PyROOT ile Veri Analizi Temelleri

TH ve TF Sınıf Aileleri

TH SINIFI

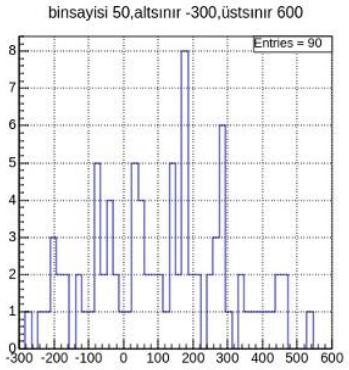
- Dışarıdan aldığı verilerin histogramlarını çizdirir.
- **ROOT.TH1D():** 1 boyutlu histogram çizdirmek için kullanılır
- **ROOT.TH2D():** 2 boyutlu histogram çizdirmek için kullanılır
- **ROOT.TH3D():** 3 boyutlu histogram çizdirmek için kullanılır

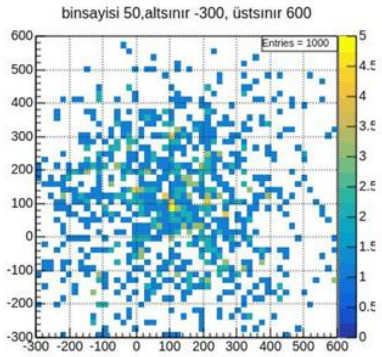
TF SINIFI

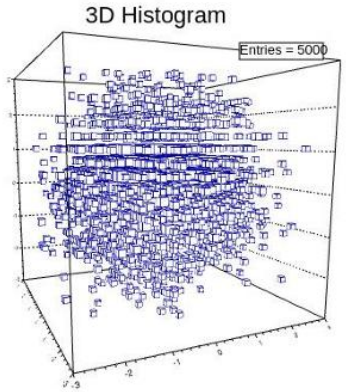
- Matematiksel fonksiyonların grafiğini çizdirir.
- **ROOT.TF1():** Tek değişkenli fonksiyonların grafiğinitek boyutta çizdirmek için kullanılır
- **ROOT.TF2():** İki değişkenli (z=f(x,y)) fonksiyonların grafiğini iki boyutta çizdirmek için kullanılır. İkinci değişkeni sıfır varsayarak tek değişkenli fonksiyonların grafiğini de iki boyutta çizdirebilir.
- **ROOT.TF3():** Üç değişkenli (f(x,y,z)) fonksiyonların grafiğini üç boyutta çizdirmek için kullanılır.

Grafik Örnekleri

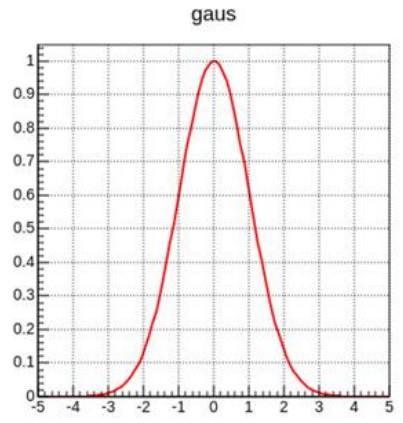
TH SINIFI

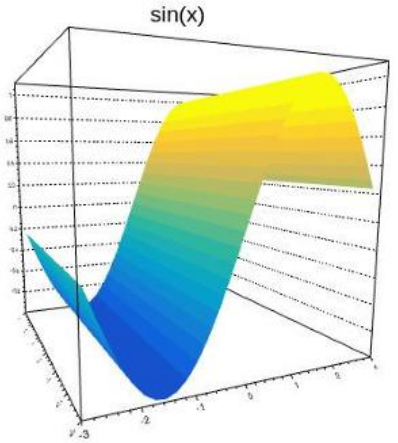


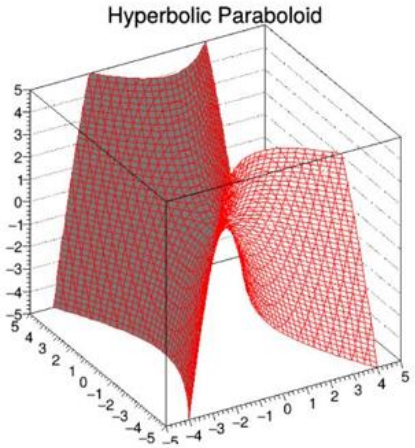


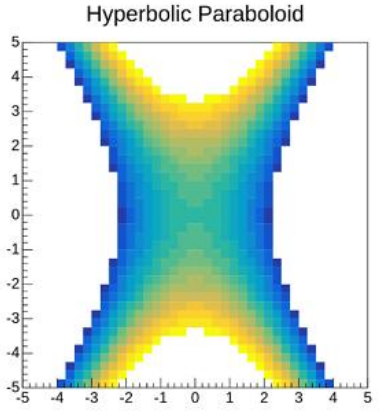


TF SINIFI



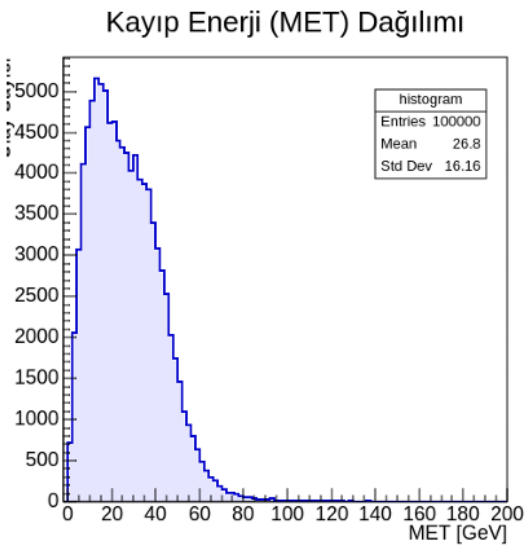


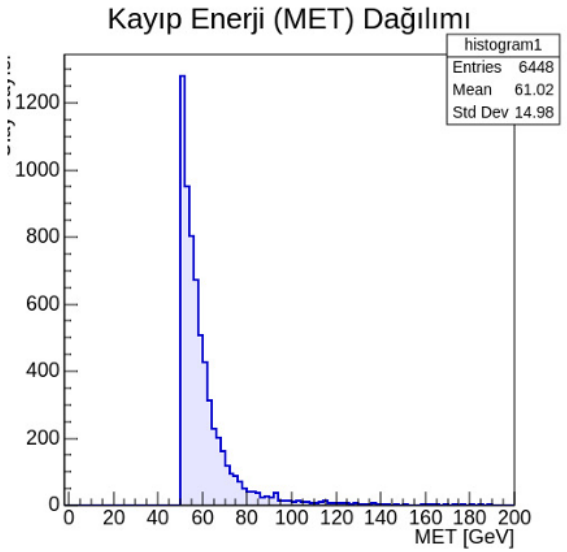


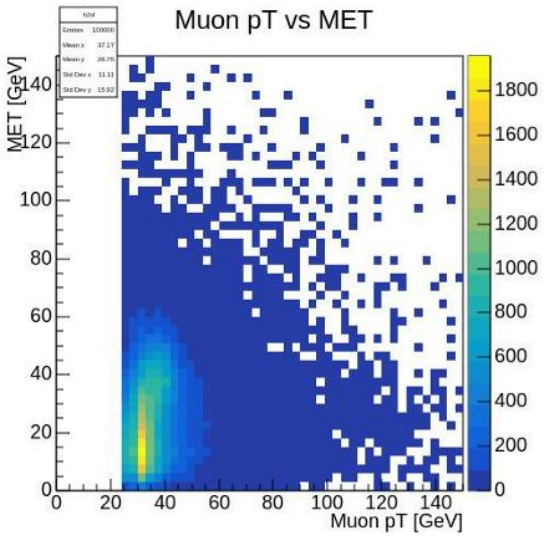


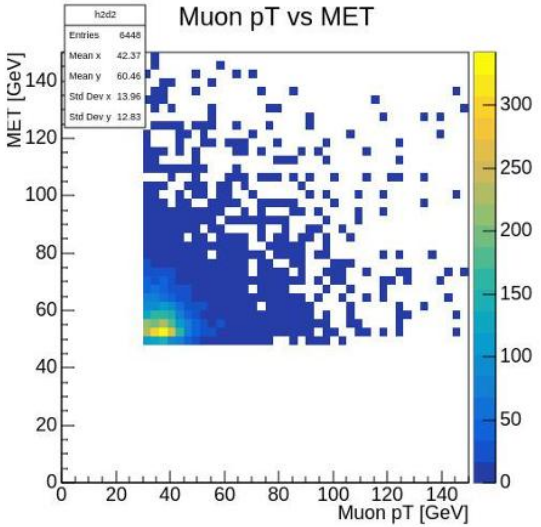
Kinematik Kesim

“W to muon and neutrino 2011” by McCauley, Thomas from CERN Open Data









Filtrelenmiş ve filtrelenmemiş veri.
Uygulanan filtre: (pt)>30 GeV ve MET>50 GeV

- Yüksüz ve etkileşimsiz parçacıklar dedektör malzemesiyle elektromanyetik etkileşime giremedikleri için dedektörde iz bırakmazlar.
- Bu parçacıklar, enine düzlemde momentumun korunduğu bilgisi sayesinde MET (Missing Transverse Energy) yani Kayıp Enine Enerji değeri ile araştırılır.
- Çarpışmanın yaşandığı boyuna ekseni dik kesen enine eksende, çarpışma öncesi ve sonrasında toplam momentumun sıfır kalması gerekir.
- Çarpışma sonucu parçacıkların enine momentumlarının toplamı sıfırdan farklı çıktığı durumlarda kayıp olan momentuma MET adı verilir.

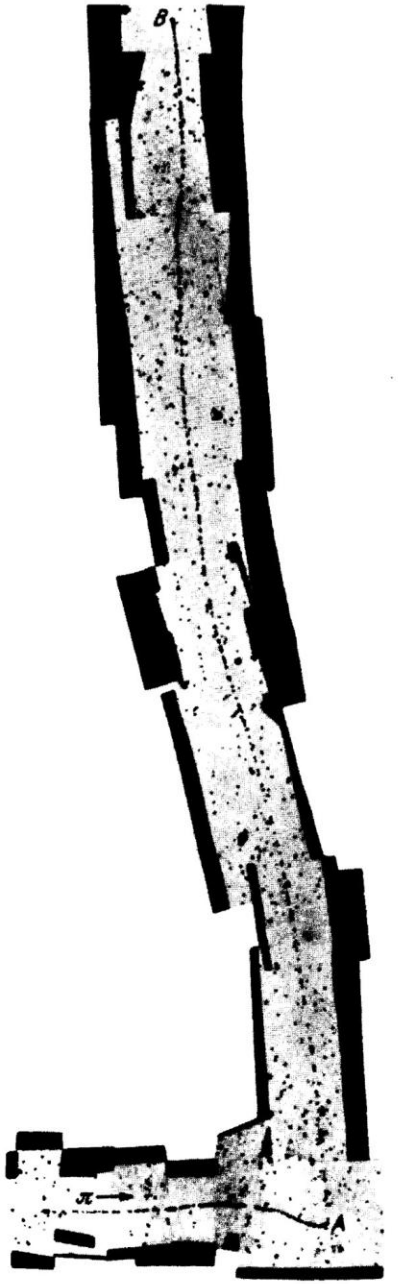
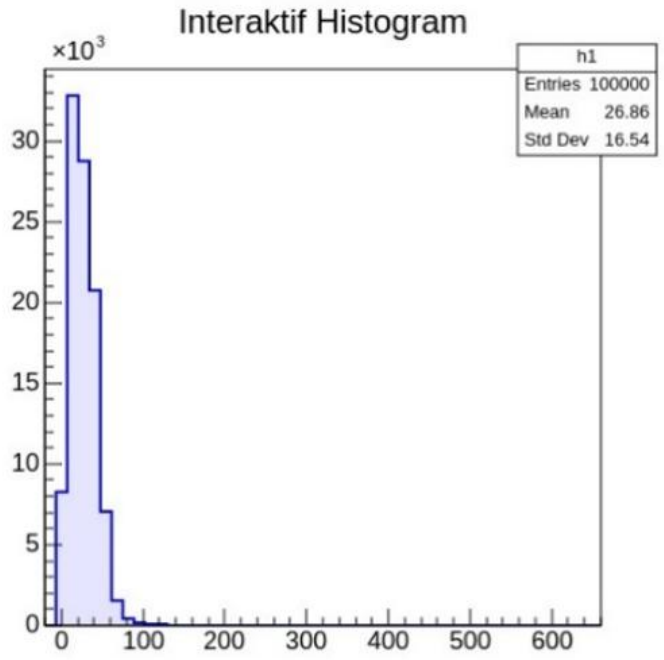


Figure 1.4 One of Powell's earliest pictures showing the track of a pion in a photographic emulsion exposed to cosmic rays at high altitude. The pion (entering from the left) decays into a muon and a neutrino (the latter is electrically neutral, and leaves no track). Reprinted by permission from C. F. Powell, P. H. Fowler, and D. H. Perkins, *The Study of Elementary Particles by the Photographic Method* (New York: Pergamon, 1959). First published in *Nature* **159**, 694 (1947).

İnteraktif Histogram

1 boyutlu histogramda x ekseninin limitleri ve bin sayısının kullanıcı tarafından dışarıdan belirlenmesi




```
bin_sayisi = int(input("Bin sayısını girin : "))
x_min = float(input("X Minimum değerini girin : "))
x_max = float(input("X Maksimum değerini girin ): "))
c1=r.TCanvas("canvas","Cern Open Data Analizi",500,500)
h1=r.TH1D("h1","Interaktif Histogram",bin_sayisi, x_min, x_max)

h1.SetCanExtend(r.TH1.kAllAxes)

for deger in df['MET']:
    h1.Fill(deger)
h1.SetLineColor(r.kBlue + 1)
h1.SetLineWidth(2)
h1.SetFillColorAlpha(r.kBlue, 0.1)
h1.Draw()
c1.Draw()
```

```
Bin sayısını girin :  50
X Minimum değerini girin :  -20
X Maksimum değerini girin ):  150
```

Değişmez Kütle (Invariant Mass) Hesabı

$$\mathrm{M} = \sqrt{2p_{T1}p_{T2}(\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2))}$$

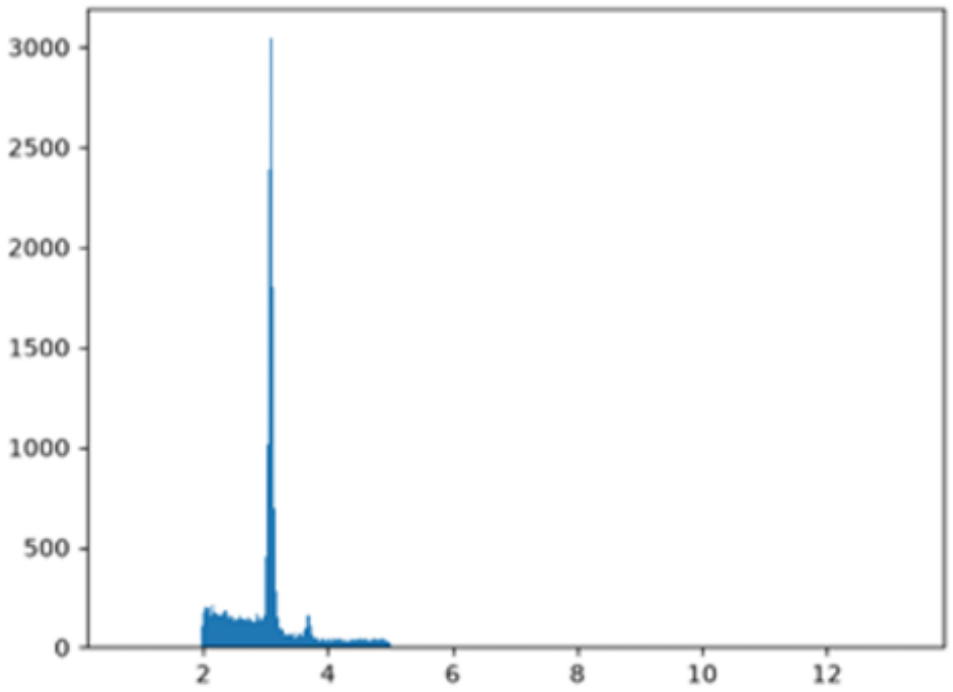
- Özel görelilikte enerji ve momentum gözlemcinin hızına bağlı olarak değişir. Ancak bu ikisinin kombinasyonu olan invariant mass, tüm referans sistemlerinde aynı kalır.
- Kısacası gözlemcinin hareketinden etkilenmeyen değişmez bir kütle ölçeği sunması ve özel görelilik ilkeleriyle uyumlu olması bu parametrenin analizlerde kullanmak için tercih edilmesini sağlar.
- Sinyal ve arka plan gürültüsünün ayrılmasında büyük rol oynar. Bozunma sonucu oluşan ürünlerin histogramı çıkarıldığında arka plan gürültüsünü oluşturan rastgele süreçler yayvan bir eğri oluştururken gerçek bir parçacığın üretildiği yerde sivri bir tepe/pik oluşur. Bu pik noktası, o parçacığın kütlesini ele verir.

Event Weight

- Invariant mass histogramlarının event sayılarının ağırlık katsayısı ile çarpılmasıyla uygulanan normalize etme işlemidir. Temel olarak üç ana amaçla uygulanır:
- Simülasyon verilerinin olay sayılarını deney verilerinin olay sayılarına normalize etmek: Simülasyonda, gerçekte yaşanandan çok daha fazla sayıda event görülebilir. Simülasyonda üretilen olay sayılarına ,olayların gerçekleşme ihtimaline (cross-section) doğru orantılı bir şekilde veriye weight katsayıları uygulanır.

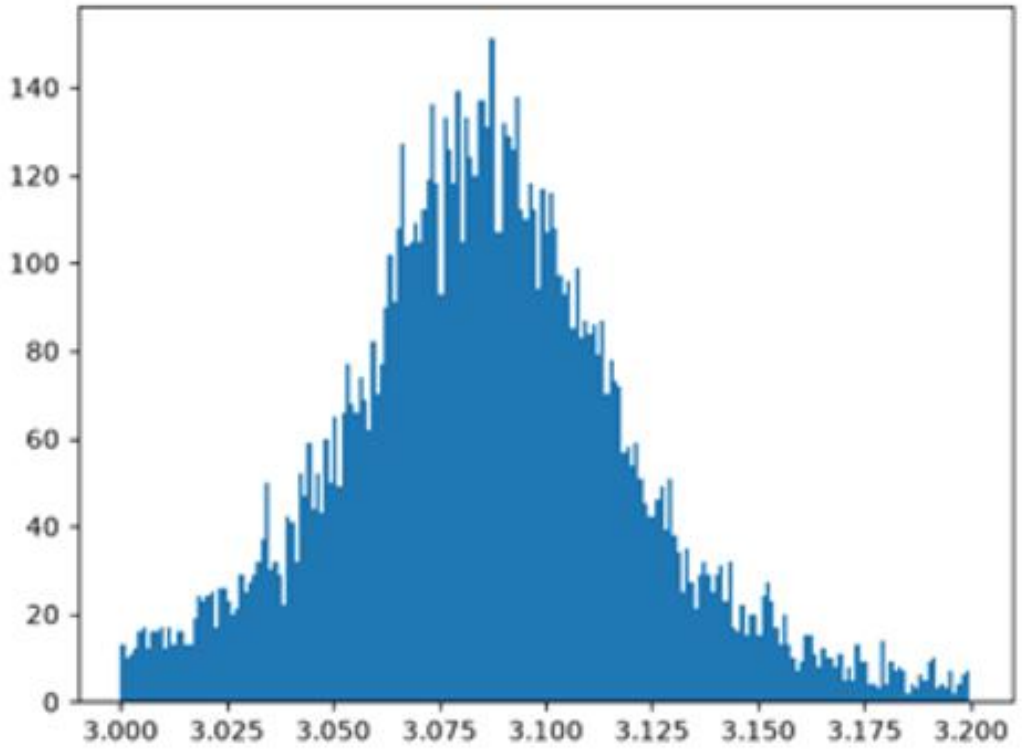
- Dedektör verimlilik düzeltmeleri (scale factors): Gerçek dedektörler simülasyonların aksine her parçacığı kusursuz yakalayamaz, bu sebeple simülasyondaki verimlilik ile dedektörün verimliliği arasındaki farkı kapatmak için her event, scale factor (düzeltme faktörü) ile çarpılır.
- Monte Carlo simulasyonlarındaki teorik belirsizlikleri, hataları ve eksiklikleri hesaplayıp hata bantlarını hesaplamak için kullanılır.

İstatistiksel Analiz



Invariant Mass Histogramı

Histogramın tamamına bakmak, tüm spektrumda bir uyumsuzluk olup olmadığını anlamamıza yardımcı olur. Verinin düzgün fit edilmesinde ve simülasyon verilerinin gerçek veriyi ne kadar iyi temsil ettiğini test etmede işe yarar.



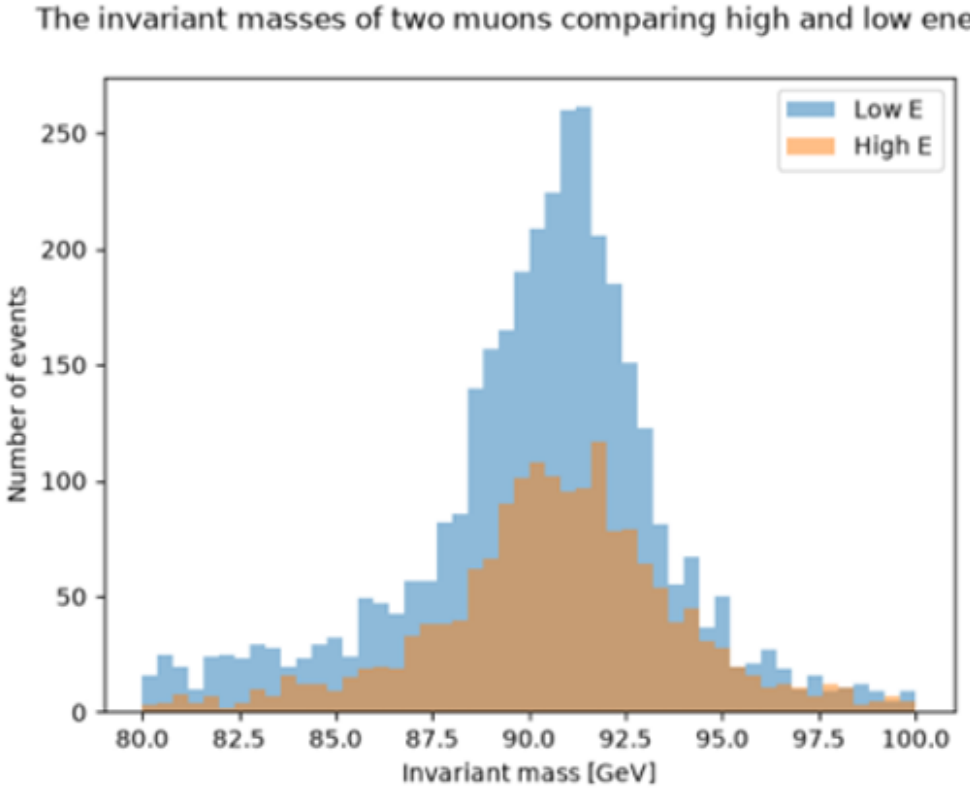
Invariant Mass Histogramının Tepe Noktası

Doğrudan aradığımız parçacığın kaç adet üretildiğini verir. Pik bölgesine özel olarak uygulanan fit işlemleri ise invariant mass ve dedektör çözünürlüğünün hassas bir şekilde ölçülmesini sağlar.

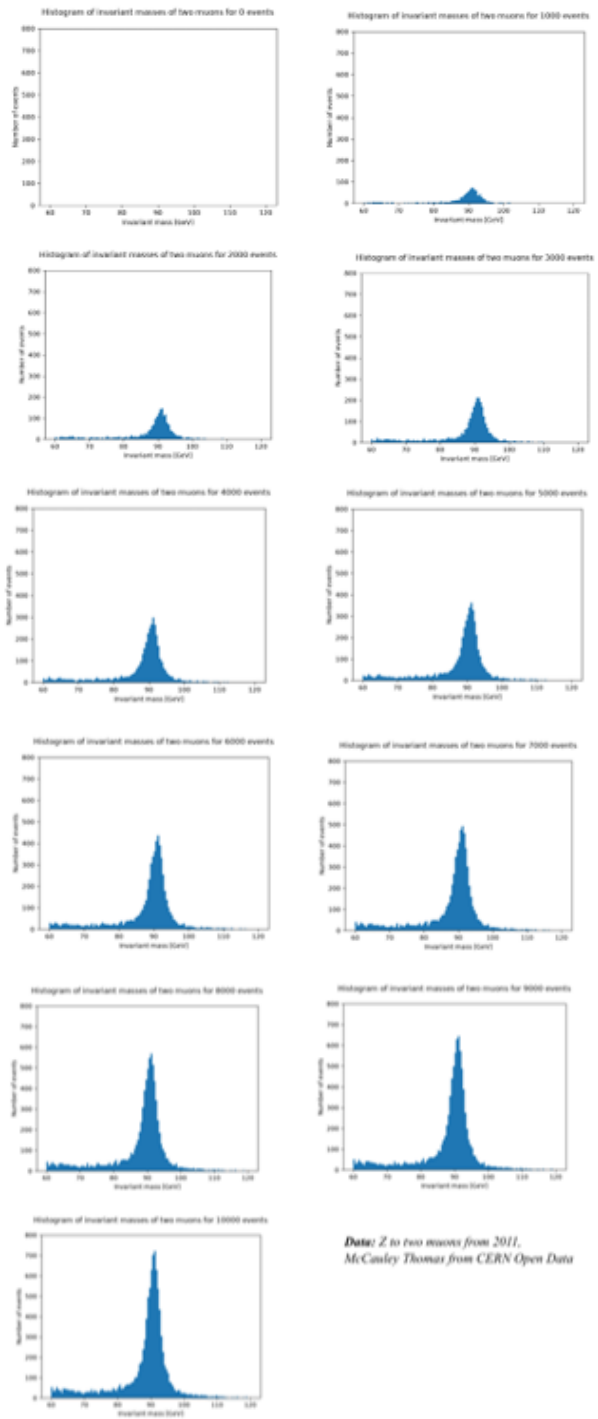
	Mean	**Variance**	**Standard Deviation**
Masses	3.0843736344537818	0.3921691908895595	0.6262341342417862
Peak masses	3.09342552933628	0.0012908143684853 98	0.03592790515025052

Üst Üste Bindirilmiş Histogramlar

- Histogramları kıyaslama amaçlı kullanılır.



150 GeV sınırının altı düşük enerji bölgesi, üstü yüksek enerji bölgesi alınarak histogramdaki tepe noktalarının kıyaslanması



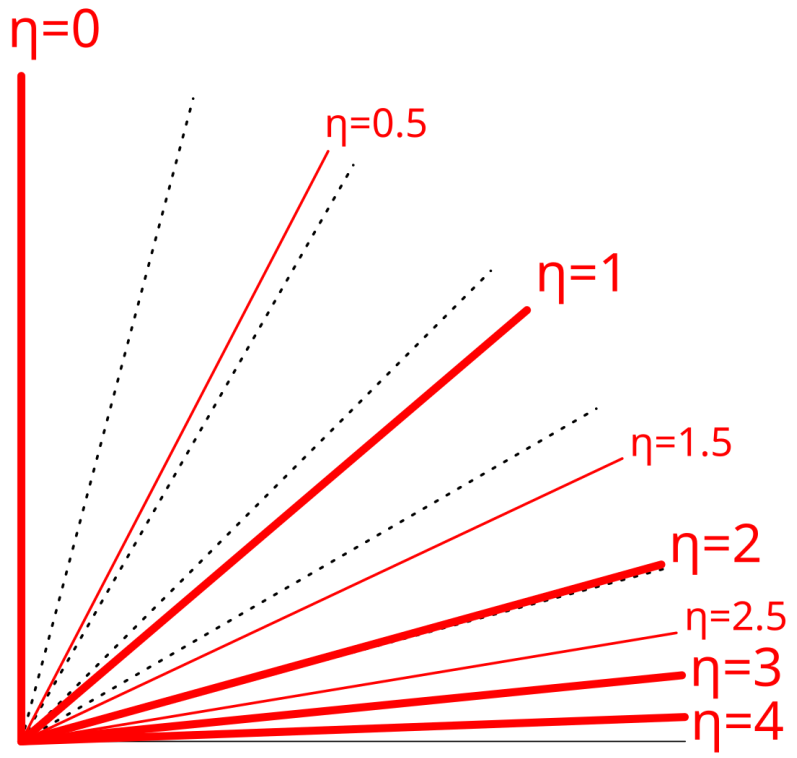
Data: Z to two muons from 2011, McCauley Thomas from CERN Open Data

Data Selection

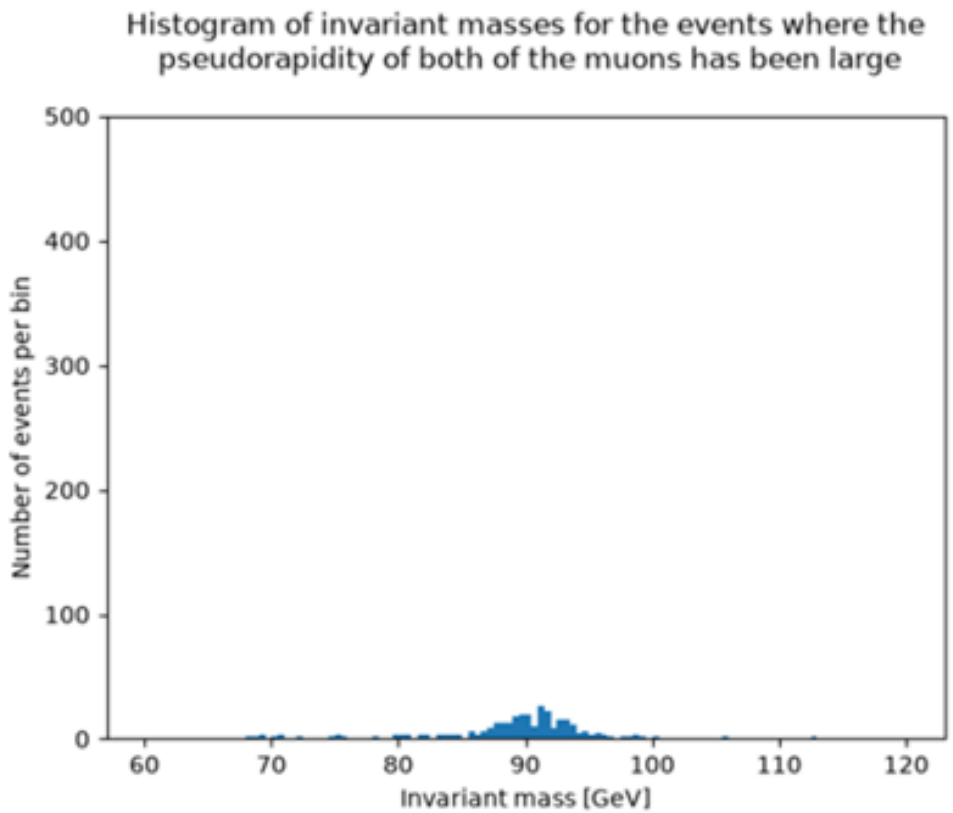
Olay sayısını aşama aşama arttırarak parçacık keşfi yapabilmek için kaç olaya ihtiyaç duyduğumuzu anlamak ve gürültünün anlamlı sinyalden ayrıştığı noktayı görmek.

Pseudorapidity Çözünürlüğü Analizi

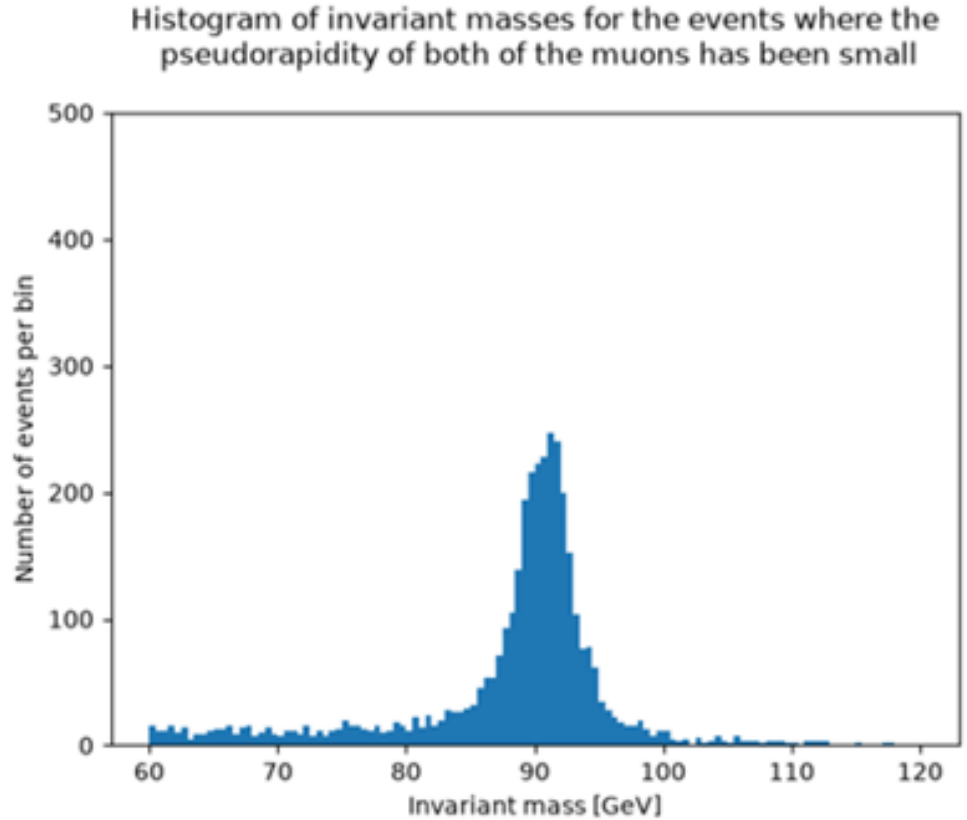
Yalancı hızlılık çözünürlüğü, dedektörün ileri ve uç bölgelerinde ne miktarda açısal duyarlılık kaybı yaşandığını belirlemek için kullanılan bir uzaysal koordinattır. Eksen ve parçacık arasındaki açıyı tanımlar. Açı daraldıkça sapmalar artar.



Histogram for large η events (|η | > 1.52)



Histogram for small η events (|η | < 0.45)

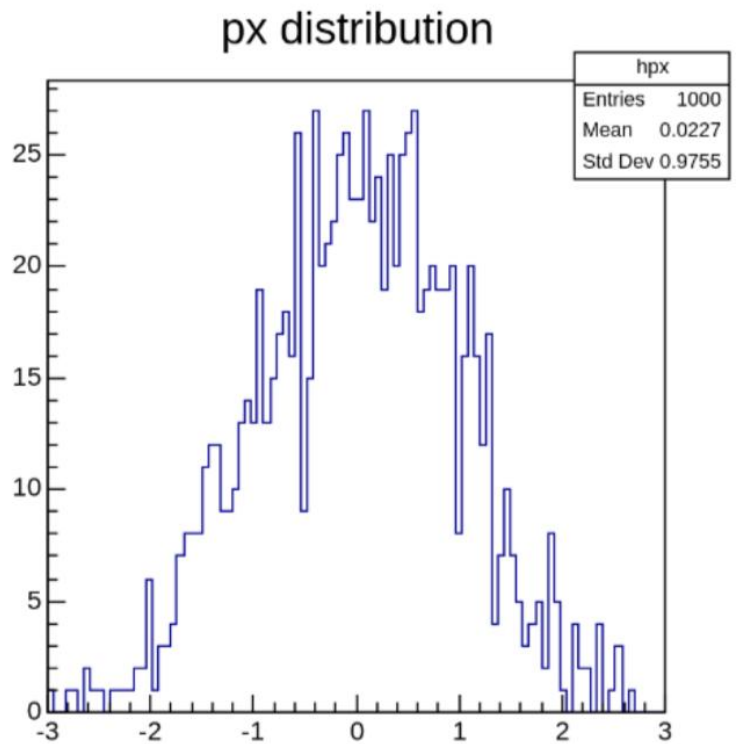


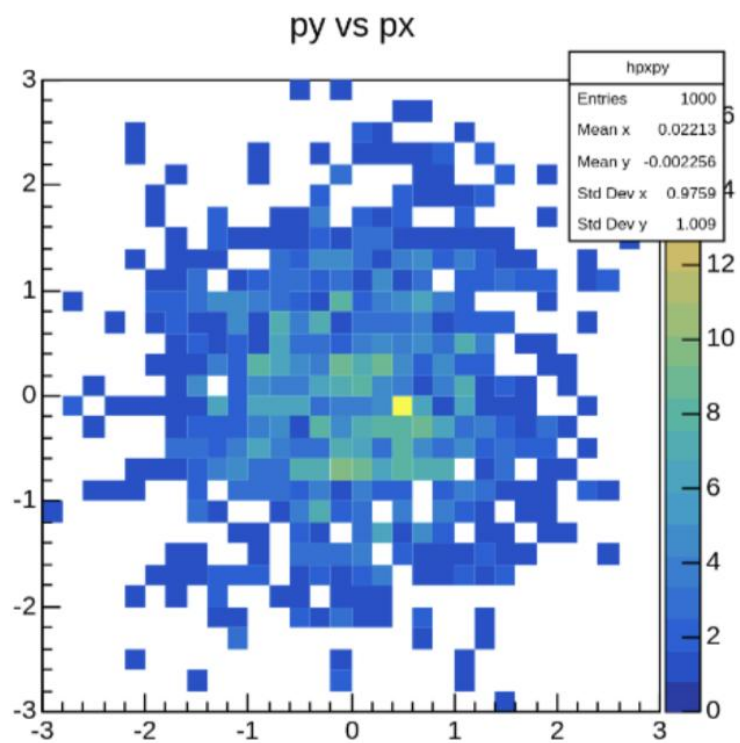
TTree Yapıları

Sıfırdan Ttree yapısı oluşturmak:

1. ROOT.Tfile() metodu ile yeni bir dosya oluşturuldu
2. ROOT.Ttree() metoduyla bir TTree objesi tanımlandı
3. Array() fonksiyonuyla branch tanımlandı ve Branch() metoduyla tree yapısına eklendi. (Bu örnekte tanımlanan branchler px ve py)
4. Bu örnekteki eventler for döngüsü içerisinde numpy.random.normal() kullanılarak rastgele üretildi.
5. Dosya Close() metoduyla kapanarak ROOT.TFile.Open() metoduyla okuma modunda tekrar açıldı.
6. Tanımlanan tree objesi Get() metoduyla geri çağırılarak bir değişkene atandı.
7. Event sayısı tree.GetEntries() metoduyla yazdırılarak bir değişkene atandı.
8. Eventler bir for döngüsü içerisinde tree.GetEntries(i) metodu kullanılarak hafızaya aktarıldı.
9. 1 boyutlu px ve 2 boyutlu px py histogramları çizdirilidi.

1D histogram of px distribution and 2D histogram of py vs px distrubiton. Data is randomly generated using numpy library:





Csv uzantılı veriyi Ttree yapısına dönüştürmek:

1. Veri dosyası bir Pandas DataFrame'ine yüklendi
2. DataFrame objesi ROOT.RDF.FromPandas() metoduyla RDataFrame formatına dönüştürüldü.
3. Veri, Snapshot() metoduyla bir root dosyası içerisine Ttree formatında kaydedildi.

i- Data format before as a csv file:

```
          Run      Event       pt     eta     phi  Q     chiSq     dxy  \
0      173389  489963747  29.3153 -0.1393  1.3561  1    0.8456 -0.0600
1      173389  490024840  44.1461 -1.3590 -0.5390  1    1.2943 -0.0126
2      173389  490102189  42.7895 -1.0193  2.3390 -1    0.8078 -0.0840
3      173389  490092121  31.2215 -0.1270 -0.4889 -1  109.9490  0.0723
4      173389  490193981  36.9668  1.8557  1.3772  1    1.0597 -0.0676
...       ...        ...     ...     ...     ... ..       ...     ...
99995  171369   55073997  89.2292  1.1032  0.7624 -1    0.5443 -0.0168
99996  171369   55221533  37.6914 -1.3926 -0.6669 -1    0.6061  0.0832
99997  171369   55273989  31.2467 -0.1335  0.3830 -1    0.8857  0.0592
99998  171369   55354925  42.7469 -0.3049 -0.5734 -1    1.4464  0.0818
99999  171369   55346997  36.4015  2.0057  2.0813  1    1.3507 -0.0880

            iso      MET  phiMET
0        0.0000  30.6670 -2.1308
1       26.3423  27.9521 -2.6686
2        2.9748   5.1565  0.3061
3       56.4897   4.2143 -0.1166
4       41.4910  20.9963 -1.6251
...         ...      ...     ...
99995    0.2016  19.2938  2.3639
99996    0.5965   7.5065  2.1850
99997   23.6527  23.0953 -3.1020
99998   24.3723  16.4557  2.1887
99999    0.5617  48.3986 -1.2394

[100000 rows x 11 columns]
```

ii- Data format after converting to a ROOT TTree:

```
Toplam Satır Sayısı: 100000
******************************************************************************
*Tree    :tree      : tree                                                   *
*Entries :   100000 : Total =         8829609 bytes  File  Size =    3673467 *
*        :          : Tree compression factor =   2.40                       *
******************************************************************************
*Br    0 :Run       : Run/L                                                  *
*Entries :   100000 : Total  Size=     802645 bytes  File Size  =       7487 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression= 107.09     *
*.............................................................................*
*Br    1 :Event     : Event/L                                                *
*Entries :   100000 : Total  Size=     802705 bytes  File Size  =     388040 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   2.07     *
*.............................................................................*
*Br    2 :pt        : pt/D                                                   *
*Entries :   100000 : Total  Size=     802615 bytes  File Size  =     435303 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   1.84     *
*.............................................................................*
*Br    3 :eta       : eta/D                                                  *
*Entries :   100000 : Total  Size=     802645 bytes  File Size  =     374399 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   2.14     *
*.............................................................................*
*Br    4 :phi       : phi/D                                                  *
*Entries :   100000 : Total  Size=     802645 bytes  File Size  =     417811 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   1.92     *
*.............................................................................*
*Br    5 :Q         : Q/L                                                    *
*Entries :   100000 : Total  Size=     802585 bytes  File Size  =      56008 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=  14.32     *
*.............................................................................*
*Br    6 :chiSq     : chiSq/D                                                *
*Entries :   100000 : Total  Size=     802705 bytes  File Size  =     368358 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   2.18     *
*.............................................................................*
*Br    7 :dxy       : dxy/D                                                  *
*Entries :   100000 : Total  Size=     802645 bytes  File Size  =     303953 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   2.64     *
*.............................................................................*
*Br    8 :iso       : iso/D                                                  *
*Entries :   100000 : Total  Size=     802645 bytes  File Size  =     444986 *
*Baskets :       26 : Basket Size=      32000 bytes  Compression=   1.80     *
*.............................................................................*
*Br    9 :MET       : MET/D                                                  *
```

$t\bar{t}$ Cross-Section Measurements

CMS data analysis tutorial: Exercise 1

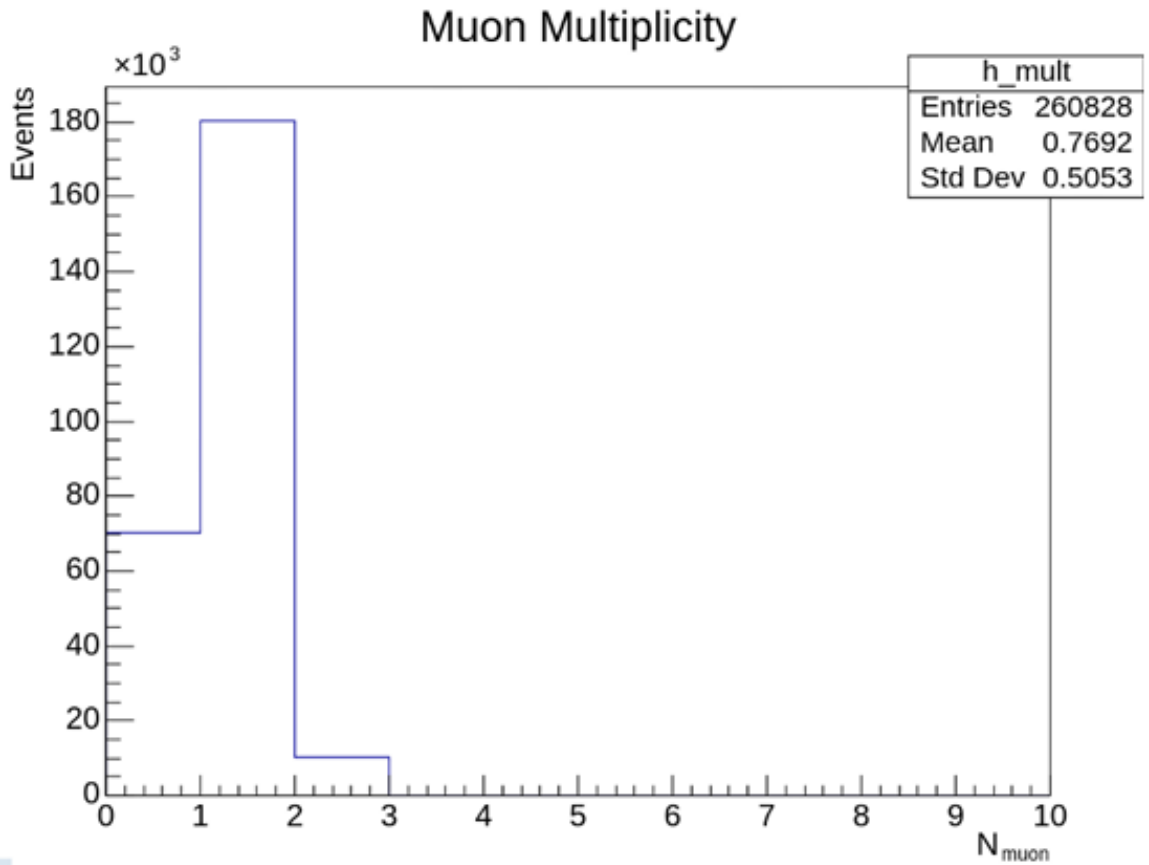
Jun.-Prof. Dr. Christian Sander, Dr. Alexander Schmidt

September 2014

1 Warmup

The trigger for this tutorial selects events which contain one or more muons as discussed in the documentation and explanation.

- Find out how often there is more than one isolated, reconstructed muon in data (histogram of the muon multiplicity)! Where could these additional muons come from?
- Calculate the invariant mass of two muons of opposite charge (manually and/or using the ROOT functionality of adding two fourvectors)! Only use isolated muons.
- Display the invariant mass distribution of two muons in a histogram (hint: try different axis ranges)!
- Compare your results to MC simulation (display simulation and data in the same histogram). Make sure you select triggered events only for the simulated samples!



N_{muon}= 0 bin has nearly 70.000 events. These are events that do not contain any isolated muons.

N_{muon}= 1 bin has nearly 180.000 events. These are events that contain a single isolated muon.

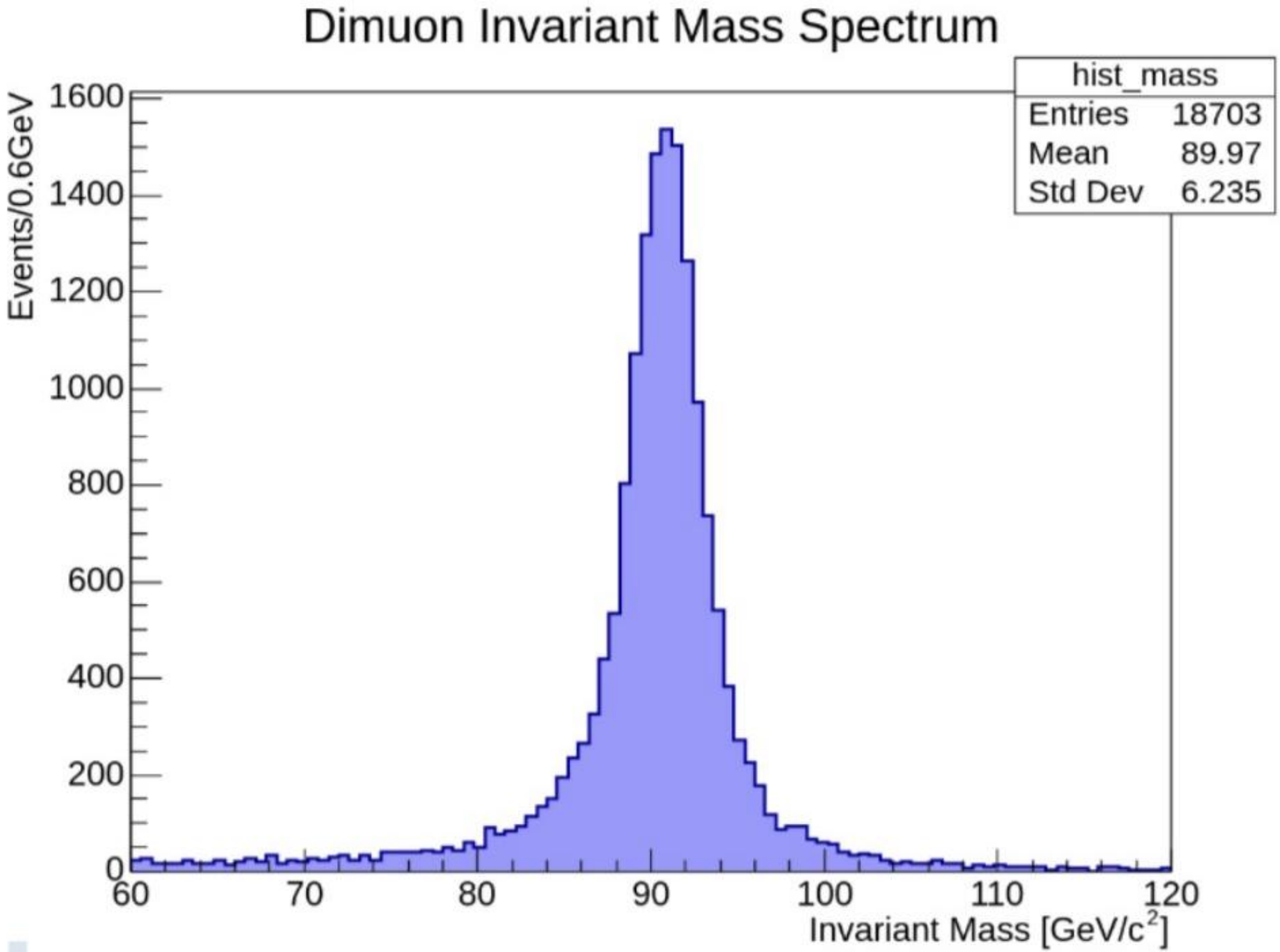
N_{muon}= 2 bin has nearly 10.000 events. These are events that contain two isolated muons.

$N_{\text{muon}} \geq 3$ bin do not contain any events.

Invariant Mass

1. ROOT.RDataFrame() ile .AsNumpy() metodu beraber kullanılarak veriden müonun x,y,z düzlemlerindeki momentumu, enerjisi, yükü ve tek çarpışmada kaç müon bulunduğu bilgileri veriden çekildi, her bir branch bir numpy array olarak dönüştürülerek array'lerin tamamı bir pandas DataFrame'ine atandı.
2. Nmuon (çarpışmada bulunan müon sayısı) 2 ve müon çiftinin yükleri toplamı sıfır olacak şekilde filtreler uygulandı.
3. Aynı branch/array içindeki işlemler için .apply() metodu kullanıldı.
4. Kullanılan invariant mass formülü:

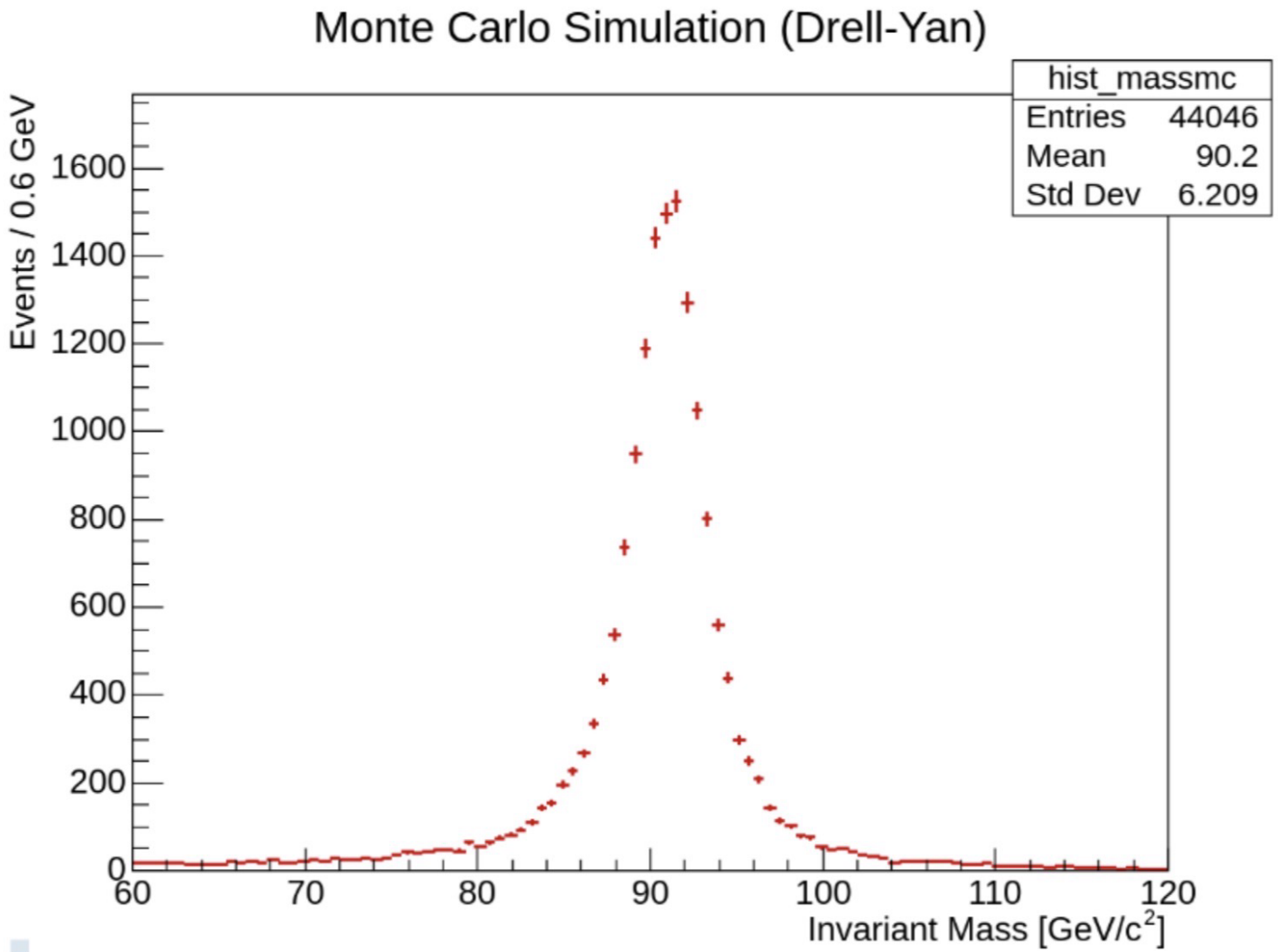
$$M = \sqrt{E_{sum}^2 - \left(P_{x,sum}^2 + P_{y,sum}^2 + P_{z,sum}^2\right)}$$


Dimuon Invariant Mass Spectrum
hist_mass
Entries 18703
Mean 89.97
Std Dev 6.235
Events/0.6GeV
1600
1400
1200
1000
800
600
400
200
0
60
70
80
90
100
110
120
Invariant Mass [GeV/c^2]

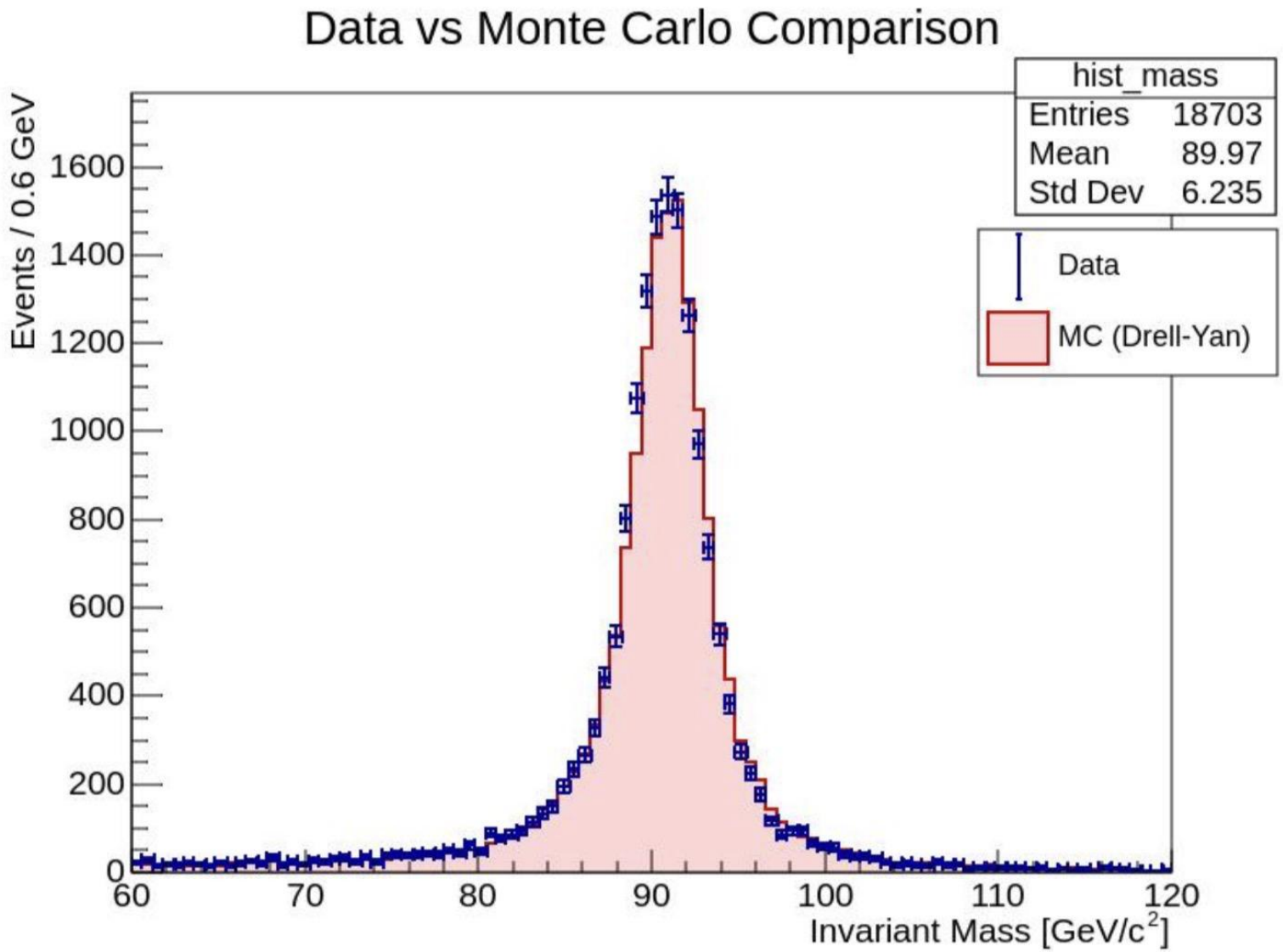
Sonuçları MC Simülasyonları ile Karşılaştırma

1. Gerçek veri için uygulanan adımlar ve filtreler birebir simülasyon verisine de uygulandı.
2. MC histogramı, .Scale() metodu ile deneysel verinin alındığı toplam alana normalize edildi.

Invariant mass distribution of two muons obtained from Monte Carlo simulation:



Overlaid histogram of data and Monte Carlo simulation:



Beni dinlediğiniz için teşekkürler.

YEDEK SLAYTLAR

Sıfırdan Ttree yapısı oluşturma:

[1]:
```
from array import array
import numpy as np
import ROOT as r
```

[2]:
```
f_out=r.TFile("tree11.root","RECREATE")
t_out=r.TTree("t1","Egzersiz Agaci")
px=array('f',[0.])
py=array('f',[0.])
t_out.Branch("px",px,"px/F")
t_out.Branch("py",py,"py/F")
for _ in range(1000):
    px[0]=np.random.normal(0,1)
    py[0]=np.random.normal(0,1)
    t_out.Fill()
f_out.Write()
f_out.Close()
print("Test dosyasi (tree11.root) oluşturuldu.")
```

```
Test dosyasi (tree11.root) oluşturuldu.
```

[3]:
```
%jsroot on
f=r.TFile.Open("tree11.root")
tree=f.Get("t1")
hpx=r.TH1F("hpx","px distribution",100,-3,3)
hpxpy=r.TH2F("hpxpy","py vs px", 30,-3,3,30,-3,3)

nentries=tree.GetEntries()
for i in range(nentries):
    tree.GetEntry(i)
    hpx.Fill(tree.px)
    hpxpy.Fill(tree.px,tree.py)

canvas=r.TCanvas("c1","Egzersiz Cizimi", 800,400)
canvas.Divide(2,1)
canvas.cd(1)
hpx.Draw()
canvas.cd(2)
hpxpy.Draw("COLZ")
canvas.Update()
canvas.Draw()
```

Csv uzantılı veriyi Ttree yapısına dönüştürme:

[2]:
```
import ROOT as r
import pandas as pd
```

[3]:
```
df=pd.read_csv("Wmunu.csv")
```

[4]:
```
rdf=r.RDF.FromPandas(df)
rdf.Snapshot("tree","Wmunu.root")
print("Ttree oluşturuldu")
```

```
Ttree oluşturuldu
```

[5]:
```
f=r.TFile.Open("Wmunu.root")
tree=f.Get("tree")
tree.Print()
print("Toplam Satır Sayısı:", tree.GetEntries())
```

1. **Finding out how often there is more than one isolated, reconstructed muon in data (histogram of the muon multiplicity).**
   - Given data named "data.root" has been opened using ROOT.TFile.Open() method.
   - Names of branches has been listed using the script below:

```
for branch in tree.GetListOfBranches():
    print(branch.GetName())
```

   - A function was defined to detect isolated muons. In this function, iso stands for isolation value, pt stands for transverse momentum and relcut stands for threshold value for relative isolation.

```
def isolatedmuon(pt,iso,relcut=0.05):
 if pt==0:
     return False
 return(iso/pt) < relcut
```

   - A for loop was used to process each event**.** Within the loop; necessary variables were defined, Transverse momentum (p_T) was calculated using the formula:

$$p_T = \sqrt{p_x^2 + p_y^2}$$

   and the previously defined function was applied to select isolated muons.

```
import math

for event in tree:
    n_isolated=0
    for i in range(event.NMuon):
        px=event.Muon_Px[i]
        py=event.Muon_Py[i]
        pt=math.sqrt(px**2+py**2)
        iso=event.Muon_Iso[i]

        if isolatedmuon(pt,iso):
            n_isolated +=1
        h_mult.Fill(n_isolated)
```

2. **Calculating the invariant mass of two muons of opposite charge (manually and/or using the ROOT functionality of adding two fourvectors), only using isolated muons.**

For this task, the bin $N_{\text{muon}} = 2$ was selected to work with, since the objective was to calculate the invariant mass of two isolated muons. ROOT.RDataFrame() was used together with the .AsNumpy() method to convert the required branches into NumPy arrays in order to work with the data as a Pandas DataFrame. Then the filters $N_{\text{muon}} = 2$ and sum of charges=0 were applied, since the charges needs to be of opposite signs. For calculations within same arrays/branches, .apply() method was used.

```
file = r.TFile.Open("data.root")
tree = file.Get("events")
```

```
df_root=r.RDataFrame("events","data.root")
data_dict=df_root.AsNumpy(columns=["Muon_Px","Muon_Py","Muon_Pz","Muon_E","NMuon","Muon_Charge"])
df=pd.DataFrame(data_dict)
print(df.head())
```

```
        Muon_Px       Muon_Py       Muon_Pz       Muon_E  NMuon Muon_Charge
0    [4.859496]  [-30.239874]   [137.77649]  [141.13979]      1        [-1]
1  [-23.376602]  [-21.324467]  [-21.486786]  [38.247765]      1         [1]
2            []            []            []           []      0          []
3            []            []            []           []      0          []
4     [21.0246]   [15.292481]    [38.08255]   [46.11058]      1         [1]
```

```
filtered_df1=df[(df['NMuon']==2)]
f_df2=filtered_df1[(filtered_df1['Muon_Charge'].apply(lambda x: sum(x)==0))]
```

For the calculation of invariant mass, the formula below was used.

$$M = \sqrt{E_{sum}^2 - \left(P_{x,sum}^2 + P_{y,sum}^2 + P_{z,sum}^2\right)}$$

The two four-vectors mentioned were defined using the energy and the x, y, z components of momentum.

Calculation script for invariant mass:

```
#TOTAL VALUES:
E=f_df2['Muon_E'].apply(lambda x: x[0]+ x[1])
px=f_df2['Muon_Px'].apply(lambda x: x[0]+x[1])
py=f_df2['Muon_Py'].apply(lambda x: x[0]+x[1])
pz=f_df2['Muon_Pz'].apply(lambda x: x[0]+x[1])

invariant_mass=np.sqrt(E**2-(px**2+py**2+pz**2))
print(invariant_mass)
```

```
37         88.946831
92         40.915272
109        89.886681
150        92.508911
158        87.331520
             ...
469270    103.110374
469294     90.998978
469308     93.526100
469329      0.396518
469373     60.848289
Length: 18703, dtype: float32
```

4. **Comparing results to MC simulation (displaying simulation and data in the same histogram).**
   The same processes were applied to the Monte Carlo simulation data. The only addition was normalizing the Monte Carlo simulation histogram to the total area of the experimental data using the .Scale() method:

```
if h_mc.Integral() > 0:
    h_mc.Scale(h_im.Integral() / h_mc.Integral())
```