BCGS Intensive Week
“From Hits to Higgs”

ROOT mini tutorial

- Starting, exiting root
- Drawing a function
- Using macros
- Creating, filling, drawing a histogram
- Using the TBrowser
- ROOT resources
- Fitting a histogram
- Creating publication-style LHC plots

Eckhard von Törne

universitätbonn

ROOT (http://root.cern.ch)

Created for Data Analysis of LHC experiments.

Public code for many platforms (any linux/ MAC).

ROOT is:

- interactive program with a C++ interpreter to encourage use of interactive coding and macros.
- library of C++ classes for particle physics.
- Here we use it mainly for working with histograms

Starting, exiting ROOT

- The tutorial material is found in the project subfolder **`ROOTtutorial`**
- Open a terminal and cd to this directory
- From the command line type **`root`**
- Now you can use the command line to pass commands to ROOT (= ROOT command line)
- quick start: **`root –l`**
- exiting root **`.q`** from the root command line
- Starting root with a file: **`root –l filename.root`**

Drawing a function

- From the ROOT command line type
- **`TF1 func("func","sin(x)/x",0.,15.);`**
- **`func.Draw();`**

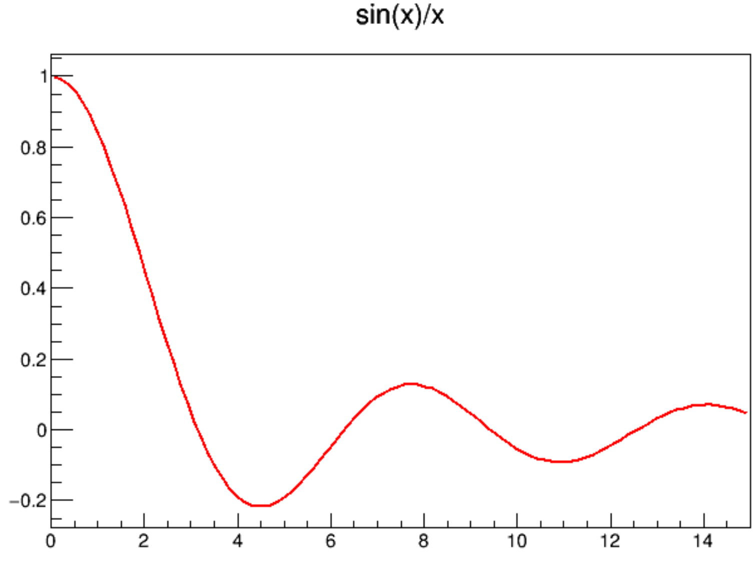
- ROOT nomenclature:
  - ROOT Classes all start with a capital “T“ (TF1, TH1D, TVector3, TLorentzVector,…).
  - Example: TF1 is a 1-dimensional function
  - Functions are all capitalized, example , Draw()

- With the TF1 command a 1-dimensional function object is created and drawn with the second command.

Using macros

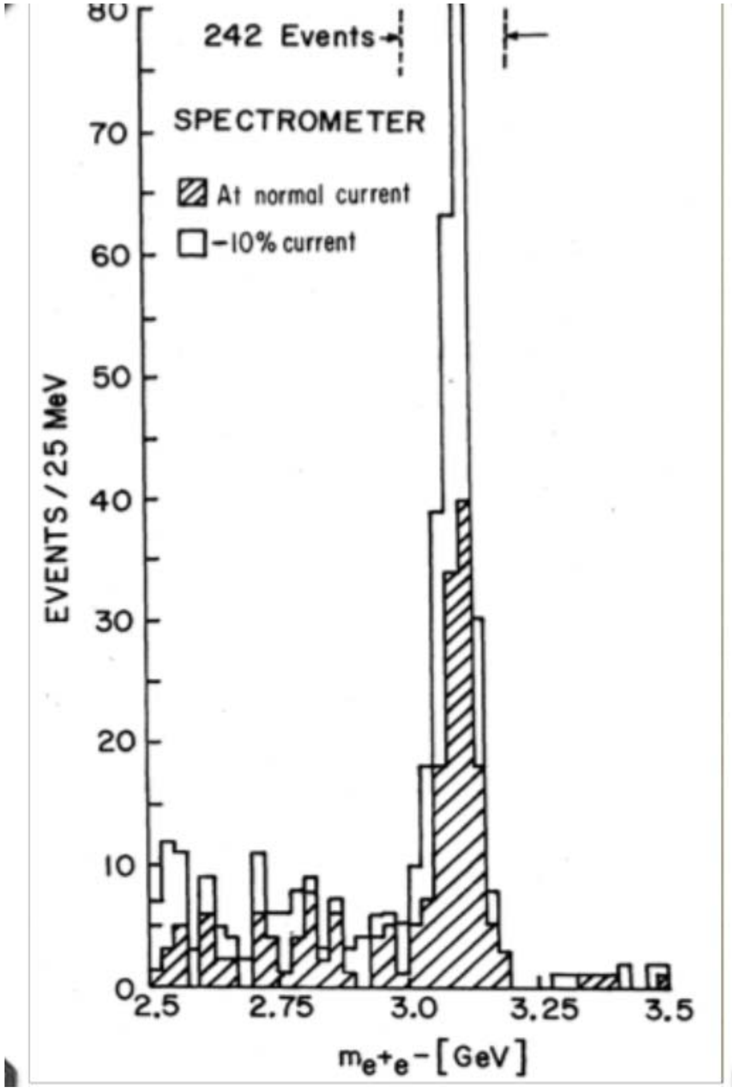
- From the root command line type:
- **`.x func.C`**
- Content of file **`func.C`**

```
// example for a root macro. The standard extension of root macros is .C
//
// root macros are implemented as functions
// the name of the function must match the filename
//
// macros are called from the root command line:  .x func.C
void func()
{
   TF1* func= new TF1("func","sin(x)/x",0.,15.);
   func->Draw();
   c1->SaveAs("func.jpg"); // c1 is the name of
}
```

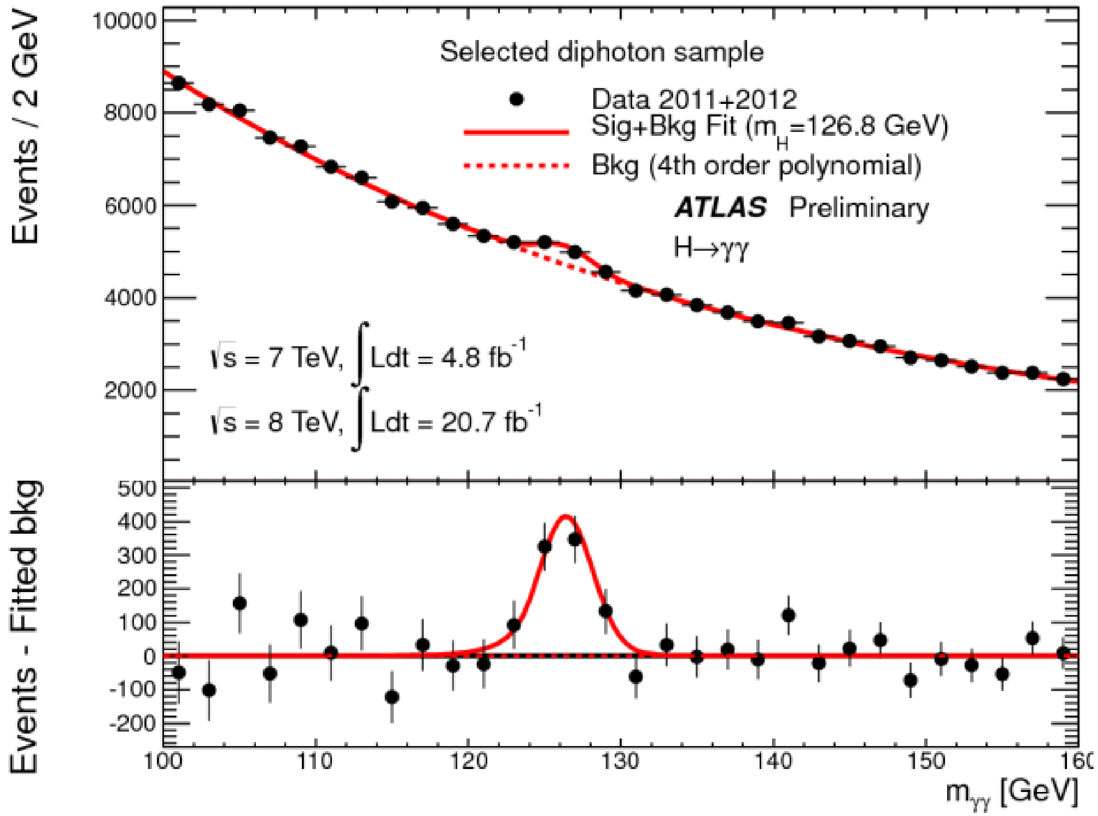


Histograms

hand-drawn histogram



Histogram created with root



Creating, filling, drawing a histogram

Execute macro histo1.C: .x histo1.C+g

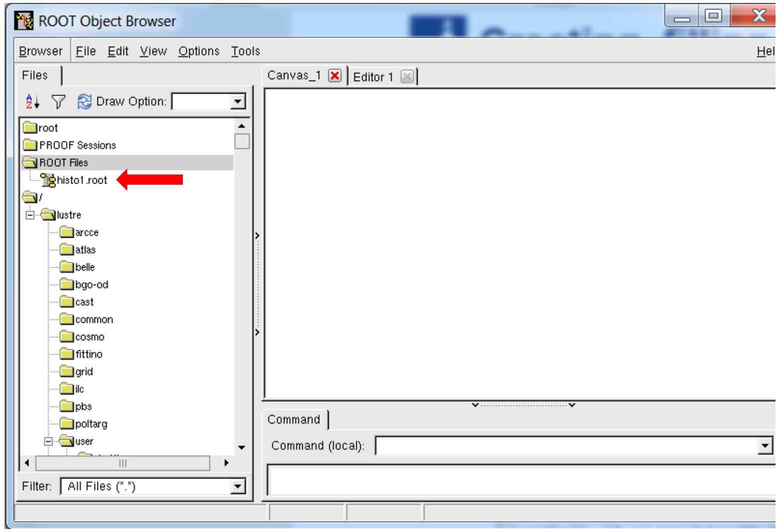
For a more robust macro execution add a „+g“ to the filename (triggers creation of a shared library)

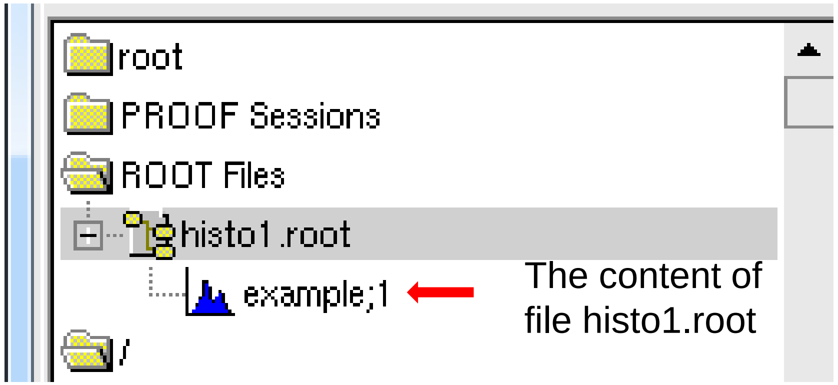
```
void histo1()
{
   cout << "simple example on how to use histograms"<<endl;

   TFile* outfile = new TFile("histo1.root","RECREATE");
   h = new TH1D("example", "example",50,0.,1.);

   for (int i=0;i<1000;i++){
      double x=0.001*i;
      double y = x*x;
      h->Fill(y);
   }

   h->Write();

   h->Draw();
}
```

This approach can be quite clumsy if you create many histograms.
During the intensive week we will use a short-cut approach to handling histograms

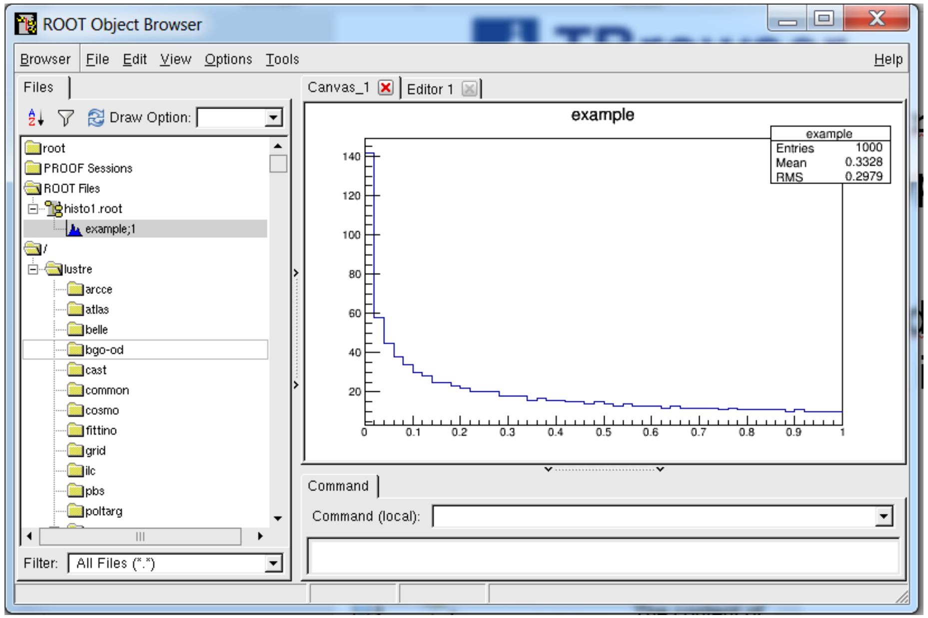
TBrowser

- Start root to look at a file: **`root –l histo1.root`**
- type: **`new TBrowser`**
- You get a new window
- The file histo1.root is listed near the top. Double-click it.





The content of file histo1.root



Now double click the histogram icon
The histogram is displayed inside the browser

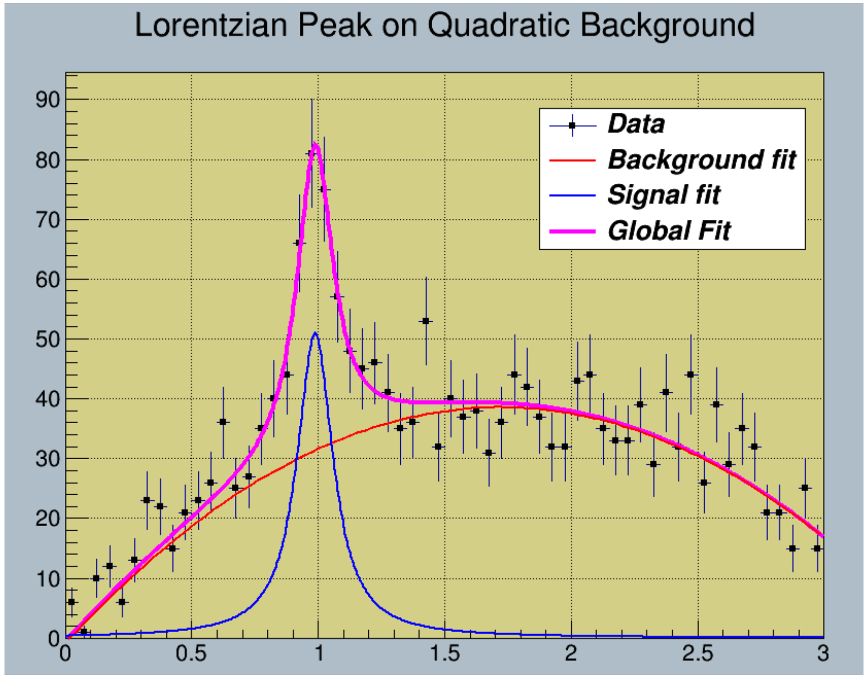
ROOT resources

- Most resources are online.
- See the root main page, specifically the tutorials and the howTOs:
- https://root.cern.ch/doc/master/group__Tutorials.html
- https://root.cern.ch/howtos
- Most tutorials are also available as part of the ROOT distribution in $ROOTSYS/tutorials
- Every class in ROOT has a web-site. Try a web-search on CERN ROOT TF1

Fitting a histogram

- From the root website copy macro FittingDemo.C to the tutorial folder
- Alternatively from the command line:
- `cp $ROOTSYS/tutorials/fit/FittingDemo.C .`
- Execute the macro.

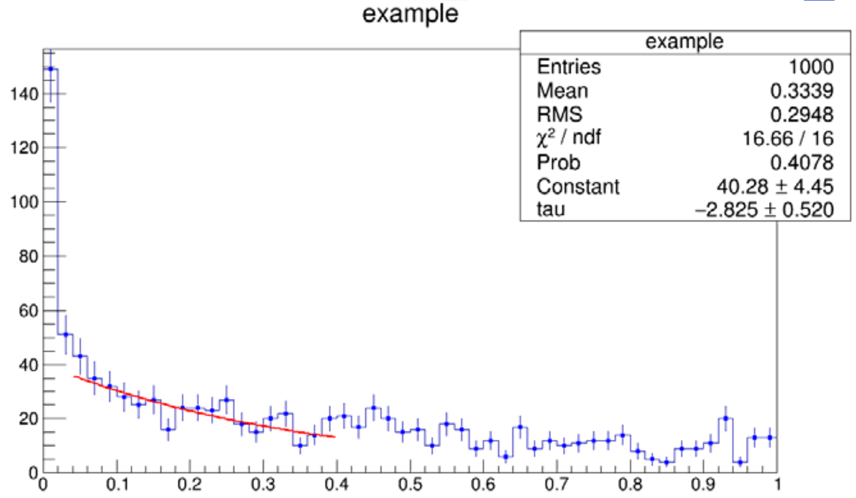
  The macro performs a χ^2 fit of a parameterized function to the histo. The function has a signal peak + polynomial background.



Fitting a histogram II

- .x fitExample.C+g   This fits a user defined fucntion to a histo
- Obviously the expoential does not describe the complete range

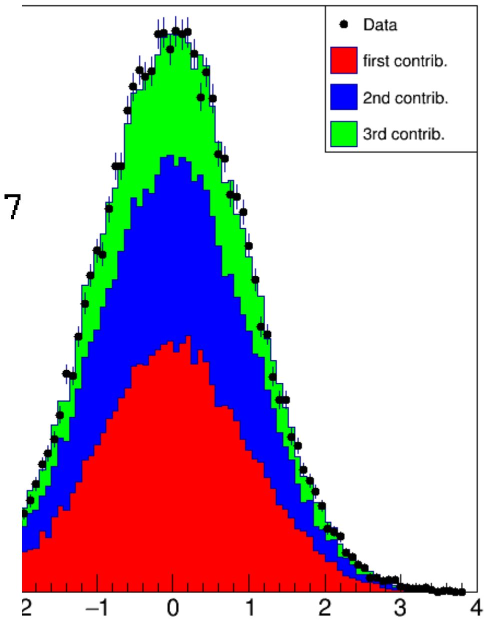
```
double fitf(Double_t *x, Double_t *par)
{
   double fitval = par[0]*TMath::Exp(par[1]*x[0]);
   return fitval;
}

void fitExample()
{
   gStyle->SetOptFit(11111);
   double xmin=0.04, xmax=0.4; // fit range
   int npara = 2;  // number of parameters to 1
   TString hname="example";

   // obtaining the histogram
   TH1D* hist = (TH1D*) gROOT->FindObject(hnam
   if (!hist){
      cout << "opening file histo1.root" << en
      TFile *file = TFile::Open("histo1.root").
      hist = (TH1D*) file->Get(hname);
   }

   TCanvas *c1 = new TCanvas("c1","the fit canvas",500,400);
// Creates a Root function based on function fitf above
   TF1 *func = new TF1("fitf",fitf,xmin,xmax,npara);
// Sets initial values and parameter names
   func->SetParameters(100.,-1.); // initial parameters before mini
mization
   func->SetParNames("Constant","tau");
// Fit histogram in range defined by function
   hist->Fit(func,"r,e"); // r: fit in function range, e: use MINOS
 for error calc.
   hist->Draw("e,same");
```



Stacked plots with THStack

- Execute the macro hstack.C+g

```
TCanvas *hstack() {
// Example of stacked histograms: class THStack
// based on Rene Bruns hstack.C

   THStack *hs = new THStack("hs","Stacked 1D histograms");

   //create 1-d histograms
   TH1D *h1st = new TH1D("h1st","test hstack",100,-4,4);
   h1st->FillRandom("gaus",20000);
   hs->Add(h1st);
   TH1D *h2st = new TH1D("h2st","test hstack",100,-4,4);
   h2st->FillRandom("gaus",15000);
   hs->Add(h2st);
   TH1D *hdata = new TH1D("hdata","test hstack",100,-4,4);
   hdata->FillRandom("gaus",10000+15000+20000);

   TCanvas *cst = new TCanvas("cst","stacked hists",10,10,7
   hs->Draw();
   hdata->Draw("e,same");
   TLegend* legend = new TLegend(0.7,0.8,0.9,0.9);
   legend->AddEntry(hdata,"Data","P");
   legend->AddEntry(h1st,"first contrib.","f");
   legend->AddEntry(h2st,"2nd contrib.","f");
   legend->Draw();
   return cst;
}
```



BACKUP