



# **The framework ROOT and its use in particle physics**

## Session 1: ROOT basics

Instructor: Gustavo Loachamin





# Logistics

- The workshop is divided in 4 sessions, each session we will explore some functionalities of the ROOT framework.
- During each session, an exercise will be proposed to be solved together.
- After the session is over, a exercise will be proposed to be solved by yourselves.
- Each exercise will be marked over 10 points and the final proposed exercise is worth 40 points, you are encourage to work with other people but the you must submit your own work!



# Organization

- **Objective** → Introduce the basic concepts of the ROOT CERN framework and apply them in the analysis of a data set from the LHCb.
- **Session 1** → ROOT basics.
- **Session 2** → Graphical tools: histograms, functions.
- **Session 3** → Fitting with RooFit.
- **Session 4** → Intro to Particle Physics, LHCb open data exercise

# LINUX basic commands

- If you came this far it's because you were able to install the ROOT CERN framework.
- However some of you might not be familiar with the terminal and basic LINUX commands.
- Here are some:
  - ls -> Show files in current directory.
  - cd -> Enter and leave a directory.
  - mv -> move or change name of a file.
  - rm -> delete a file.
  - mkdir -> create a directory.
  - cp -> copy a file.
  - tar -> extract or compress files.
  - pwd -> show the current directory.

# Text editor

- Text file editors are important for the creation of Macros.
- Here are some:

Vi/VIM editor

*Gedit editor*

VSCode

Atom editor

Pico editor

Kate/Kwrite

Eclipse

Jed editor

Leaf Pad

Medit text editor

*Nano editor*

*Sublime text editor*

GNU emacs

Brackets editor

Bluefish

Notepad ++

gVIM editor

*Geany editor*

Light Table

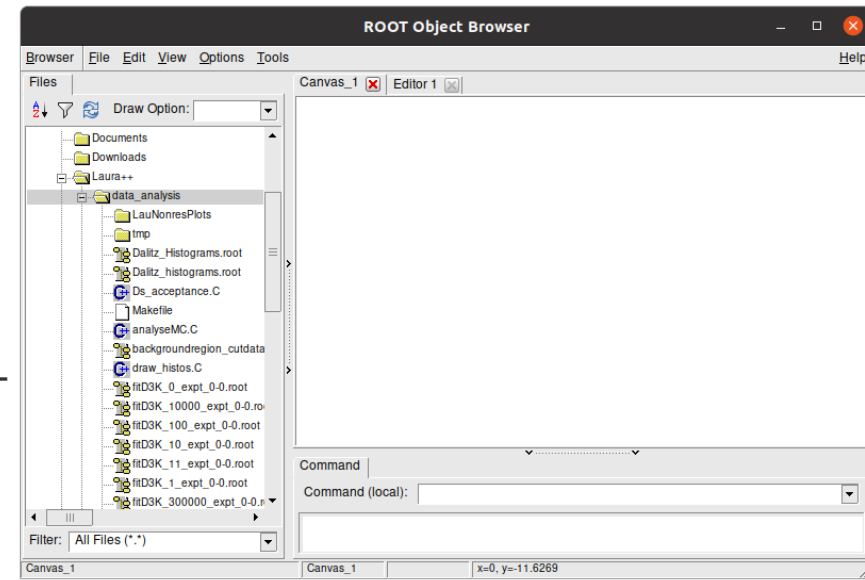
CodeLite

- Everyone is free to choose the text editor that suits you better but for the first in-class and after class exercises I ask that you use **gedit**.
- For more info: <https://www.javatpoint.com/linux-text-editors>

# The ROOT framework

Widely used in high-energy particle physics experiments.

- Possesses a graphical user interface for interactive analysis (GUI).
- It can store and process large files (GB).
- Object oriented programming mainly based on C++
- Several tools like random number generations, fit methods (Minuit), Neural Network framework



```
-----
| Welcome to ROOT 6.20/04                                     https://root.cern |
| (c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers |
| Built for linuxx8664gcc on Apr 01 2020, 08:28:48             |
| From tags/v6-20-04@v6-20-04                                 |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.' |
|-----
root [0] █
```

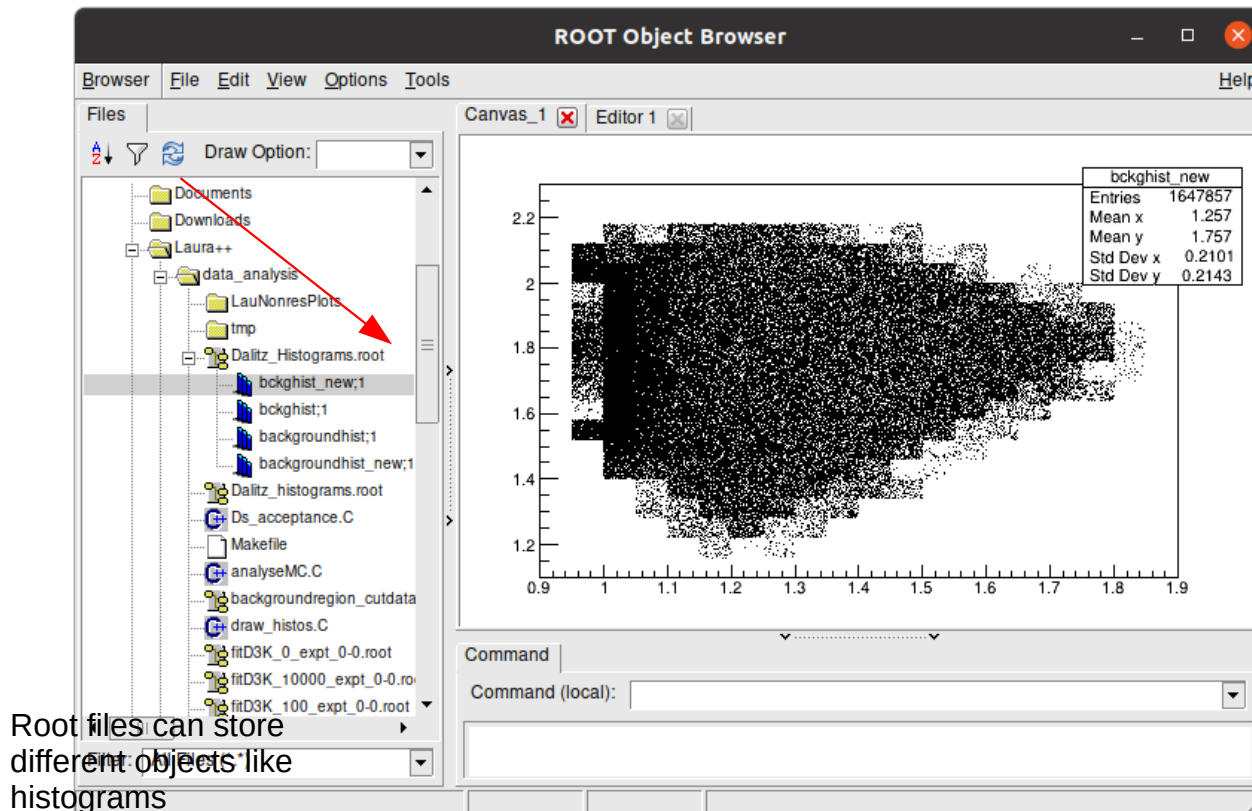
6 / 17

[https://root.cern/get\\_started/](https://root.cern/get_started/)  
<https://root.cern.ch/root/html/doc/guides/primer/ROOTPrimer.html>  
<https://root.cern.ch/doc/master/>

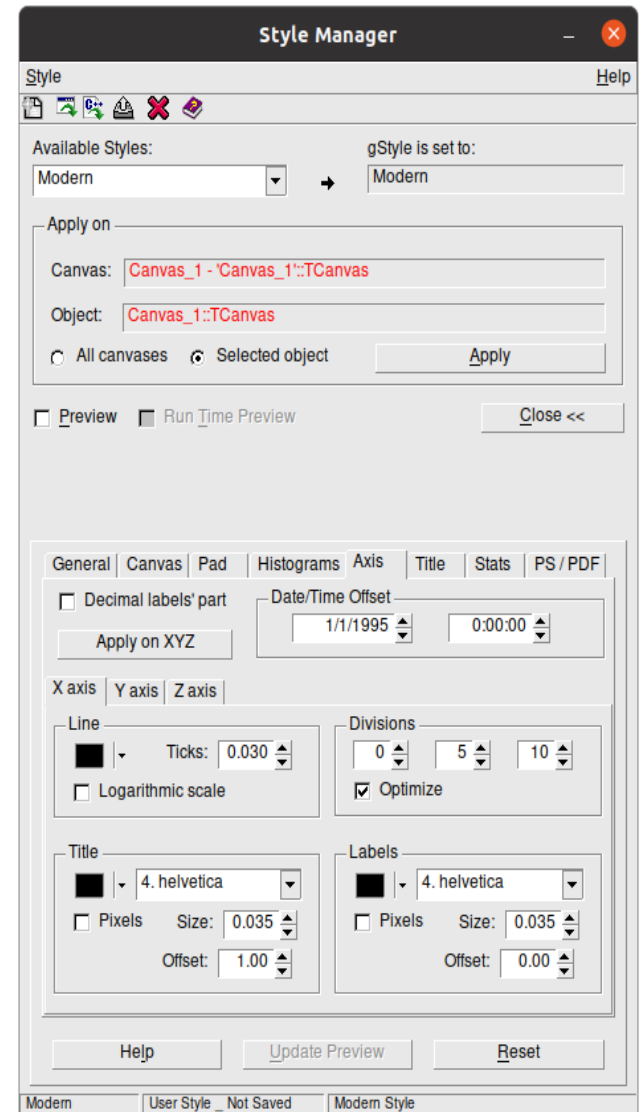
Command line Interpreter CINT  
Basic commands →  
**root** -> open the CINT  
**.q** -> close the CINT

# GUI (Graphic User Interface)

>>new TBrowser  
>>Tbrowser b



Now we play with the GUI!





# C++ Basic operations

**cout <<**  
**Arithmetic: +,-,/,\*,++,--**  
**Assignment: +=,-=,\*=,/=**  
**Comparison: ==,!=,>,<,>=,<=**  
**Logical: &&, ||, !**  
**Strings: +, .size(), .length(), []**



# Arrays and vectors

## Arrays

```
double numbers[3] = {3,4,5};
```

```
string words[3] = {"cv", "df", "dgf"};
```

## Vectors

```
vector<double> a = {3,4,5.6};
```

```
vector<double> b;
```

```
b.push_back(3.4);
```

# For, while loops

- **While loops**

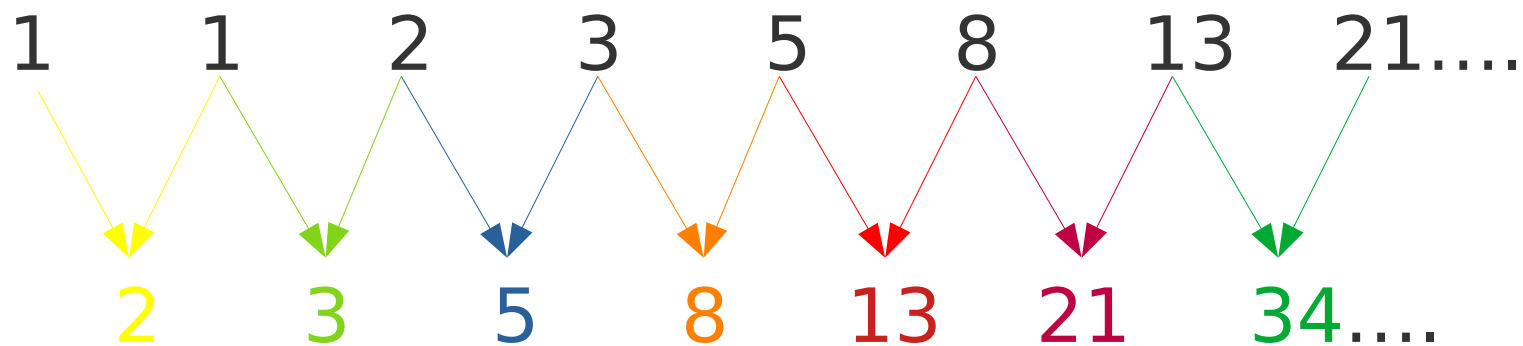
```
while (condition) {  
    // code block to be executed  
}  
//*****  
int i = 0;  
while (i < 5) {  
    cout << i << "\n";  
    i++;  
}
```

- **For Loops**

```
for (int i = 0; i <= 10; i++) {  
    cout << i << "\n";  
}
```

**Now we try a little exercise!**

# Fibonacci Sequence



$$n_1=1 \quad n_2=1 \quad n_3=2 \quad n_4=3 \quad \dots \quad n_i=n_{i-1} + n_{i-2}$$

# Macros

Files that are executable and contain C++ or ROOT methods and functions.

These files end in .C (usually)

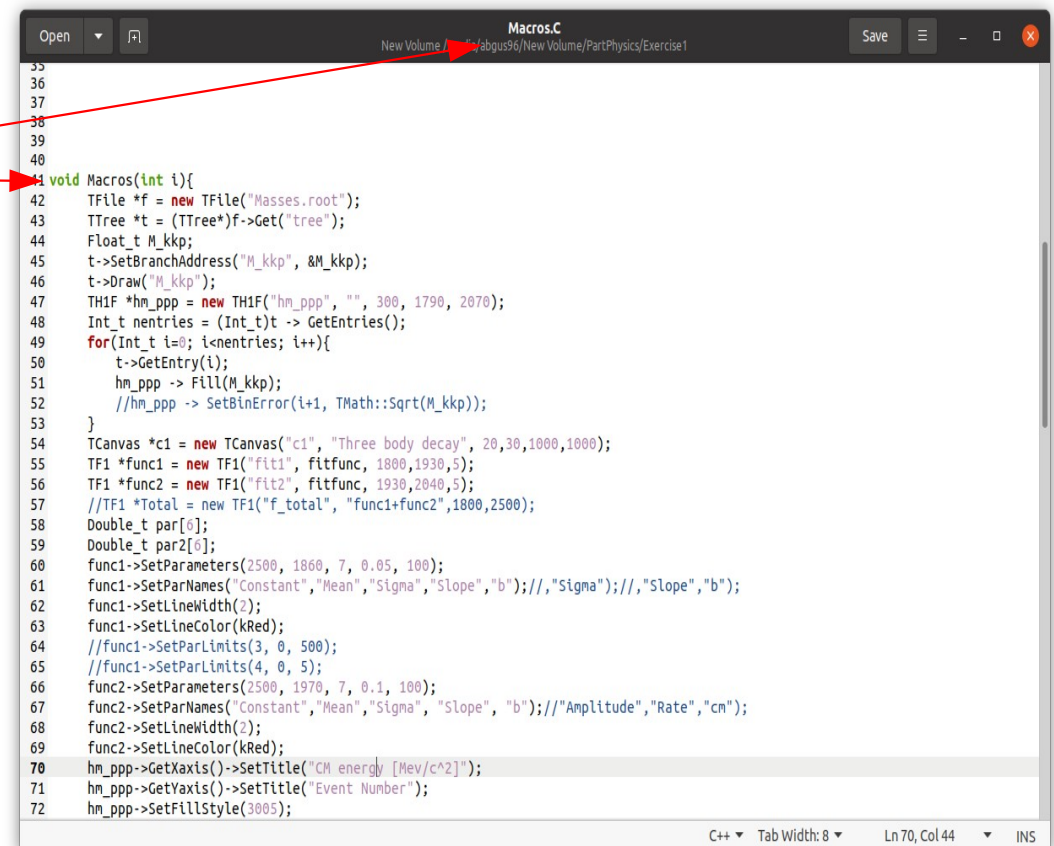
.L macro.C

Load a Macro

.X macro.C

Execute a Macro

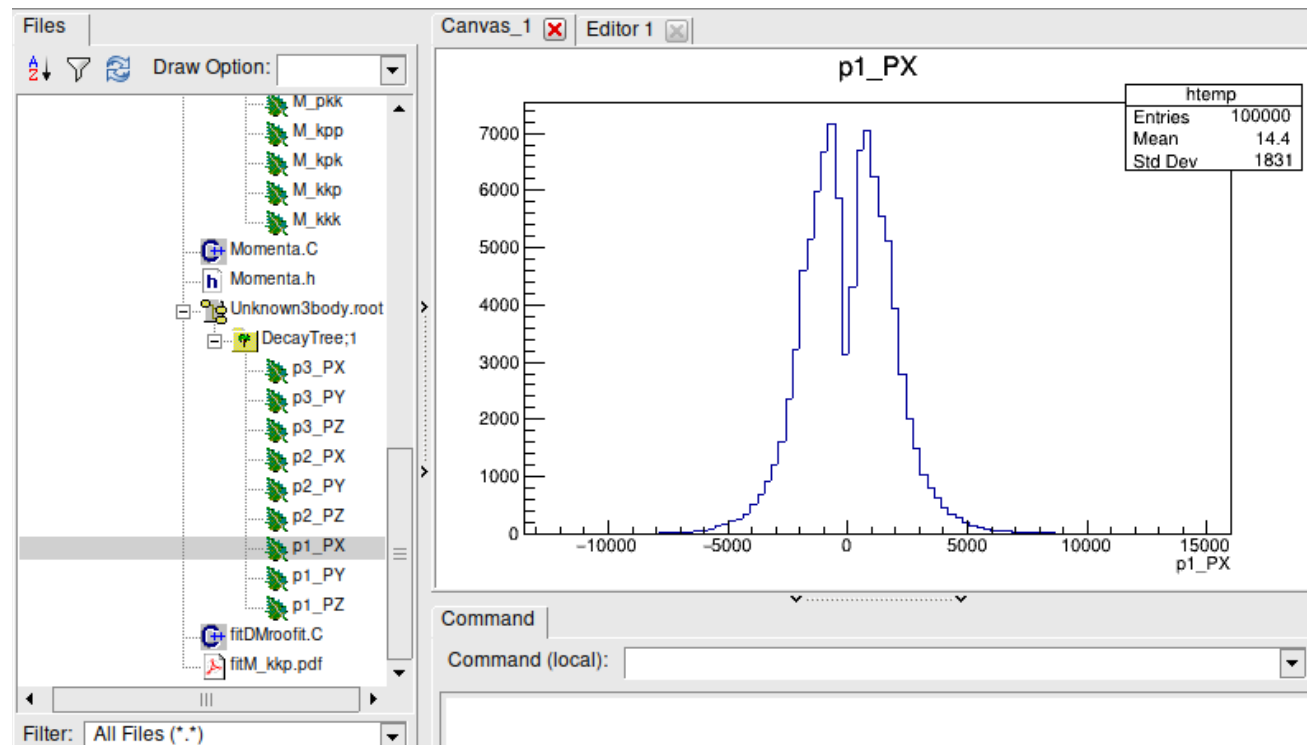
In order to execute a Macro (.X), it must have the same name as the main function!



```
35
36
37
38
39
40
41 void Macros(int i){
42     TFile *f = new TFile("Masses.root");
43     TTree *t = (TTree*)f->Get("tree");
44     Float_t M_kkp;
45     t->SetBranchAddresses("M_kkp", &M_kkp);
46     t->Draw("M_kkp");
47     TH1F *hm_ppp = new TH1F("hm_ppp", "", 300, 1790, 2070);
48     Int_t nentries = (Int_t)t->GetEntries();
49     for(Int_t i=0; i<nentries; i++){
50         t->GetEntry(i);
51         hm_ppp->Fill(M_kkp);
52         //hm_ppp->SetBinContent(i+1, TMath::Sqrt(M_kkp));
53     }
54     TCanvas *c1 = new TCanvas("c1", "Three body decay", 20,30,1000,1000);
55     TF1 *func1 = new TF1("fit1", fitfunc, 1800,1930,5);
56     TF1 *func2 = new TF1("fit2", fitfunc, 1930,2040,5);
57     //TF1 *Total = new TF1("f_total", "func1+func2",1800,2500);
58     Double_t par[0];
59     Double_t par2[0];
60     func1->SetParameters(2500, 1860, 7, 0.05, 100);
61     func1->SetParNames("Constant", "Mean", "Sigma", "Slope", "b");//, "Sigma");//, "Slope", "b");
62     func1->SetLineWidth(2);
63     func1->SetLineColor(kRed);
64     //func1->SetParLimits(3, 0, 500);
65     //func1->SetParLimits(4, 0, 5);
66     func2->SetParameters(2500, 1970, 7, 0.1, 100);
67     func2->SetParNames("Constant", "Mean", "Sigma", "Slope", "b");//, "Amplitude", "Rate", "cn");
68     func2->SetLineWidth(2);
69     func2->SetLineColor(kRed);
70     hm_ppp->GetXaxis()->SetTitle("CM energy [Mev/c^2]");
71     hm_ppp->GetYaxis()->SetTitle("Event Number");
72     hm_ppp->SetFillStyle(3005);
```

# ROOT files

- ROOT is an object oriented programming framework, so it contains different classes (objects).
- They can store ROOT objects like Trees, histograms, graphs, etc.
- ROOT can read data from different sources like files, databases.
- Data sets can be saved as Ttrees which are similar to “tables” (not only with numbers but complex objects).



# Open/save files

TFile f("rootfile.root","CREATE"); // create a new file.

TFile f ("sdfg.root", "update"); //update the contents of a file

TFile f ("sdfg.root", "recreate"); //format a file and open it

TFile f ("sdfg.root", "read"); //read a file

f.Close(); //Close a file

# TTree

- ROOT is an object oriented programming framework which means that it contains many classes with their own methods.
- Important objects: Trees, Ntuples, graphs, functions, histograms:
  - TTrees->Save data in a table, can hold all type of data (ntuples).
- How to fill a TTree (or ntuple)?, here is one method:

```
double a, b; //define the variables to be written
TTree *DecayTree = new TTree("DecayTree", "a simple tree"); //define a Ttree
DecayTree->Branch("a", &a); //set the branch address for a
DecayTree->Branch("b", &b); //set the branch address for b
a =5; //assign a value to a
b =4; //assign a value to b
DecayTree->Fill(); //Fill the branches of a tree
TFile *f = new TFile("test.root", "recreate");
DecayTree->Write();
f->Close();
```

## Read a TTree

- There might exist different methods to read a TTree from root files. Here is one:

```
TFile *f = new TFile("test.root", "READ"); //read file
TTree *tree = (TTree*)f->Get("DecayTree"); //read Ttree
double a,b; //define variable that are gonna hold data.
tree->SetBranchAddress("a", &a); //set branch address a
tree->SetBranchAddress("b", &b); //set branch address b
tree->GetEntry(0); //get the first event
```



# TRandom

- A class used to generate random numbers with different distributions
- Documentation about TRandom can be found here: <https://root.cern.ch/doc/master/classTRandom.html>
- Generate numbers;

```
TRandom *event = new TRandom();  
double a = event→BreitWigner(3,5);
```

- Now we try our first exercise!