Queen Mary University of London

Graduate Computing Course

Gitanjali Poddar
7/11/2024

Course Outline

https://indico.ph.qmul.ac.uk/event/2175/

2024 Graduate Lectures (II)

31 Oct 2024, 12:00 → 7 Nov 2024, 14:00 Europe/London

Bancroft Building

Gitanjali Poddar (Queen Mary)

Description Zoom link: https://cern.zoom.us/j/8157769491?pwd=TWRtN0s0ZXVpa05sK293WHM3R1pVUT09

Thursday 31 October

12:00 → 14:00 **Linux and Git** — G.13

Wednesday 6 November

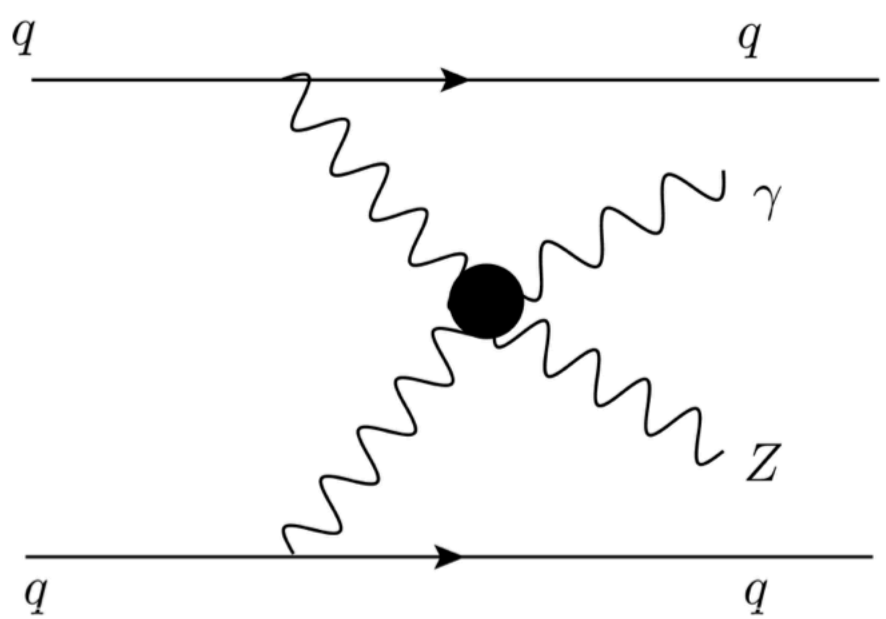
15:00 → 17:00 **C++ and ROOT** — 1.03 (Bancroft Building)

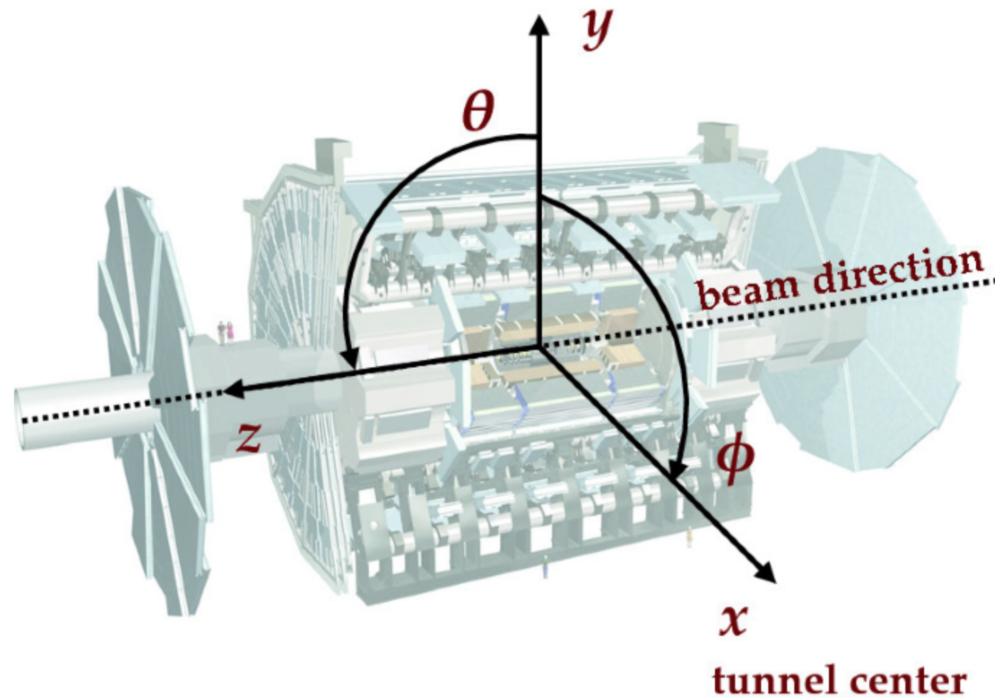
Thursday 7 November

12:00 → 14:00 **More ROOT and PyROOT** — 1.03 (Bancroft Building)

Part I: Intro to ROOT (II)

Introduction





- Analysis signature: $qq \to Z(\to l^+l^-)\gamma qq$
  - Final state objects: 2 quark jets, 2 leptons and 1 photon
- Data is in `events.root`
  - Contains `tree` which has p_T, E, η, ϕ of each final state object and event number

$\eta = -\ln[\tan(\theta/2)]$

Goals

- Compute variables $m_{jj}, p_T^{Z\gamma}, \Delta\phi(j_1, j_2)$ and store in a tree (`Compute.C`)
- Plot histogram of m_{jj} with binning {500, 800, 1200, 2000} GeV for $p_T^{Z\gamma}$>50 GeV. Draw another histogram half its size on the same plot (`Hist_Draw.C`)
- Some other exercises..

Compute.C

```
//local variables of new tree
Double_t m_jj, pt_llpho, dphi_jj;

//creating new tree
TFile *file = new TFile("computed.root","RECREATE");
TTree *tree_new = new TTree("tree", "computed_tree");

tree_new->Branch("eventnumber", &evt);
tree_new->Branch("mass_jj", &m_jj);
tree_new->Branch("pt_llpho", &pt_llpho);
tree_new->Branch("d_phi_jj", &dphi_jj);

//initialising Lorentz vectors for computation
TLorentzVector pho(0., 0., 0., 0.);
TLorentzVector lep1(0., 0., 0., 0.);
TLorentzVector lep2(0., 0., 0., 0.);
TLorentzVector jet1(0., 0., 0., 0.);
TLorentzVector jet2(0., 0., 0., 0.);

//computing quantities for new tree
for (int i=0; i<tree_original->GetEntries(); i++)
{
    tree_original->GetEntry(i);

    pho.SetPtEtaPhiE(pt_pho, eta_pho, phi_pho, e_pho);
    lep1.SetPtEtaPhiE(pt_lep1, eta_lep1, phi_lep1, e_lep1);
    lep2.SetPtEtaPhiE(pt_lep2, eta_lep2, phi_lep2, e_lep2);
    jet1.SetPtEtaPhiE(pt_jet1, eta_jet1, phi_jet1, e_jet1);
    jet2.SetPtEtaPhiE(pt_jet2, eta_jet2, phi_jet2, e_jet2);

    m_jj=(jet1+jet2).M();
    pt_llpho=(lep1+lep2+pho).Pt();
    dphi_jj=jet1.DeltaPhi(jet2);

    //filling new tree
    tree_new->Fill();
}
```

Defining tree

Creating branches of tree

Creating Lorentz vector objects

Updating Lorentz vector objects

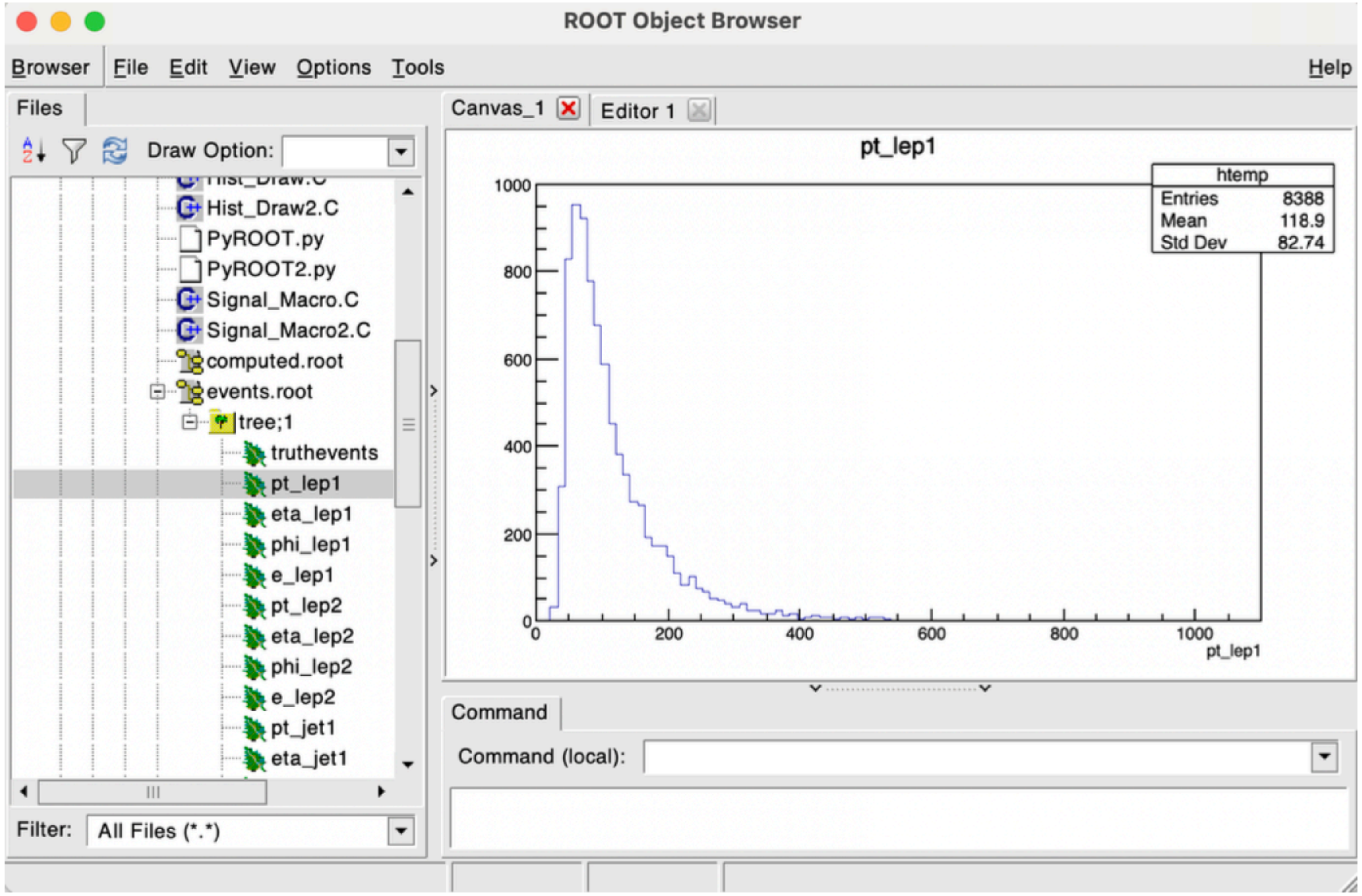
Computing required variables

Filling tree: all branches get filled at the same time

https://root.cern.ch/doc/master/classTLorentzVector.html

TBrowser

- `TBrowser` is useful to read and visualise ROOT files
  - Type "`TBrowser t`" in ROOT shell



Exercise

- Run "`Compute.C`". It generates "`computed.root`"
- In a new macro, compute m_Z, and add it as a branch of the tree in `computed.root`

Solution: Compute2.C

```
//updating tree
TFile file("computed.root", "UPDATE");
TTree *tree_updated=(TTree*)file.Get("tree");

TBranch *new_branch= tree_updated->Branch("mass_Z", &m_Z);

//initialising Lorentz vectors for computation
TLorentzVector pho(0., 0., 0., 0.);
TLorentzVector lep1(0., 0., 0., 0.);
TLorentzVector lep2(0., 0., 0., 0.);
TLorentzVector jet1(0., 0., 0., 0.);
TLorentzVector jet2(0., 0., 0., 0.);

//computing quantities for updated tree
for (int i=0; i<tree_original->GetEntries(); i++)
{
    tree_original->GetEntry(i);

    pho.SetPtEtaPhiE(pt_pho, eta_pho, phi_pho, e_pho);
    lep1.SetPtEtaPhiE(pt_lep1, eta_lep1, phi_lep1, e_lep1);
    lep2.SetPtEtaPhiE(pt_lep2, eta_lep2, phi_lep2, e_lep2);
    jet1.SetPtEtaPhiE(pt_jet1, eta_jet1, phi_jet1, e_jet1);
    jet2.SetPtEtaPhiE(pt_jet2, eta_jet2, phi_jet2, e_jet2);

    m_Z=(lep1+lep2).M();

    //filling new tree
    new_branch->Fill();
}
```

Use UPDATE option for file

Fill the new branch only and not entire tree

Hist_Draw.C

```
//creating histogram
int bins=3;
float binning[4];
binning[0]=500;binning[1]=800; binning[2]= 1200; binning[3]=2000;

TH1D *hist=new TH1D("h1", "p_{T}^{Z#gamma}>50 GeV", bins, binning);
hist->Sumw2();
```

- Create histogram with unequal binning
- Title of histogram has `LateX` style with #
- `Sumw2()` enables calculation of errors

```
//creating new histogram whose integral is half of hist
TH1D *hist_half= (TH1D*) hist->Clone();
hist_half->Scale(0.5);
hist_half->Sumw2();
```

- `Clone()` enables copying histogram

Hist_Draw.C (II)

```
//Plotting histograms
gStyle->SetOptStat(0);
TCanvas *c=new TCanvas("c","c");

hist->SetLineColor(kRed);
hist->SetLineWidth(1);

hist_half->SetLineColor(kBlue);
hist_half->SetLineStyle(kDashed);

hist->GetYaxis()->SetRangeUser(0, 1500);
hist->GetYaxis()->SetTitle("Events");
hist->GetXaxis()->SetTitle("m_{jj} (in GeV)");

hist->Draw();
hist_half->Draw("same");

auto legend = new TLegend(0.7,0.7,0.9,0.9);
legend->AddEntry(hist,"Original","l");
legend->AddEntry(hist_half,"Half","l");
legend->Draw();

c->Update();
c->SaveAs("Histograms.png");
c->Clear();
c->Close();
```

Removes statistics box from plot

Set histogram plotting options

Set histogram axes options

“same” option to superimpose histogram on plot

Update and save histograms after drawing

https://root.cern.ch/doc/master/classTColor.html

https://root.cern.ch/doc/master/classTCanvas.html

Hist_Draw.C (III)

```
auto legend = new TLegend(0.7,0.7,0.9,0.9);
legend->AddEntry(hist,"Original","l");
legend->AddEntry(hist_half,"Half","l");
legend->Draw();
```

- `auto` is a C++ keyword that automatically assigns a type to a variable. It is usually used for complicated objects.
  Note: you could also use `TLegend *legend=new TLegend(…)`

- `TLegend(x1,y1,x2,y2)`: position of legend on the plot

https://root.cern.ch/doc/master/classTLegend.html

Exercise II

- Run `Hist_Draw.C`. It generates "`Histograms.png`"
- In a new macro, plot m_Z:
  - Binning for m_Z: {60, 70, 85, 90, 95} GeV
  - Keep statistics box
  - Save as "`Histograms2.pdf`"

Solution in `Hist_Draw2.C`

Summary

- ROOT to work on analysis:
    - Create, read and update files
    - Create, read and update trees
    - Create, read and plot histograms
- Many other ROOT features available- documentation online
- Tip: work on existing code if possible..

Part II: PyROOT

Introduction

- ROOT has Python bindings
  - `import ROOT`
- Run a .py file as `python file.py`
- **Task 1**: compute m_Z from `events.root`, fill it in a histogram ranging from 60-100 GeV with 5 bins, and save it to `output.root`
- **Task 2**: read the histogram from the `output.root` file and plot it

```python
import ROOT

#reading file and getting tree
file  = ROOT.TFile("events.root","READ")
tree  = file.Get("tree")

#creating Lorentz vector objects
lep1 = ROOT.TLorentzVector()
lep2 = ROOT.TLorentzVector()
jet1 = ROOT.TLorentzVector()
jet2 = ROOT.TLorentzVector()
pho = ROOT.TLorentzVector()

#creating histogram
mZ = ROOT.TH1D("h1","m_{Z}",5,60,100)
mZ.Sumw2()

#reading tree and computing
for i in range(0,tree.GetEntries()):

    tree.GetEntry(i)

    pt_lep1  = getattr(tree,"pt_lep1")
    pt_lep2 = getattr(tree,"pt_lep2")
    eta_lep1  = getattr(tree,"eta_lep1")
    eta_lep2 = getattr(tree,"eta_lep2")
    e_lep1  = getattr(tree,"e_lep1")
    e_lep2 = getattr(tree,"e_lep2")
    phi_lep1  = getattr(tree,"phi_lep1")
    phi_lep2 = getattr(tree,"phi_lep2")

    lep1.SetPtEtaPhiE(pt_lep1, eta_lep1, phi_lep1, e_lep1)
    lep2.SetPtEtaPhiE(pt_lep2, eta_lep2, phi_lep2, e_lep2)

    dilepton = lep1 + lep2
    mZ.Fill(dilepton.M()) #filling histogram

outHistFile = ROOT.TFile.Open("output.root","RECREATE")
outHistFile.cd()
mZ.Write()
```

```python
import ROOT

#reading file and getting histogram
file  = ROOT.TFile("output.root","READ")
hist  = file.Get("h1")

hist.SetStats(0) #turn off statistics box
hist.SetTitle("")
hist.SetLineColor(ROOT.kRed)
hist.SetLineWidth(2)
hist.GetYaxis().SetTitle("Events")
hist.GetXaxis().SetTitle("m_{Z} (in GeV)")

c1 = ROOT.TCanvas("canvas")
c1.cd()
hist.Draw()
c1.Print("hist.png")
```

`PyROOT.py` and `PyROOT2.py`: hopefully comments are self-explanatory!

https://root.cern/manual/python/

Summary

- **Exercise**: try to recreate `Compute.C` and `Hist_Draw.C` in PyROOT
- `import ROOT` is the main command to work in PyROOT
- You can work with PyROOT or ROOT (with C++)
    - Note: in general, more documentation exists for the latter