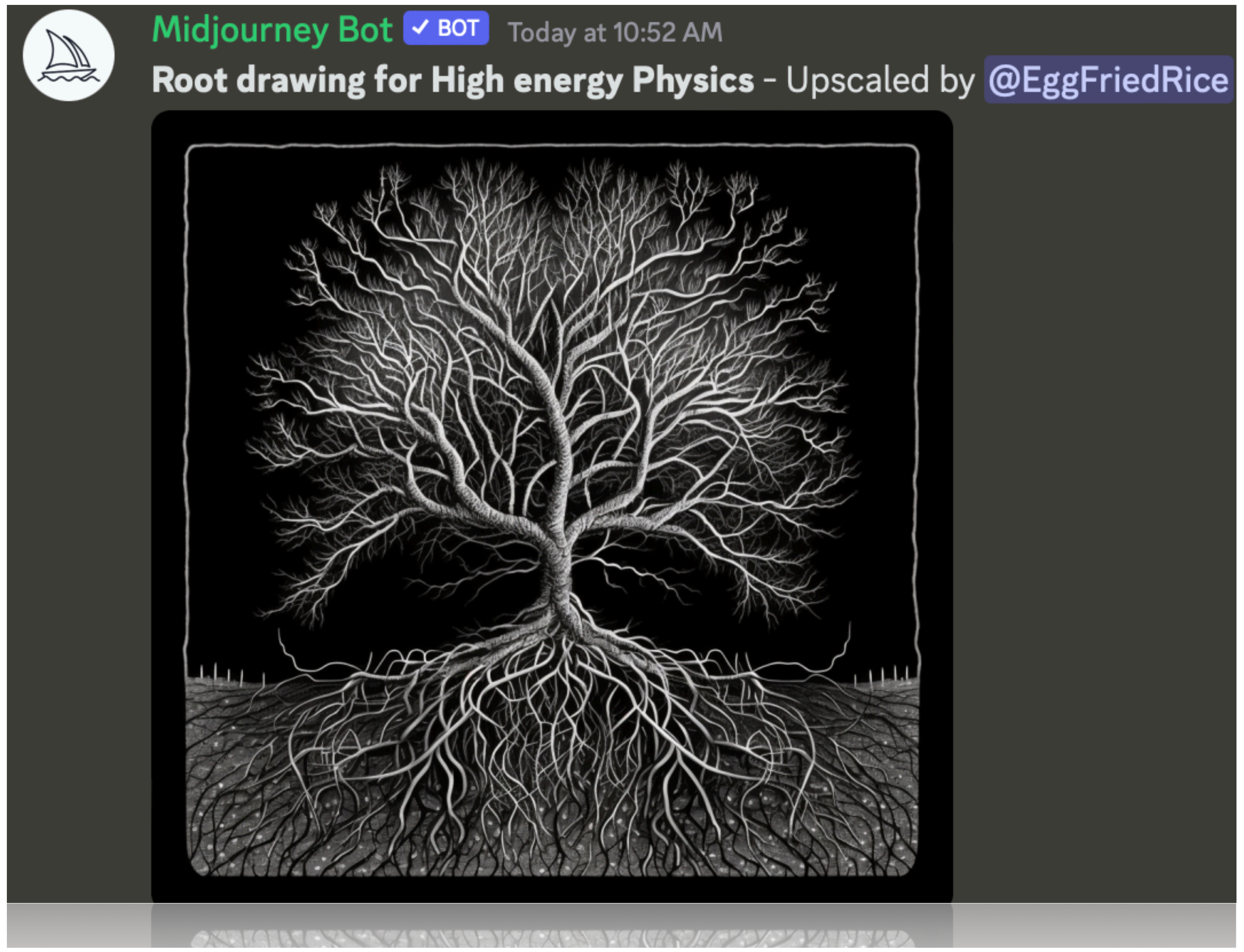
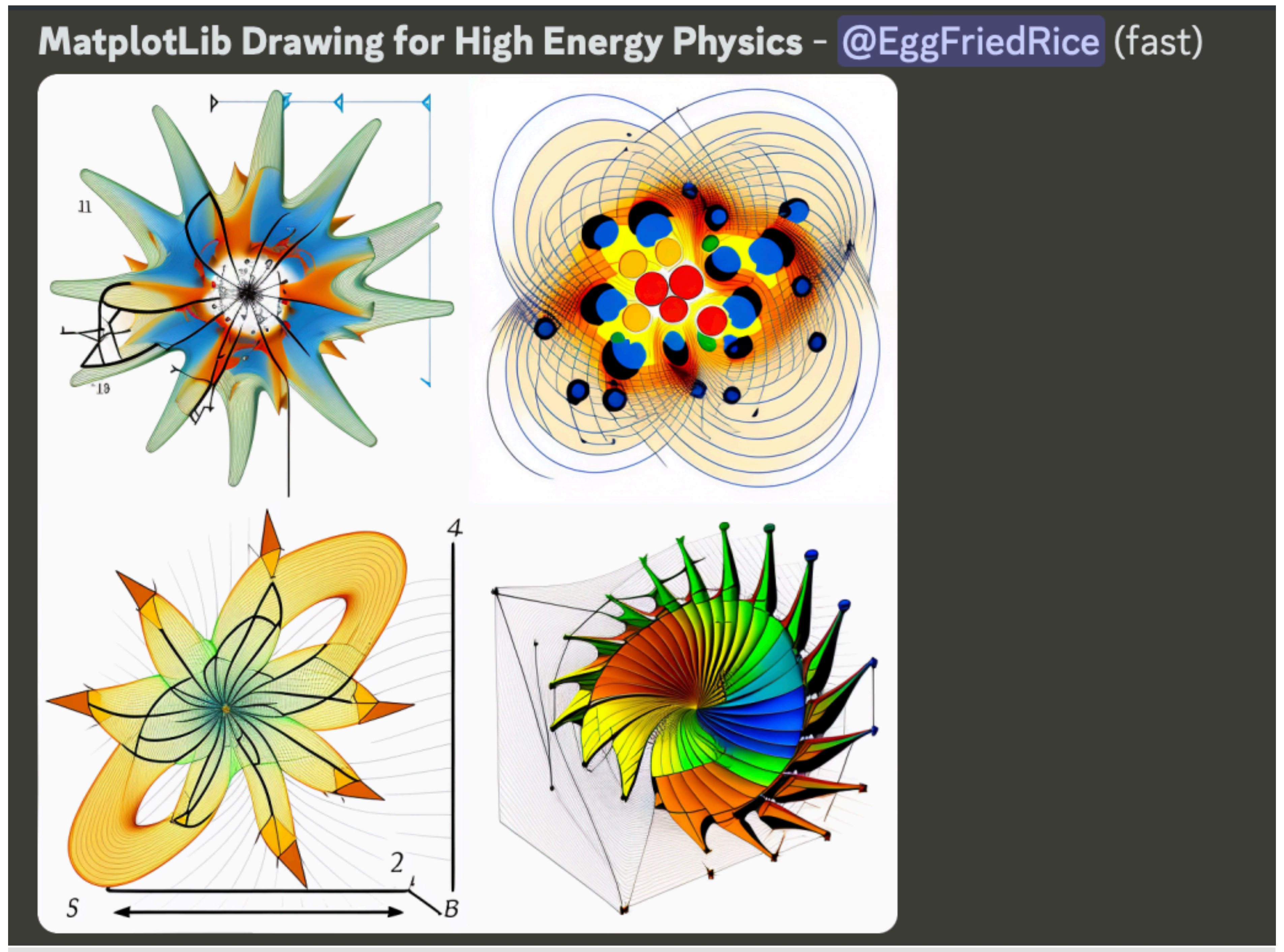

Midjourney Bot ✓ BOT Today at 10:52 AM

Root drawing for High energy Physics - Upscaled by @EggFriedRice


MatplotLib Drawing for High Energy Physics - @EggFriedRice (fast)

ROOT/MatPlotLib Tutorial

—How to draw in High Energy Physics

Yvonne Ng and Tadej Novak

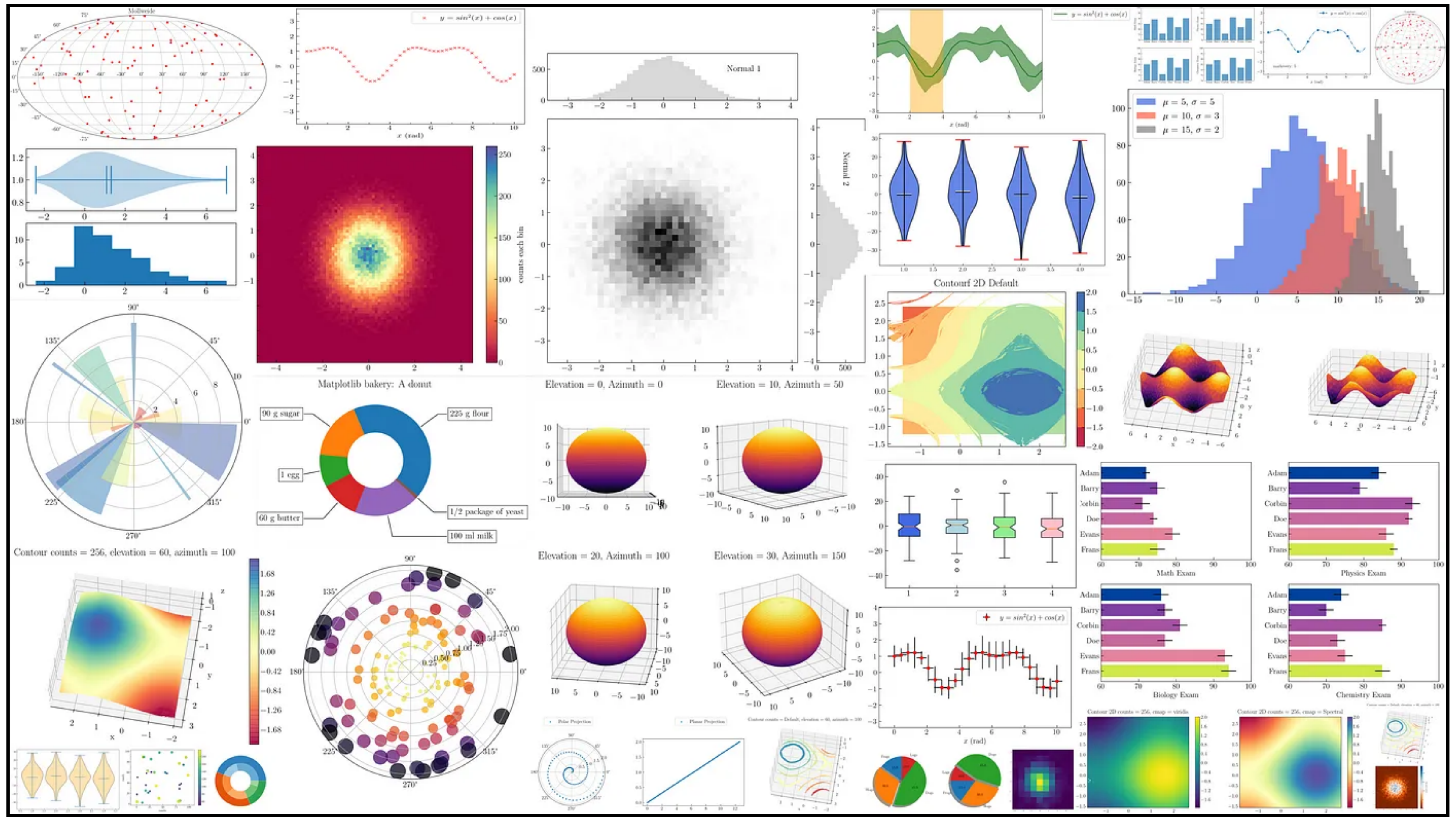
DESY Terascale School 2023

6th March 2023

Drawing in HEP

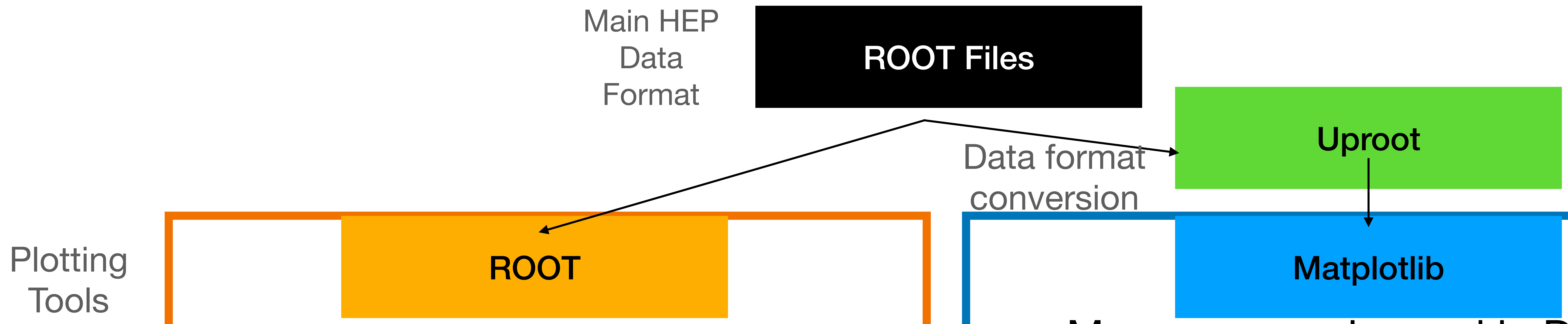
—Why is it important?

- Experimental result in form of many dimension data in HEP (Up to ~Petabytes)
- Drawing/plotting
  = Smart information reduction
  + Visual Representation of Data
- In the Drawing process:
  - Emphasis characteristic of the data
  - Analyzing features
  - Compare results
  - Aid decision making
  - A way to understand and present information


Matplotlib plotting visualization

How do I draw in HEP?

—The two main players:



ROOT

- Traditional/Official Drawing tool in HEP
- Powered by CERN
- Recently with RDataFrame support
- c++ based (python wrapper version pyROOT available)

Matplotlib

- More commonly used in Data science/ outside of HEP
- Becoming more popular within the HEP community
- Python based tool
- Can be used for HEP plotting with tools like uproot to convert root files into numpy array

I. Root Tutorial

- Setting up ROOT
- ROOT `TBrowser` for simple manipulation
- PyROOT

Slides adapted from Claudia Seitz’s tutorial in 2015

What is ROOT?

ROOT
Data Analysis Framework

- ROOT is a powerful tool in HEP analysis. Three main functions:

    1. Data storage structure/format: Tree-Branch base data structure for event based format common in HEP

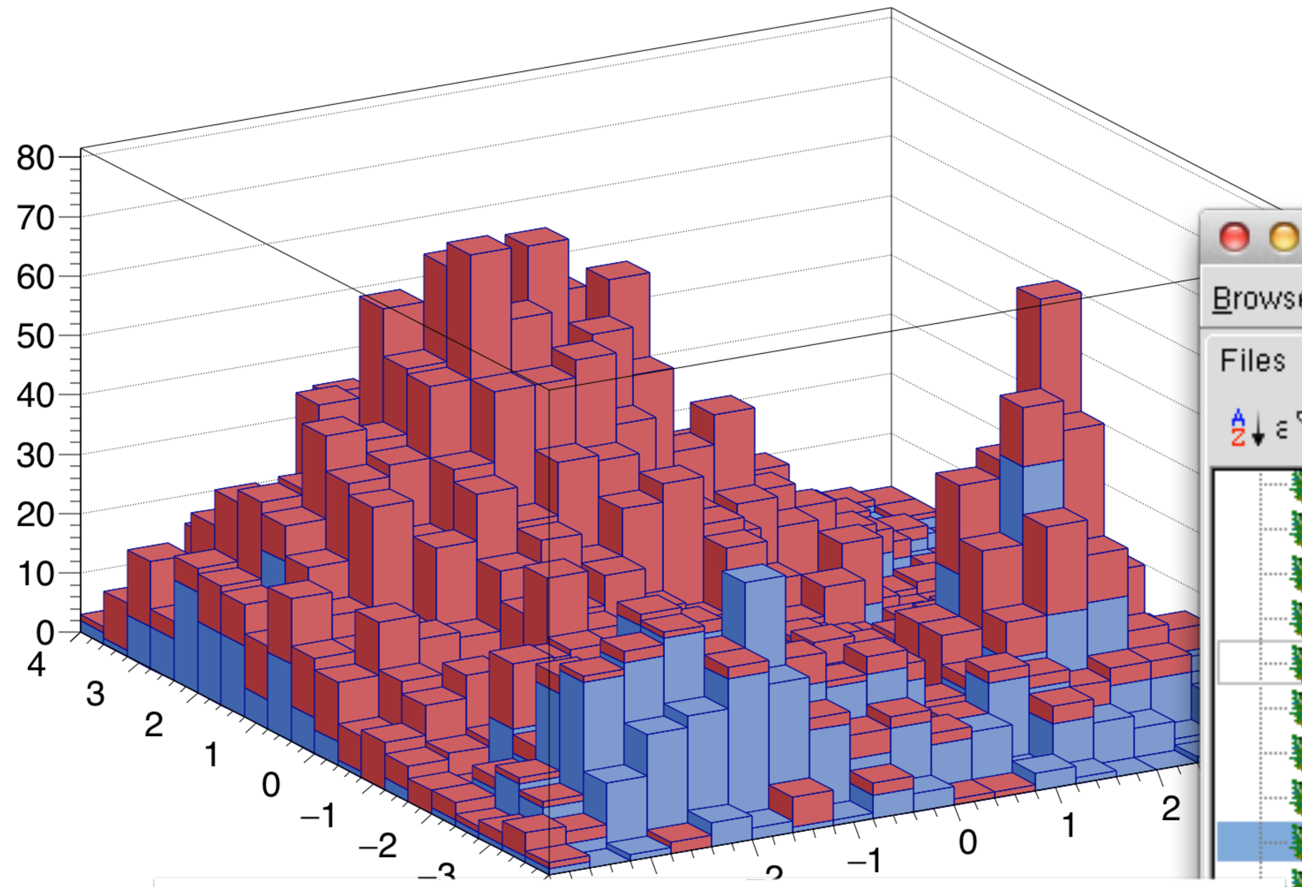
    2. Analysis

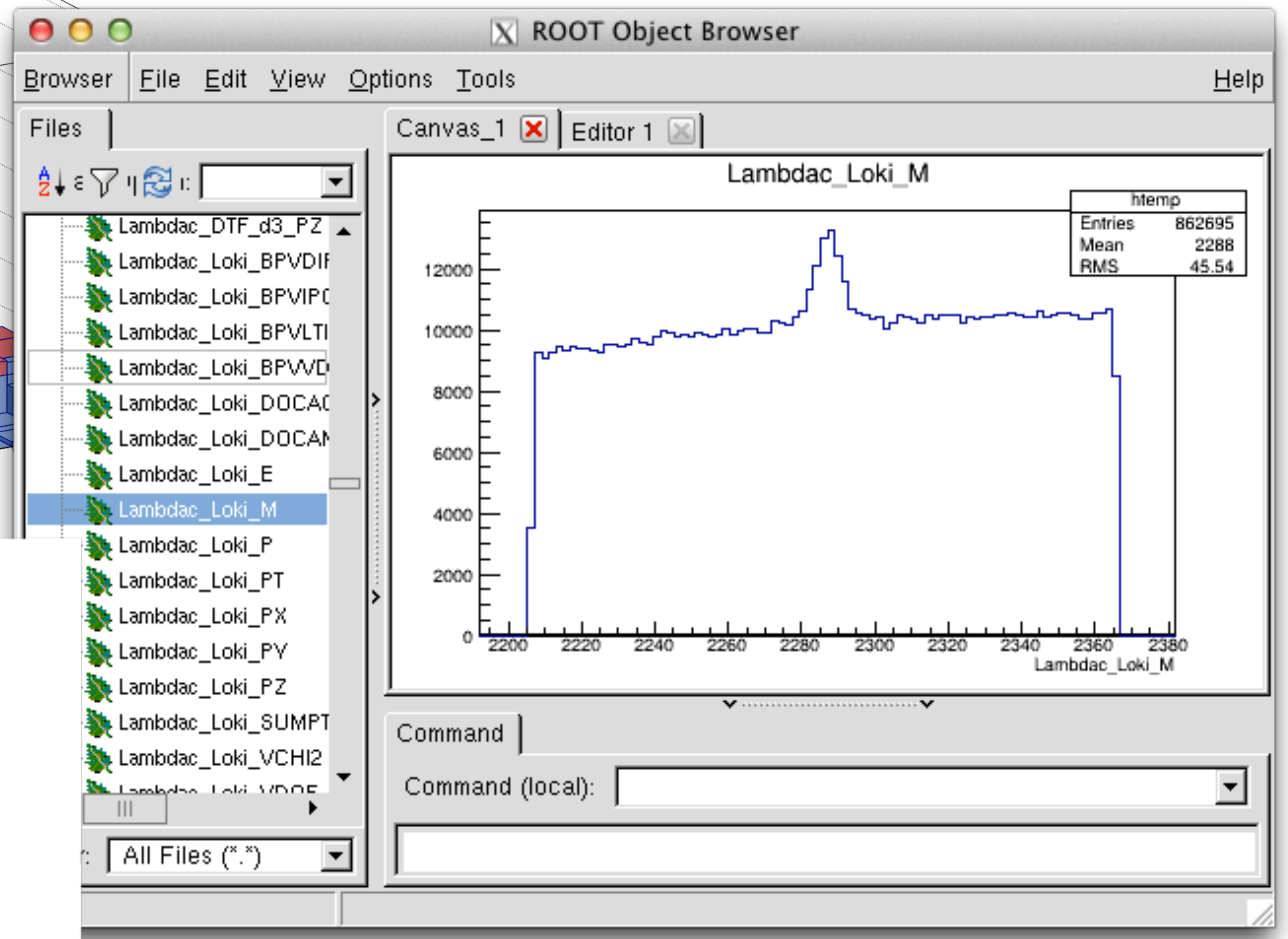
        - Math libraries

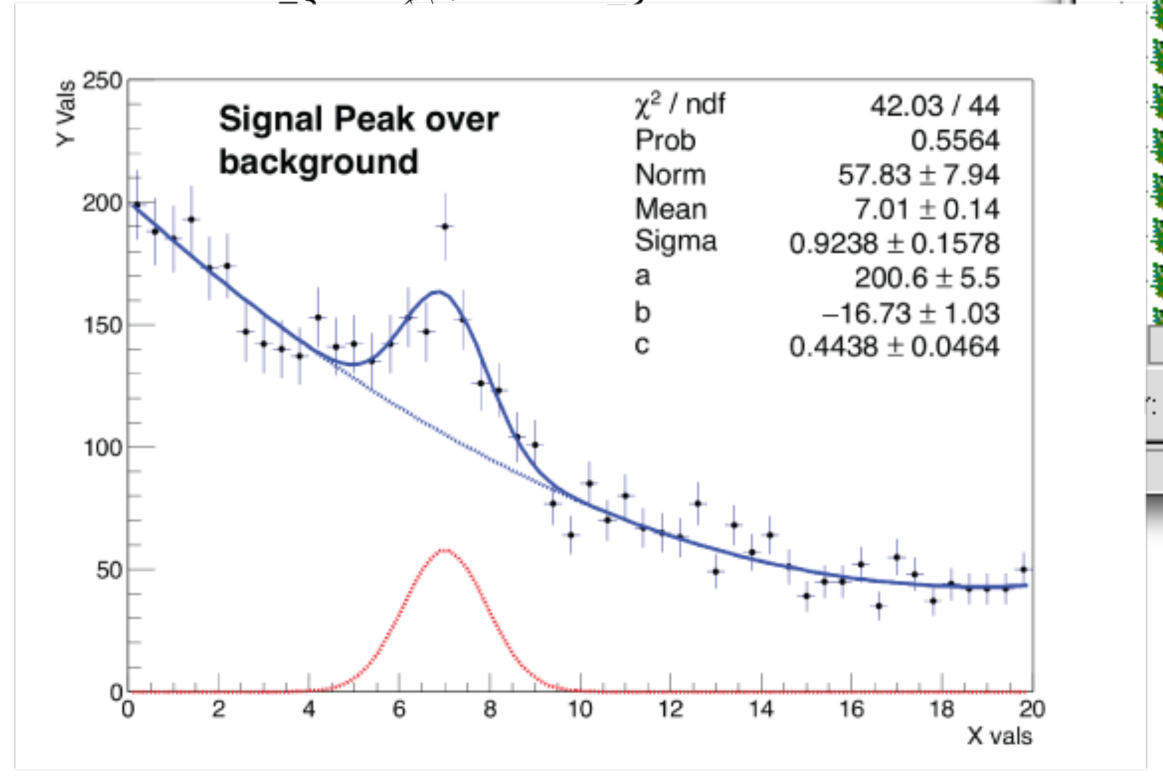
        - stats library (e.g. `RooFit/RooStats`)

        - machine learning libraries (e.g. `TMVA`)

    3. Plotting (Drawing)







ROOT and Plotting

- ROOT is a powerful tool in HEP analysis. Three main functions:

    1. **Data storage structure/format:** Tree-Branch base data structure for event based format common in HEP

    2. Analysis

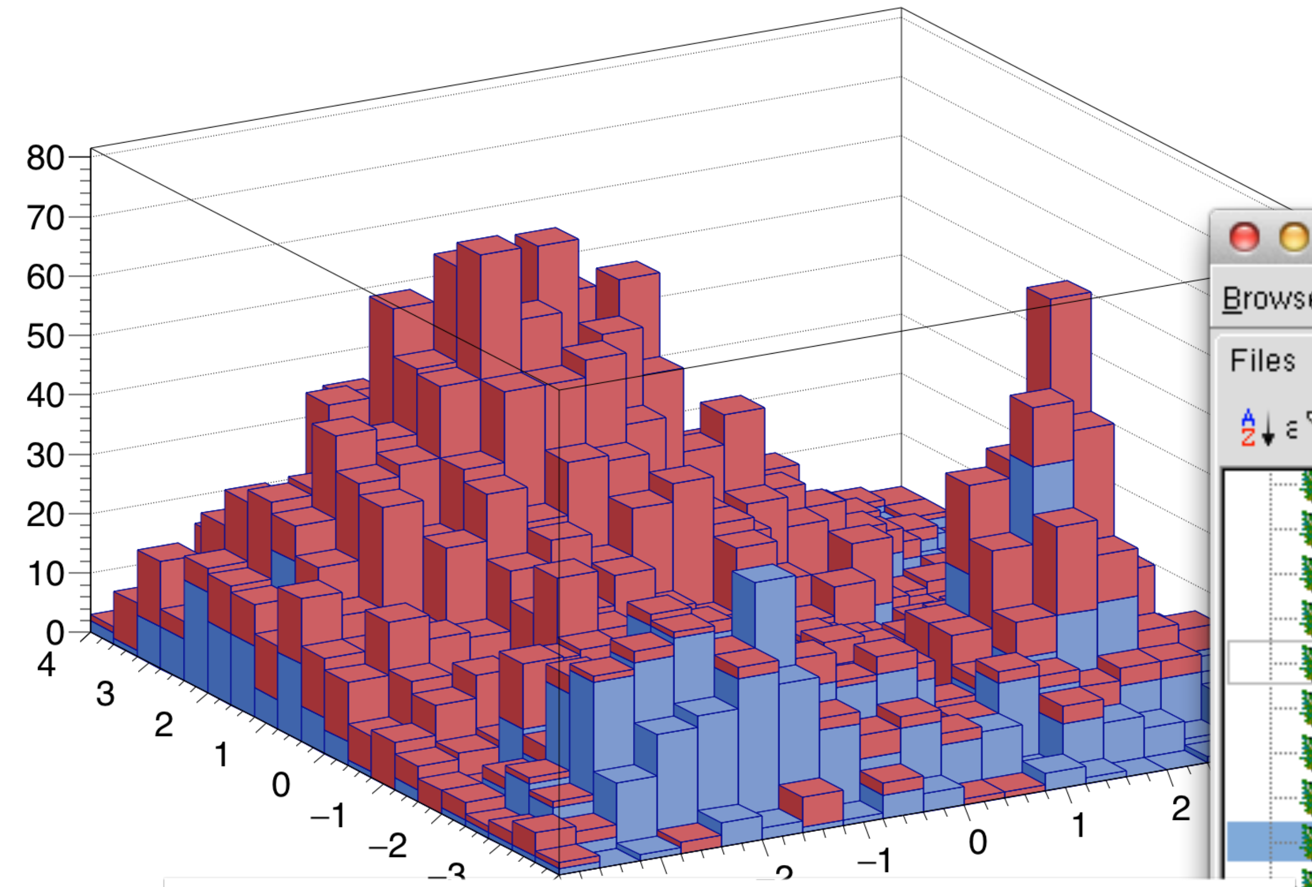
        - Math libraries

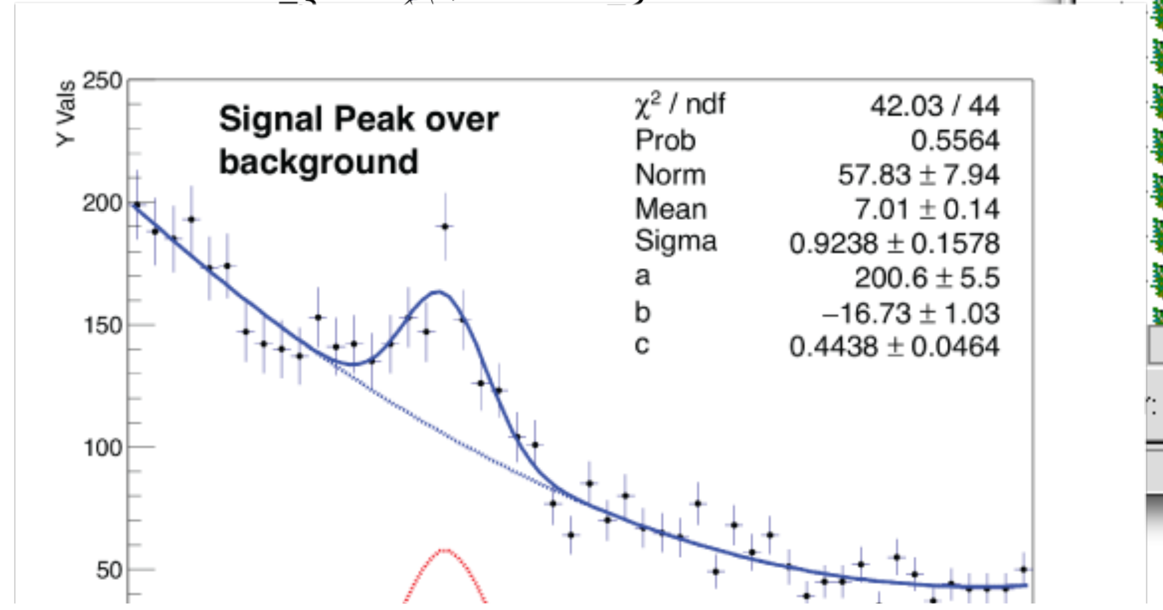
        - stats library (e.g. `RooFit/RooStats`)

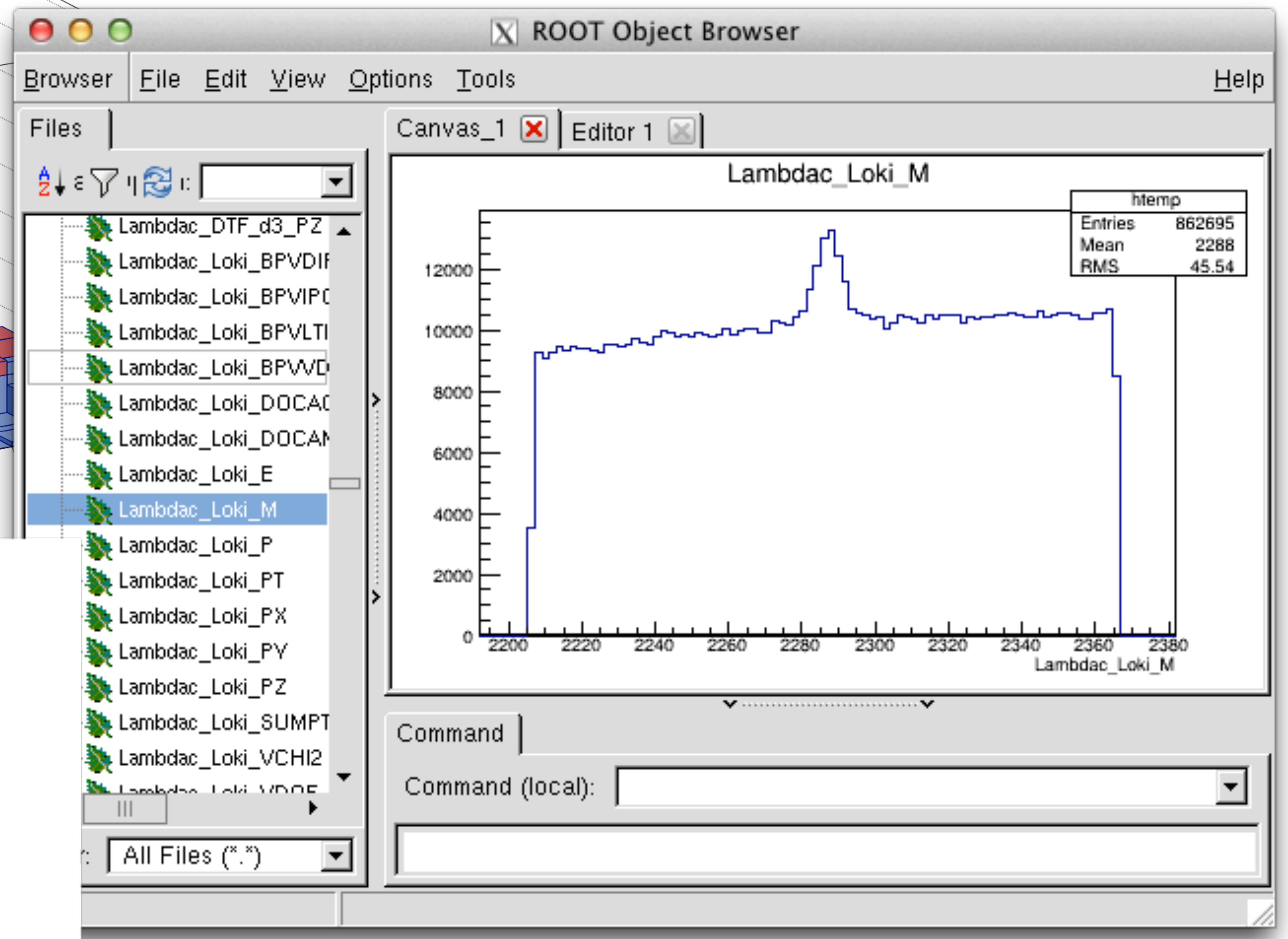
        - machine learning libraries (e.g. `TMVA`)

    3. **Plotting (Drawing)**

ROOT
Data Analysis Framework





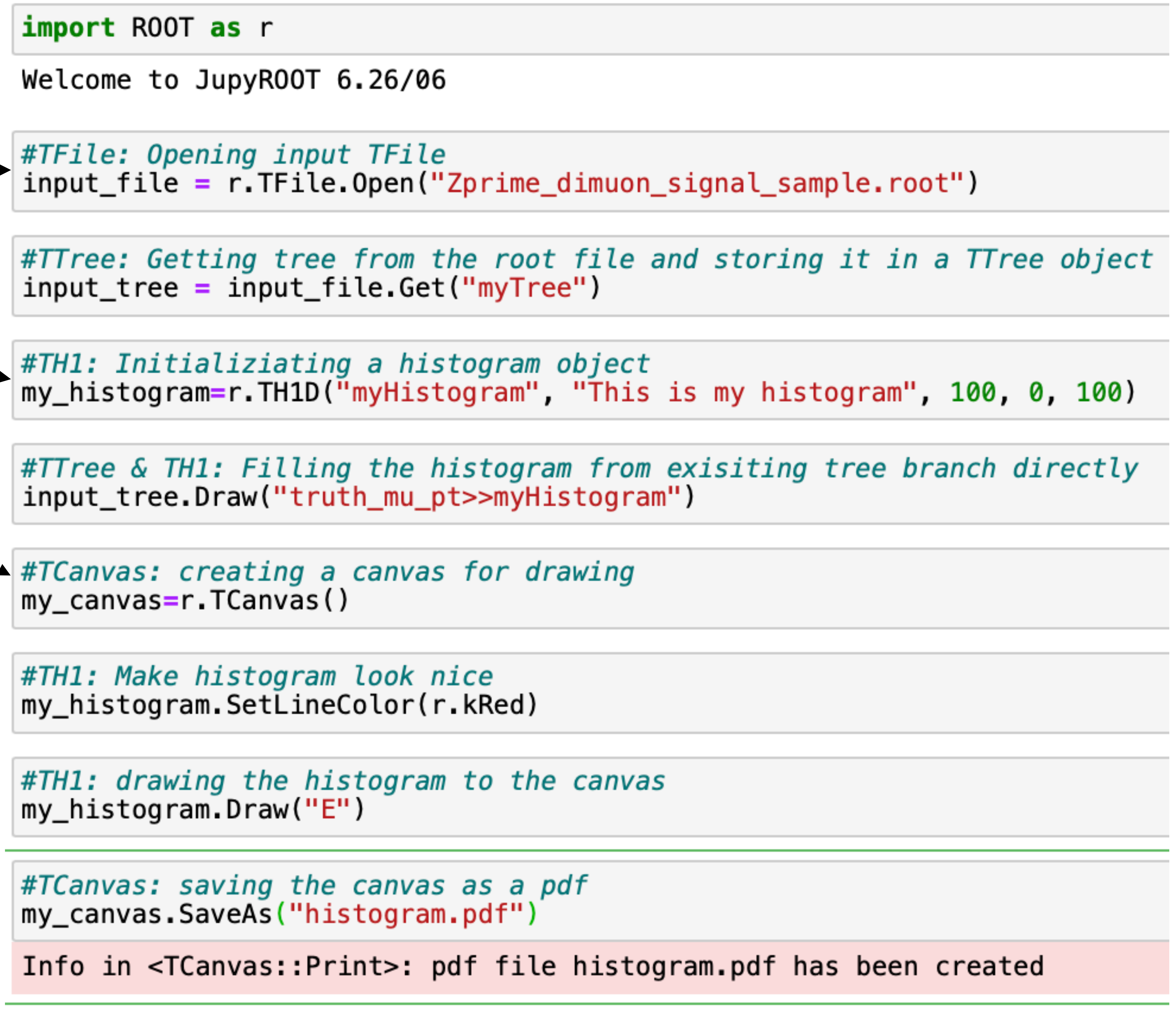


ROOT is a c++ based tool, this tutorial will focus on the python wrapper (PyROOT)

ROOT as a object based tool

- **`TObject`:** base class for all ROOT objects
- **`TFile`**: class for reading/writing root files
- **`TTree`**: basic storage format in ROOT
- **`TH1`**: base class for1-,2-,3-D Histograms
- **`TCanvas`**: class for graphical display
- **`TStyle`**: class for style of histograms, axis, title, markers, etc...
- **`TGraph`**: class of graphic object based on x and y-arrays
- **`TF1`**: base class for functions

Complete list: https://root.cern/doc/master/classes.html



```
import ROOT as r
```

```
Welcome to JupyROOT 6.26/06
```

```
#TFile: Opening input TFile
input_file = r.TFile.Open("Zprime_dimuon_signal_sample.root")
```

```
#TTree: Getting tree from the root file and storing it in a TTree object
input_tree = input_file.Get("myTree")
```

```
#TH1: Initializiating a histogram object
my_histogram=r.TH1D("myHistogram", "This is my histogram", 100, 0, 100)
```

```
#TTree & TH1: Filling the histogram from exisiting tree branch directly
input_tree.Draw("truth_mu_pt>>myHistogram")
```

```
#TCanvas: creating a canvas for drawing
my_canvas=r.TCanvas()
```

```
#TH1: Make histogram look nice
my_histogram.SetLineColor(r.kRed)
```

```
#TH1: drawing the histogram to the canvas
my_histogram.Draw("E")
```

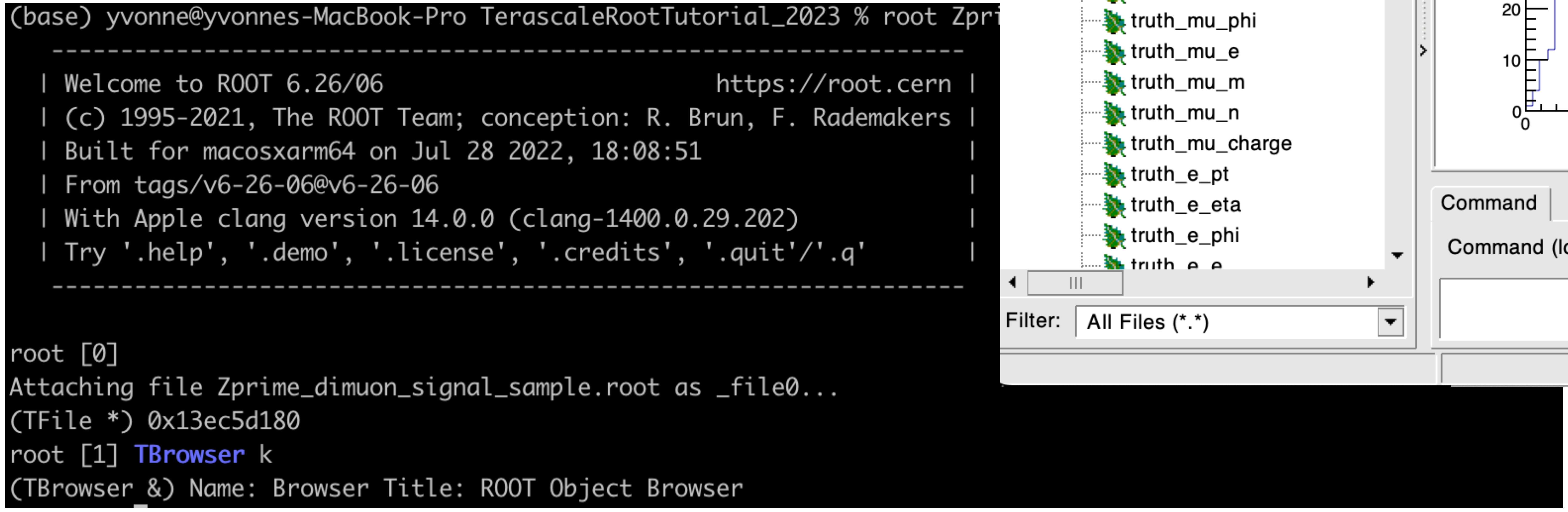
```
#TCanvas: saving the canvas as a pdf
my_canvas.SaveAs("histogram.pdf")
```

```
Info in <TCanvas::Print>: pdf file histogram.pdf has been created
```

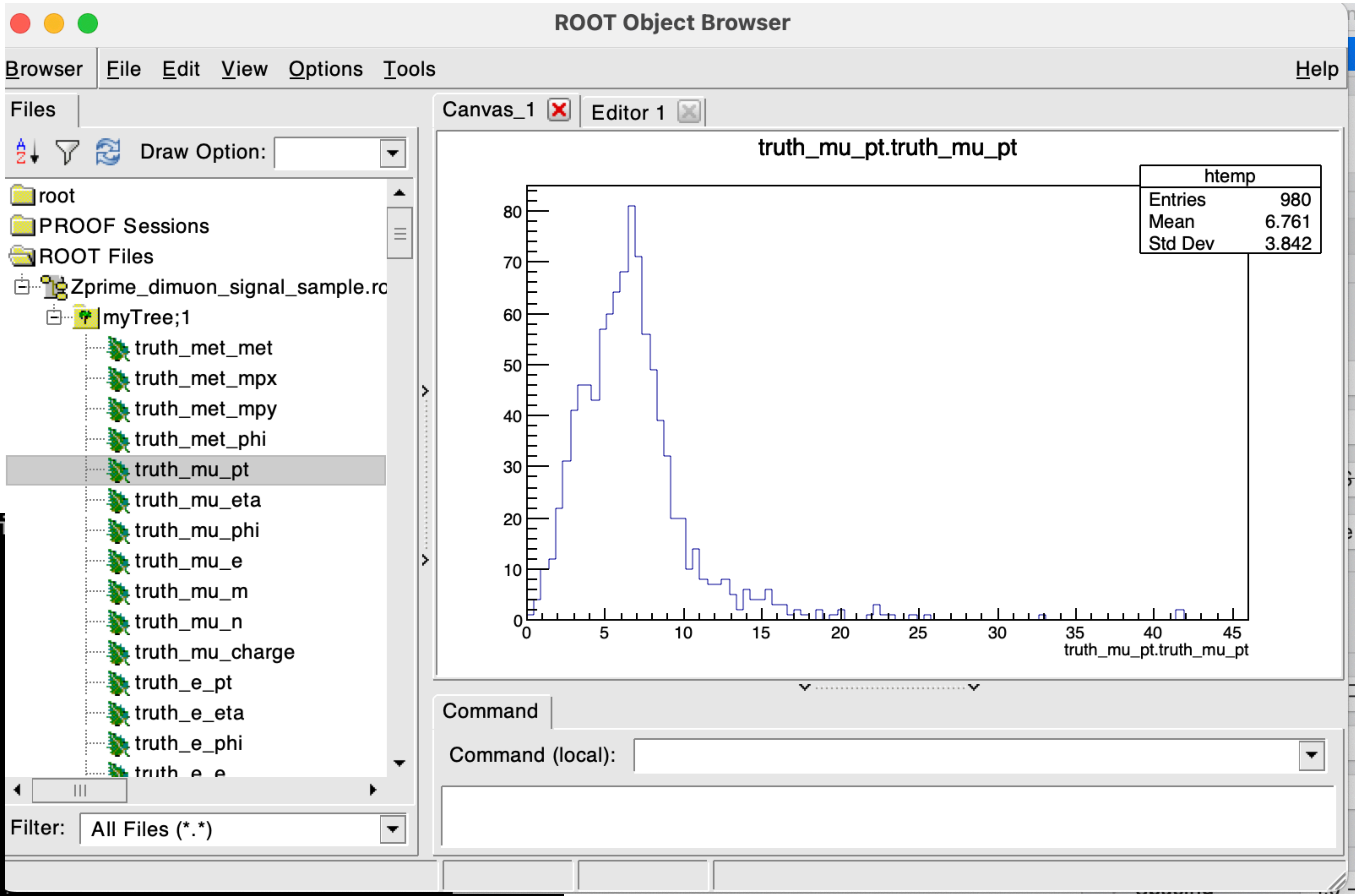
TBrowser

—Drawing option manipulation with GUI/command line

- `TBrowser` can be used for ROOT file inspection. Run:
  - `$ root Zmumu.root`
    (use “root —web=off" for later versions)
  - `$ TBrowser newBrowser`



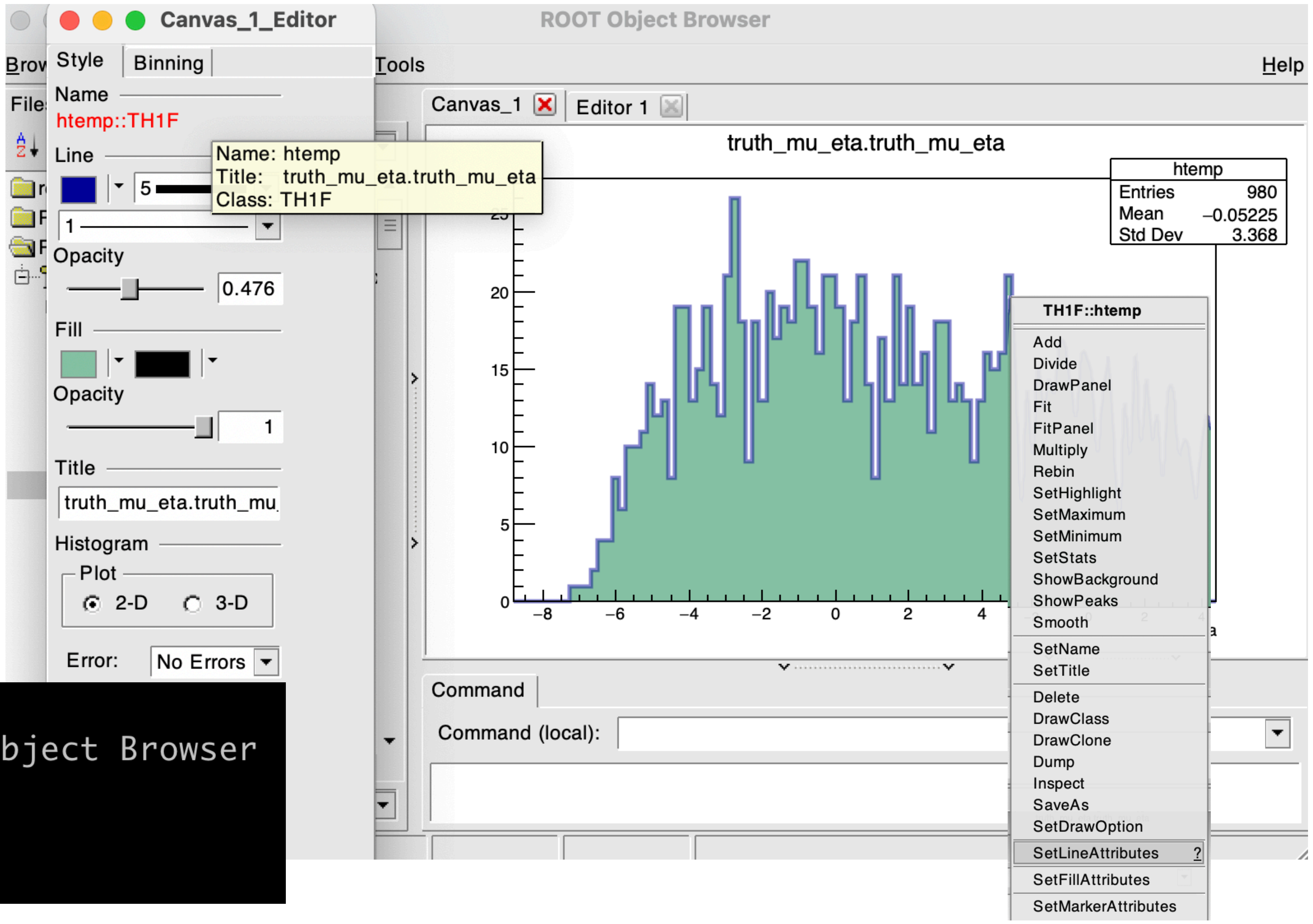
```
(base) yvonne@yvonnes-MacBook-Pro TerascaleRootTutorial_2023 % root Zpri
   ------------------------------------------------------------------
  | Welcome to ROOT 6.26/06                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for macosxarm64 on Jul 28 2022, 18:08:51                   |
  | From tags/v6-26-06@v6-26-06                                      |
  | With Apple clang version 14.0.0 (clang-1400.0.29.202)            |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0]
Attaching file Zprime_dimuon_signal_sample.root as _file0...
(TFile *) 0x13ec5d180
root [1] TBrowser k
(TBrowser &) Name: Browser Title: ROOT Object Browser
```



TBrowser Drawing Manipulation

—Example: Changing histogram outlook

- **GUI**:
  Right click on the histogram
  - Choose **`SetLineAttribute`**
    - Change line color, set width, set Fill color
- **C**


Canvas_1_Editor

Name: htemp::TH1F

Line: 5, 1

Opacity: 0.476

Fill

Opacity: 1

Title: truth_mu_eta.truth_mu

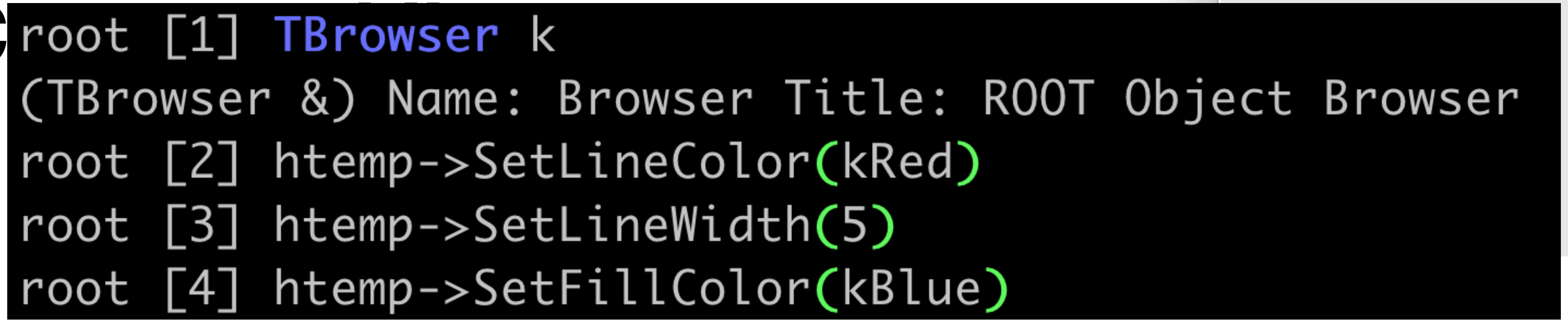
Histogram — Plot: [✓ 2-D] 3-D

Error: No Errors

Name: htemp
Title: truth_mu_eta.truth_mu_eta
Class: TH1F

htemp	
Entries	980
Mean	−0.05225
Std Dev	3.368

Command (local):


```
root [1] TBrowser k
(TBrowser &) Name: Browser Title: ROOT Object Browser
root [2] htemp->SetLineColor(kRed)
root [3] htemp->SetLineWidth(5)
root [4] htemp->SetFillColor(kBlue)
```

Same effect! `TBrowser` GUI is a good choice for quick plot making.

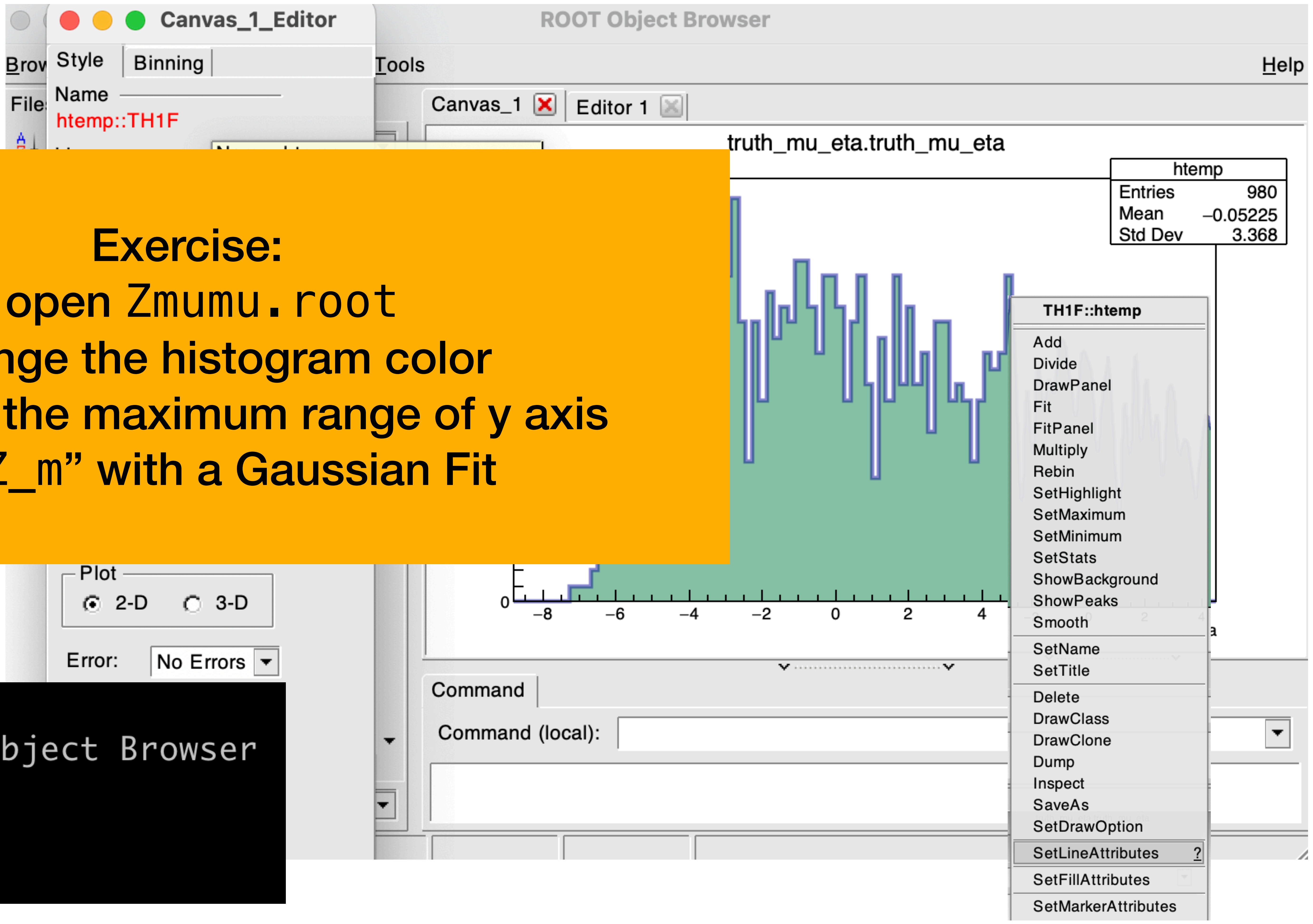
TBrowser Drawing Manipulation

—Example: Changing histogram outlook

- **GUI**:
  Right click on the
  - Choose **`SetLineAttri`**
    - Change line c
      width, set Fill color
- **C**

Exercise:
- open `Zmumu.root`
- change the histogram color
- changing the maximum range of y axis
- fit "`Z_m`" with a Gaussian Fit

```
root [1] TBrowser k
(TBrowser &) Name: Browser Title: ROOT Object Browser
root [2] htemp->SetLineColor(kRed)
root [3] htemp->SetLineWidth(5)
root [4] htemp->SetFillColor(kBlue)
```



Same effect! `TBrowser` GUI is a good choice for quick plot making.

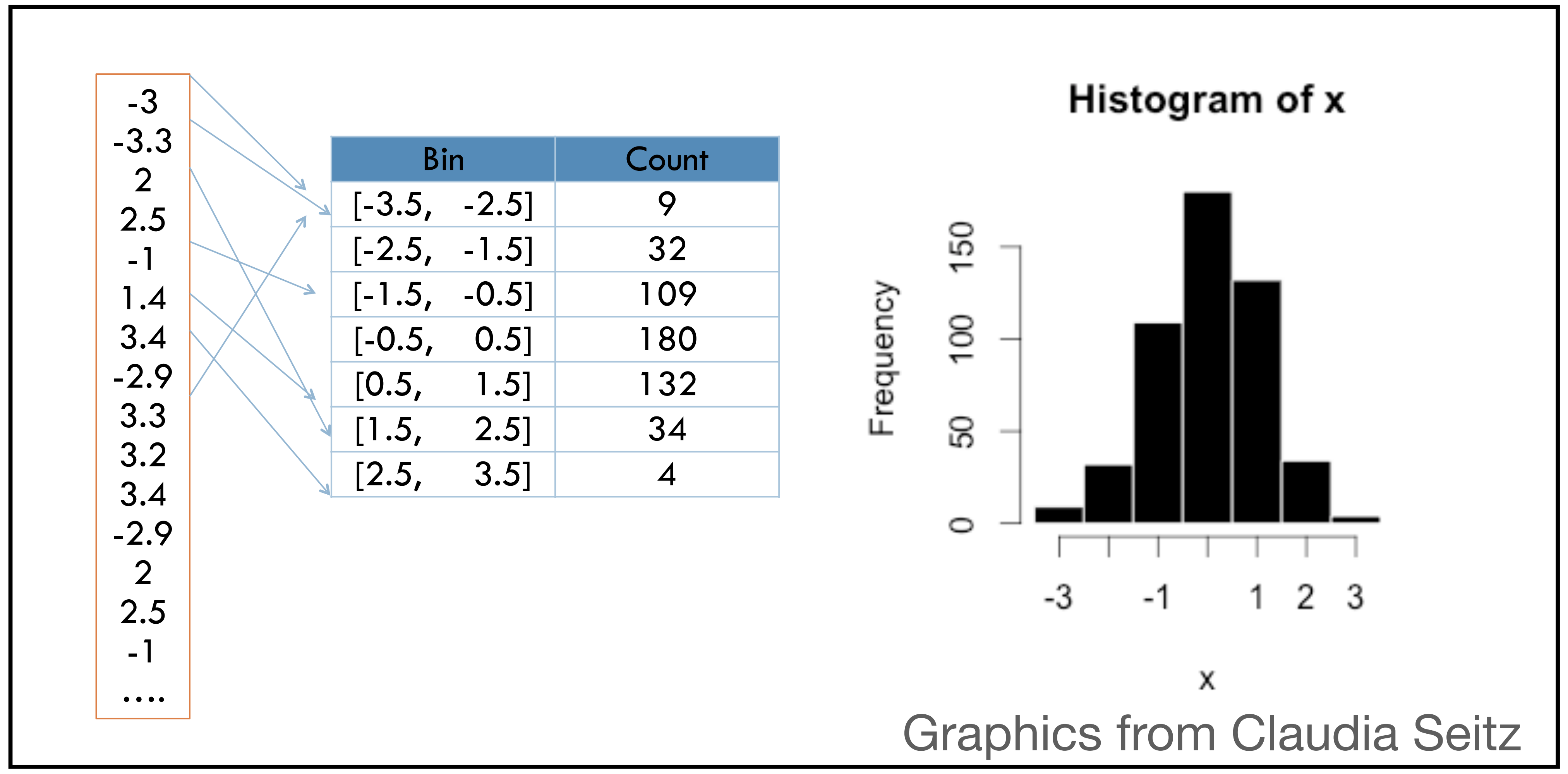
Classes: TFile and TTree

- `TFile` is basic I/O format in ROOT.
- Open an existing file (read only):
    - `file = TFile.Open("Zmumu.root")` (or `TFile::Open("Zmumu.root")` in C++)
- Make a new file:
    - `file = TFile("Zmumu.root", "OPTION")`
    - OPTION = “RECREATE” (replace file), “UPDATE” (append to file)
- Files can contain directories, histograms and trees (ntuples) etc. → for analysis data we usually use `TTree`.
- Tree has “entries” (e.g. collision events), each with identical data structure.
- Can contain floats, integers, or more complex objects (whole classes, vectors, etc…).

Classes: Histograms

—Plotting with ROOT

- When you are interested in the distribution/frequency of the data
- ROOT Classes : TH1 (1D histogram), TH2 (2D histogram), TH3 (3D histogram)
  - Child classes: TH1F (1D single-precision floating-point histogram), TH1D (1D double-precision) TH2I (1D integer histogram)



Bin	Count
[-3.5, -2.5]	9
[-2.5, -1.5]	32
[-1.5, -0.5]	109
[-0.5, 0.5]	180
[0.5, 1.5]	132
[1.5, 2.5]	34
[2.5, 3.5]	4

Graphics from Claudia Seitz

(some) histogram drawing options:

Option	Explanation
"E"	Draw error bars.
"HIST"	When an histogram has errors it is visualized by default with error bars. To visualize it without errors use the option "HIST".
"SAME"	Superimpose on previous picture in the same pad.
"TEXT"	Draw bin contents as text.
Options just for TH1	
"C"	Draw a smooth Curve through the histogram bins.
"E0"	Draw error bars. Markers are drawn for bins with 0 contents.
"E1"	Draw error bars with perpendicular lines at the edges.
"E2"	Draw error bars with rectangles.
"E3"	Draw a fill area through the end points of the vertical error bars.
"E4"	Draw a smoothed filled area through the end points of the error bars.

Histogram: Sorting information into bins
Creating information about the frequency/distribution of the data
Correlation information between different variables in 2D or higher dimensions

Interactive tutorial

- Will present the next part interactively
- Also available at https://www.desy.de/~tadej/tutorial/
    - File that we will use: https://www.desy.de/~tadej/tutorial/Zmumu.root
    - `$ wget https://www.desy.de/~tadej/tutorial/Zmumu.root`
- Simple steps to get the code running:
    - `$ ssh school01@naf-school01.desy.de`
    - make the python file e.g. `ExerciseHist.py`, later you execute with `python3 ExerciseHist.py`
    - to copy files: `$ scp school01@naf-school01.desy.de:/path/to/file /local/path/to/file`
- All ROOT macros should start with

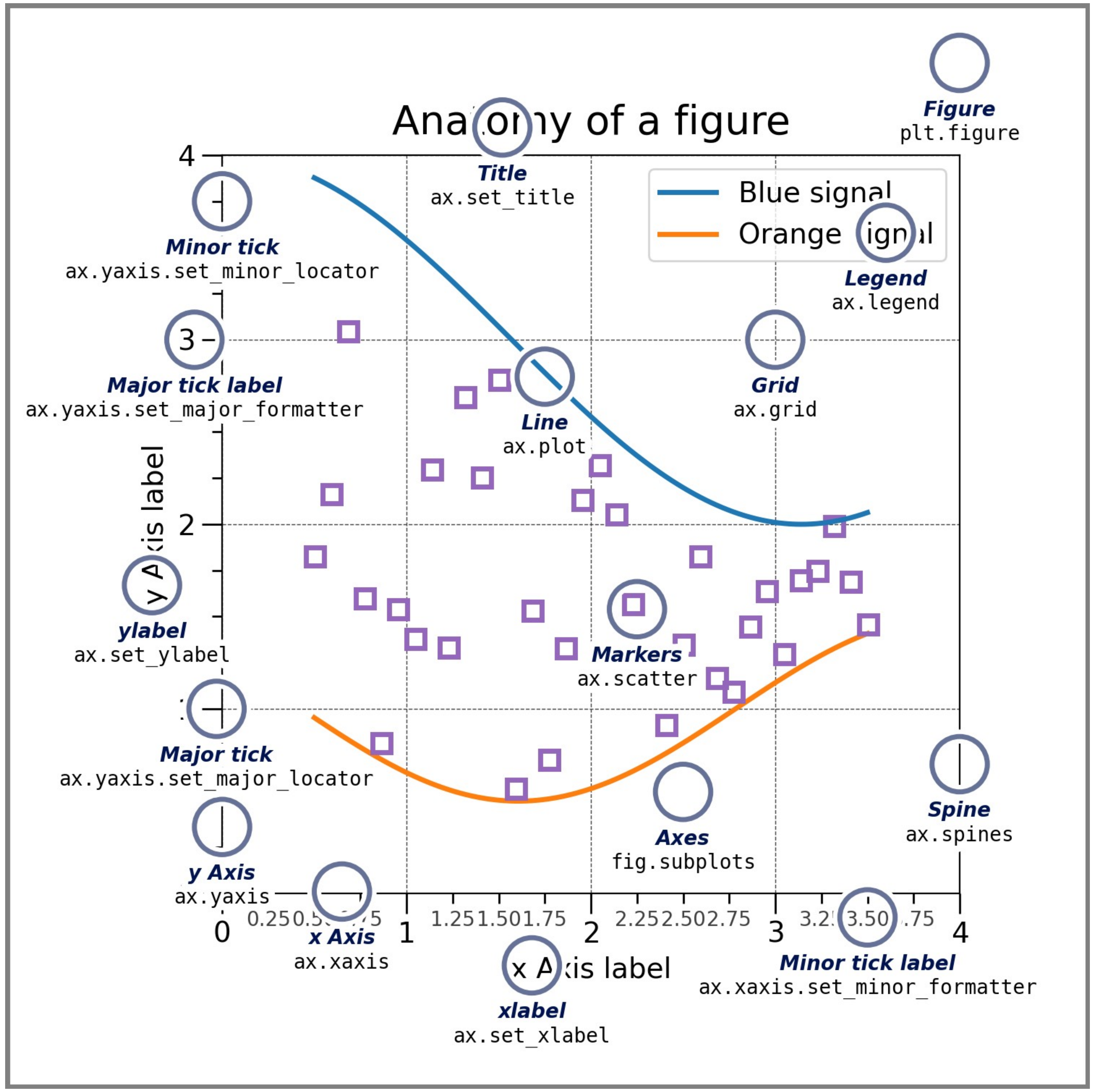
```
import ROOT
ROOT.gROOT.SetBatch(True)
```

II. Matplotlib Tutorial

- Uproot: Changing root files to Numpy arrays
- Drawing with Matplotlib

What is Matplotlib?

- Plotting tool used widely in data science and other fields outside of HEP
- With recent development of supporting library, it has become a popular tools to be used in HEP
  - **Uproot**: Converts ROOT input files to awkward arrays, panda dataframes, python array, numpy arrays, for matplotlib uses
  - **mplhep**: matplotlib to ROOT styling
  - **Scipy**: contain function/stats for mathematics manipulations
- Advantages: Large support community+guides online, large set of styling options
- Disadvantages: Not designed for HEP uses



Uproot

—Converting root files into numpy arrays

PyROOT Reading a File (TFile &TTree)

```
>>> import ROOT
>>> f = TFile("basic.root")
>>> t = f.Get(“myTree”)

>>> for i_entry, entry in t.GetEntries():
>>>     if (i_entry<10):
>>>         print(entry[“truth_jet_pt”])
```

Uproot Reading a File

```
>>> import uproot
>>> d = uproot.open(“./basic.root”)
>>> t = MyUprootTFile[“myTree"]


>>> #Convert to Python Array
>>> truth_jet_pt_array = 
MyUprootTTree[“truth_jet_pt"].array()

>>> #Checking the uproot tree type
>>> print(truth_jet_pt_array)
[[], [], [], [], [], [], [], [], ..., [], 
[], [], [57.3, 20.5], [], [], [], []]
>>> print(type(truth_jet_pt_array))
<class 'awkward.highlevel.Array'>
```

Histogram

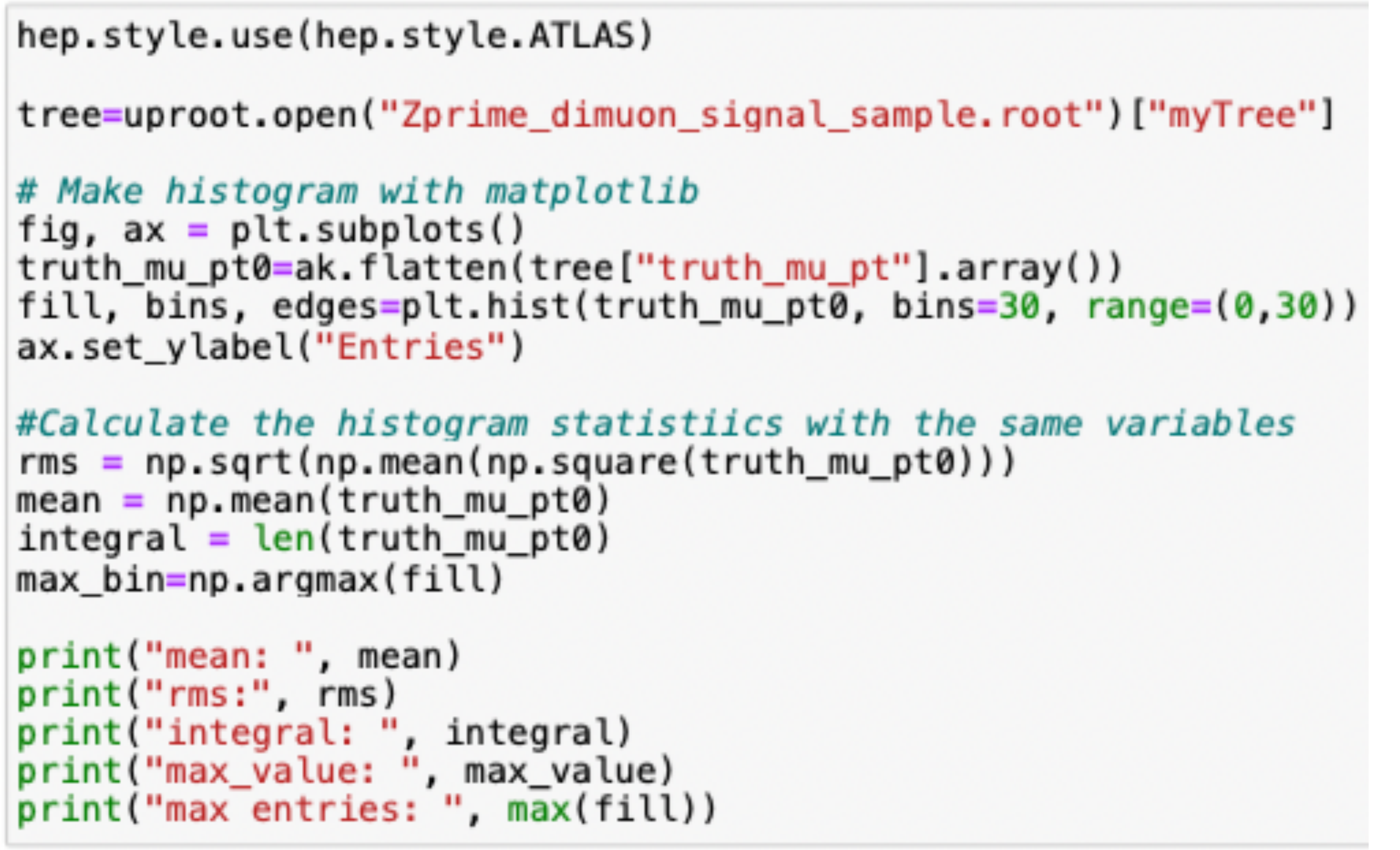
—hist in Matplotlib

Write a python macro ExerciseHist.py

1. Create a histogram with 30 bins ranging from 0. to 30. with title/x-axis label "x"
2. Fill the histogram at the following branch: “truth_mu_pt[0]” in “MyTree”.
3. Draw the histogram.
4. Calculate the mean value and the rms and show it on the screen.

   ```
   print mean, rms
   ```
5. Calculate the integral of the histogram.
6. Identify the bin with the maximum number of entries.
7. Calculate the maximum bin content.
8. Set the y-axis label to "entries".
9. Set the line color of the histogram to red.
10. Run with

    ```
    python -i ExerciseHist.py
    ```



```
hep.style.use(hep.style.ATLAS)

tree=uproot.open("Zprime_dimuon_signal_sample.root")["myTree"]

# Make histogram with matplotlib
fig, ax = plt.subplots()
truth_mu_pt0=ak.flatten(tree["truth_mu_pt"].array())
fill, bins, edges=plt.hist(truth_mu_pt0, bins=30, range=(0,30))
ax.set_ylabel("Entries")

#Calculate the histogram statistiics with the same variables
rms = np.sqrt(np.mean(np.square(truth_mu_pt0)))
mean = np.mean(truth_mu_pt0)
integral = len(truth_mu_pt0)
max_bin=np.argmax(fill)

print("mean: ", mean)
print("rms:", rms)
print("integral: ", integral)
print("max_value: ", max_value)
print("max entries: ", max(fill))
```

Histogram

—hist in Matplotlib

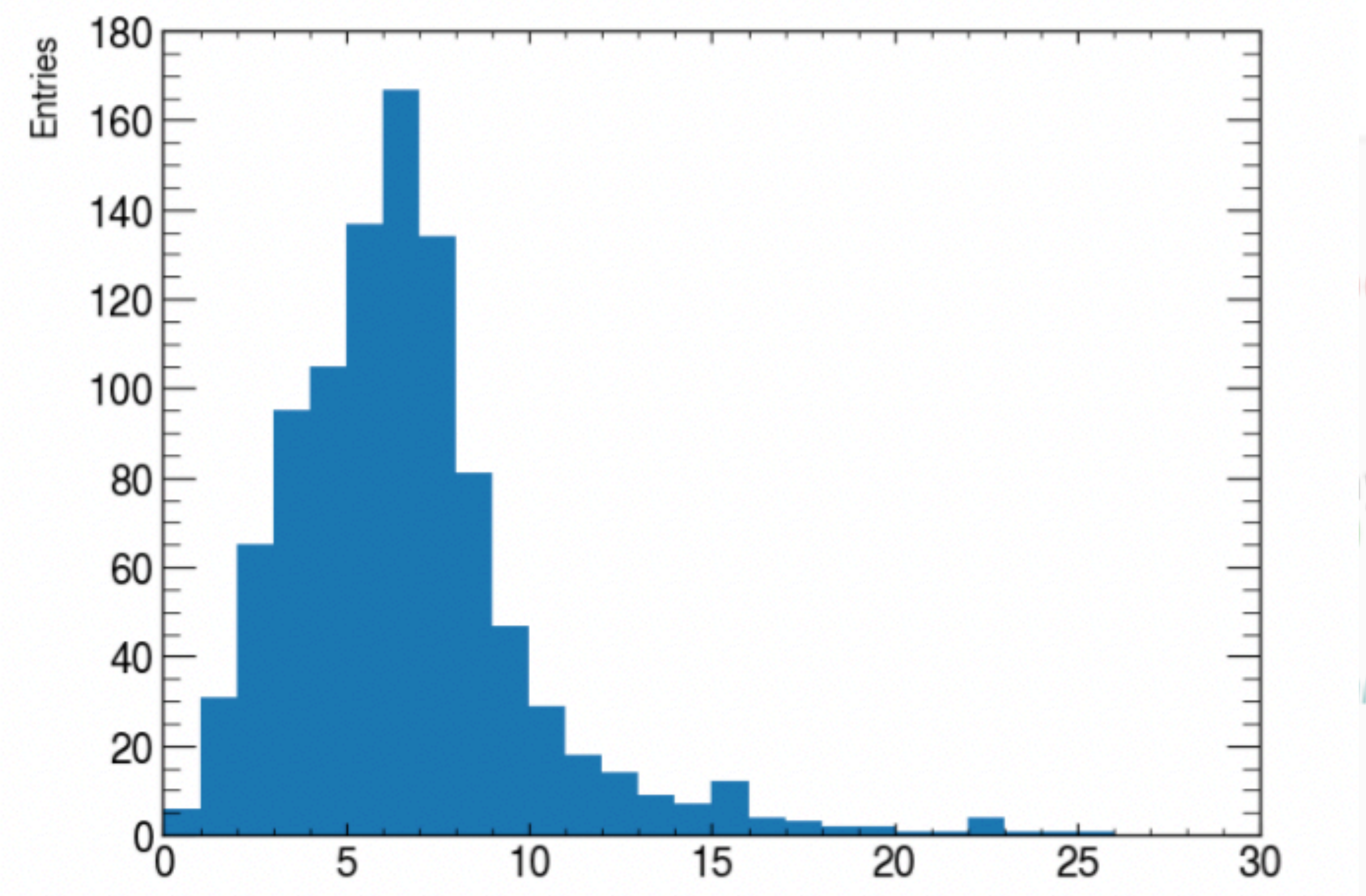
Write a python macro ExerciseHist.py

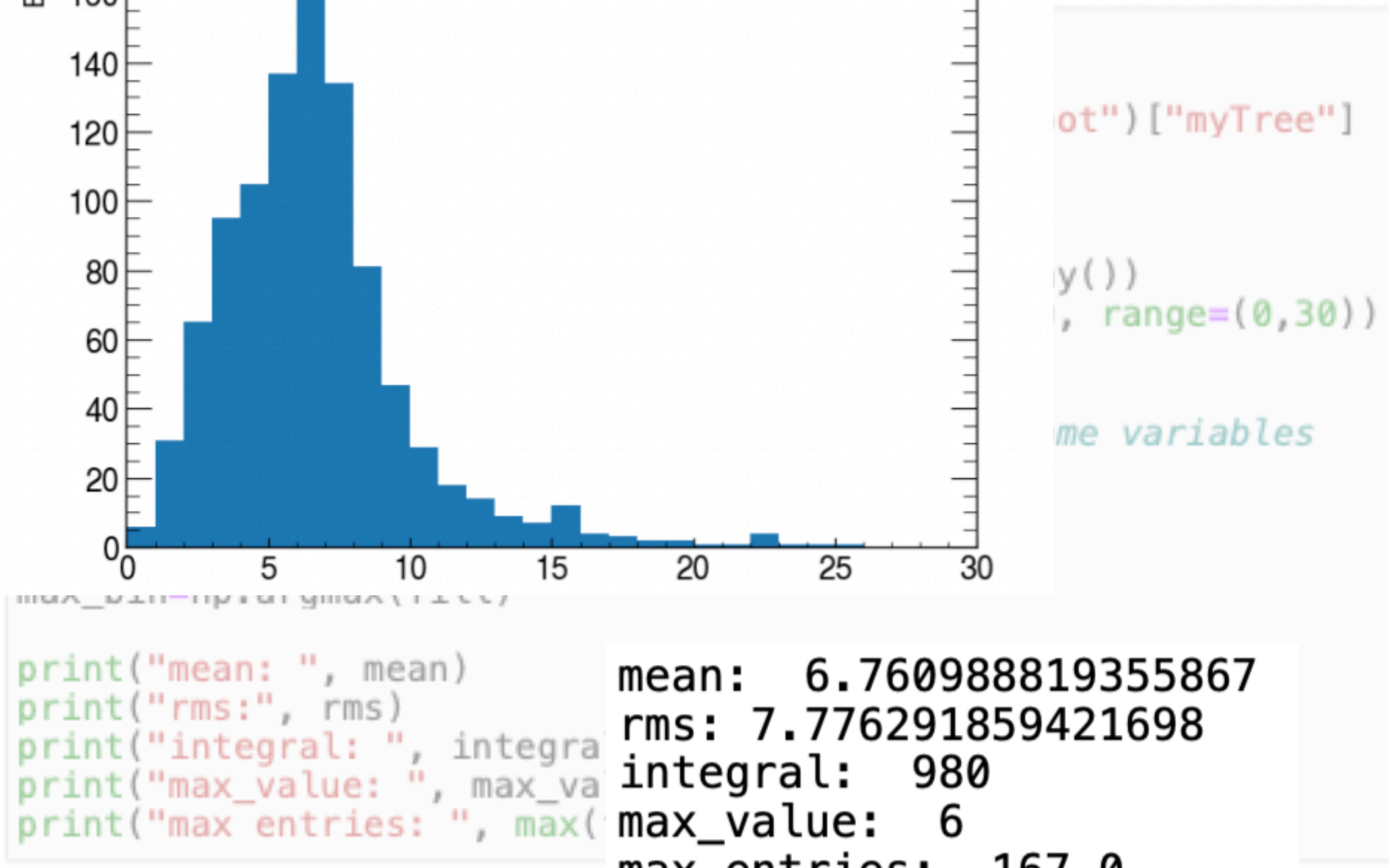
1. Create a histogram with 30 bins ranging from 0. to 30. with title/x-axis label "x"
2. Fill the histogram at the following branch: “truth_mu_pt[0]” in “MyTree”.
3. Draw the histogram.
4. Calculate the mean value and the rms and show it on the screen.

```
        print mean, rms
```

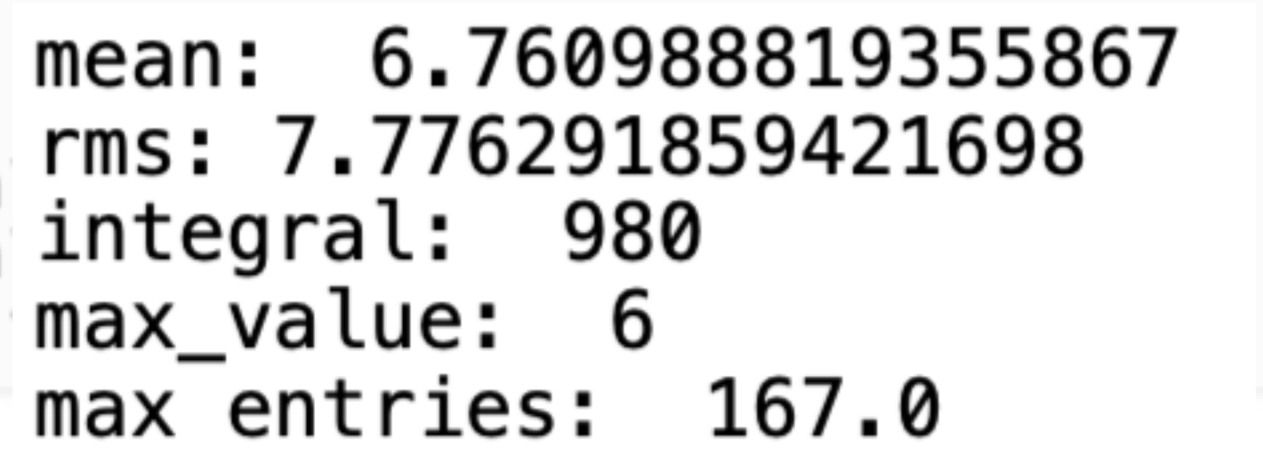
5. Calculate the integral of the histogram.
6. Identify the bin with the maximum number of entries.
7. Calculate the maximum bin content.
8. Set the y-axis label to "entries".
9. Set the line color of the histogram to red.
10. Run with

```
python -i ExerciseHist.py
```





```
ot")["myTree"]

y())
, range=(0,30))

me variables

print("mean: ", mean)
print("rms:", rms)
print("integral: ", integra
print("max_value: ", max_va
print("max entries: ", max(
```



```
mean:  6.760988819355867
rms: 7.776291859421698
integral:  980
max_value:  6
max entries:  167.0
```

Histogram
—hist in Matplotlib

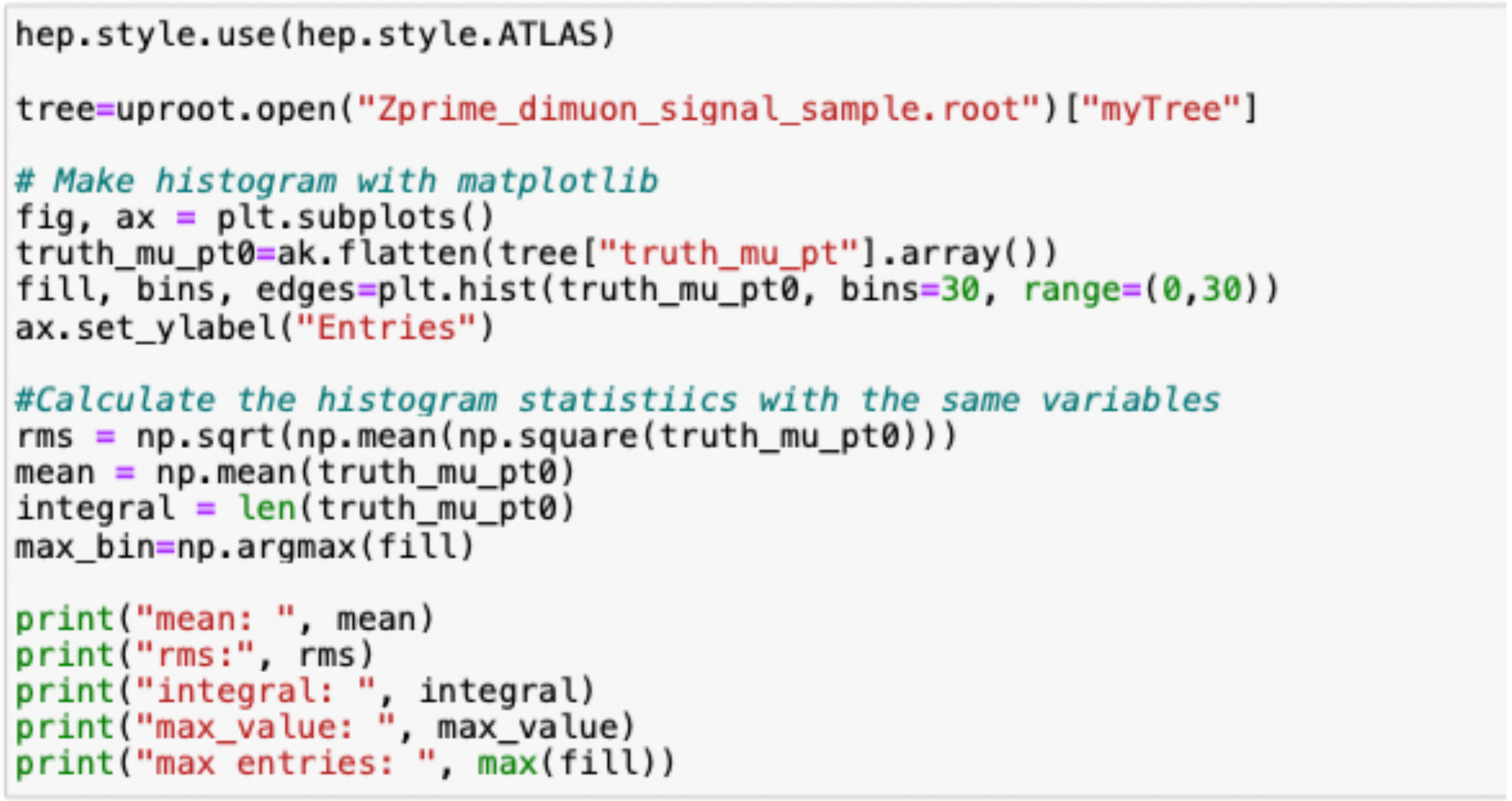
Write a python macro ExerciseHist.py

1. Create a histogram with 30 bins ranging from 0. to 30. with title/x-axis label "x"
2. Fill the histogram at the following branch: “truth_mu_pt[0]” in “MyTree”.
3. Draw the histogram.
4. Calculate the mean value and the rms and show it on the screen.

```
print mean, rms
```

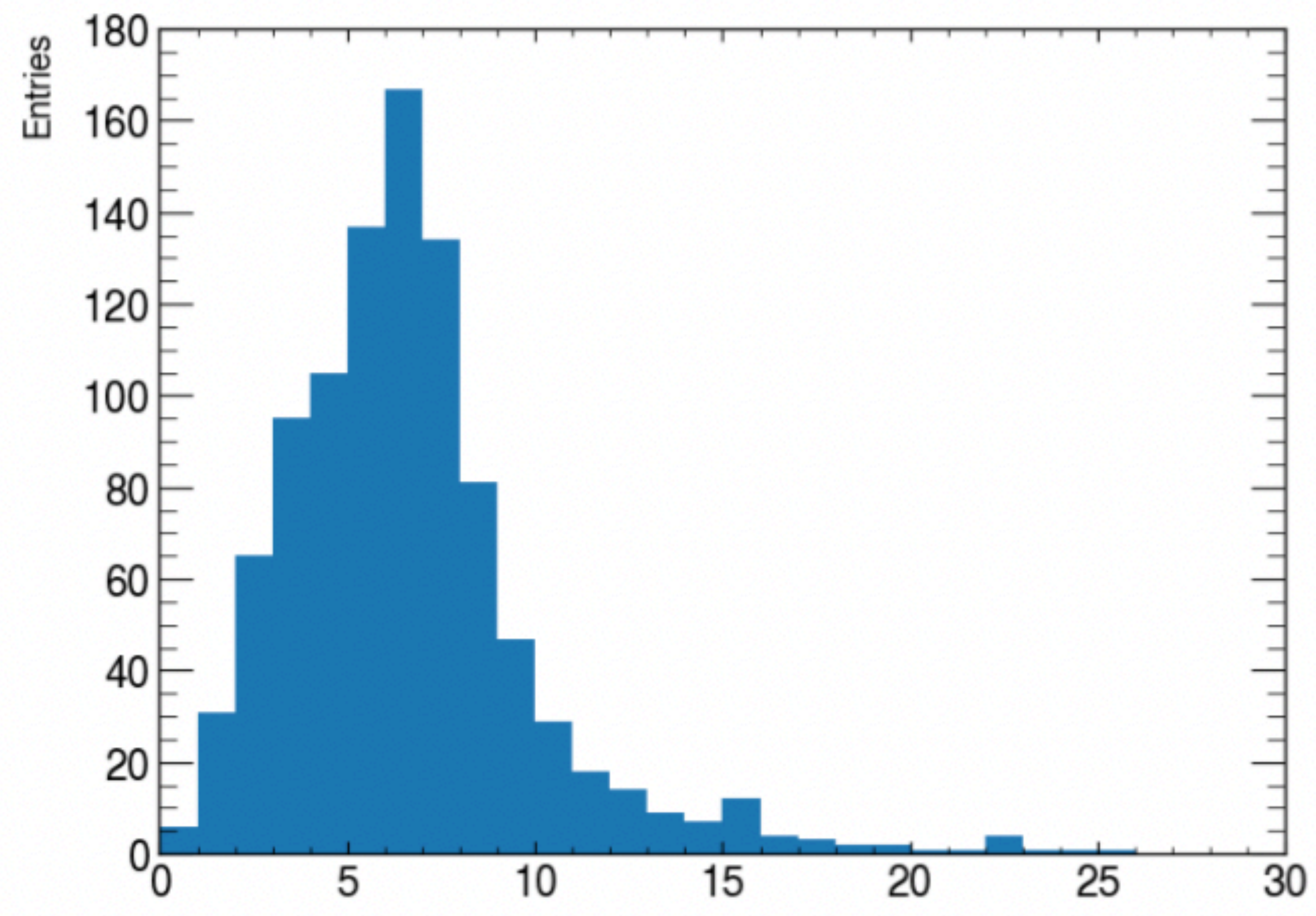
5. Calculate the integral of the histogram.
6. Identify the bin with the maximum number of entries.
7. Calculate the maximum bin content.
8. Set the y-axis label to "entries".
9. Set the line color of the histogram to red.
10. Run with

```
python -i ExerciseHist.py
```



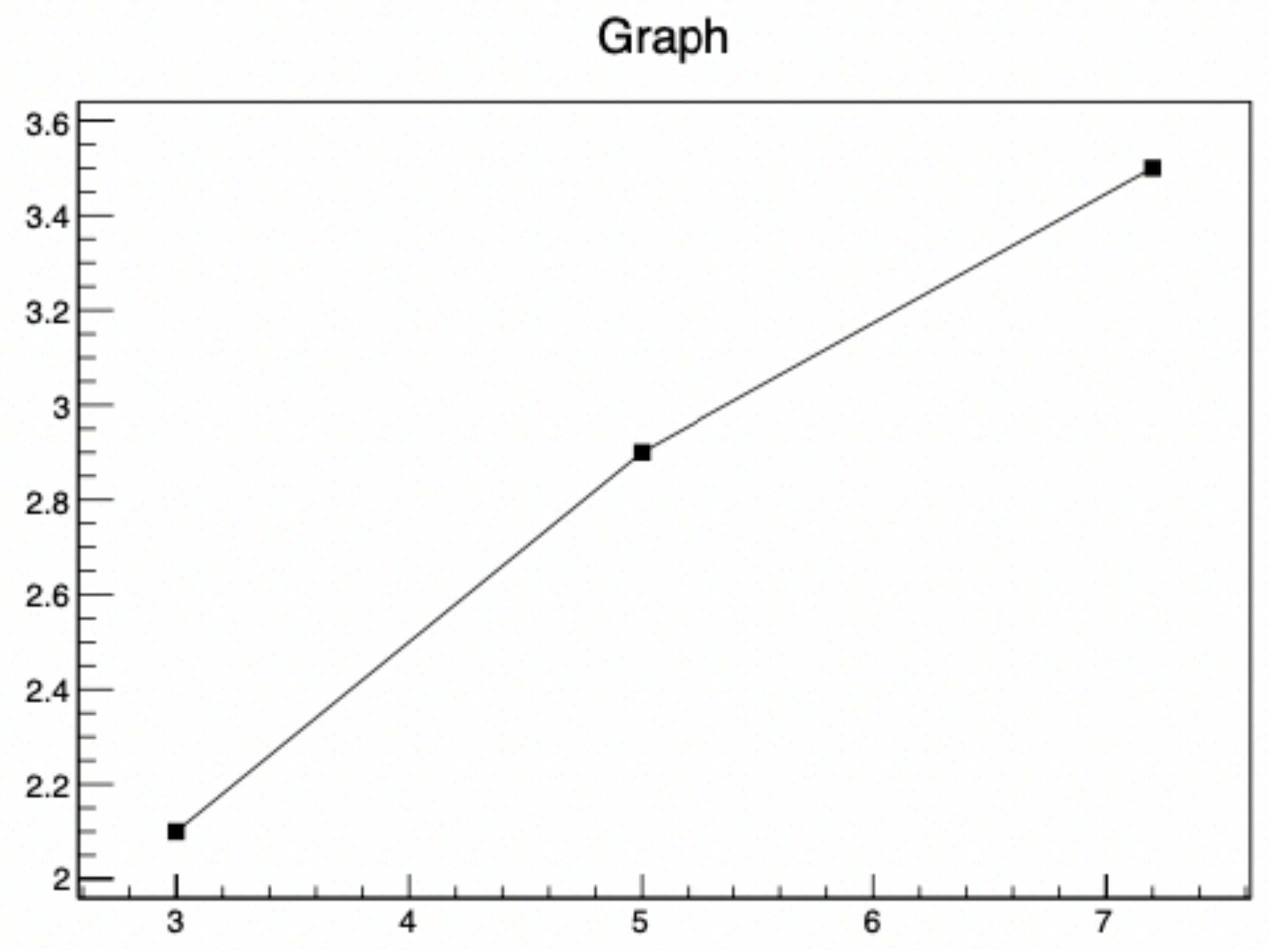
```
hep.style.use(hep.style.ATLAS)

tree=uproot.open("Zprime_dimuon_signal_sample.root")["myTree"]

# Make histogram with matplotlib
fig, ax = plt.subplots()
truth_mu_pt0=ak.flatten(tree["truth_mu_pt"].array())
fill, bins, edges=plt.hist(truth_mu_pt0, bins=30, range=(0,30))
ax.set_ylabel("Entries")

#Calculate the histogram statistiics with the same variables
rms = np.sqrt(np.mean(np.square(truth_mu_pt0)))
mean = np.mean(truth_mu_pt0)
integral = len(truth_mu_pt0)
max_bin=np.argmax(fill)

print("mean: ", mean)
print("rms:", rms)
print("integral: ", integral)
print("max_value: ", max_value)
print("max entries: ", max(fill))
```

```
mean:  6.760988819355867
rms: 7.776291859421698
integral:  980
max_value:  6
max entries:  167.0
```



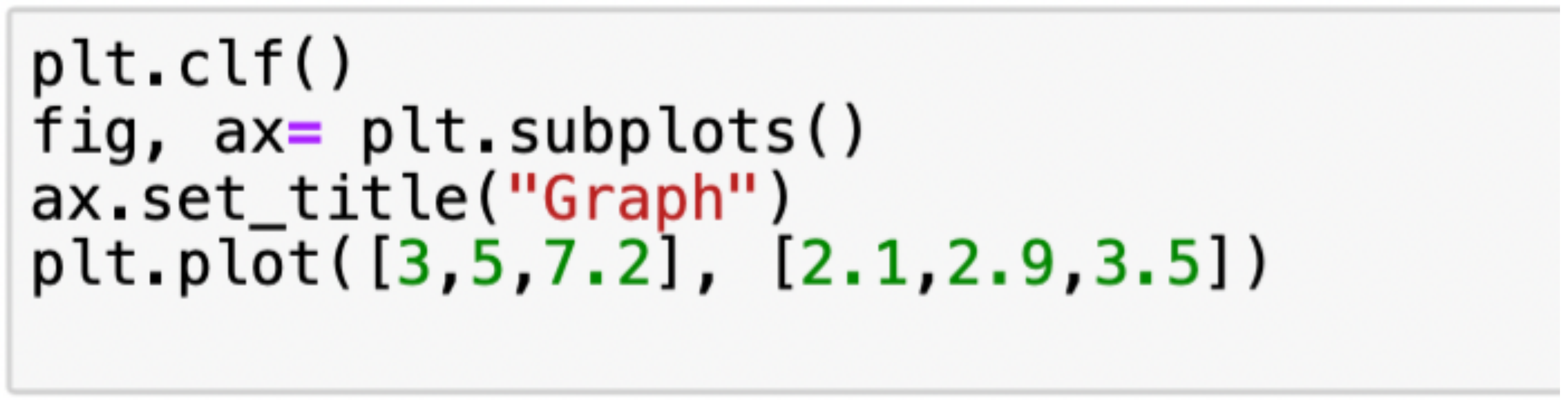
Making Graph with Matplotlib
—Root version

```
>>> from ROOT import *
>>> #create graph with 3 points
>>> graph = TGraph(3)
>>> #set three points of the graph
>>> graph.SetPoint(0, 3.0, 2.1)
>>> graph.SetPoint(1, 5.0, 2.9)
>>> graph.SetPoint(2, 7.2, 3.5)
>>> #set styles
>>> graph.SetMarkerStyle(21)
>>> graph.SetMarkerSize(1)
>>> #Draw axis (A), points (P), and line (L)
>>> graph.Draw("APL")
```

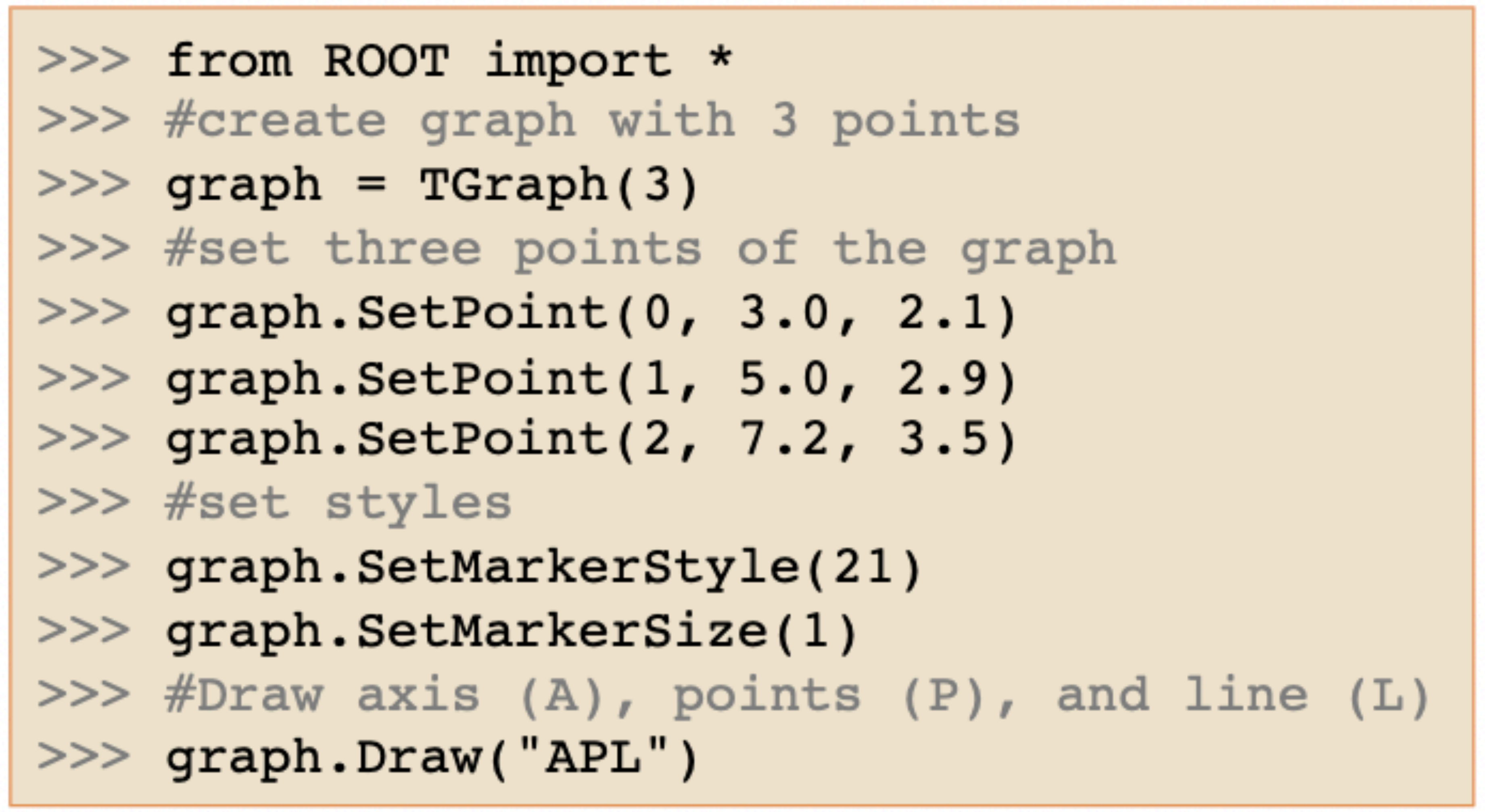


Making Graph with Matplotlib

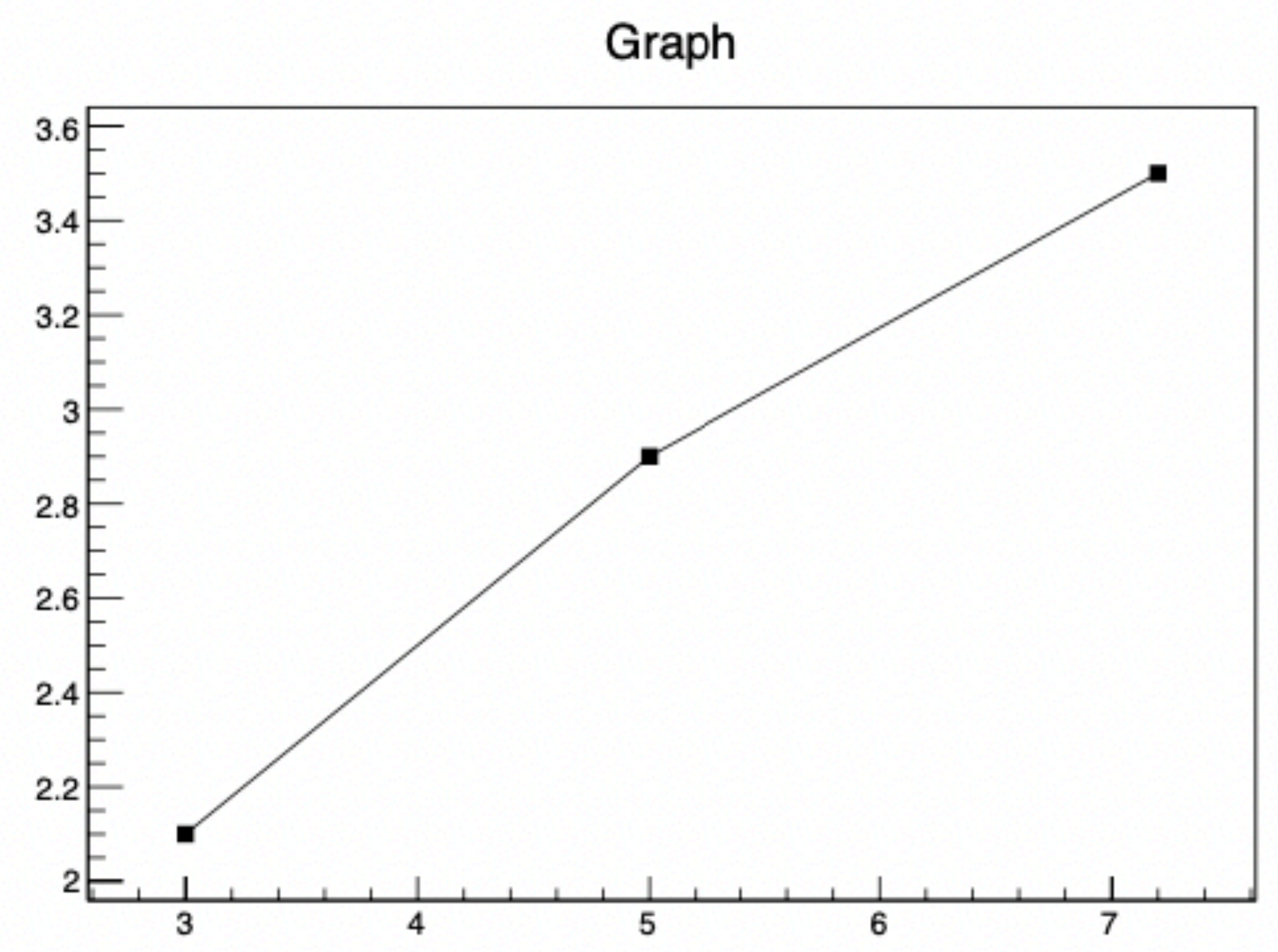
—Matplotlib equivalent

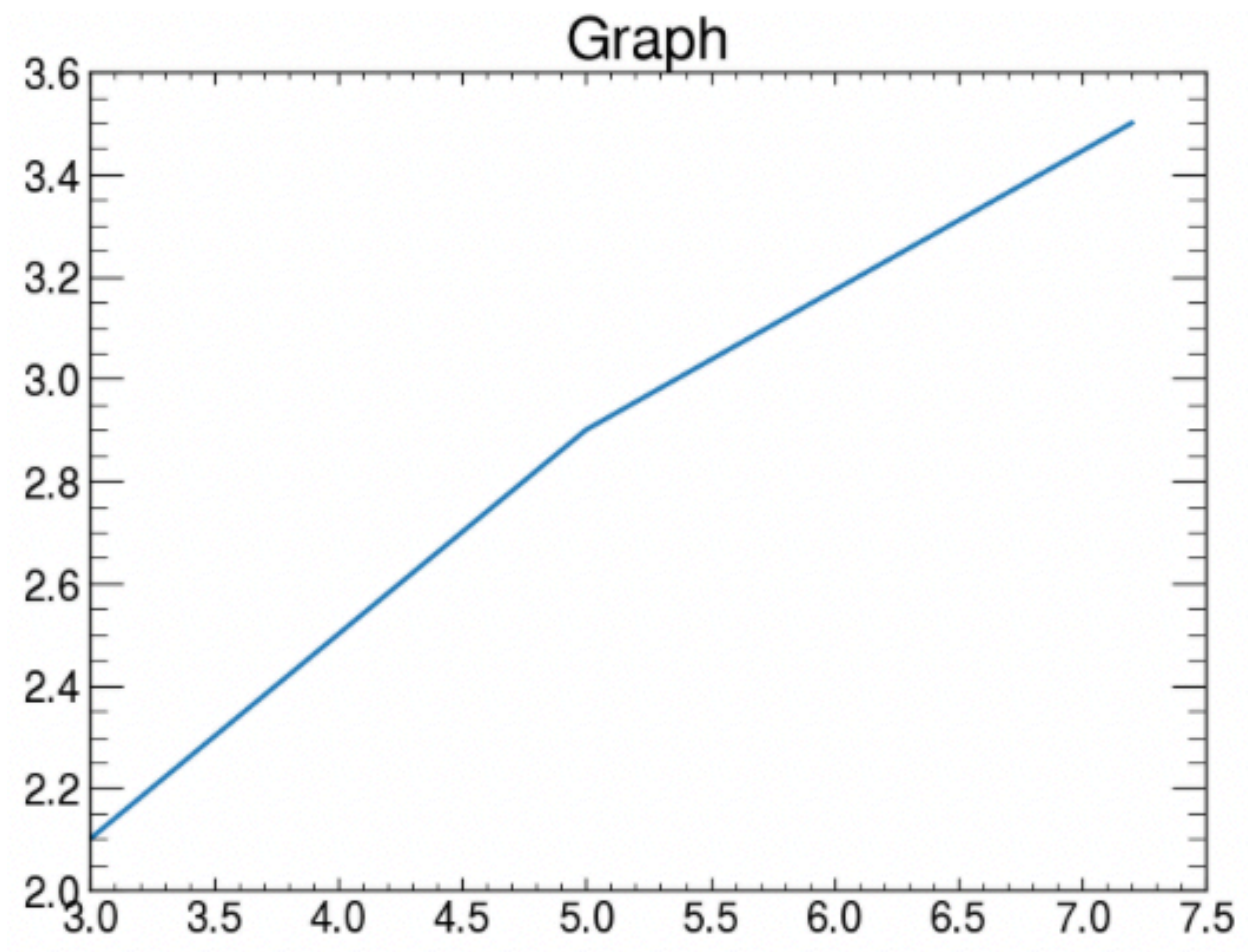


```
plt.clf()
fig, ax= plt.subplots()
ax.set_title("Graph")
plt.plot([3,5,7.2], [2.1,2.9,3.5])
```



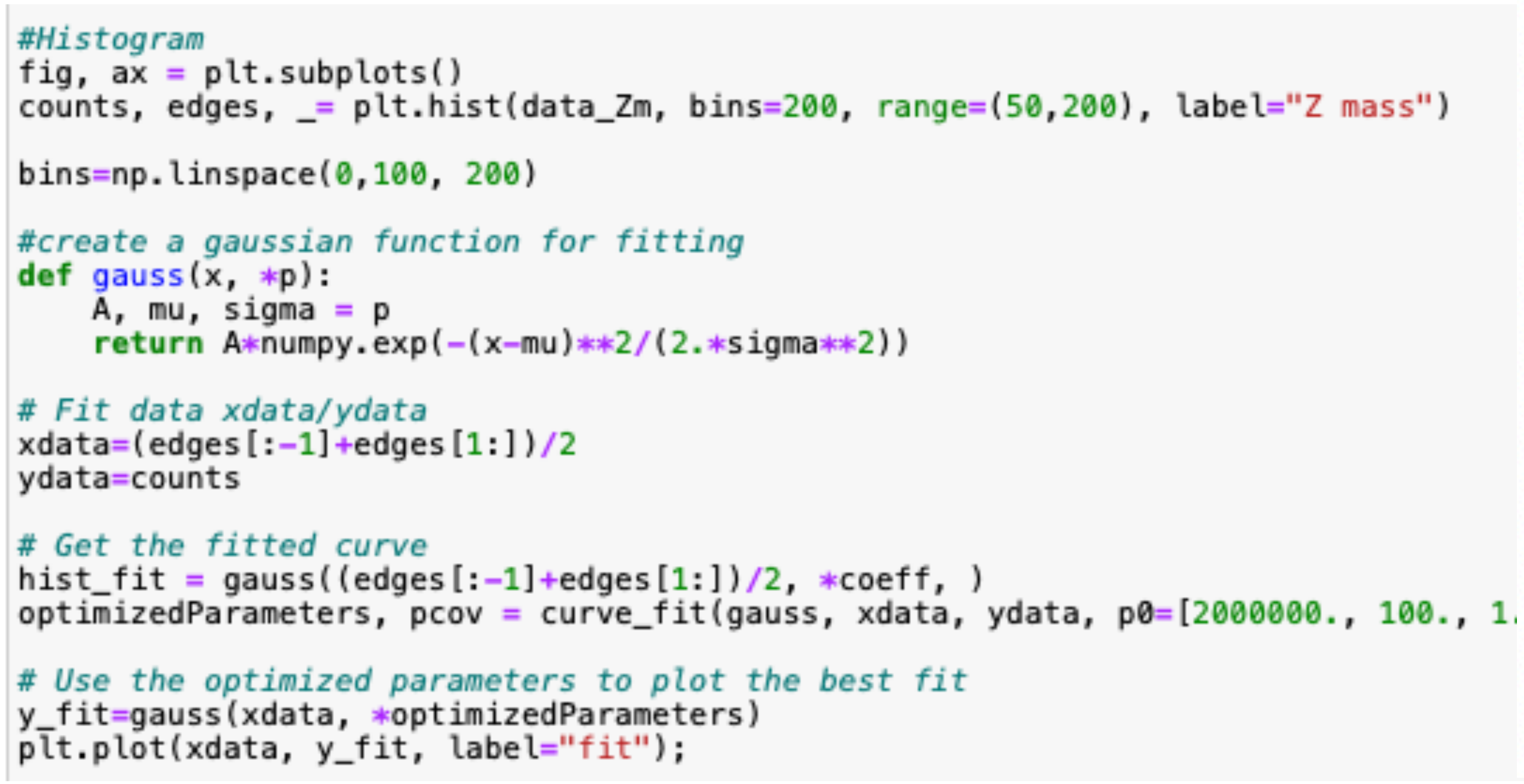
```
>>> from ROOT import *
>>> #create graph with 3 points
>>> graph = TGraph(3)
>>> #set three points of the graph
>>> graph.SetPoint(0, 3.0, 2.1)
>>> graph.SetPoint(1, 5.0, 2.9)
>>> graph.SetPoint(2, 7.2, 3.5)
>>> #set styles
>>> graph.SetMarkerStyle(21)
>>> graph.SetMarkerSize(1)
>>> #Draw axis (A), points (P), and line (L)
>>> graph.Draw("APL")
```



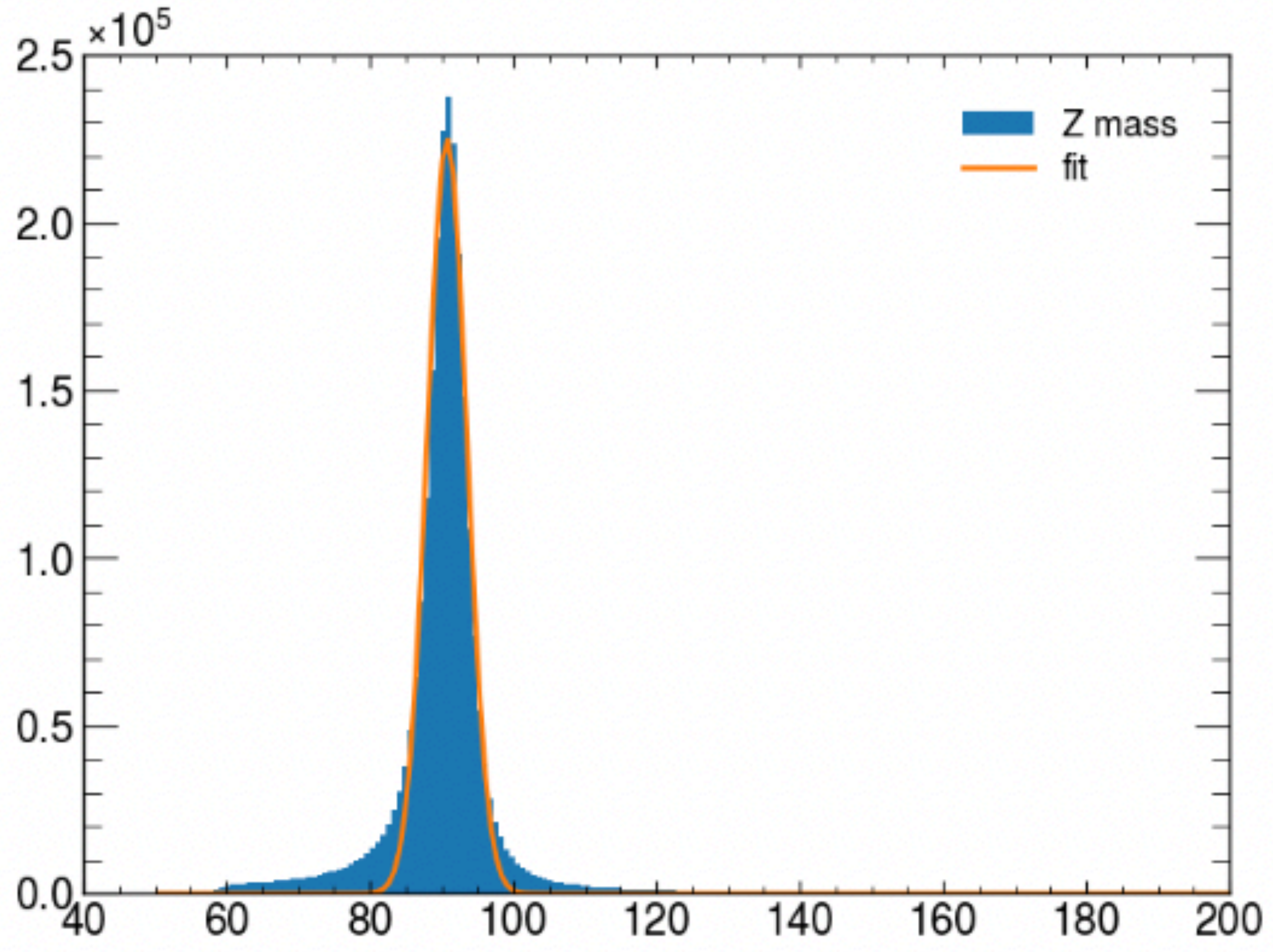


Fitting the Z-mass peak
—Fit function example

- Using the curve_fit function from scipy
- Define the Gaussian Function to be fitted
- Perform fit with the optimizer

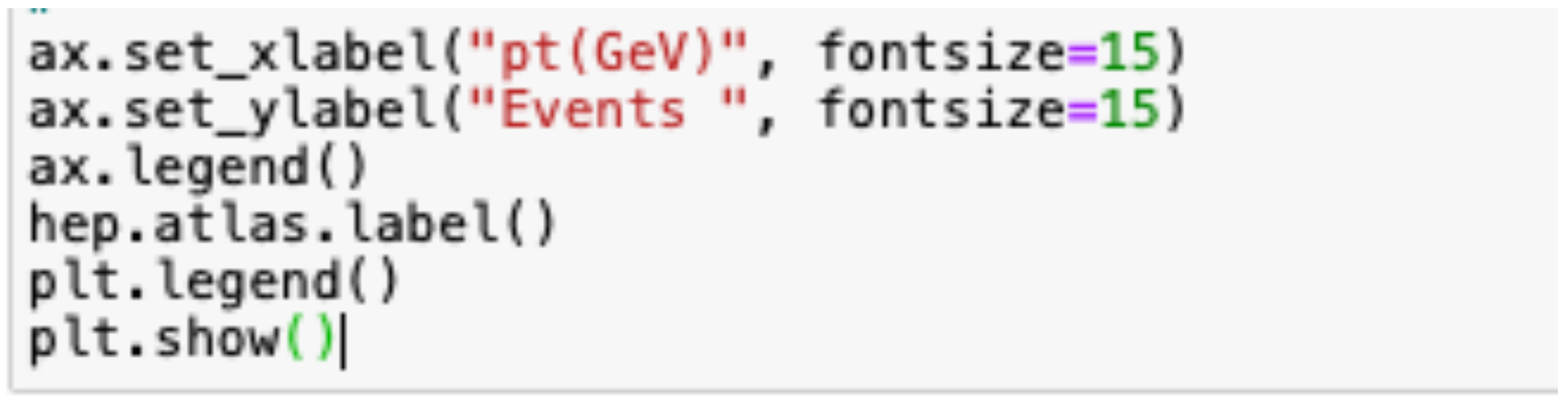


```
#Histogram
fig, ax = plt.subplots()
counts, edges, _= plt.hist(data_Zm, bins=200, range=(50,200), label="Z mass")

bins=np.linspace(0,100, 200)

#create a gaussian function for fitting
def gauss(x, *p):
    A, mu, sigma = p
    return A*numpy.exp(-(x-mu)**2/(2.*sigma**2))

# Fit data xdata/ydata
xdata=(edges[:-1]+edges[1:])/2
ydata=counts

# Get the fitted curve
hist_fit = gauss((edges[:-1]+edges[1:])/2, *coeff, )
optimizedParameters, pcov = curve_fit(gauss, xdata, ydata, p0=[2000000., 100., 1.

# Use the optimized parameters to plot the best fit
y_fit=gauss(xdata, *optimizedParameters)
plt.plot(xdata, y_fit, label="fit");
```

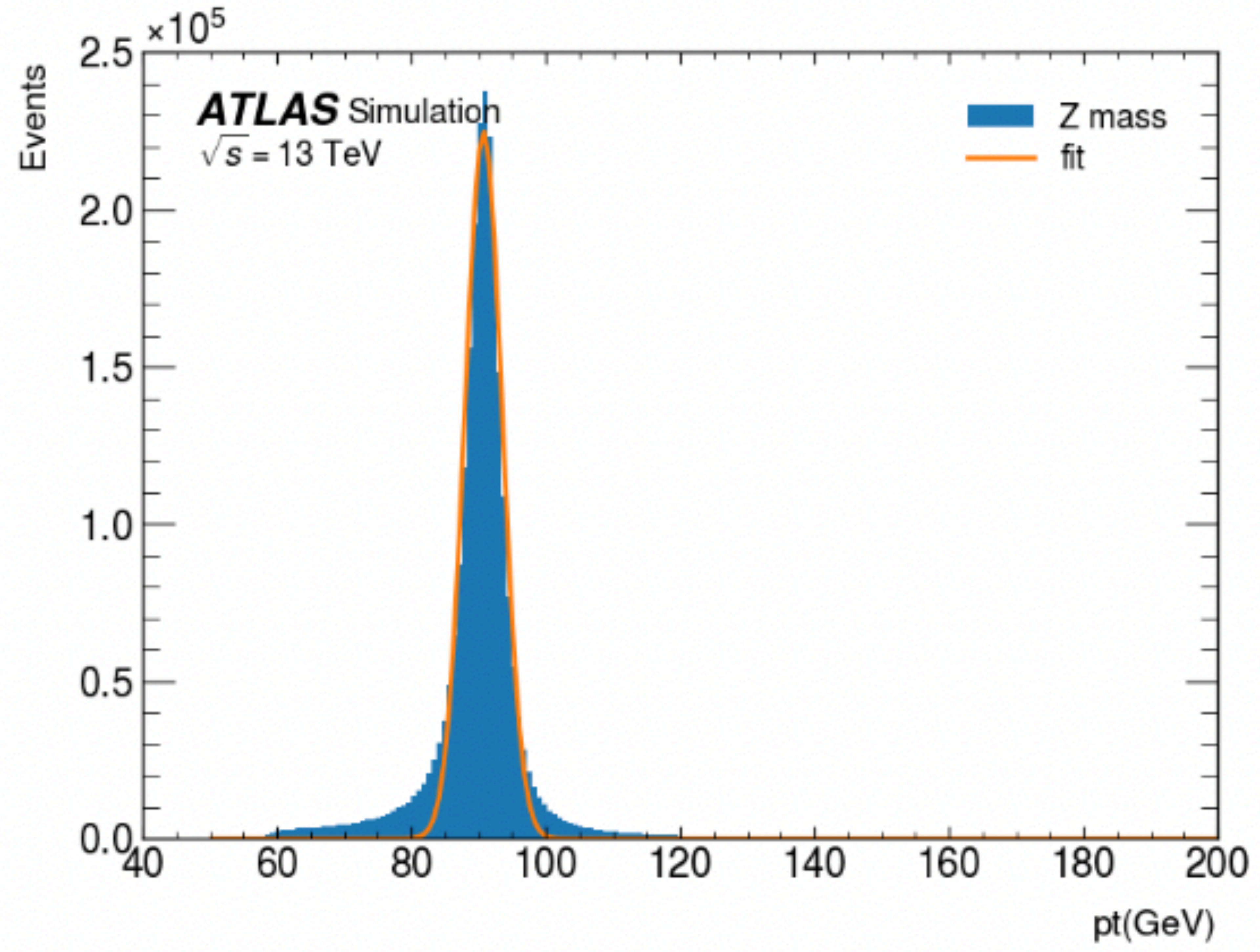


Beautifying plots

- Using the mplhep library: Create root style matplotlib plots
- ATLAS/CMS Label
    - Easily switch between ATLAS/CMS styles
    - `>mplhep.style.use(mplhep.style.ATLAS)`
- Legend
    - Set graph/hist/function label by the label option (label="histogram")
    - Display legend by calling
        - `>ax.legend()`
- X/y axis:
    - `ax.set_ylabel("y label")`
    - `ax.set_xlabel("x label")`



```
ax.set_xlabel("pt(GeV)", fontsize=15)
ax.set_ylabel("Events ", fontsize=15)
ax.legend()
hep.atlas.label()
plt.legend()
plt.show()
```



Back up

References

- https://hsf-training.github.io/hsf-training-matplotlib/aio/index.html
- https://github.com/nsmith-/mpl-hep/blob/master/binder/gallery.ipynb

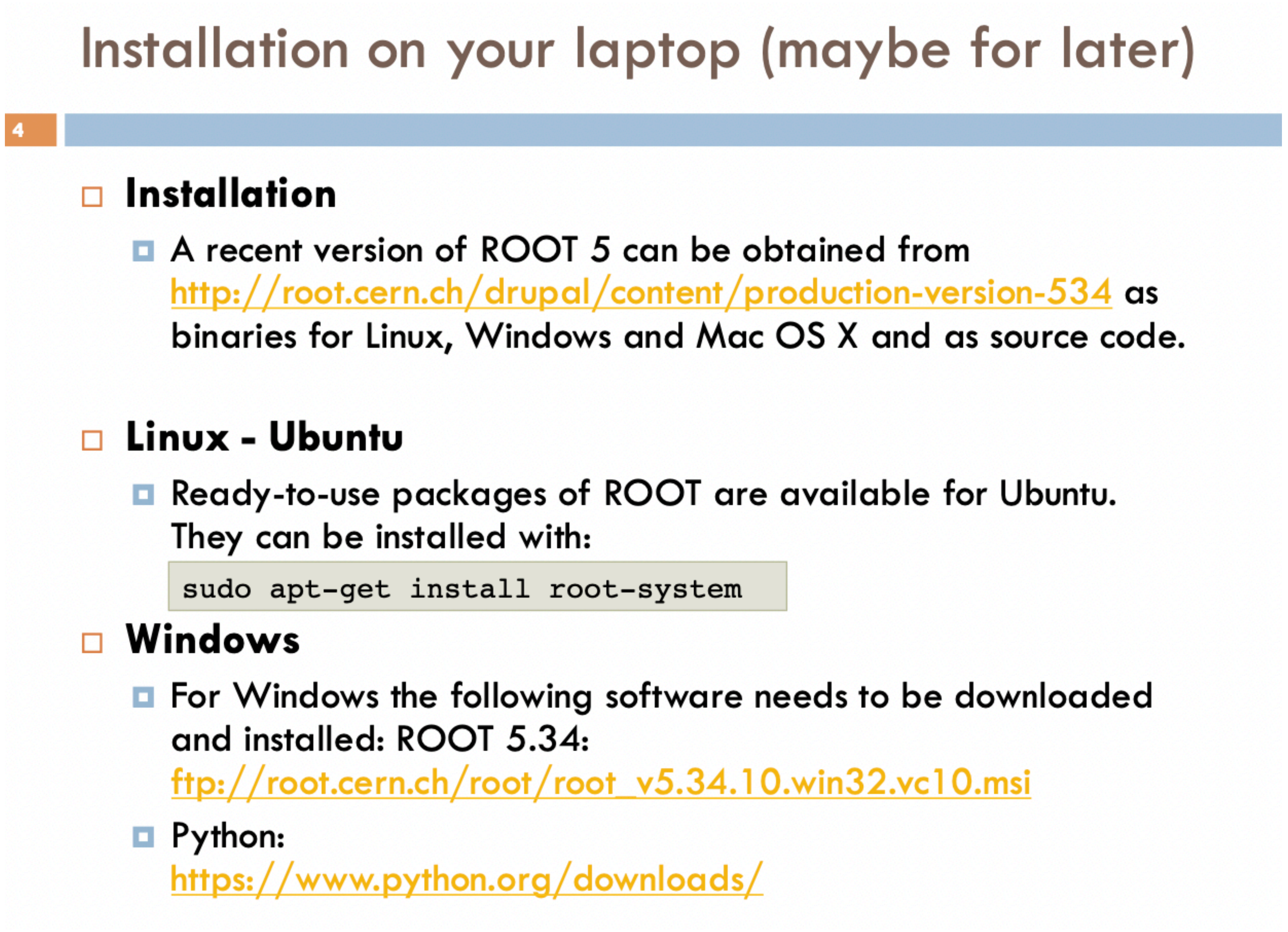
Setting up ROOT

Need update From Isabell and Claudia

- Some technical details
    - ! Connect to either eduroam or the school network: ! Name: terascale
    - ! WPA/WPA2-PSK: XxPWjNH7
    - ! Code examples throughout the talk with colors Execute this! Some example code!
    - ! All will get school accounts for naf
    - ! Example: ssh -X -Y school30@naf-school02.desy.de
    - ! Setup the needed software
    - module avail!
    - module load root/5.34!

- Laptop installation in backup:
- Docker installation in backup

Installation guide for laptop



Installation on your laptop (maybe for later)

- **Installation**
  - A recent version of ROOT 5 can be obtained from http://root.cern.ch/drupal/content/production-version-534 as binaries for Linux, Windows and Mac OS X and as source code.

- **Linux - Ubuntu**
  - Ready-to-use packages of ROOT are available for Ubuntu. They can be installed with:

```
sudo apt-get install root-system
```

- **Windows**
  - For Windows the following software needs to be downloaded and installed: ROOT 5.34: ftp://root.cern.ch/root/root_v5.34.10.win32.vc10.msi
  - Python: https://www.python.org/downloads/

Slides from Claudia Seitz

Exercise

—Histogram Drawing

Write a python macro ExerciseHist.py

1. Create a histogram with 10 bins ranging from 0. to 100. with title/x-axis label "x"
2. Fill the histogram at the following numbers: 11.3, 25.4, 18.1
3. Fill the histogram with the square of all integers from 0. to 9. (Hint: A simple loop will save you from typing several lines of code)
4. Draw the histogram.
5. Calculate the mean value and the rms and show it on the screen.

```
        print mean, rms
```

6. Calculate the integral of the histogram.
7. Identify the bin with the maximum number of entries.
8. Calculate the maximum bin content.
9. Set the y-axis label to "entries".
10. Set the line color of the histogram to red.
11. Run with

```
python -i ExerciseHist.py
```

- One dimensional histogram TH1F.
- Constructor of a histogram: TH1F::TH1F(const char* name, const char* title, Int_t nbinsx, Double_t xlow, Double_t xup).
- Fill a histogram: Int_t TH1F::Fill(Double_t x)
- Draw a histogram: void TH1F::Draw(Option_t* option = "")
- Mean of a histogram: Double_t TH1F::GetMean(Int_t axis = 1) const
- RMS of a histogram: Double_t TH1F::GetRMS(Int_t axis = 1) const
- Mode of a histogram: Int_t TH1F::GetMaximumBin() const
- Get the bin content of a histogram: Double_t TH1F::GetBinContent(Int_t bin) const
- Integral of a histogram: Double_t TH1F::Integral(Option_t* option = "") const
- Y-axis used to draw the histogram: TAxis* TH1F::GetYaxis() const
- Access axis and set label void TAxis::SetTitle(char*)
- Change line color of the histogram: void TAttLine::SetLineColor(Color_t lcolor).
  The color index for red is named kRed.

Change to drawing muonPT[0] and muonPT[1]