TESHEP 2025 Hands-on Exercesis

Introduction

Arantza Oyanguren Campos, Olena Mezhenska, Dominik Babal, **Rafał Staszewski**

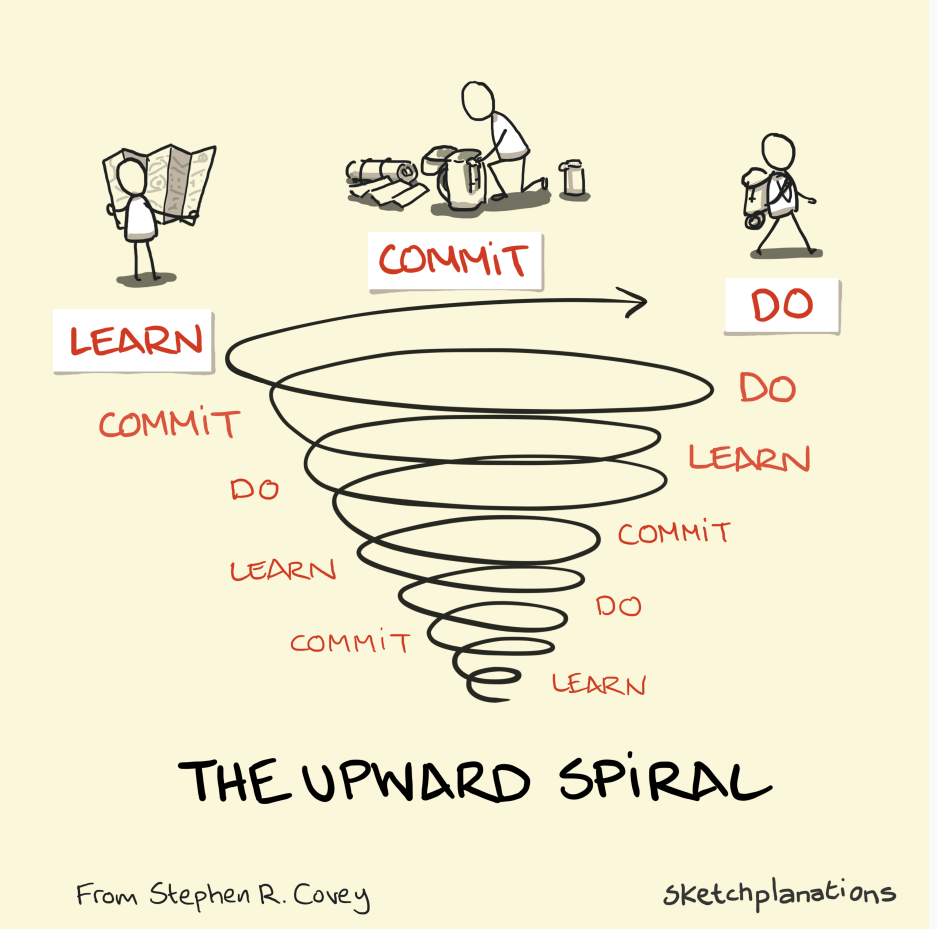
12 July 2025

Goals of the hands-on exercises

The goal is **not** to finish the exercise … but rather to learn something new

Things to learn include:

- Elements and concepts of statistical data analysis
- ROOT framework (with RDataFrame)
- Optimizing selection
- Unfolding



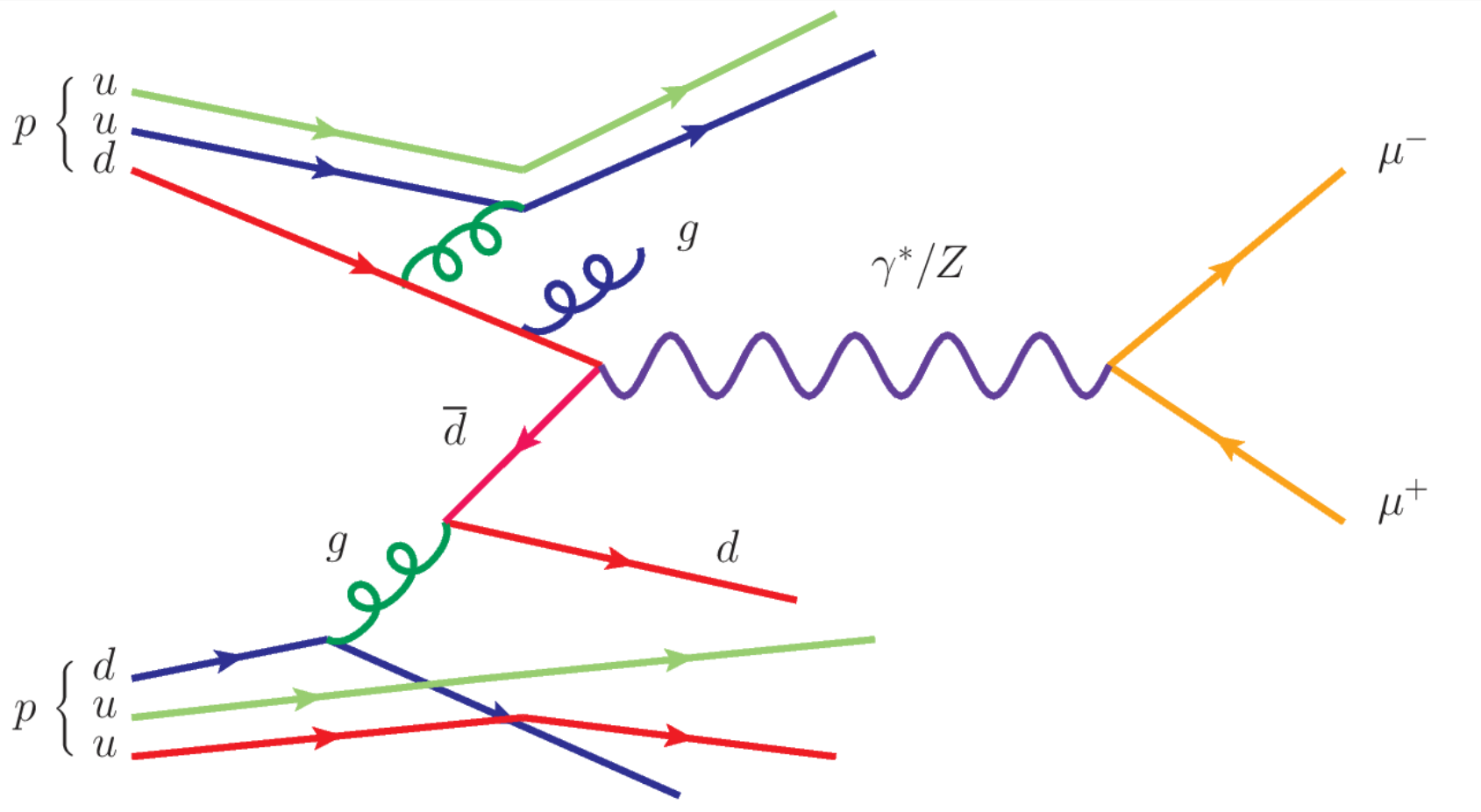
ROOT

Useful links:

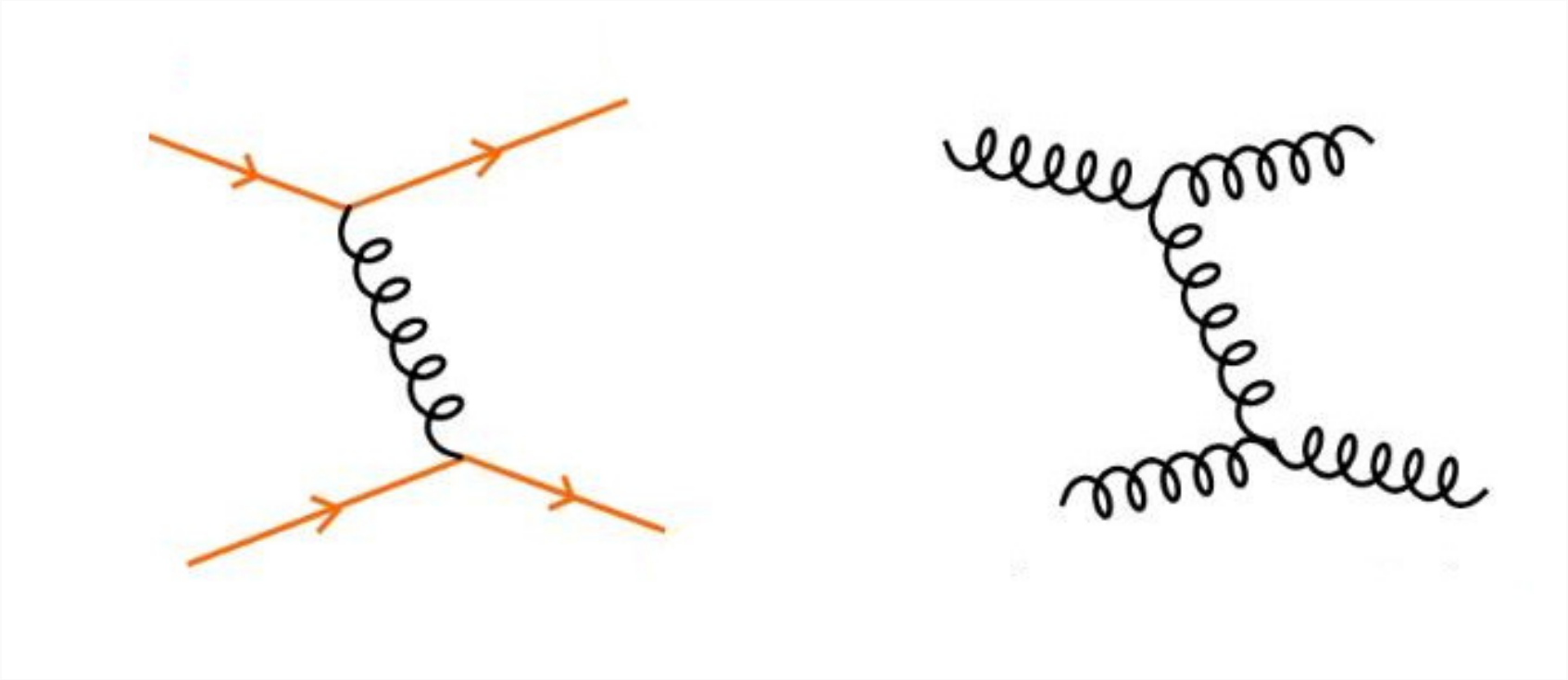
`https://root.cern/learn`

`https://root.cern/doc/master`

Our signal process: $Z \to e^+e^-$ in pp interaction

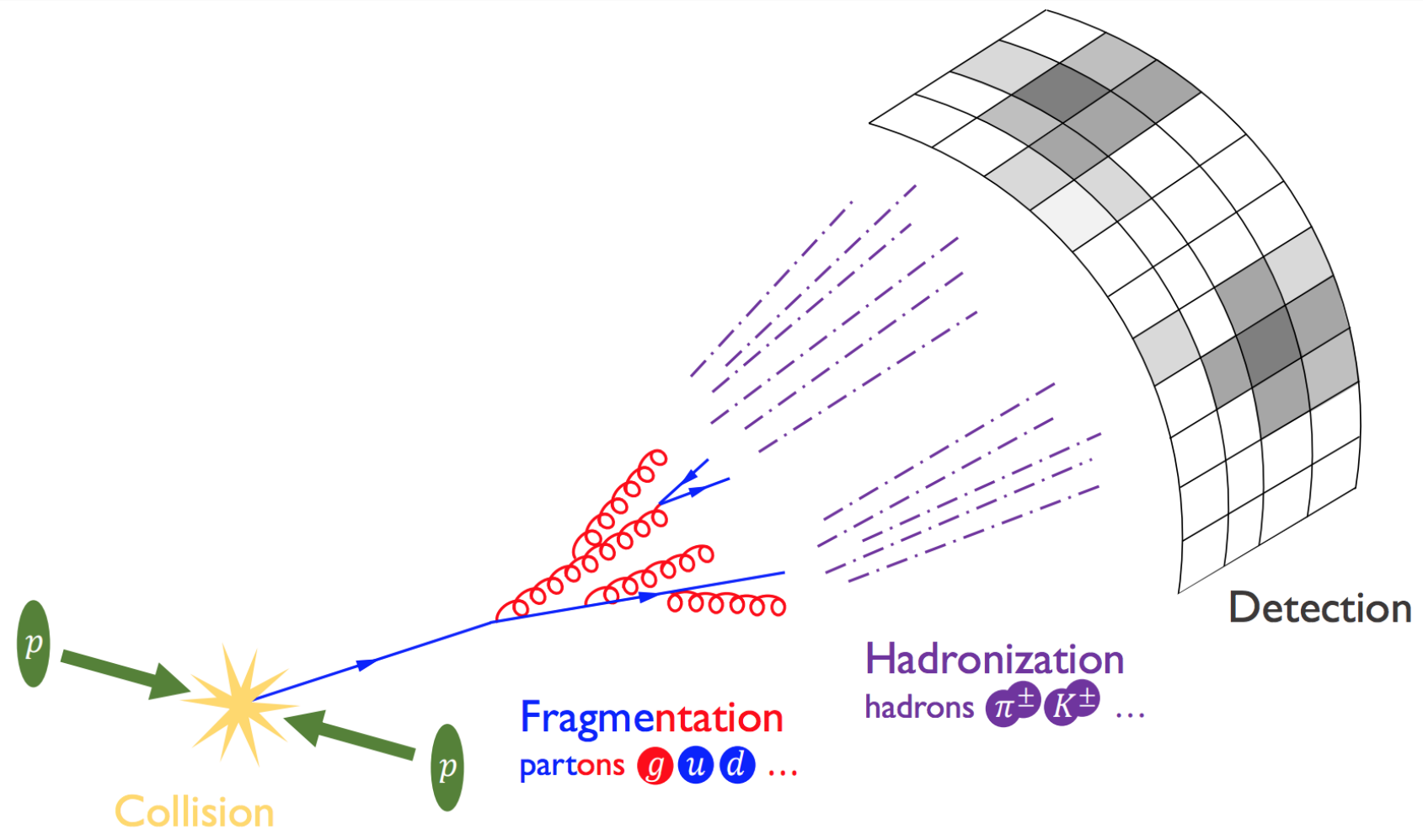


Background process



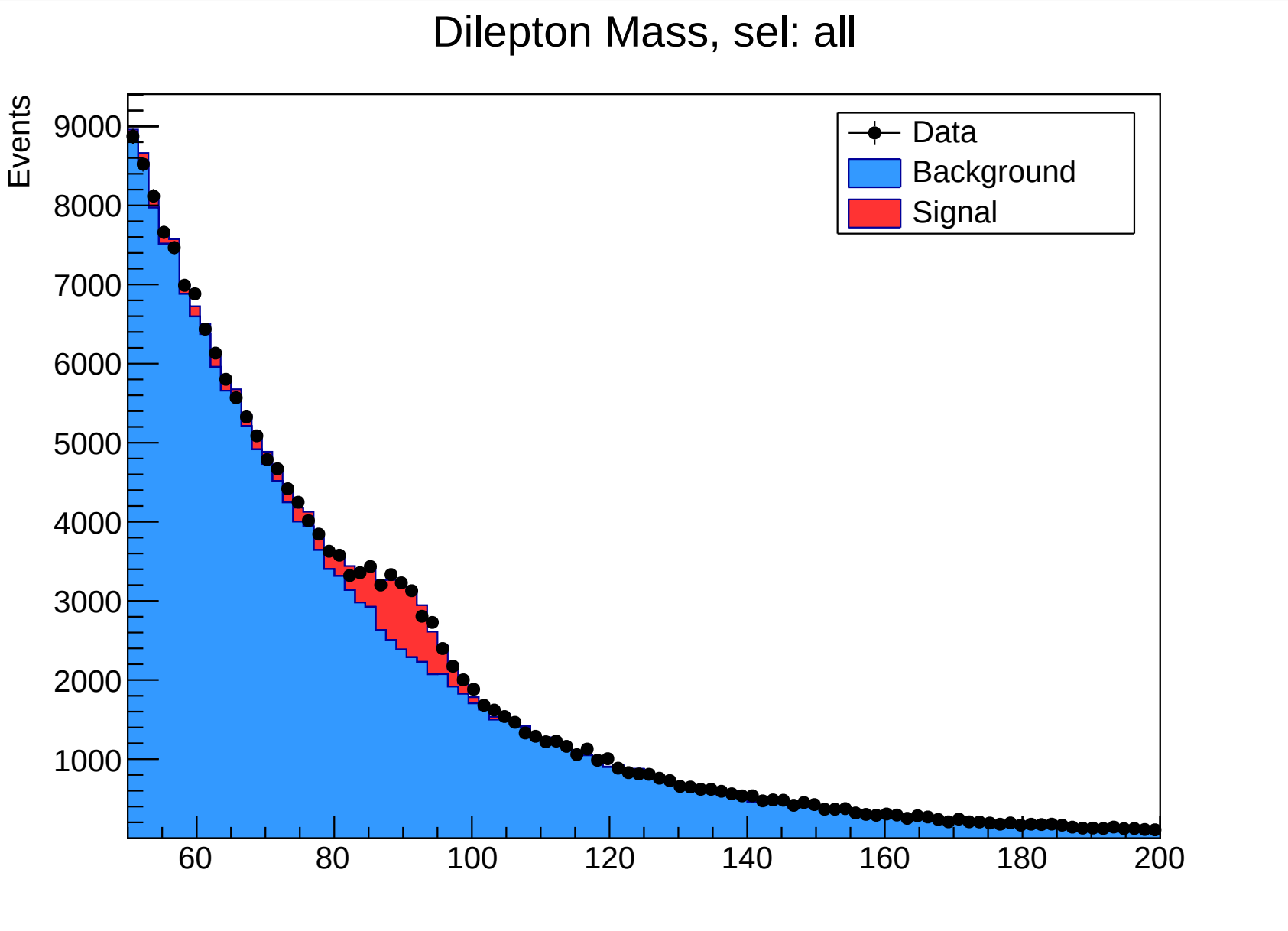
From: cronodon.com

Background process

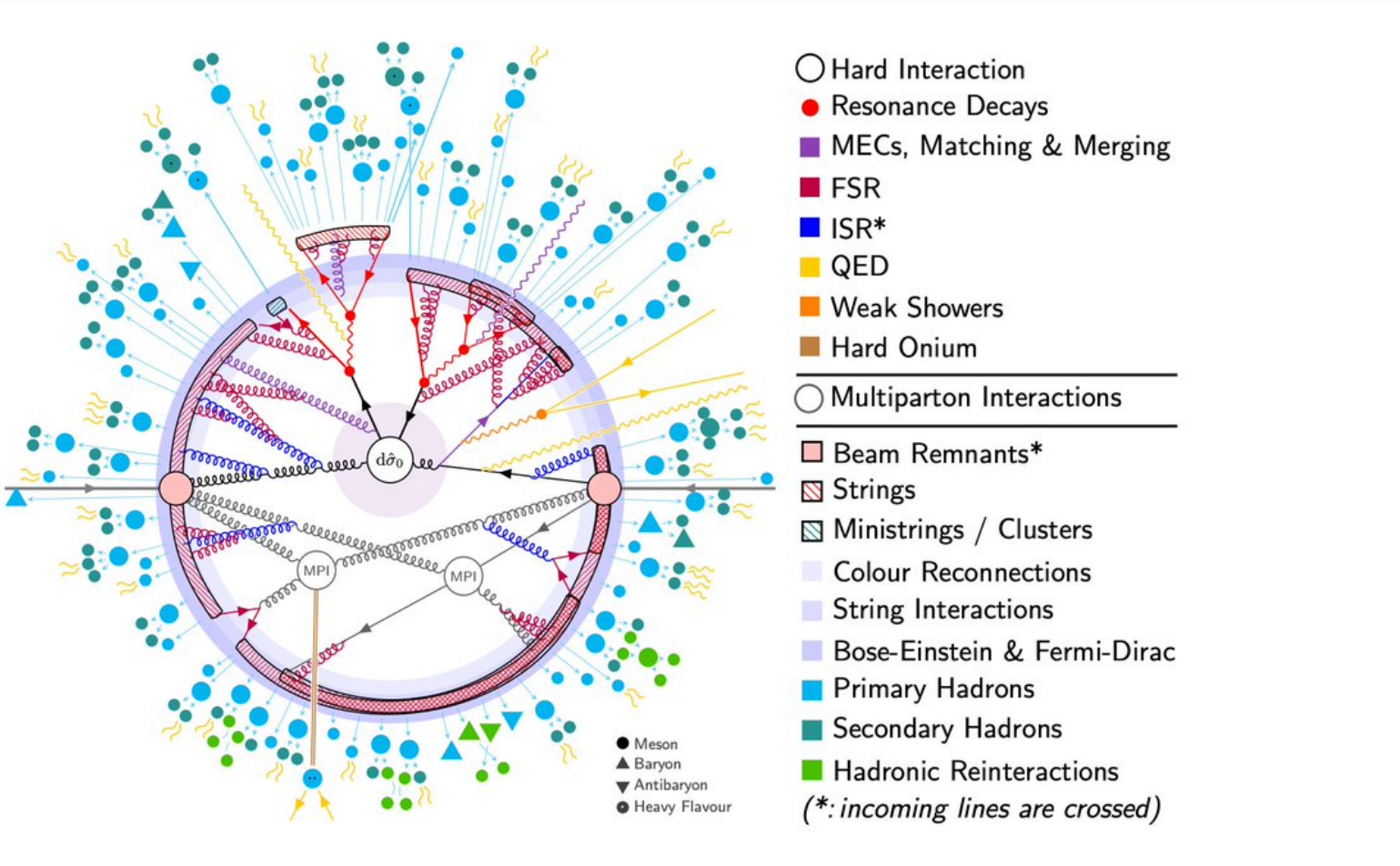


From: www.ericmetodiev.com

Histograms


Dilepton Mass, sel: all
Events
9000
8000
7000
6000
5000
4000
3000
2000
1000
60
80
100
120
140
160
180
200
Data
Background
Signal

Monte Carlo event generator (Pythia)



https://x.com/profdvp/status/1613179796372000768/

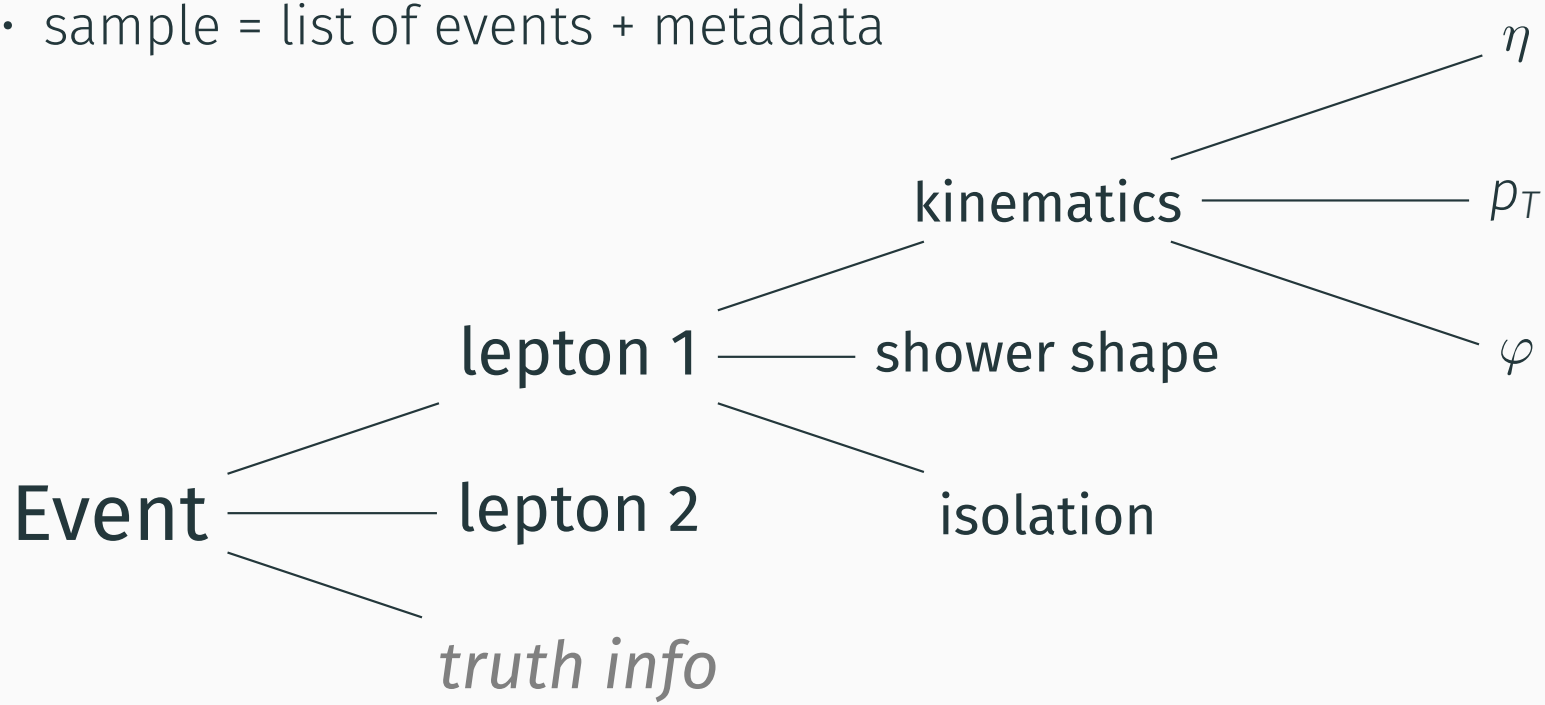
Simulation

- Event (list of particles) generated with Pythia for $Z \to ee$ or jets
- Full simulation – simulation of how each particle passes through the material of the detector, how it interacts, what other particles are produces, how much energy is left, response of electronics, ...
- Here, we perform *fast simulation*
  - For each particle, check if it is considered an electron candidate:
    - 95% for genuine electrons (5% inefficiency)
    - 40% for pions (fake electrons)[1]
  - Simulate the response of the detector with some effective parametrizations
  - Store events to a ROOT file

[1] 40% is unrealistically large, but it makes the exercise more interesting

Data structure

- sample = list of events + metadata



- Truth information present only in MC samples, not in data
- You can check *generate.py* to see how generation works.
- You can run it with *sig.cmnd* or bkg.cmnd. We will provide the samples, so this is not needed. But you can try.

Task 1 – inspect data file

Download `test.root`, for example:

```
wget https://rafal.web.cern.ch/teshep2025/test.root
```

Show how to open file with TBrowser and see distributions

```
root --web=off test.root
```

```
new TBrowser
```

Inspect what variables are available and what distributions they have.

Run StartViewer from TBrowser or from command line:

```
ntuple->StartViewer()
```

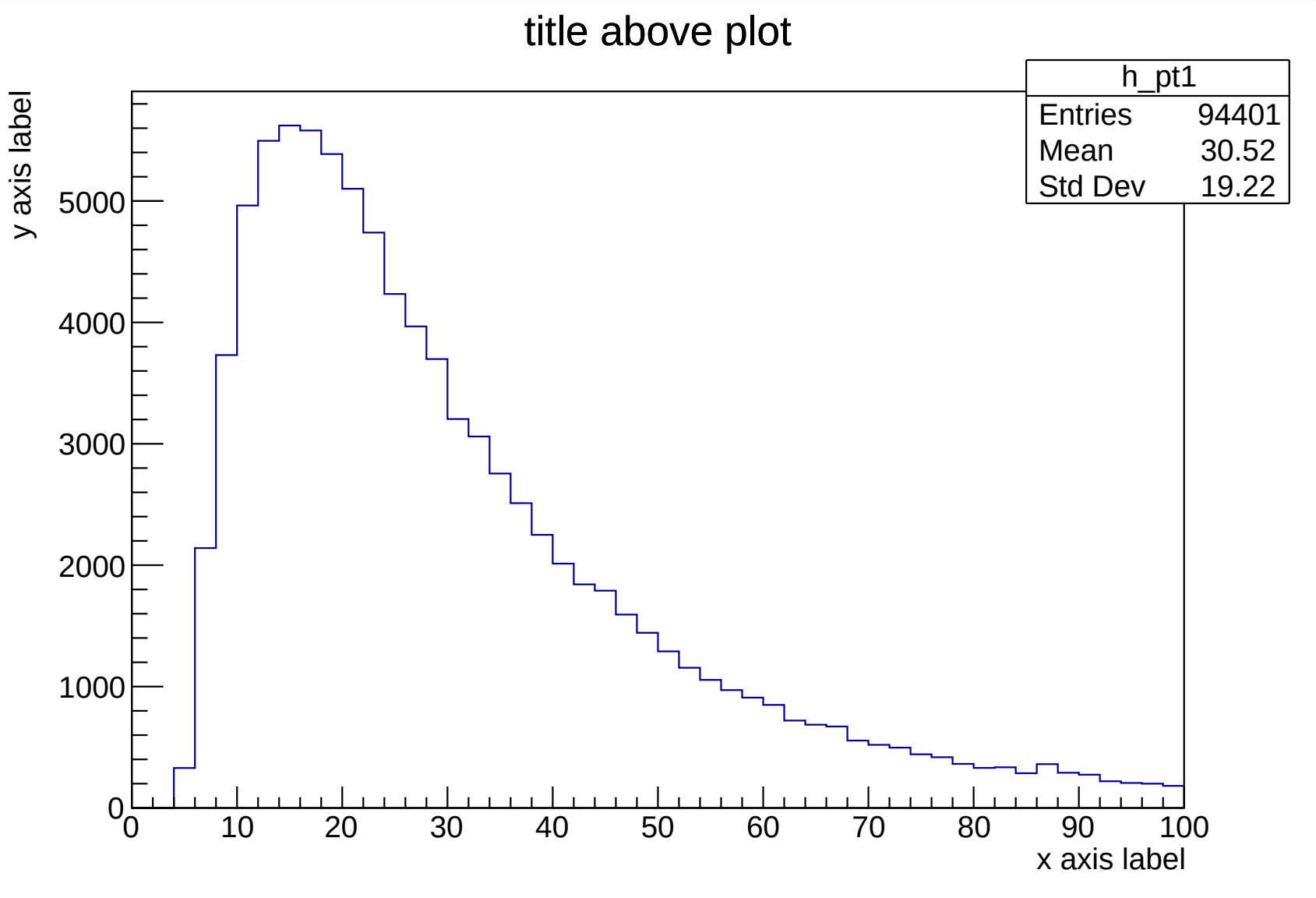
With TreeViewer You can also inspect correlations between variables.

Task 2 – RDataFrame

```python
#!/usr/bin/env python3
import ROOT

# Create RDataFrame
df = ROOT.RDataFrame("ntuple", "test.root")

# Book a histogram of pt1
h_pt1 = df.Histo1D(("h_pt1", "title above plot;x axis label;y
↪  axis label", 50, 0, 100), "pt1")

# Save histogram to output file
c = ROOT.TCanvas()
h_pt1.Draw()
c.SaveAs("example1.pdf")
```

Add 2 more histograms. Choose proper ranges.

Task 2 – RDataFrame

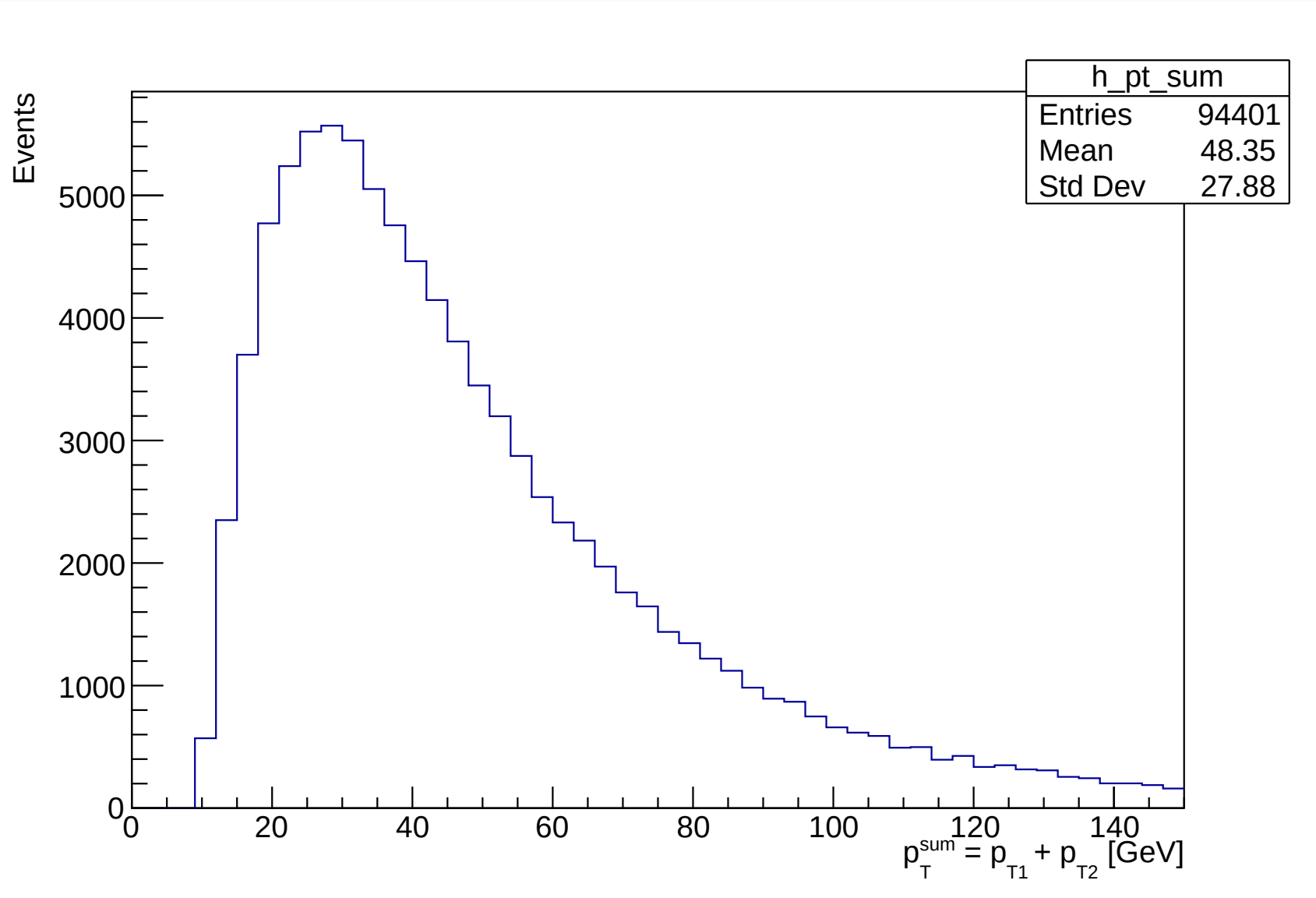


Task 2 – RDataFrame

```python
#!/usr/bin/env python3
import ROOT

df = ROOT.RDataFrame("ntuple", "test.root")

# Define a new variable
df = df.Define("pt_sum", "pt1 + pt2")

h_pt_sum = df.Histo1D(("h_pt_sum", ";p_{T}^{sum} = p_{T1} +
↪  p_{T2} [GeV];Events", 50, 0, 150), "pt_sum")

c = ROOT.TCanvas()
h_pt_sum.Draw()
c.SaveAs("example2.pdf")
```

Create a histogram of $|\vec{p}_{T,1} + \vec{p}_{T,2}|$
(this is not the same as $p_{T,1} + p_{T,2} = |\vec{p}_{T,1}| + |\vec{p}_{T,2}|$!)

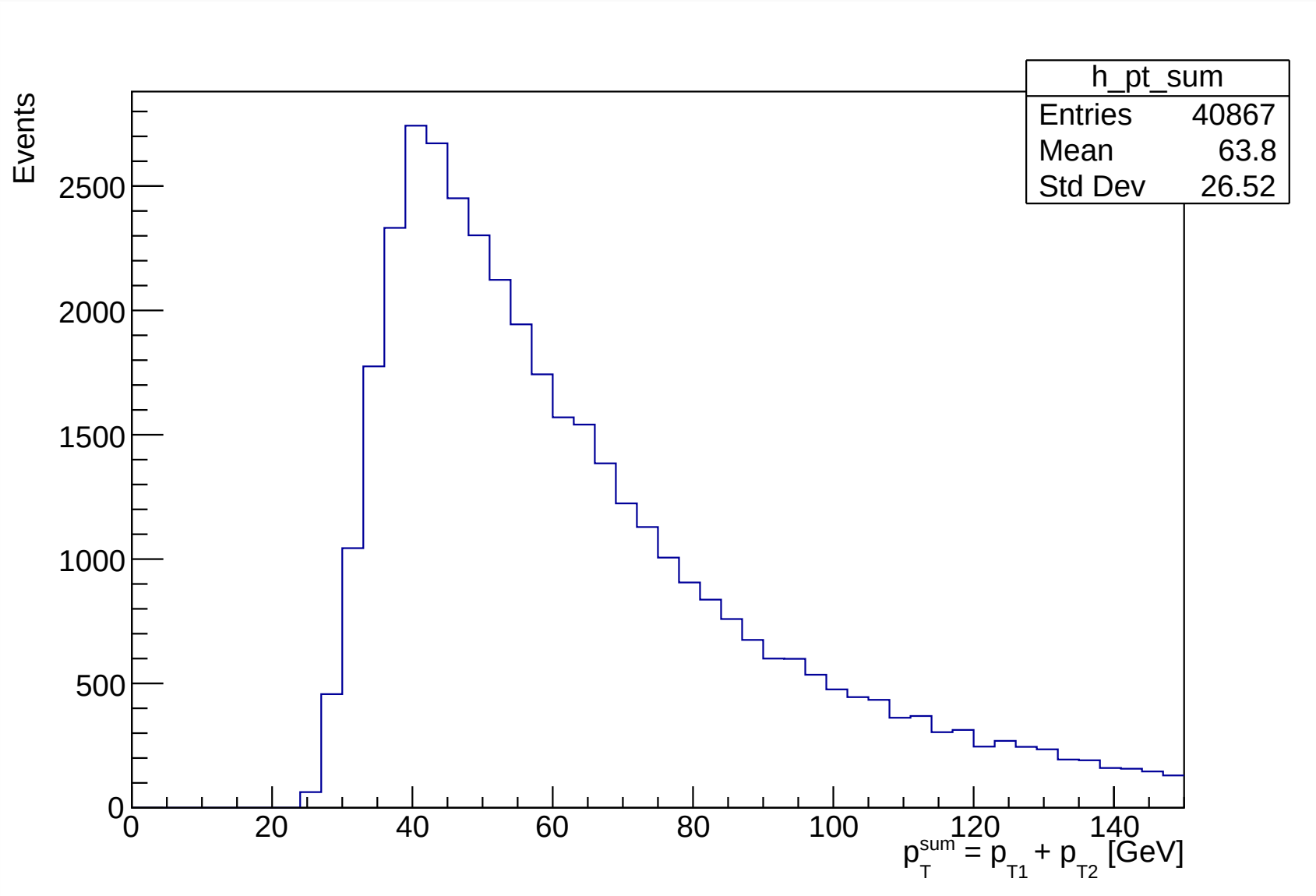
Task 2 – RDataFrame



Task 2 – RDataFrame

```python
#!/usr/bin/env python3
import ROOT

df = ROOT.RDataFrame("ntuple", "test.root")
df = df.Define("pt_sum", "pt1 + pt2")

# Event selection
df_sel = df.Filter("pt1 > 20 && iso2 > 200")

h_pt_sum = df_sel.Histo1D(("h_pt_sum", ";p_{T}^{sum} = p_{T1} +
↪  p_{T2} [GeV];Events", 50, 0, 150), "pt_sum")

c = ROOT.TCanvas()
h_pt_sum.Draw()
c.SaveAs("example3.pdf")
```

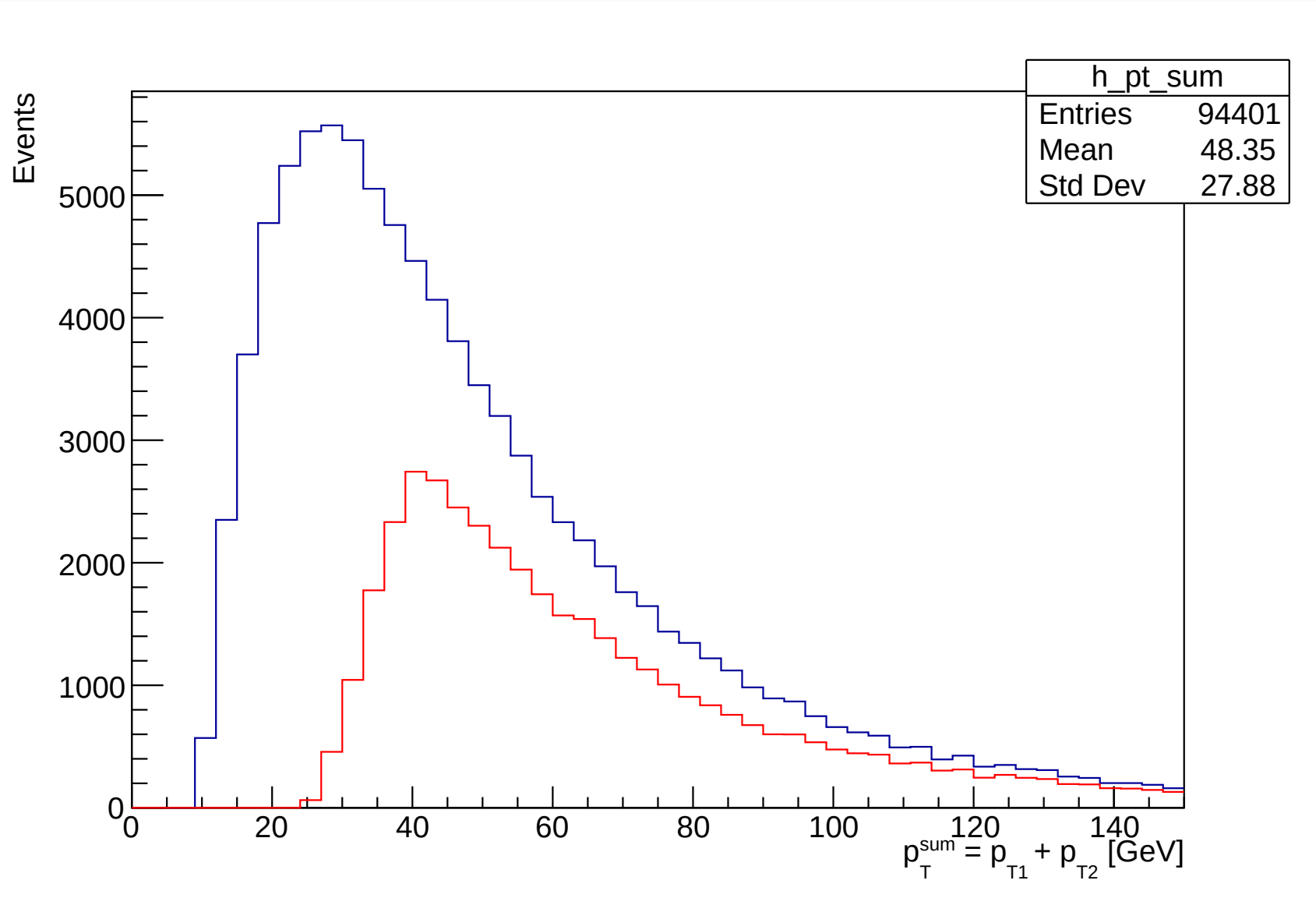
Task 2 – RDataFrame



Task 2 – RDataFrame

```python
#!/usr/bin/env python3
import ROOT

df = ROOT.RDataFrame("ntuple", "test.root")

df = df.Define("pt_sum", "pt1 + pt2")
df_sel = df.Filter("pt1 > 20 && iso2 > 200")
h_pt_sum = df.Histo1D(("h_pt_sum", ";p_{T}^{sum} = p_{T1} +
↪  p_{T2} [GeV];Events", 50, 0, 150), "pt_sum")
h_pt_sum_sel = df_sel.Histo1D(("h_pt_sum_sel", ";p_{T}^{sum} =
↪  p_{T1} + p_{T2} [GeV];Events", 50, 0, 150), "pt_sum")

c = ROOT.TCanvas()
h_pt_sum.Draw()

# Draw on top
h_pt_sum_sel.Draw("same")

# Change line color
h_pt_sum_sel.SetLineColor(2)

c.SaveAs("example4.pdf")
```

Task 2 – RDataFrame



Task 2 – RDataFrame

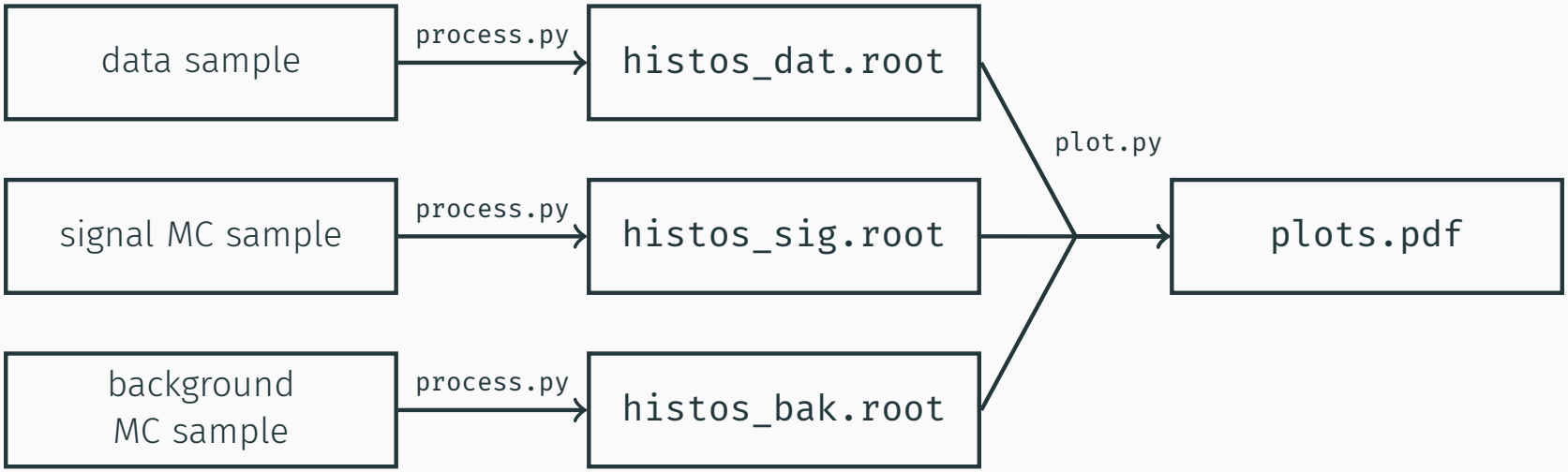
```python
#!/usr/bin/env python3

import ROOT

df = ROOT.RDataFrame("ntuple", "data.root")
df = df.Define("pt_sum", "pt1 + pt2")
df_sel = df.Filter("pt1 > 20 && iso2 > 200")
h_pt_sum = df.Histo1D(("h_pt_sum", ";p_{T}^{sum} = p_{T1} +
↪  p_{T2} [GeV];Events", 50, 0, 150), "pt_sum")
h_pt_sum_sel = df_sel.Histo1D(("h_pt_sum_sel", ";p_{T}^{sum} =
↪  p_{T1} + p_{T2} [GeV];Events", 50, 0, 150), "pt_sum")

# Save histograms to output file
f = ROOT.TFile("example5.root", "recreate")
h_pt_sum.Write()
h_pt_sum_sel.Write()
f.Close()
```

Task 2 – RDataFrame

```python
#!/usr/bin/env python3

import ROOT

# Open file
f = ROOT.TFile("example5.root")

# Get histograms
h_pt_sum = f.Get("h_pt_sum")
h_pt_sum_sel = f.Get("h_pt_sum_sel")

# Save histogram to output file
c = ROOT.TCanvas()
h_pt_sum.Draw()
h_pt_sum_sel.Draw("same")
h_pt_sum_sel.SetLineColor(2)
c.SaveAs("example5.pdf")
```

Change the line style and width.

Task 3 – TESHEP Analysis Framework



Make configuration: just run **`make`**

Task 3 – Makefile

```
SERVER=https://rafal.web.cern.ch/teshep2025/

all: plots.pdf

download: data.root sig.root bkg.root

data.root:
	wget $(SERVER)/data.root

sig.root:
	wget $(SERVER)/sig.root

bkg.root:
	wget $(SERVER)/bkg.root

hist_sig.root: process.py sig.root
	./process.py sig.root hist_sig.root --MC

hist_bkg.root: process.py bkg.root
	./process.py bkg.root hist_bkg.root --MC

hist_data.root: process.py data.root
	./process.py data.root hist_data.root

plots.pdf: hist_data.root hist_sig.root hist_bkg.root
	./plot.py hist_data.root hist_sig.root hist_bkg.root plots.pdf
```

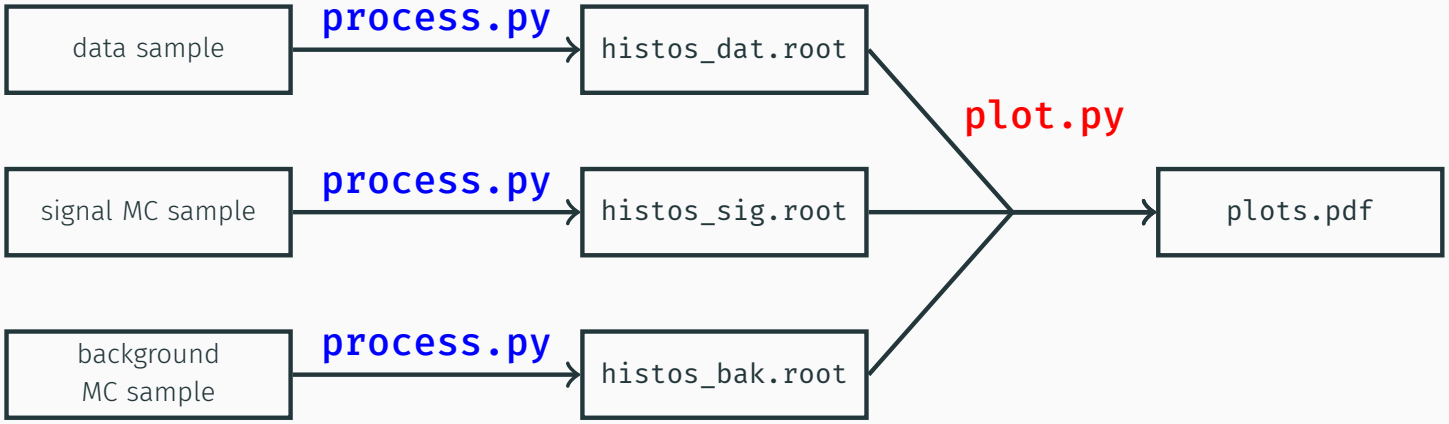
Task 3 – automated creation of histograms – `process.py`

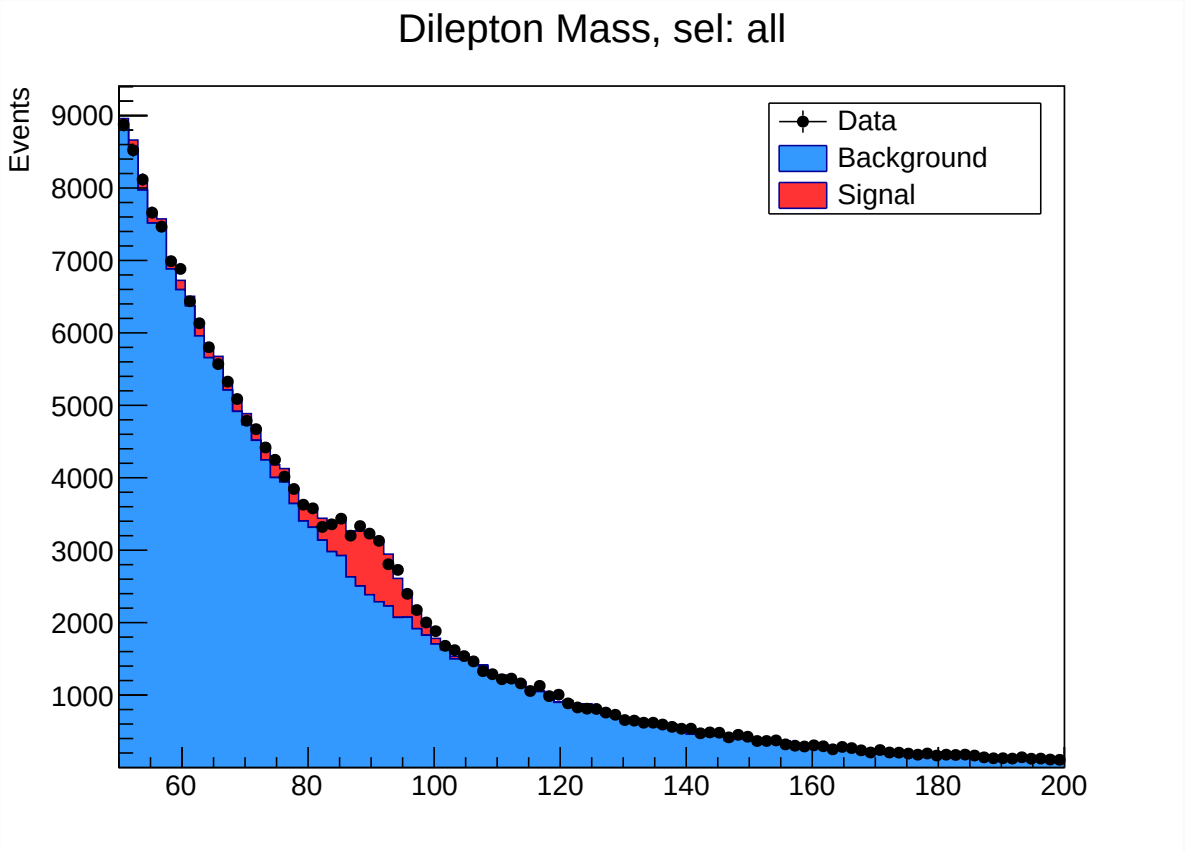
```
ROOT.gInterpreter.Declare('#include "Math/Vector4D.h"')
ROOT.gInterpreter.Declare("""
float compute_mass(float pt1, float eta1, float phi1,
                   float pt2, float eta2, float phi2) {
    ROOT::Math::PtEtaPhiMVector v1(pt1, eta1, phi1, 0.000511);
    ROOT::Math::PtEtaPhiMVector v2(pt2, eta2, phi2, 0.000511);
    return (v1 + v2).M();
}
""")

# Load input file and tree
df = ROOT.RDataFrame("ntuple", input_file)
df = df.Define("mass", "compute_mass(pt1, eta1, phi1, pt2, eta2,
↪  phi2)")
if isMC:
    df = df.Define("true_mass", "compute_mass(true_pt1, true_eta1,
    ↪  true_phi1, true_pt2, true_eta2, true_phi2)")
```

Task 3 – automated creation of histograms – `process.py`

```python
# Define selection criteria
variables = {
    "pt1":        ("Lepton 1 p_{T}", "p_{T1} [GeV]", "Events", 100, 0, 200),
    "pt2":        ("Lepton 2 p_{T}", "p_{T2} [GeV]", "Events", 100, 0, 200),
    "iso1":       ("Lepton 1 Isolation", "Isolation_{1}", "Events", 100, 0, 1000),
    "iso2":       ("Lepton 2 Isolation", "Isolation_{2}", "Events", 100, 0, 2000),
    "w_shower1":  ("Shower Width 1", "w_{shower1}", "Events", 100, 0, 0.1),
    "w_shower2":  ("Shower Width 2", "w_{shower2}", "Events", 100, 0, 0.1),
    "mass":       ("Dilepton Mass", "M_{ee} [GeV]", "Events", 20, 80, 100),
    "true_mass":  ("True dilepton Mass", "M_{ee} [GeV]", "Events", 20, 80, 100),
}

selections = {
    "all": ("", ["mass", "true_mass", "iso1", "w_shower1"]),
    "test": ("iso1 > 300 && iso2 < 300", ["mass"]),
}
```

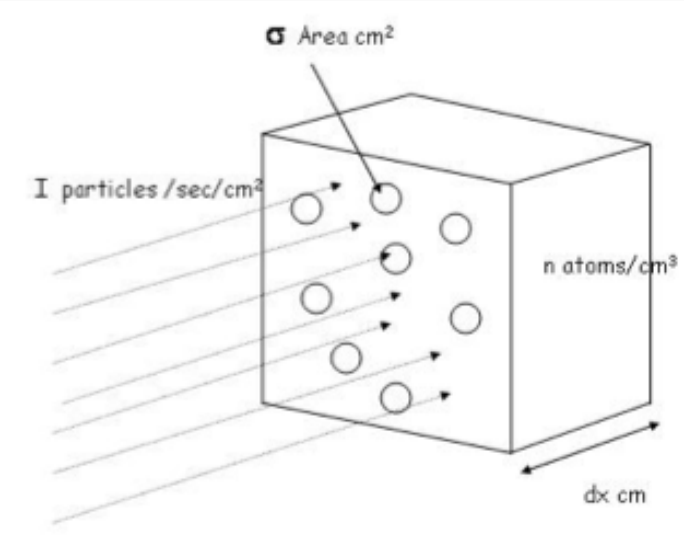
Task 3 – automated plotting





Task 3 – cross section and luminosity

Cross section: measure of the probability of a process



$$N = \sigma \cdot L$$

Instantaneous luminosity: measures of data-taking rate. For colliders:

$$\mathcal{L} = \frac{N_1 N_2 f n_b}{4\pi\sigma_x\sigma_y}$$

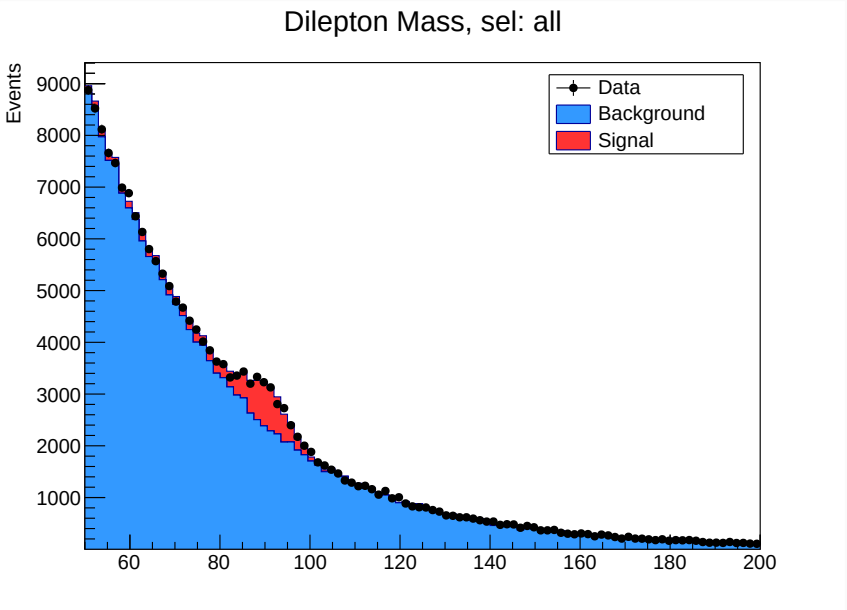
Integrated luminosity: measure of amount of data collected:

$$L = \int \mathcal{L}(t)\,\mathrm{d}t$$

Task 3 – comparing MC to data – normalization

Data sample

- Histogram directly shows number of events



MC samples

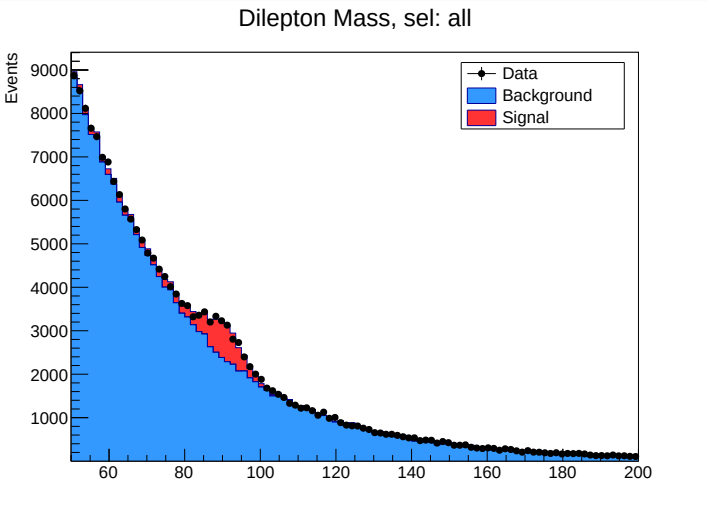
- Stacked on top of each other
- We want the histogram to show the **expected** number of events for the **luminosity of the data sample**
- Have to be appropriately normalized (scaled):

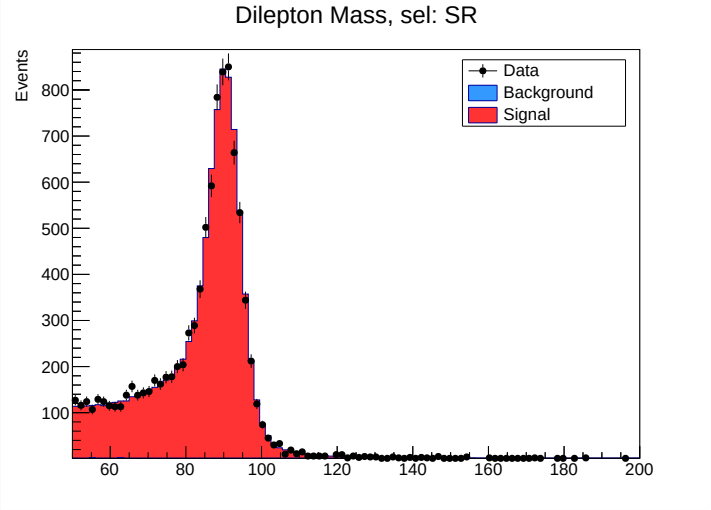
$$n_i \quad \rightarrow \quad n_i \cdot \frac{\sigma L}{\sum_i n_i}$$

Note 1: In some cases, the vertical scale is not ”Events”, but $\mathrm{d}\sigma/\mathrm{d}\,\mathrm{sth}$.
Note 2: MC events may have weights. Then, scaling is (a little) more complicated.

Task 3 – define the signal region (SR)

- Look at distributions of different variables
- Define selection that will remove background events while keeping signal events





How to get source files

You will get instruction sheets.
Please follow them.

It is encouraged to work in pairs

Do not hesitate to contact us:

- if you are stuck at any point
- the tasks are trivial to you and you would prefer something more difficult

TESHEP 2025 Hands-on Exercises

Task 0 – check if your ROOT installation works

- Download `check.py`, for example
  `wget https://rafal.web.cern.ch/teshep2025/check.py`
- Execute:
  `python3 check.py`
- If it works, it should display:
  `Info in <TCanvas::Print>:  file check.png has been created`

Task 1 – manual inspection of ROOT data file

- Download `test.root`, for example:
  `wget https://rafal.web.cern.ch/teshep2025/test.root`
- Show how to open file with TBrowser and see distributions
  `root --web=off test.root`
  `new TBrowser`
- Inspect what variables are available and what distributions they have
- Run StartViewer from TBrowser or from command line:
  `ntuple->StartViewer()`
- Make a correlation plot between two variables

Task 2 – RDataFrame

- Get the tarball with files:
  `wget https://rafal.web.cern.ch/teshep2025_examples.tar.gz`
- Unpack
  `tar xvzf teshep2025_examples.tar.gz`
- Copy `test.root` to the `teshep2025_examples` directory
- Run each example, for example
  `python3 example1.py`
- Open the produced plot
- Look at each code and see if you can understand what it does
- In order to make sure you understand:
  - In example 1 – add 2 more histograms
  - In example 2 – add histogram of histogram of the e^+e^- pair transverse momentum $|\vec{p}_{T,1}+\vec{p}_{T,2}|$. Keep in mind that $|\vec{p}_{T,1}+\vec{p}_{T,2}| \neq p_{T,1}+p_{T,2} = |\vec{p}_{T,1}|+|\vec{p}_{T,2}|$.
  - In example 3 – add a requirement of $p_{T,2}$ being above 50 GeV
  - In example 4 – change line style (check ROOT documenation, Google or ChatGPT)
  - In example 5 – produce two more plots with one histogram on each

Task 3 – data & MC analysis

- Get the TESHEP analysis framework:
  `wget https://rafal.web.cern.ch/teshep2025_framework.tar.gz`
  `tar xvzf teshep2025_framework.tar.gz`
  `cd teshep2025_framework`
  `make`
- Open `plots.pdf`
- Look at distributions of different variables
- Define selection that will remove background while keeping signal

Acknowledgments

RS is supported in part by the National Science Center (NCN), Poland, SONATA BIS grant no. 2021/42/E/ST2/00350.