ROOT Tutorial
Hands-on Session

Andrea Di Simone
Uni Freiburg

Aim of this session:

You will (hopefully) become familiar with the CINT interpreter and the most common ROOT objects. You are highly encouraged to experiment both with the command line and the GUI.

1. The CINT interpreter

Start the CINT by typing root at the system prompt. Now you are at the CINT prompt. Try issuing some c/C++ commands

```
int a
cout<<a
cin>>a
a
```

As you can see, CINT executes your commands after you type them. You can also do more complex things, like loops

```
for(int i=0;i<10;i++){
    cout<<i<<endl;
}
```

As you may know, in c/C++ you MUST type a “;” at the end of each line. CINT allows you to omit it, with the exception of lines fed to a loop. Try experimenting, with loops, both with and without the trailing “;” on the lines into them.

2. Our first histograms

Create a 1D histogram and fill it with random numbers. Then Draw it

```
TH1F myHisto(“myHisto”,”test histo”,100,0,100)
for(int k=0;k<1000;k++){
    myPR.Fill(gRandom.Uniform(0,100));
}
myHisto.Draw()
```

gRandom is very useful when you need random variables. There are several distributions available. Try to type gRandom and the hit the “tab” key. This is one of the features of CINT: it will present you a list of possible choices when hitting the “tab” key. In this case you see all the methods of gRandom. Choose one of them, open the parenthesis and try hitting tab again. In this case, you have an explanation of the signature of the method you chose.

Now go back to the histogram you just plotted, and start playing with it. Try for example to change the axis title (you may want to get the X axis the set its title) or to change the fill color.

Now that you are acquainted with the command line controls, let's see what we can do with the GUI. Go with the mouse on top of any bin. The mouse pointer will change shape. Now right-click and see the list of possible options. Experiment.

Let's try some algebraic operations with histograms. Create two TH1F and set all the contents of the bins to 1. You can force a given bin to have a given value by using SetBinContent:

```
for(int i=0;i<histo->GetNbinsX();i++){

    histo.SetBinContent(i,1);

}
```

Now add them and check that the sum is ok:

```
TH1F histoTot=histo1+histo2
```

Something else: in the menu bar of the graphic window, click on “View” and then “Editor”. Experiment with what you see...

Now it's time to play with 2D histos. Create a 2D histogram and Fill it. Follow the previous example, just remember that you need more parameters both in the constructor and in the fill method. Also, experiment with different gRandom distributions.

Before moving to the next session, go with the mouse on one of the axes of any of your histograms. The pointer will become a small hand. Now click and drag, then release and see what happens. Now try to right-click on the axis, and experiment with the available options. At the end, be sure you unzoom the histo to its original range. As a last exercise, try to zoom again without the help of the GUI, just by using the command line. You will need to get the axis you want to zoom and set its axis.

3. Profiles

Along the same lines of what you have done with histograms, create two profiles, and fill them with constant values. You can do this either by forcing with SetBinContent or by filling enough times with

```
myProfile.Fill(gRandom.Uniform(xmin,xmax),1)
```

Now add the two profiles. What do you see? What's the difference wrt the histograms?

4. Graphs

Graphs are slightly different from histograms and profiles both in the way you create them and fill them. Take your time, create some graphs, set their points to the values you choose, draw them. Experiment with different drawing options (“apl”,”ap”,”al”). Try plotting two graphs on the same canvas using for example “same apl” on the second one. If you see something weird, try removing “a” from the second drawing's options. Do you understand why this happens?

5. Functions

Define a function plotting the sin with different frequencies. Something like sin(ϖt), plus a free normalization:

TF1 myF(“myF”,”[0]*sin([1]*x)”,0,10)
myF.SetParameter(0,1)
myF.SetParameter(1,10)

Experiment with different values of the frequencies, and try drawing on the same plot two versions of your function with different frequencies. Use “same” as drawing option. Why does this not work? Try using DrawClone(“same”) instead of Draw(“same”). Do you understand why this happens?

Now create a 1D histo with proper range and nbins and force its bin contents to be the ones foreseen by your function. You may need a loop on the bins of the histogram where you call

```
myF.Eval(histo.GetBinCenter(i))
```

Now let's see if everything matches. Fit the resulting histogram with the original function

```
histo.Fit(myF,”VL”,””,0,10)
```

The first argument is the function to be used by the fit, the second are fitting options (ignore them for the time being), the third are optional graphic options, the last two are the range of the fit process. Compare the fitted parameters with the ones you used for the generation.

If you have enough time, try fitting with one of the predefined functions. In this case you will call them by name (“gaus”, “exp”,”pol0”) in the first argument of the Fit method. The easiest way is to fit with a gaussian one part of your sinusoidal distribution: you only need to choose a proper range for the fit.

6. Ntuples

Now, use the file

```
ROOTTutorial.dat
```

Look into it. There should be four columns. Create an Ntuple to describe its contents and fill it by reading the file:

```
TNtuple nt(“nt”,”nt”,”x:y:z”)
nt.ReadFile(“ROOTTutorial.dat”)
```

Now experiment with it. Try the Print(), Scan(), GetEntries(). Open the first parenthesis then hit “tab” for a list of needed parameters.

When you are familiar with the structure, start inspecting it. Draw the distributions of each variable. Then combinations of them, then use cuts/conditions.

You may want to use custom binning for the histograms (both 1D and 2D). Just do something like:

```
nt.Draw("x1>h1(10,0,100)")
```

Try to guess what the distributions must be and test your guess with the Fit method.

7. Save everything

Most likely our time now is over. Create a file and append to it all you want to save. Including the Ntuple.

```
TFile f("myFile.root","RECREATE")
f.Append(&nt)
f.Write()
```

You can still do things on your histograms/graphs/ntuple. When you are done, write again and close the file

```
f.Write()
f.Close()
```

You will see that the objects written to file disappeared from memory. Now quit ROOT (type ".q") and start again. Open the file and retrieve one object:

```
TFile f1("filename.root","READ")
f1.ls()
Tntuple * nt=(TNtuple*)f1.Get("nt")
nt.Draw("x1")
```

Quit and restart again. This time launch this command

```
TBrowser b
```

In the resulting window, click on the file you created and see what it's inside. Double-clicking on some of the objects may be a good idea.