Welcome package - week 3

ROOT Tutorial

Sreelakshmi Sindhu, Ishan Pokharel, Chris Scheulen

---

1 Introduction

Aim of this tutorial is to give you a first hand experience on using root to perform some basic data analysis that will come in use during your Bachelor thesis. Today we will start with the first part which includes, reading event attributes from a TTree, filling it to a histogram and writing it to an output file.

2 Writing a ROOT macro

- **Step 1: Creating the outline**
  - The main function in a ROOT macro should have the same name as the file name. For example, if your file name is EventLoopExample.cxx" the the main function should be called EventLoopExample()
  - Within the macro the main function can look something like this:

```
void EventLoopExample()
   {
   std::cout << "Testing␣my␣ROOT␣macro" << std::endl;
   }
```

  - Run the macro using ROOT:

```
root -l "EventLoopExample.cxx"
```

  - Also, include appropriate header files of ROOT classes you will use. For Example:

```
    #include "TFile.h"
```

- **Step 2: Set the input and output arguments**

- We want the macro to read the input file: "/afs/cern.ch/user/i/ispokhar/public/sampleNtuple/" and store the information that we need in a new file.
- Declare the input file path and the output file path. For Example:

```
void EventLoopExample()
   {

   std::string pathToInputFile = "";
   std::string pathToOutputFile = "";
   std::cout << "Reading␣from␣" << pathToInputFile <<
    "␣and␣writing␣to␣" << pathToOutputFile << std::endl;

   }
```

- Declare TChain and add the file to the TChain.

```
TChain Chain_A("nominal");
```

- Read events from the nominal tree.

```
Chain_A.Add(pathToInputFile);
```

- **Step 3: Identifying the variables, declaring them and setting branch address** Next we have to identify the branch address for the variables we are interested in and link their branch address. For example, for the photon p_T :
  - Declare a variable that will hold the object kinematics as a pointer: (make sure that the datatype is matched)

```
std::vector<float> *v_pt_A =0;
```

  - Declare the branch that holds the information in the ntuple:

```
TBranch *b_pt_A =0;
```

  - Now link the branch to the variable where the information has to be stored temporarily, using TTree::SetBranchAddress.

```
Chain_A.SetBranchAddress("ph_pt", &v_pt_A, &b_pt_A);
```

- **Step 4: Declare histograms:**
  - Our aim is to store the information from the ntuple to a histogram. For that first we have to declare them as TH1 objects:

```
TH1F* pt_A = new TH1F("A_pt", "", 20, 0, 400);
```

  - Here the arguments are as follows: (name of the histogram, title of the histogram, number of bins, starting point of first bin, ending point of last bin)
- **Step 5: Loop over the events**
  - Now write a simple loop over all events.
  - Within the loop fill the histograms
  - Event information can be obtained within the loop using:

```
  Chain_A.GetEntry(ievent);
```

  - The number of events in the tree can be accessed using:

```
int entries_A = Chain_A.GetEntries();
```

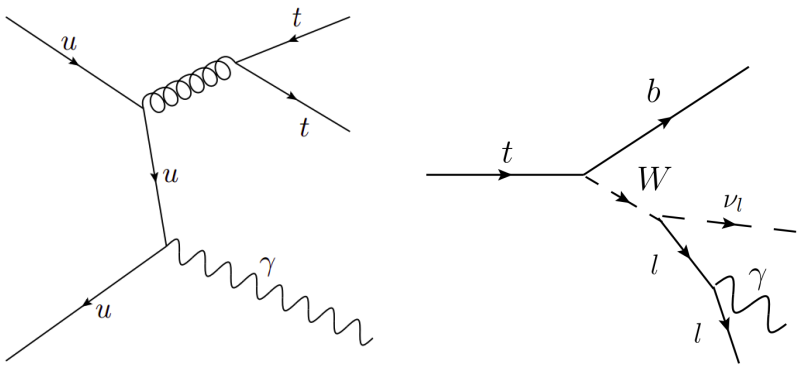
- **Step 6: Write the histogram to output file**
  - The final step is to create a TFile to store the histogram.
  - cd() into the file → Write the histogram →Close the file

```
TFile* outFile = new TFile(outPath + "outFile.root", "RECREATE");
outFile->cd();
pt_A->Write("pt_A");
outFile->Close();
```

- **Step 7: Additional assignments**
  - Write more histograms to the TFile: Specifically, add, "ph_eta" and "ph_phi".
  - Fill a 2D Histogram with photon p_T on x-axis and η on the y-axis.

```
TH2F* hist2D_A = new TH2F("pTvsEta_A", "", 20, 0, 400, 20, -2.5,
   2.5);
```

– Add another input file: "/eos/user/a/ankirchh/ttgamma-analysis/mini-n tuples/l-jets/mc16d_TOPQ1_ttgamma_LO_dec.412114.root"
This file contains events with FSR photons. The file you worked with till now contains $t\bar{t}\gamma$ events with ISR photons.



(a) Photon radiated from the initial state particle (ISR)

(b) Photon radiated from final state particle (FSR)

Figure 1: Feynman diagrams of $t\bar{t}\gamma$ production

– Add another event loop and write, "ph_pt", "ph_eta" and "ph_phi" for the FSR photons to the same TFile.

- Outlook: These histograms will be used in tomorrow's session.

3 Plotting using PYROOT

In this session we will focus on plotting these histograms and saving them as png/pdf files. It is usually useful to overlay histograms and take ratios to do some comparative study. We will use PYROOT to do this in this session but this can also be done using ROOT based on C++.

3.1 Making a simple plot

First access the root file that we created using Tfile and get the required histogram using the Get command. Create a canvas, now draw the histogram using the Draw option ans Save the file. A simple example is below:

```
import ROOT
from ROOT import TCanvas, TFile

Hist_file = TFile("path_to_file")
Hist_name = Hist_file.Get("name_of_the_histogram")
canvas = TCanvas("name", "title", width, height)
Hist_name.Draw("HIST")
canvas.SaveAs("file_name.pdf")
```

3.2 Making it presentable

The plots can be made easier to understand by adding axis-labels, titles, picking colours and adding texts. These can be done using using TLegend, TGaxis, TColor, TH1F, TPad.. Remember to import appropriate header files for each of these classes. Details on the arguments required can be found in the reference guide. Some examples:

```
Hist_name.GetYaxis().SetTitle("Events")
```

3.3 Making ratio plots

You can make several pads within the canvas using TPad. An additional pad can be added for showing the ratio plot.

4 Performing a fit