ROOT Tutorial

Andrés Delgado

Practical session for bachelor project

April 21th 2022

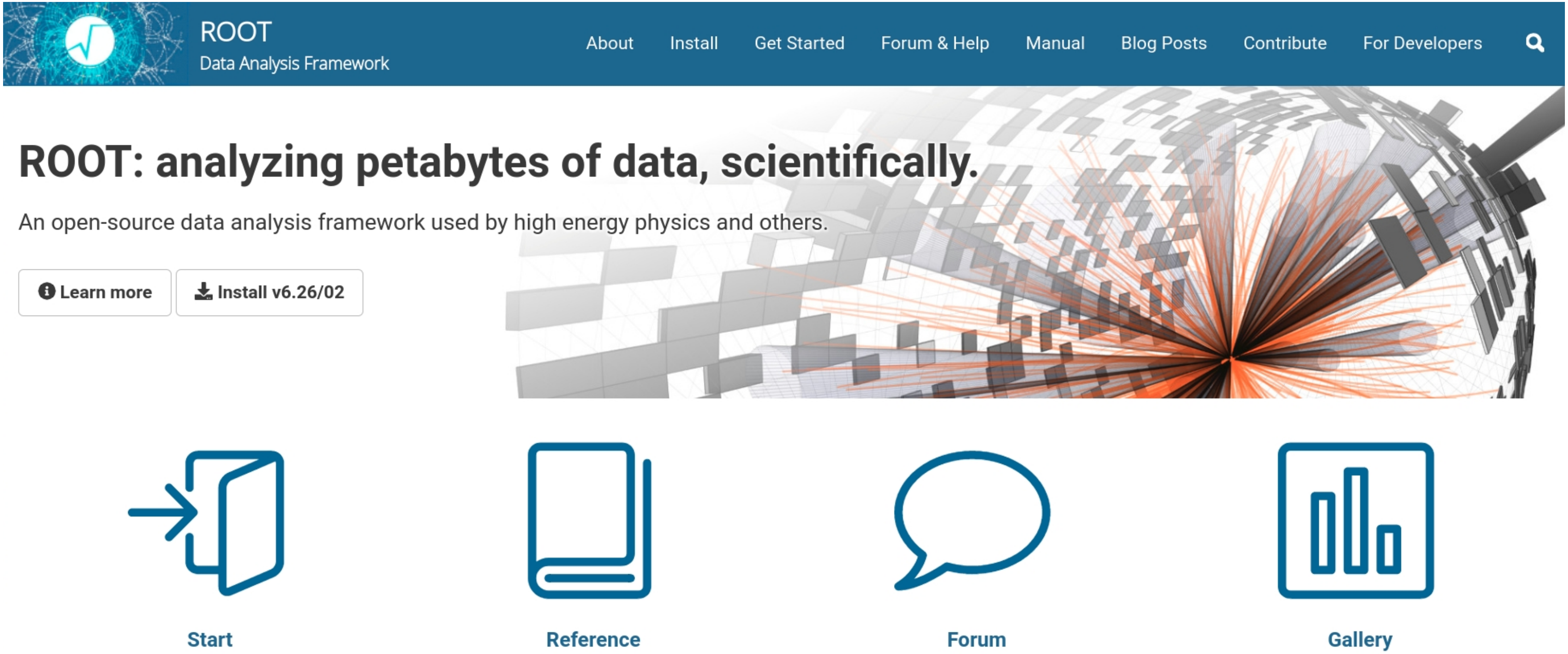
Contents

- **1st part**
  - Create a TGraph
  - Create a TH1D
  - Write simple code
  - Pointers
  - Writting a TTree
  - Reading a TTree
- **2nd part**
  - Read simulation files
  - Write a function
  - Fitting of data

Nomenclature:
- Blue: you type it
- Red: indicate in what enviroment you are

Macros and slides are here https://github.com/andresgd17/Bachelor_project.git

Google: ROOT CERN

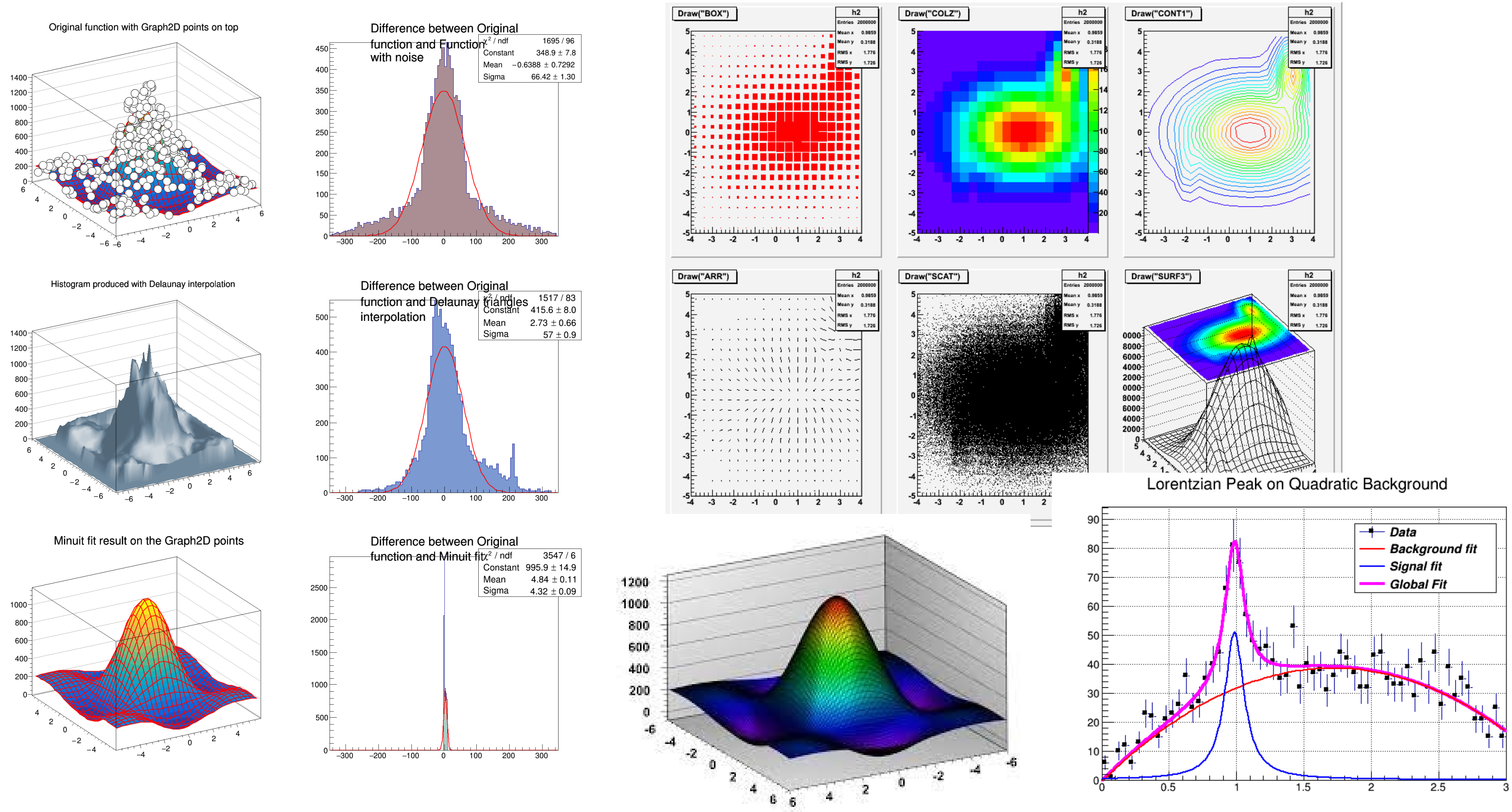


ROOT in a nutshell

- **ROOT is a large Object-Oriented data handling and analysis framework**
  - C++ interpreter
  - Efficient object store scaling from kB's to PB's
- **Extensive set of multi-dimensional histograming, data fitting, modeling and analysis methods.**

Ref: Tutorial from Luca Fiorini

ROOT: Graphics



ROOT in a nutshell

- **The user can interact with ROOT using:**
  - graphical user interface, (probably don’t work in rug cluster)
  - the command line, or
  - scripts.
- **The command and scripting language is C++**

ROOT in the RUG cluster

- **Connect to cluster:**
  - ssh <snumber>@peregrine.hpc.rug.nl
  - https://wiki.hpc.rug.nl/_media/peregrine/additional_information/peregrine_course.pdf
- **Load ROOT module in the cluster**
  - $ module spider ROOT
  - $ module load ROOT

ROOT prompt

- **Starting ROOT**
  - $ root
- **In the ROOT prompt you can type:**
  - root [ ] 5+4
  - root [ ] TMath::Pi()
  - root [ ] float x[5] = {1, 2, 3, 4, 5}
  - root [ ] float y[ ] = {1, 4, 9, 16, 25}
  - root [ ] int i = 3
  - root [ ] cout << i << endl;

TGraph

- **Creating a Graph with TGraph in the ROOT prompt:**
  - root [ ] TGraph gr(5,x,y)
  - root [ ] gr.Draw("alp")
  - root [ ] TGraph gr2
  - root [ ] gr2.SetPoint(0,1,1)
  - root [ ] gr2.SetPoint(1,2,4)
  - root [ ] gr2.Draw("alp")
  -

TH1D

- **Creating a histogram of 1 dimension:**
  - root [ ] TH1D histo("name", "title", 10, 0, 100)
  - root [ ] histo.Fill(2)
  - root [ ] histo.Draw()
  - root [ ] histo.Fill(50, 4)
  - root [ ] histo.Draw("histo")
  -

Creating a macro.C

- **macro.C is the script containts your code**
- **To execute your script macro.C:**
  - System prompt: $ root macro.C
  - ROOT prompt : root [ ] .x macro.C
- **Useful options (root --help or root.help):**
  - System prompt: $ root -b -q macro.C
  - ROOT prompt : root [ ] .L macro.C

Pointers

- **A pointer is a variable that stores address of another variable, .i.e., point to the address of the memory location.**
- A pointer declaration use * and has the form: data_type * name;
  - Variable declaration: int i = 8;
  - Pointer declaration: int * i = new int(8); (To print the value type: *i)
- Allocate memory with operator new just for pointers:
- We access to members of pointers using: ->
- You can get the address of variable using: &

Pointers

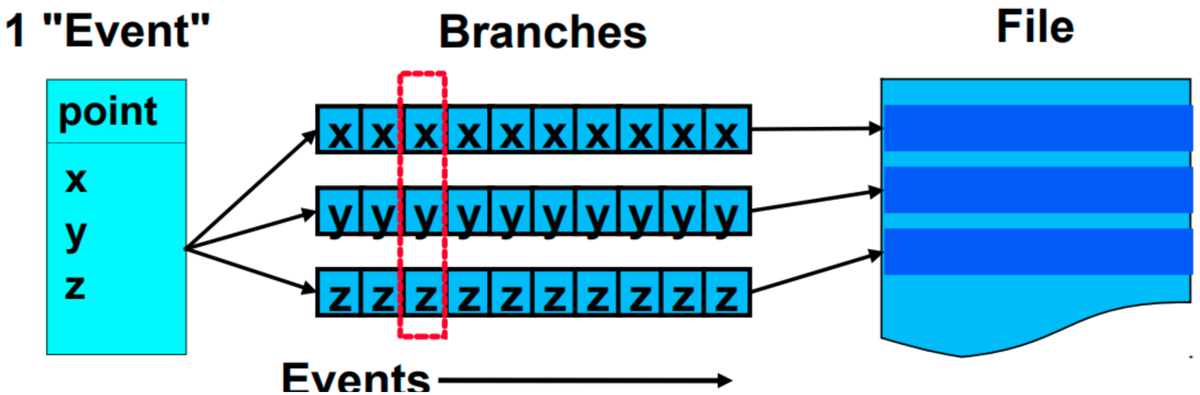
- **Create macro_pointers.C by modifying:**
  - TGraph gr(5,x,y) → TGraph *gr = new TGraph(5,x,y)
  - gr.Draw("alp") → gr->Draw("alp")

Break

- Everything is ok?
- Any questions?

TTree

- **Trees have been designed to support very large collections of objects.**
- **Trees allow direct and random access to any entry**
- **Trees can be browsed via TBrowser**
- **The class TTree is the main container for data storage**
  - It can store any class and basic types (e.g. Float_t)
  - When reading a tree, certain branches can be switched off → speed up of analysis when not all data is needed



Ttree - writing

- **Root files have extension <filename>.root**
  - root [ ] TFile *file = new TFile("myRootFile.root", "recreate")
  - root [ ] TTree *tree = new TTree("tree_name", "tree_title")
  - root [ ] float energy
  - root [ ] tree->Branch("Energy", &energy)
  - root [ ] tree->Print()
  - root [ ] energy = 50
  - root [ ] tree->Fill()
  - ...

Ttree - writing

- ...
- root [ ] tree->Print()
- root [ ] tree->Scan("Energy")
- root [ ] energy = 100
- root [ ] tree->Fill()
- root [ ] tree->Print()
- root [ ] tree->Scan("Energy")
- root [ ] file->Write()
- root [ ] file->Close()

Don’t forget to write all objects in the file!!

Ttree - reading

- **Reading myRootFile.root:**
  - root [ ] TFile *file = new TFile("myRootFile.root", "read")
  - root [ ] .ls
  - root [ ] tree_name->Print()
  - root [ ] tree_name->Scan("Energy")
  - root [ ] tree_name→Draw("Energy")
- **Check readtree.C:** Script to read a tree
  - $ root readtree.C

Small assignment

- **Try to type by yourself all the codes again to familiarize with them.**
- **Create a script called writetree.C to create a Ttree:**
  - You can use the same code from Ttree – writing section :)
  - Try to create another branches: i.e. velocity, momentum, etc, and fill them with random numbers :)

2nd part of tutorial

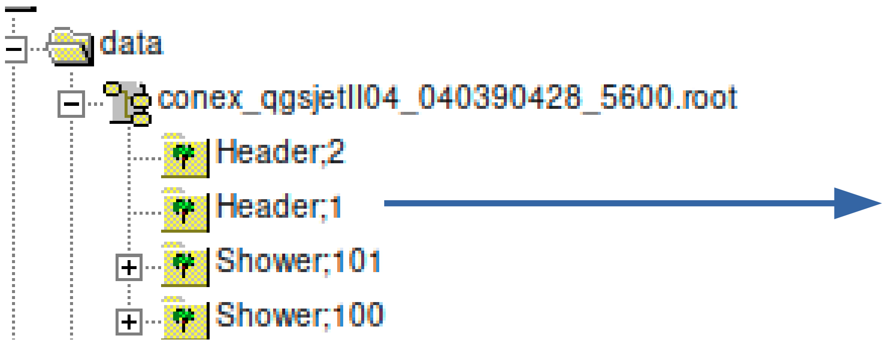
- **Read simulations files:**
  - conex_qgsjetII04_040390428_5600.root

Seed number

ID of primary particle:
- Gamma: 0
- Proton: 100
- Iron: 5600

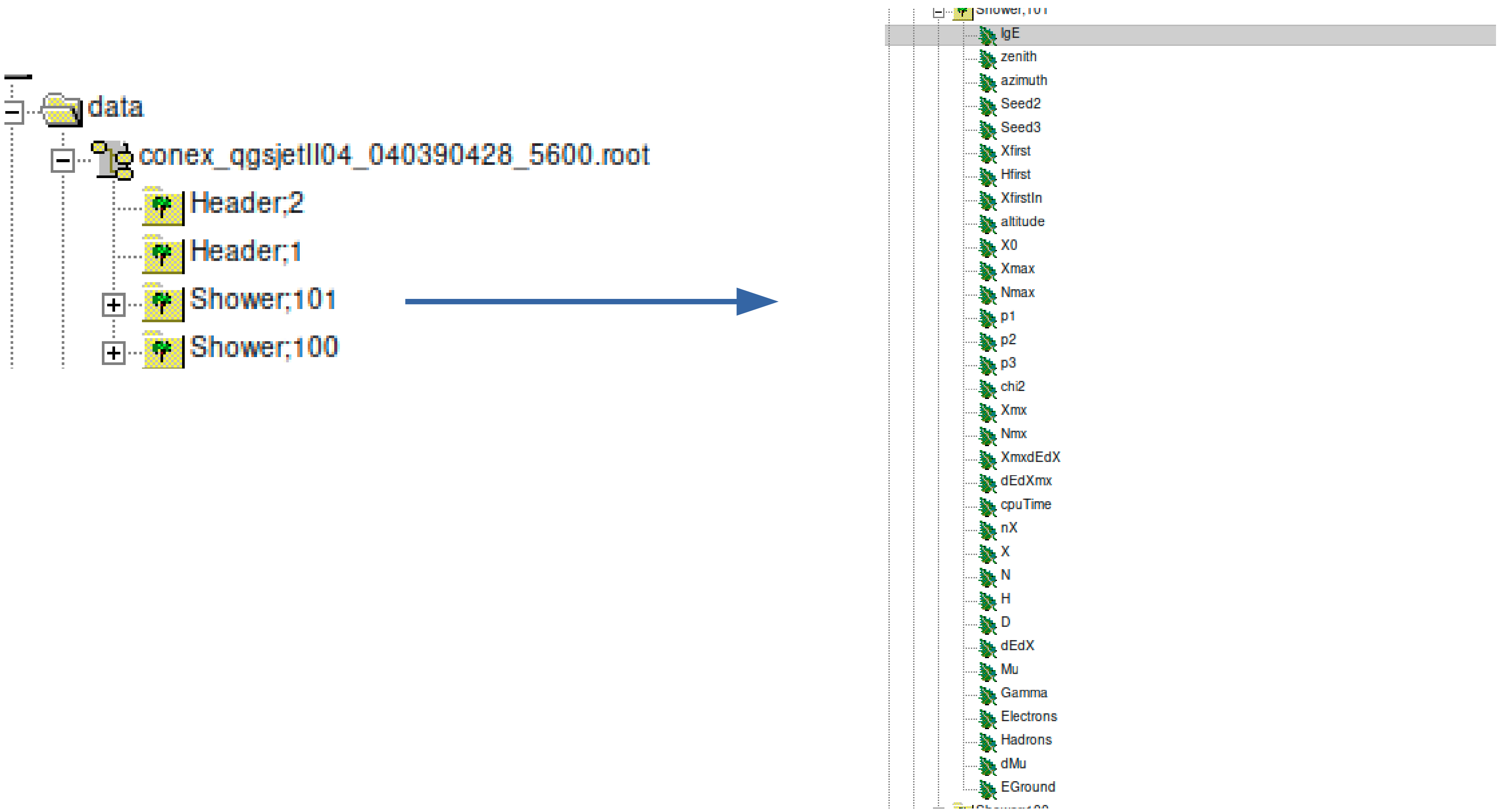
  - 12 files in total:
    - 4 files for each kind of primary particle: gamma, proton and iron
      - Each of the 4 files for each energy: 10 TeV, 50 TeV, 100 TeV, 300 TeV

Data files


data
conex_qgsjetII04_040390428_5600.root
Header;2
Header;1
Shower;101
Shower;100


data
conex_qgsjetII04_040390428_5600.root
Header;2
Header;1
Seed1
Particle
Alpha
lgEmin
lgEmax
zMin
zMax
SvnRevision
Version
OutputVersion
HEModel
LEModel
HiLowEgy
hadCut
emCut
hadThr
muThr
emThr
haCut
muCut
elCut
gaCut
lambdaLgE
lambdaProton
lambdaPion
lambdaHelium
lambdaNitrogen
lambdaIron
Shower;101
Shower;100

Data files


data
conex_qgsjetII04_040390428_5600.root
Header;2
Header;1
Shower;101
Shower;100
lgE
zenith
azimuth
Seed2
Seed3
Xfirst
Hfirst
XfirstIn
altitude
X0
Xmax
Nmax
p1
p2
p3
chi2
Xmx
Nmx
XmxdEdX
dEdXmx
cpuTime
nX
X
N
H
D
dEdX
Mu
Gamma
Electrons
Hadrons
dMu
EGround

Important information

- **Tree**: Shower;101
  - **Branches**:
    - LgE: log10(Energy /eV) e.g: lgE = 14 → Energy=10^{14} eV
    - nX: number of points of "X", "N",
    - X: slant depth array (g/cm^2)
    - N: array of number of charged particles
- **For more information check README file in data folder**

Plot profiles

- **Use plotProfile.C script to plot TGraph of charged particles profiles with legend.**
- **Let’s plot one profile:**
  - ROOT prompt:
    - $ root
    - root [ ] .x plotProfile.C("../data/conex_qgsjetII04_040390428_5600.root",4)
  - System prompt:
    - $ root "plotProfile.C(\"../data/conex_qgsjetII04_040390428_5600.root\",4)"

TChain

- **A chain is a collection of files containing TTree objects.**
  - TChain Shower("Shower");
  - Shower.Add("../data/conex_xxx_.root");
  - Shower.Add("../path/conex_xxx_.root")
- **Check chain.C**

Assignment

- Open the conex_xxx_.root files and explore the contents.
- Plot the profile of charged particles, electrons, hadrons and muons (in the same TGraph) for each particle of some energy: 3 plots in total for each profile.
- Fix the Y-axis title in the profile plot (search how to do on root fórum, google, ...)

Write a function

- Check plotFitFunction.C
  - Define second-order polynomial function
  - Fit profile around the maximum
- Assignment:
  - Get the Xmax from the second-order polynomial fitting for each kind of air shower: 3 plots in total.
  - Plot the distribution of the Xmax in a histogram TH1D

Backup

Tree

- Branches
  - root[ ] Shower->Print()
  - root[ ] Shower->Show(5)
  - root[ ] Shower->Scan(X)
  - root[ ] Shower->Scan(N)
  - root[ ] Shower→Draw("N:X","X < 500", "*", 1, 5)
  -

condition

First shower to consider

# of showers

Variables you want to draw

Useful

- You can open files when starting ROOT by:
  - $ root file.root
- Suppose a macro myscript.C with a function myscript(string file, int i, float x)
  - $ root "plot.C(\"filename.root\",1,1)"
- Type .? to see meta commands in the ROOT prompt

Need help?

- ROOT fórum (https://root-forum.cern.ch/)
- Youtube: search for root cern <something> (could work)
- Send me an e-mail! :)