INTRODUCTION TUTORIAL

Introduction to ROOT

Adrian Bevan

YETI January 2007

Uses ROOT 5.12.00

Queen Mary
University of London

OVERVIEW

- 3 tutorials over the next two days:
  - Introduction:
    - Introduction to ROOT.
  - Multi Variate Analysis:
    - Training Neural Networks
    - Tools to calculate fisher discriminants, train neutral networks and boosted decision trees.
  - Fitting:

    Fitting in ROOT (1): writing your own PDF to fit to.

    (2): Using TMinuit

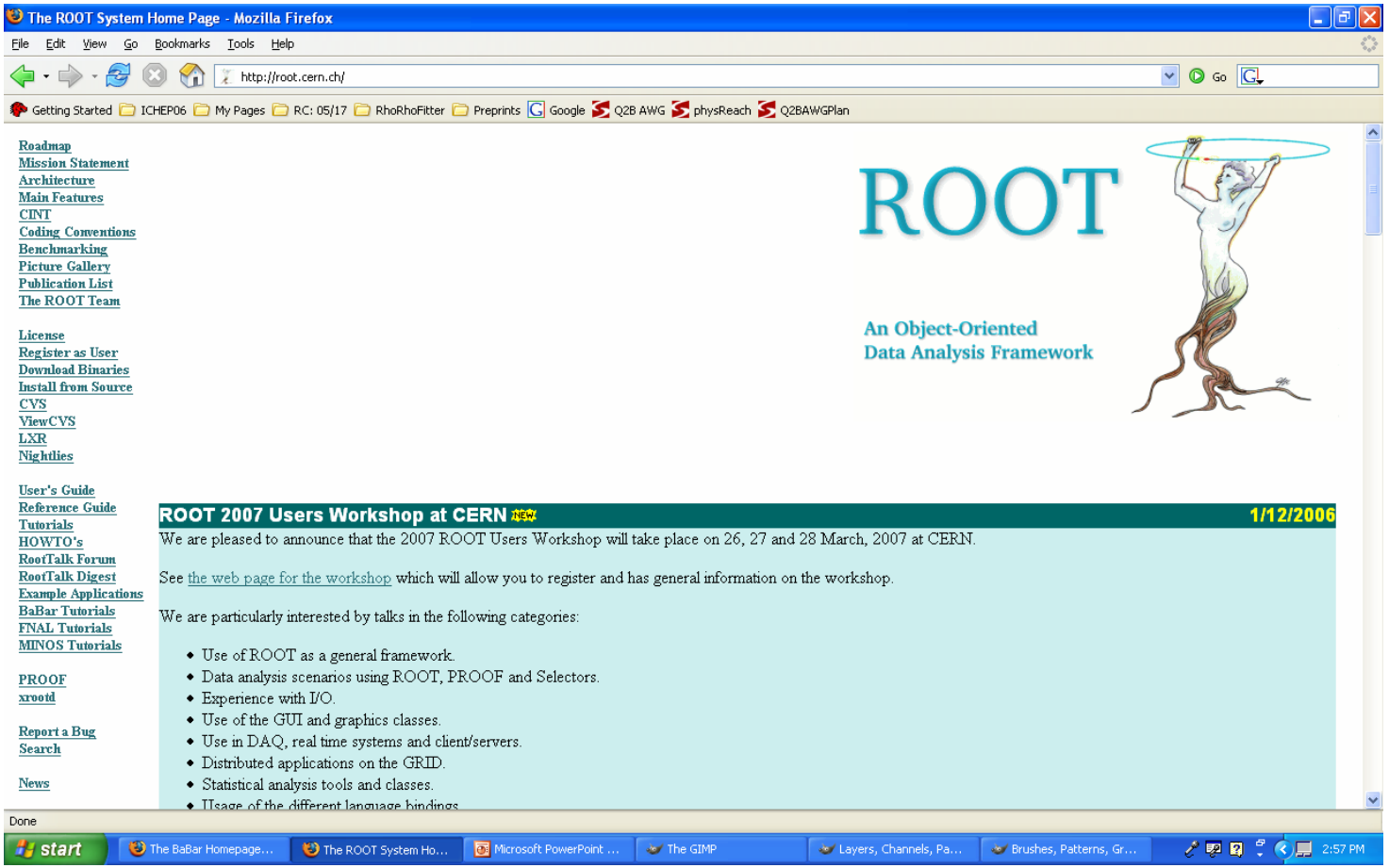
    (3): RooFit

Aims of this Tutorial

- Get started with ROOT.
- Gain familiarity with histograms, trees, and files.
- Learn how to use the Fit Panel in the ROOT GUI.

ROOT

Synopsis: ROOT is a versatile toolkit that can be used as a base for analysis, or integrated into an existing analysis framework.



More information is available at http://root.cern.ch

Getting started with ROOT

- Start a ROOT session:
  - Type the following (followed by return)

```
root
```

  - Available options:

`-l` suppress the splash screen
`-b` run in batch mode (no graphics)
`-q macro.cc` quit root when finished running the specified macro

- Finish a ROOT session
  - Type the following (followed by return)

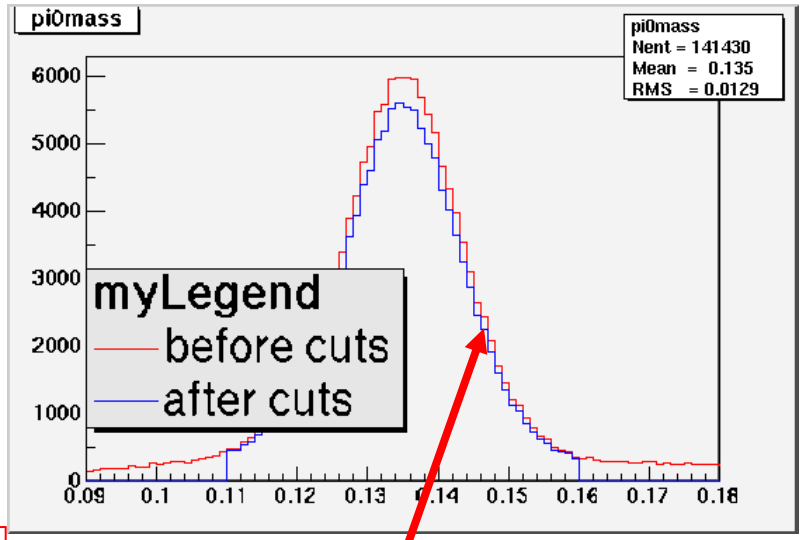
```
.q
```

This tutorial assumes that you already have a version of ROOT installed and that the appropriate environment variables have been set up.

Histograms

- A histogram is a binned representation of some data as a function of a variable.
- ROOTs data types for a histogram are
  - TH1F (floating point histogram)
  - TH1D (double precision histogram) etc.
  - If you have a histogram and want to see what it looks like you Draw() it:

```
root[10] myHist.Draw()
```



the ROOT prompt

The histogram object with variable name myHist

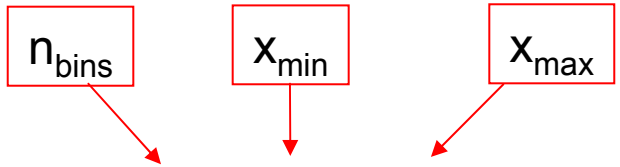
Draw() member function is called to show the Histogram on your screen

Two histograms shown with a TLegend describing the colour code used.

Histograms

Declare with:

```
TH1F h1(arguments …)
```



- Make your first 1D histogram:

```
TH1F  h_name("h_name", "h_title", 10, 0.0, 10.0);
```

h_name = key name of histo

h_title = name which appears on plotted histogram

- Now draw the (currently empty) histo:

```
h1->Draw();
```

- Fill with a few entries:

```
h1->Fill(1.);
h1->Fill(3,10.7);
```

the number to fill the histogram with *(default value is 1.0)*

x value to fill histogram at

- Try drawing the histogram when you have a few entries

```
h1->Draw();
```

Histograms

Some useful commands try out now that you've made a histogram. Try and change some of the histogram characteristics using the following:

Note the US spelling

- Line colour `h1->SetLineColor(kRed);`
- Title `h1->SetTitle("My title");`
- X axis title `h1->SetXTitle("The x axis");`
- Change x-axis range `h1->SetAxisRange(4., 15);` //zoom
- Line colours `h1->SetMarkerColor(kBlue);`
- Point size `h1->SetMarkerSize(1.);`
- Point style `h1->SetMarkerStyle(20);`
- Fill colour: (def: white) `h1->SetFillColor(kGreen);`
- Draw a filled histogram `h1->SetFillStyle(3004);` // diagonal lines
- Histogram with error bars `h1->Draw("e");` //$\sigma = \sqrt{N_{entries}}$
- Print information to screen `h1->Print()`
- Usually need to redraw histo after any changes have been made in order to update the TCanvas that shows the histogram

  `h1->Draw();`
- Draw a second histogram, `h2`, on the same canvas as another one:

  `h2->Draw("same");`

See Chapters 3 & 9 of the ROOT 5.12 User Guide for more details

Exercises: Histograms

- Make a histogram
  1. Fill the histogram with some entries and draw the histogram.
  2. Re-draw the histogram as you accumulate entries & superimpose the new result on the old histogram content.
  3. Try to modify the characteristics of the histogram (e.g. the line style and colour etc.).

Files

- Used to save trees, histograms and other ROOT objects so that you can come back and use them later.
  - Open an existing file (read only)

```
TFile myfile("myfile.root");
```

  - Open a file to replace it

```
TFile myfile("myfile.root", "RECREATE");
```

  - or append to it

```
TFile myfile("myfile.root", "UPDATE");
```

Files

- Used to save trees, histograms and other ROOT objects so that you can come back and use them later.
  - Open an existing file (read only)

```
TFile myfile("myfile.root");
```

  - Open a file to replace it

```
TFile myfile("myfile.root", "RECREATE");
```

  - or append to it

```
TFile myfile("myfile.root", "UPDATE");
```

  - To inspect (see what's in) a file just ls() it

```
myfile.ls();
```

Files

- Used to save trees, histograms and other ROOT objects so that you can come back and use them later.
  - Open an existing file (read only)

```
TFile myfile("myfile.root");
```

  - Open a file to replace it

```
TFile myfile("myfile.root", "RECREATE");
```

  - or append to it

```
TFile myfile("myfile.root", "UPDATE");
```

  - To inspect (see what's in) a file just ls() it

```
myfile.ls();
```

  - And to get something from a file, just Get() it

```
TH1F * myhist = (TH1F*)myfile.Get("myhist");
```

Files

- For example, look at the file in data/signal.root

```
root[0] TFile signal("data/signal.root")
root [1] signal.ls()
TFile**         signal.root
 TFile*         signal.root
  KEY: TH1D     cosphericity;1 cosphericity
  KEY: TH1D     photonlat;1    photonlat
  KEY: TH1D     pi0mass;1      pi0mass
   .
   .
   .
 KEY: TTree     selectedtree;1  Final variables tree
```

Open the file signal.root
(This MC is for $B^0 \rightarrow \pi^0\pi^0$ decays reconstructed within BaBar)

The key type is the root object type

Object type

Object name (use this to retrieve the object from the file)

Comment

Files

- For example, look at the file in data/signal.root

```
root[0] TFile signal("data/signal.root")
root [1] signal.ls()
TFile**         signal.root
 TFile*         signal.root
  KEY: TH1D     cosphericity;1 cosphericity
  KEY: TH1D     photonlat;1    photonlat
  KEY: TH1D     pi0mass;1      pi0mass
   .
   .
   .
 KEY: TTree    selectedtree;1  Final variables tree

root [4] TH1D * mpi0 = (TH1D*)signal.Get("pi0mass");
```

The key type is the root object type

Open the file signal.root
(This is $B^0 \rightarrow \pi^0\pi^0$ MC)

"Get" a histogram in memory

Files

- For example, look at the file in data/signal.root

```
root[0] TFile signal("data/signal.root")
root [1] signal.ls()
TFile**         signal.root
 TFile*         signal.root
  KEY: TH1D     cosphericity;1 cosphericity
  KEY: TH1D     photonlat;1    photonlat
  KEY: TH1D     pi0mass;1      pi0mass
   .
   .
   .
 KEY: TTree    selectedtree;1  Final variables tree

root [4] TH1D * mpi0 = (TH1D*)signal.Get("pi0mass");

root[5] mpi0->Draw();
```

The key type is the root object type

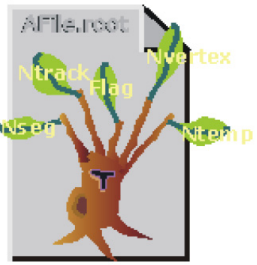
Open the file signal.root
(This is $B^0 \rightarrow \pi^0\pi^0$ MC)

"Get" a histogram in memory

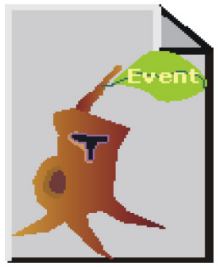
Try drawing the histogram

Trees

- A ROOT data structure filled on an entry by entry basis (e.g. candidate by candidate, event by event or using a more complicated split form [tracks, clusters etc.]).

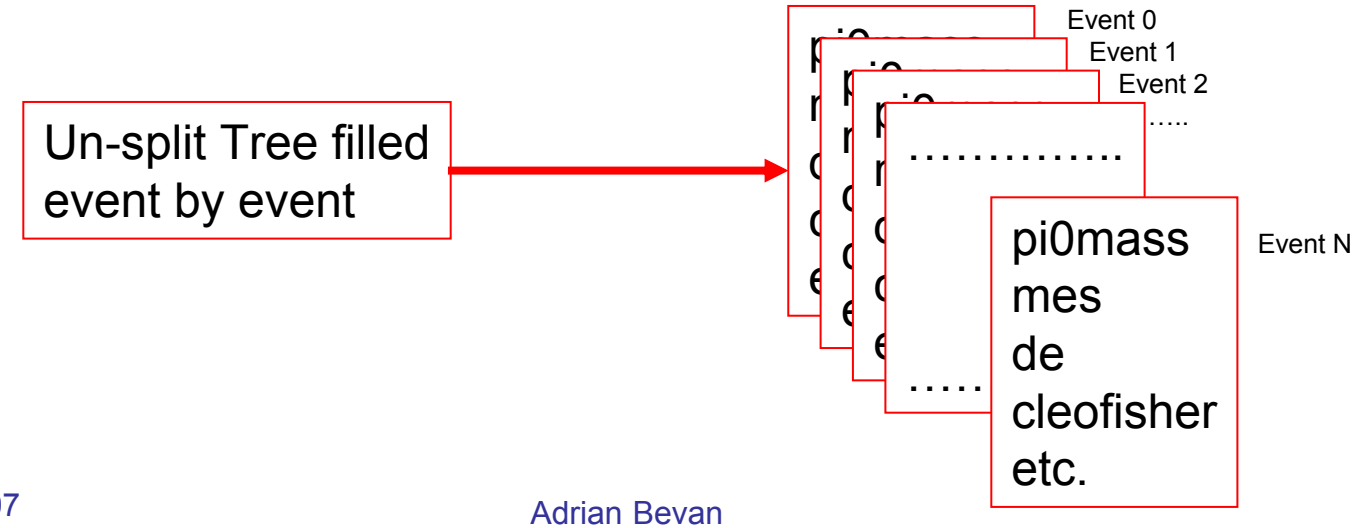


A tree that is split



A tree that is not split

- e.g.



Trees

- Get the tree from the file

```
root [6] TTree * tree = (TTree*)signal.Get("selectedtree");
```

- But what variables does it contain?
  - You can Print() the tree to show its structure

```
root [7] tree.Print()
```

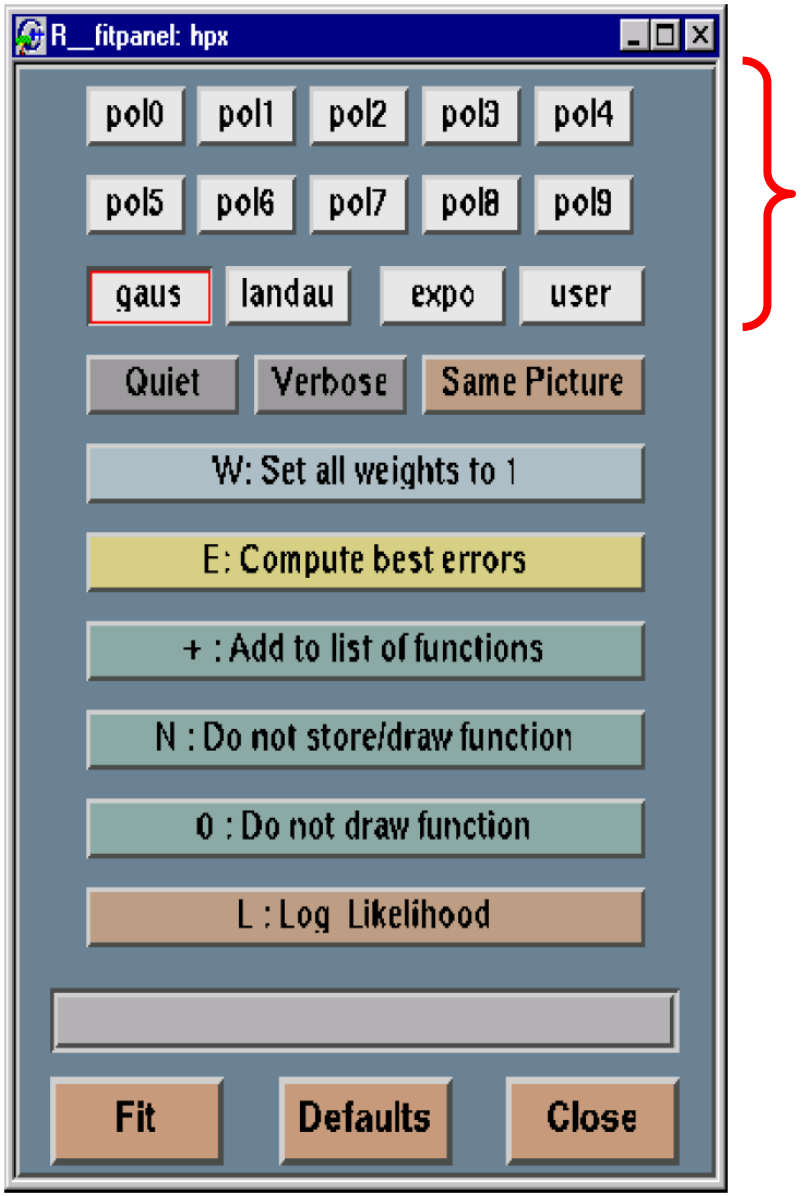
- Knowing the variable you are interested in, you can Draw() it

```
root [8] tree.Draw("mes")
```

Exercises: Files and Trees

- Open the file `~/bevan/yeti07/data/signal.root` and get the histogram `pi0mass` and tree `selectedtree` from the file.
  1. Try drawing the histogram.
  2. Print the tree structure, and try drawing
     - `mes` (the B mass)
     - `cleofish` (Fisher discriminant based on the energy flow of particles around the thrust axis of the other B-meson in the event.)
     - de (an energy difference)
  - You can try and change the characteristics of the histograms drawn from a tree (HINT use the same syntax that you would for a histogram).

The Fit Panel



- Functions specified:
  - pol0 = polynomial of order 0
  - pol1 = polynomial of order 1
  - …
  - pol9 = polynomial of order 9

$$Pn = \sum_{i=1}^{n} a_i x^i + C$$

  - gaus = Gaussian

$$G(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

  - landau = Landau function

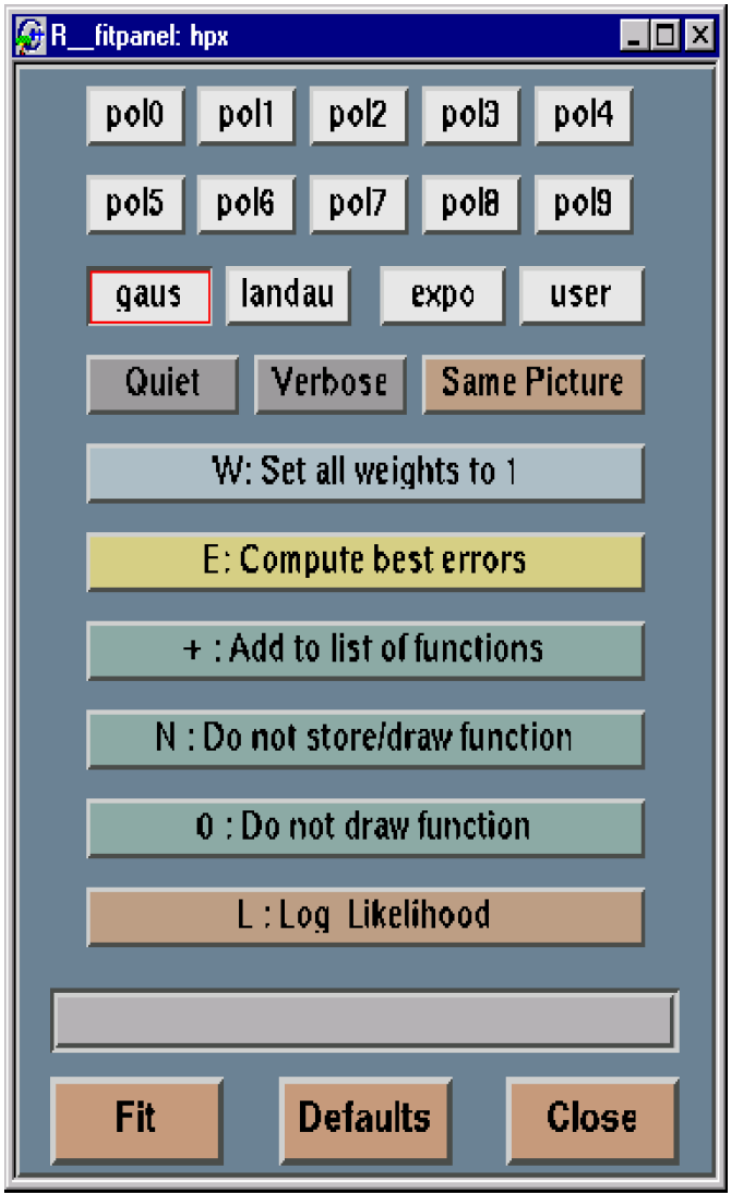
More complicated functional form
e.g. See Cowan p37 for dfn..

  - expo = exponential

$$E(x,\mu) = Ae^{\mu x}$$

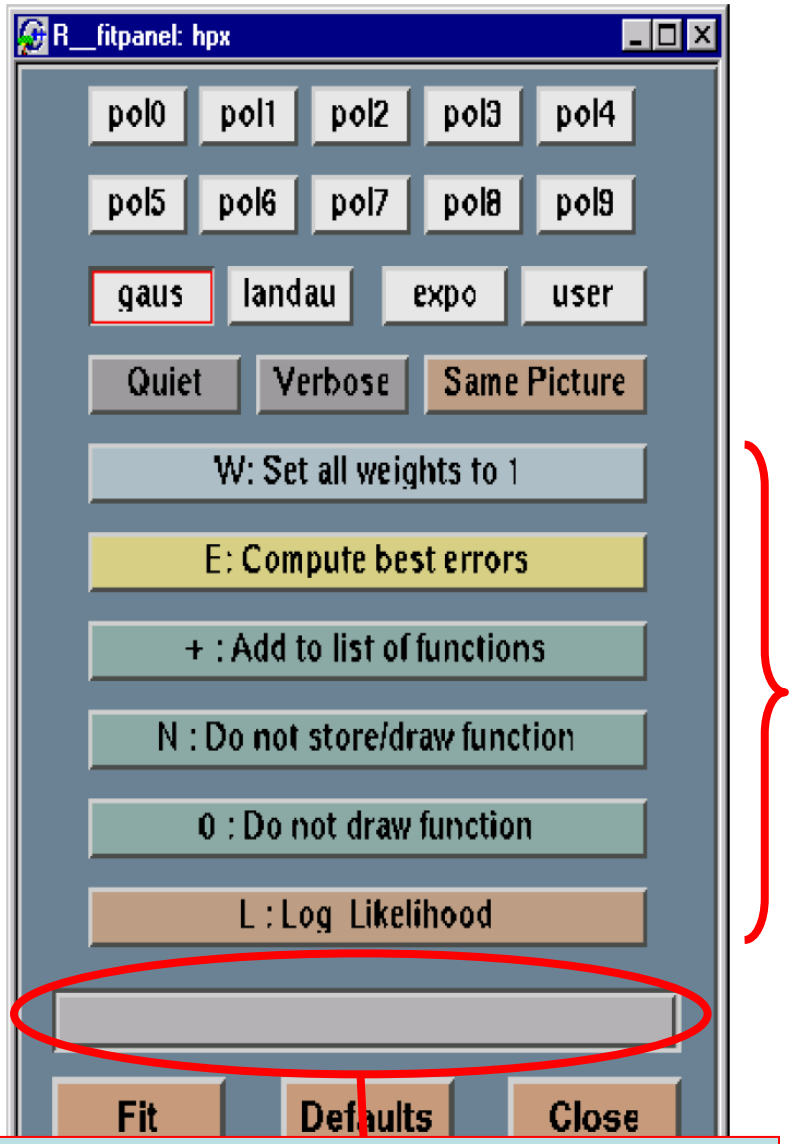
  - user = user defined

The Fit Panel



- Specify output options
  - Quiet: Switch off output of the fit to the command line.
  - Verbose: display all available information to the command line.
  - Same Picture: Overlay the curve on the histogram. The default behaviour is to replace the histogram with the shape.

The Fit Panel



Can vary fit range using the slide bar

- Fit Options:
  - Set all weights to 1: Set all errors to one (so that all entries have equal weight in the fit)
  - Compute best errors: Use MINOS to compute the best errors.
  - Add to list of functions: add this function to the list of functions fitted to the histogram.
  - Do not store/draw function:don't add the function to the list or draw it.
  - Log Likelihood: Use a log likelihood fit to the histogram. The default behaviour is to use a χ^2 fit.

Exercises: Fitting to a histogram

- Open the file `~/bevan/yeti07/data/signal.root`, and retrieve the histogram `pi0mass` from the file.
  - Try to fit various shapes (high order polynomial, Gaussian and Landau) to the histogram and look at the goodness of fit 'by eye'.
  - What is the χ^2 per degree of freedom for these fits?
    - How many degrees of freedom do the functions have?
    - What is a good fit for a given χ^2 per degree of freedom?

Exercises: Fitting to a histogram

- Open the file `~/bevan/yeti07/data/signal.root`, and retrieve the histogram `pi0mass` from the file.
  - Try to fit various shapes (high order polynomial, Gaussian and Landau) to the histogram and look at the goodness of fit 'by eye'.
  - What is the χ^2 per degree of freedom for these fits?
    - How many degrees of freedom do the functions have?
    - What is a good fit for a given χ^2 per degree of freedom?
  - What happens to the fit χ^2 values when you restrict the fit range to the central part of the histogram?

Exercises: Fitting to a histogram

- Open the file `~/bevan/yeti07/data/signal.root`, and retrieve the histogram `pi0mass` from the file.
  - Try to fit various shapes (high order polynomial, Gaussian and Landau) to the histogram and look at the goodness of fit 'by eye'.
  - What is the χ^2 per degree of freedom for these fits?
    - How many degrees of freedom do the functions have?
    - What is a good fit for a given χ^2 per degree of freedom?
  - What happens to the fit χ^2 values when you restrict the fit range to the central part of the histogram?
  - What happens to the fit results if you try and use the options:
    - Compute best errors?
    - Log Likelihood?

Summary

- You have seen how to use the basic features of
  - Histograms (THF1 etc.)
  - Trees (TTree)
  - Files (TFile)

  in ROOT.

- You can use the ROOT fit panel to fit histograms and the content of TTrees. Glen's lectures tomorrow morning will talk about the details of fitting, and we will concentrate on this topic for the third tutorial session.

- More information on the use of ROOT is available on the web:

http://root.cern.ch
http://www-root.fnal.gov/
http://www.ph.qmul.ac.uk/~bevan/GCL/computing.html
etc.