)

Right click, left click, middle click...
- Browsing and opening files
- Drawing histograms
- Draw Panel
- Fit Panel
- Adding Color and zooming
- Adding text and other objects
- Dividing the canvas
- Setting the log scale
- ....

Answer to the question: “why we have to rerun the entire kumac (plot, fit) to change the color or the size of the line ? move the title ?”

In ROOT, most objects derive from a base class TObject. This class has a virtual method Draw() so all objects are supposed to be able to be “drawn”. When an object is drawn, one can interact with it : move or transform it
**Start RooFit...**

- **Definition**: package for performing unbinned maximum likelihood fits and display the results.
  - extension of ROOT: another reason to switch to ROOT...
- **Authors**: David Kirkby and Wouter Verkerke (BaBar)
  - (← RooFitTools package)
- **Installation**:
  - from sourceforge.net (tested with Linux Redhat 7.2)
  - Code split in two packages:
    1. RooFitCore: everything but PDFs (Core code, base classes, interface to MINUIT...)
    2. RooFitModels: PDF implementations
- To load automatically the RooFit libraries when you start ROOT session, add the following commands in your roologin.cc
  (~ .pawlogon.kumac)
  ```c
  gSystem->Load("libRooFitCore.so") ;
  gSystem->Load("libRooFitModels.so") ;
  ```
- **Documentation**: http://rooFit.sourceforge.net/

BAM - Jan 23 2003  Analysis with ROOT/RooFit packages  Karim/Yang Heng
RooFit automatically takes care of the normalization (which generally depends on the parameter values and the limits on the dependent variables)

\[ \int_{\bar{x}_{\text{min}}}^{\bar{x}_{\text{max}}} g(\bar{x}, \vec{p}) d\bar{x} = 1 \]

- **Variables** (RooRealVar): parameter or observable?
  - parameter/observable interpretation is automatic and implicit when a PDF is used together with a dataset.
  - All variables that are member of the dataset are observables (limits = normalization range)
  - All other PDF variables are parameters (limits = MINUIT bounds)
Start RooFit...

- **Model**: build your PDF from
  - building blocks:
    1. RooArgusBG: Argus background shape
    2. RooBreitWigner: Breit-Wigner shape
    3. RooBifurGauss: Bifurcated Gaussian
    4. RooCBSShape: Crystal Ball function
    5. RooChebychev: Chebychev polynomial
    6. RooDecay: Simple decay function
    7. RooDstD0BG: D* background description
    8. RooExponential: Exponential function
    9. RooGaussian: Gaussian function
   10. etc....

- Complex PDFs are composed using operator classes: addition, multiplication, composition, convolution...

- but also generic PDF (RooGenericPdf) where you just write down the PDF expression as a C++ formula

- **Dataset**: binned (RootDataHist) or unbinned (RooDataSet) and the fit will be binned or unbinned according the nature of the dataset.
now you can unbinned-maximum-likelihood-fit...

- **Fit** in one command line:
  ```
  RooFitResult* r = model.fitTo(dataset,"r");
  ```
  the object RooFitResult holds complete snapshot of fit results (see `r->Print("v")`);

- **RooPlot** is an empty frame capable of holding anything plotted versus its variable

- **Other features**: Generating ToyMC, Extended Maximum-Likelihood, simultaneous fits on independent datasets...

⇒ 3 examples:

- **Simple example**: beam mass constraint fit (unbinned as simple as binned fit)

- **Mixing example from** $D^*\pi$ generated sample (Yang Heng)

- **Convoluted PDFs** (decay $\otimes$ gaussian(s))