Computational Physic School : A ROOT Guide For Beginners

ATMANI Hicham

University Mohammed VI Polytechnic, 21 Dec 2022

Overview

Root: A powerful software framework for data analysis, can be used to

- Manipulation of large amount of data.
- Classification and cleaning Data.
- Processing Data.
- Visualization Data.

Root is characterized with:

- Data structures for event-based data analysis.
- Solid Mathematical Libraries.
- Huge libraries of advanced statistical tools (RooFit, RooStats, etc.)
- Different Machine learning technics (neural networks and Boosted Decision Trees)
- powerful high-quality graphics capabilities and interfaces.

Plan

- Root Installation on google colaboratory notebook.
- A simple function definition using python (sine function).
- Definition of sine function using Root.
- Definition of Histograms using Root.
- Fitting a histograms using function.
- Reading and manipulating root files.

ROOT references:

- ROOT Website: https://root.cern
- Material online: https://github.com/root-project/training
- More material: https://root.cern/getting-started

Root Installation on google colaboratory notebook

1. Open Google drive;
2. right-click: google colaboratory

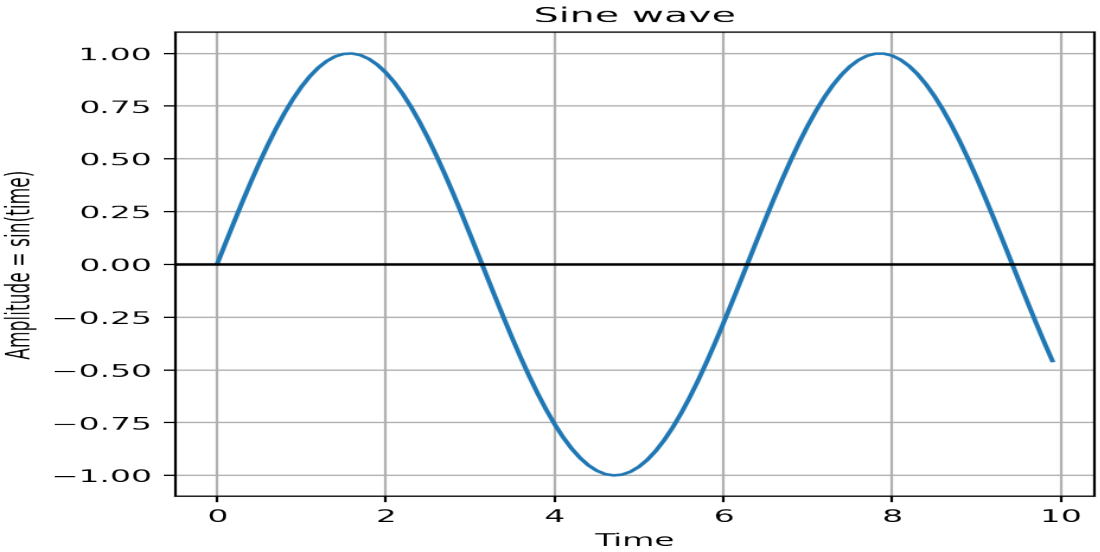
Code: Root Installation

- !wget https://github.com/MohamedElashri/HEP-ML/releases/download/ROOT/ROOT.tar.zip
- !unzip /content/ROOT.tar.zip
- !tar -xf ROOT.tar
- !apt-get install git dpkg-dev cmake g++ gcc binutils libx11-dev libxpm-dev libxft-dev libxext-dev tar gfortran subversion
- !pip install root_nump

A simple function definition using python (sine function):

```
import numpy as np
import matplotlib.pyplot as plot

# Get x values of the sine wave
time         = np.arange(0, 10, 0.1);

# Amplitude of the sine wave is sine of a variable like time
amplitude    = np.sin(time)

# Plot a sine wave using time and amplitude obtained for the sine wave
plot.plot(time, amplitude)

# Give a title for the sine wave plot
plot.title('Sine wave')

# Give x axis label for the sine wave plot
plot.xlabel('Time')

# Give y axis label for the sine wave plot
plot.ylabel('Amplitude = sin(time)')
plot.grid(True, which='both')
plot.axhline(y=0, color='k')
plot.show()

# Display the sine wave
plot.show()
```

A simple function definition using python (sine function):

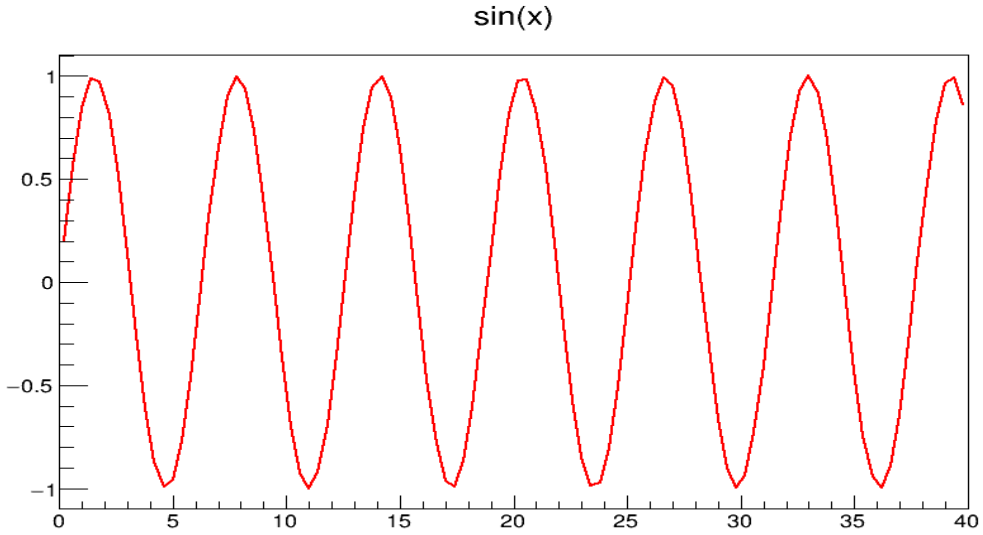


Define a function using ROOT (Part 1):

- ROOT benefits from a huge collection of mathematical libraries.
- Using ROOT allows one to reduce programming time.
- Using ROOT allows to create distributions and functions with minimal memory usage.

```
import ROOT
# TF1 : class for functions definition
# TCanvas : class for Plotting

FunctionSinus = ROOT.TF1("fa1","sin(x)",0,40)
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",900,600)
FunctionSinus.Draw()
canvas.Draw()
```

Define a function using ROOT (Part 1):

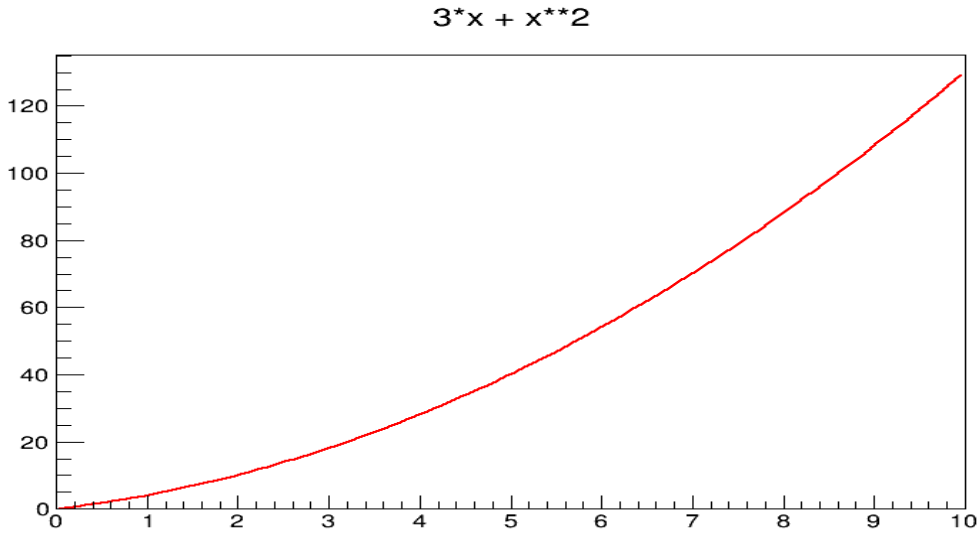


Define a function using ROOT (Part 2):

- We can define other functions using the TF1 class.
- Here is an example of f(x) $= x^2 + 3x$.

```
import ROOT
# TF1 : class for functions definition
# TCanvas : class for Plotting

FunctionSinus = ROOT.TF1("fa2","3*x + x**2",0,10)
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
FunctionSinus.Draw()
canvas.Draw()
```

Define a function using ROOT (Part 2):

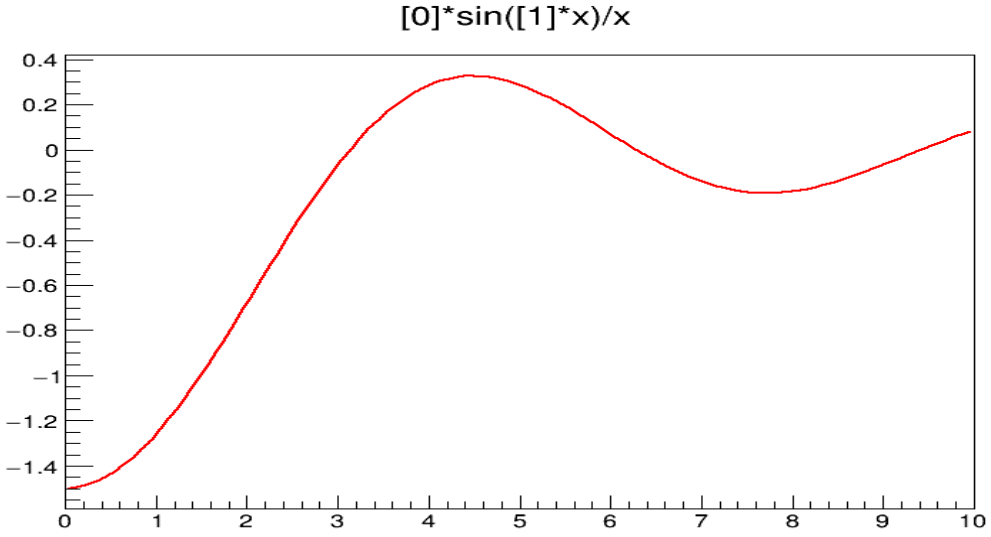


Define a function using ROOT (Part 3):

- We can also define functions with parameters that can be varied or defined later.
- We can use ROOT to calculate function parameters as well.

```
import ROOT
# TF1 : class for functions definition
# TCanvas : class for Plotting

Function = ROOT.TF1("f1","[0]*sin([1]*x)/x",0,10)
Function.SetParameter(0,1.5);
Function.SetParameter(1,-1.);

canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
Function.Draw()
canvas.Draw()

#Get function value, derivative, integral
print( "value at 4 :      ", Function.Eval(4))
print( "derivative at 4 : ", Function.Derivative(4))
print( "Integral :        ", Function.Integral(4,5))
```

Define a function using ROOT (Part 3):

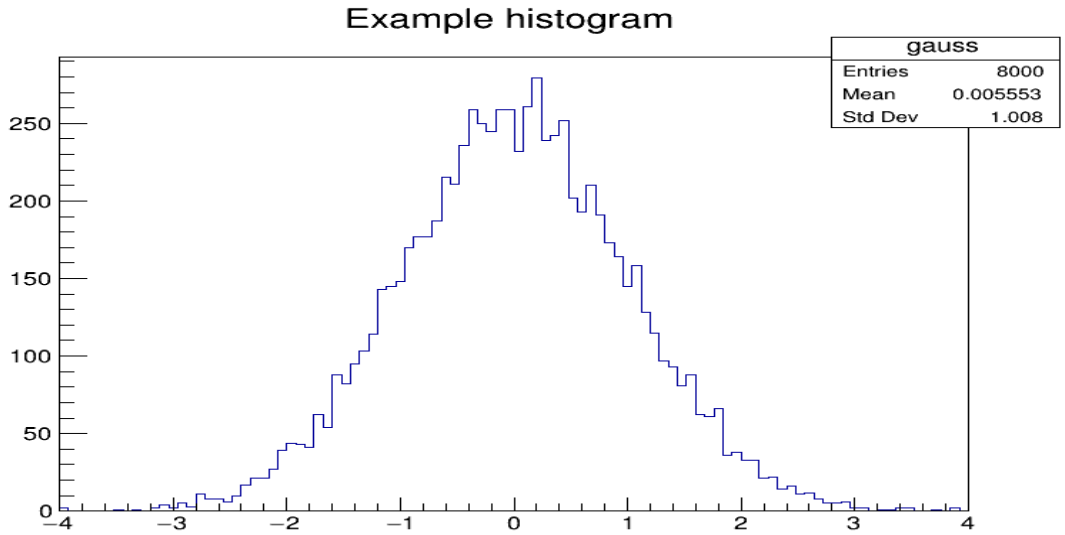


Define Histograms using ROOT:

- In addition to its functions, ROOT can also be used to define histograms.
- There are different ways to define histograms, either randomly or with a special function such as a Gaussian distribution.
- For histograms, we can define the binning, range, and number of events.

```
import ROOT
# TH1F : class for histogram definition
# TCanvas : class for Plotting

histogram = ROOT.TH1F("gauss","Example histogram",100,-4,4)
histogram.FillRandom("gaus", 8000)

canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
histogram.Draw()
canvas.Draw()
```

Define Histograms using ROOT:

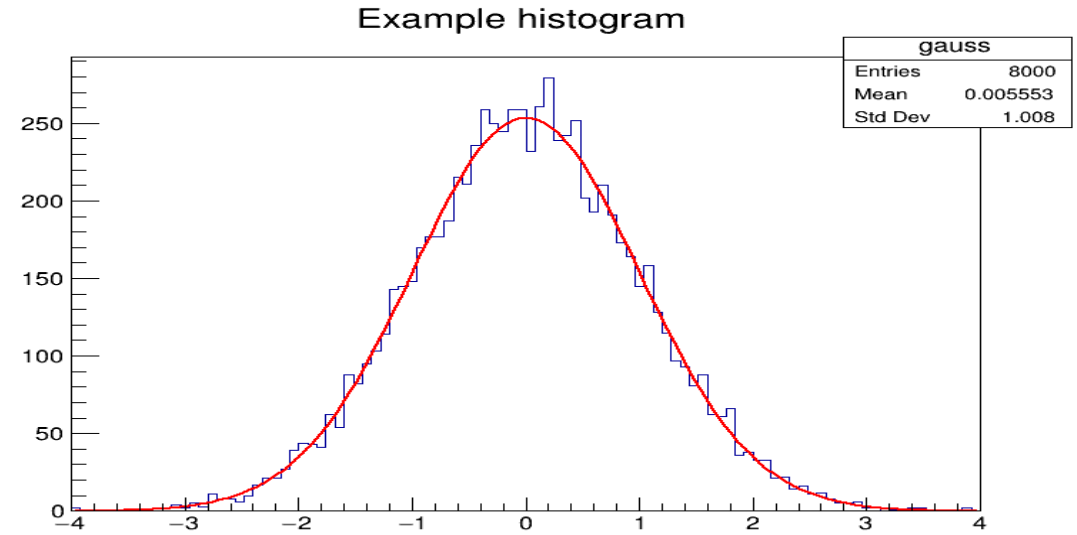


Fitting Histograms using ROOT (Part 1):

- For histograms, the statistics are often limited, so we try to fit histograms to known functions.
- ROOT offers a large library of functions and tools for fitting.
- ROOT provides a list of predefined functions for fitting, such as a Gaussian function.

```
import ROOT
# TH1F : class for histogram definition
# TCanvas : class for Plotting
# Fit : function for fitting

histogram = ROOT.TH1F("gauss","Example histogram",100,-4,4)
histogram.FillRandom("gaus", 8000)
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
histogram.Fit("gaus")
histogram.Draw()
canvas.Draw()
```

Fitting Histograms using ROOT (Part 1):

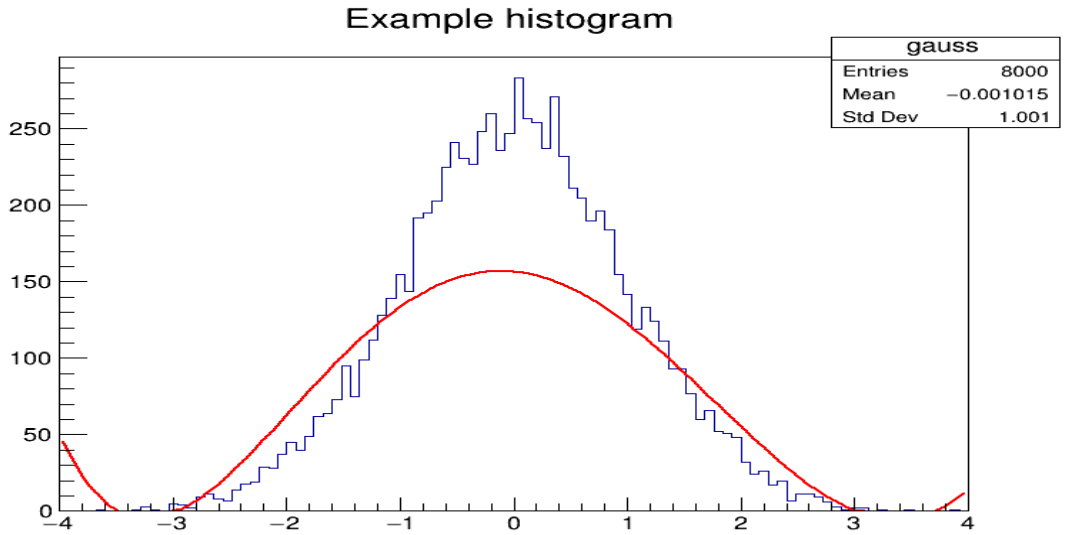


Fitting Histograms using ROOT (Part 2):

- In addition to the Gaussian function, we can use polynomial functions to fit our distributions.

```
import ROOT
# TH1F : class for histogram definition
# TCanvas : class for Plotting
# Fit : function for fitting

histogram = ROOT.TH1F("gauss","Example histogram",100,-4,4)
histogram.FillRandom("gaus", 8000)
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
histogram.Fit("pol5")
histogram.Draw()
canvas.Draw()
```

Fitting Histograms using ROOT (Part 2):

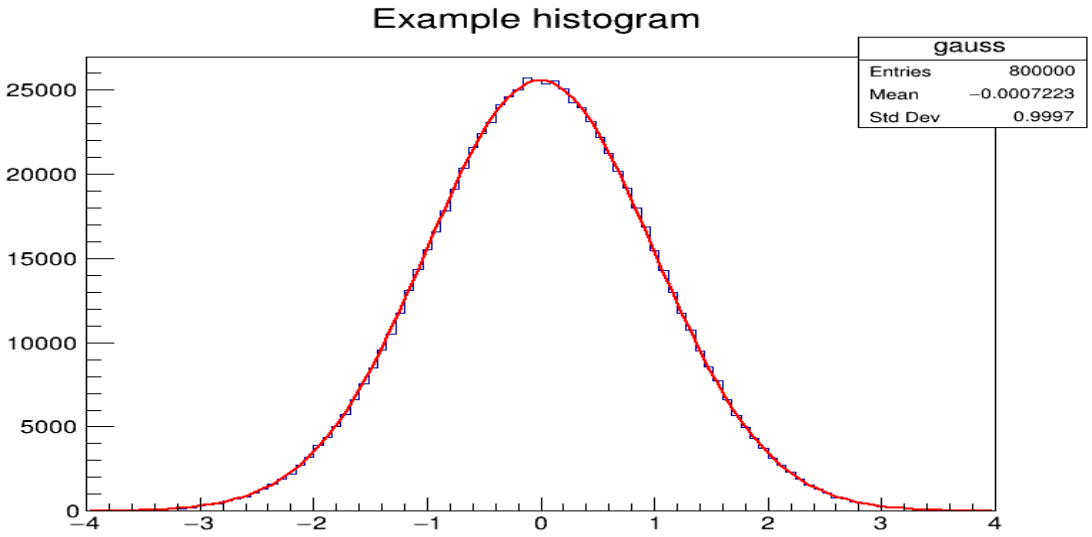


Fitting Histograms using ROOT (Part 3):

- In order to evaluate the goodness of the fit, we calculate the chi-square X-2 test between the histogram and function used for the fit.

```
import ROOT
# TH1F : class for histogram definition
# TCanvas : class for Plotting
# Fit : function for fitting

histogram = ROOT.TH1F("gauss","Example histogram",100,-4,4)
histogram.FillRandom("gaus", 800000)
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
fit    = ROOT.TF1("fit", "gaus", 100, -4,4 )

histogram.Fit(fit,"RQ")
Chi2_fit = fit.GetChisquare()
print("Chi2-fit : ", Chi2_fit)
histogram.Draw()
canvas.Draw()
```

Fitting Histograms using ROOT (Part 3):

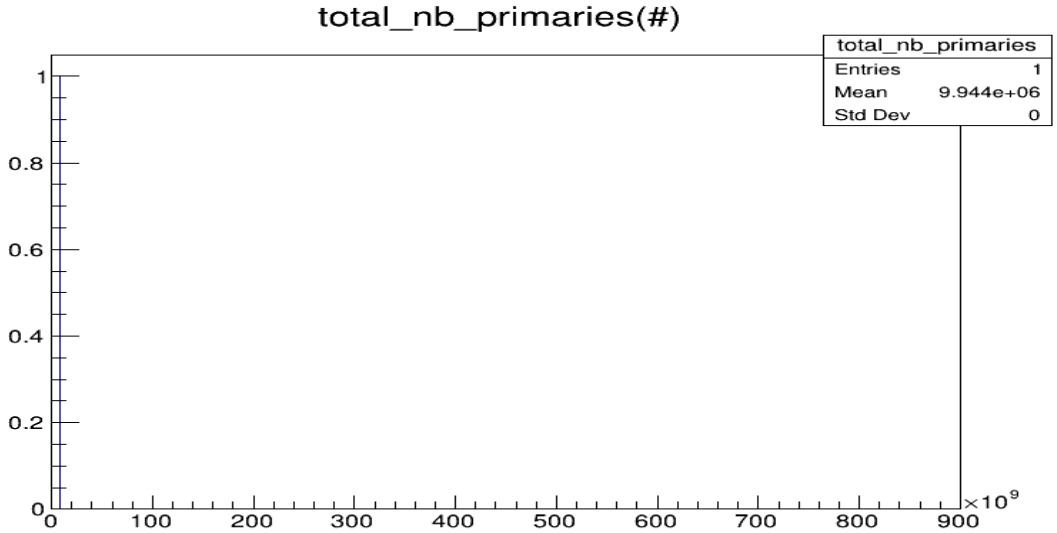


Reading ROOT file (Part 1):

- Events are stored in special files with the .root extension, either in histograms or in a tree structure.
- In each tree, we will find branches in which observables are defined.

```
import ROOT
# TFile : class for file manipulation

# Reading root file
myFile = ROOT.TFile.Open("/content/drive/MyDrive/School/Input.root")
print(myFile.ls())

# Plotting histogram from root file
Histogram = myFile.Get("total_nb_primaries")
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
Histogram.Draw()
canvas.Draw()
```

Reading ROOT file (Part 1):



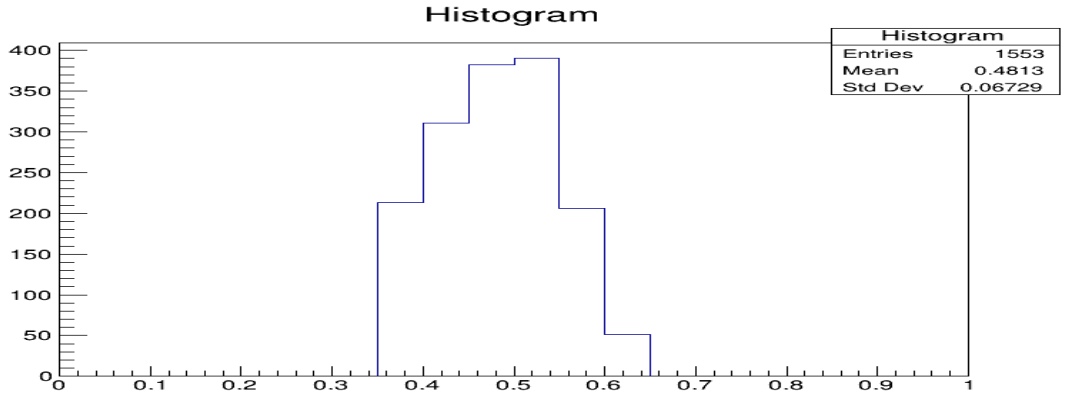
Reading ROOT file (Part 2):

- To read a tree, we can define histograms and fill them with the contents of the tree's branches.

```
import ROOT

# Reading root file
myFile    = ROOT.TFile.Open("/content/drive/MyDrive/School/Input.root")
Histogram = ROOT.TH1F("Histogram", "Histogram", 20, 0, 1)

# Reading Tree from root file
myTree = myFile.Get("delay")
for entry in myTree:
     # Now you have acess to the leaves/branches of each entry in the tree, e.g.
     Histogram.Fill(entry.energy1)

# Plotting histogram from root file
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
Histogram.Draw()
canvas.Draw()
```

Reading ROOT file (Part 2):

- To read a tree, we can define histograms and fill them with the contents of the tree's branches.



Reading ROOT file (Part 3):

- Using ROOT, we can also define two-dimensional histograms.
- We can fill a two-dimensional histogram with the contents of two different branches.

```
import ROOT

# Reading root file
myFile    = ROOT.TFile.Open("/content/drive/MyDrive/School/Input.root")
Histogram = ROOT.TH2F("Histogram", "Histogram", 20, 0.35, 0.45, 20, 0, 70)

# Reading Tree from root file
myTree = myFile.Get("delay")
for entry in myTree:
     # Now you have acess to the leaves/branches of each entry in the tree, e.g.
     Histogram.Fill(entry.energy1, entry.time1)

# Plotting histogram from root file
canvas = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
Histogram.Draw("colz")
canvas.Draw()
```

Reading ROOT file (Part 3):