


Exercícios Análise de Dados: Manipulando dados Parte II

Lucas Brasil de Cerqueira

Novembro de 2024

1 Exercício 1




exercício

Usando a idéia inicial do exercício da aula anterior que você calculou a M com a amostra do seu grupo:

Aplique cortes de seleção antes de plotar a massa, vamos exigir um limiar de p_T e η em cada objeto e depois salvar a figura da M no formato png.

1. O número de eventos é afetado?
2. Compare os plots de p_T e η antes e depois do corte .
3. Adicione as figuras e o código no git.



Fiz dois códigos. O primeiro faz toda a reconstrução da massa e plota os gráficos com e sem os cortes. Inclusive mostra o numero de eventos nos plots de massa como iremos comparar.

```
[1]: import uproot
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations

[2]: # Função para calcular a massa invariante
def invariant_mass(pt1, eta1, phi1, mass1, pt2, eta2, phi2, mass2):
    """Calcula a massa invariante de duas partículas."""
    px1 = pt1 * np.cos(phi1)
    py1 = pt1 * np.sin(phi1)
    pz1 = pt1 * np.sinh(eta1)
    e1 = np.sqrt(px1**2 + py1**2 + pz1**2 + mass1**2)

    px2 = pt2 * np.cos(phi2)
    py2 = pt2 * np.sin(phi2)
    pz2 = pt2 * np.sinh(eta2)
    e2 = np.sqrt(px2**2 + py2**2 + pz2**2 + mass2**2)

    # Soma os vetores momento
    e_total = e1 + e2
    px_total = px1 + px2
    py_total = py1 + py2
    pz_total = pz1 + pz2

    # Massa invariante
    mass_squared = e_total**2 - (px_total**2 + py_total**2 + pz_total**2)
    return np.sqrt(np.maximum(mass_squared, 0))

[*]: # Abrindo o arquivo ROOT
file = uproot.open("tree.root")
tree = file["Events"]

# Lendo as brachs
pt = tree["Muon_pt"].array()
eta = tree["Muon_eta"].array()
phi = tree["Muon_phi"].array()
mass_mu = 0.105 # GeV
```

Neste ponto preparamos tudo necessário para plotar o gráfico sem os cortes. Portanto fiz o calculo da massa e usei a função soma (sum) do numpy para me retornar os eventos sem o corte como podemos ver aqui:

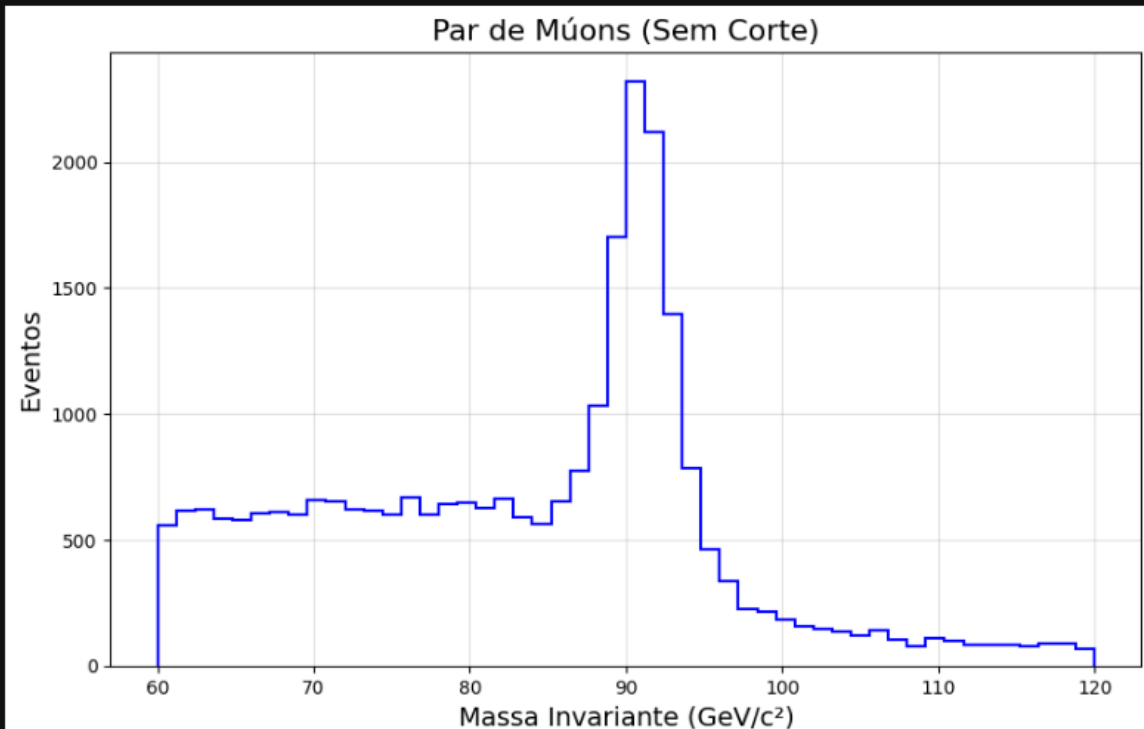
```
[4]: # Calculando a massa invariante para todos os pares de múons no evento
masses1 = []
for event_pt, event_eta, event_phi in zip(pt, eta, phi):
    # Gera todas as combinações de pares de múons no evento
    for (i, j) in combinations(range(len(event_pt)), 2):
        mass1 = invariant_mass(
            event_pt[i], event_eta[i], event_phi[i], mass_mu,
            event_pt[j], event_eta[j], event_phi[j], mass_mu
        )
        masses1.append(mass1)

# Convertendo a lista de massas para um array NumPy
masses1 = np.array(masses1)
masses1.sum()

[4]: np.float32(4501245.0)
```

Agora faremos o plot da massa SEM os cortes:

```
[5]: # Plotando o histograma
plt.figure(figsize=(10, 6))
plt.hist(masses1, bins=50, range=(60, 120), histtype='step', color='blue', linewidth=1.5)
plt.title("Par de Múons (Sem Corte)", fontsize=16)
plt.xlabel("Massa Invariante (GeV/c²)", fontsize=14)
plt.ylabel("Eventos", fontsize=14)
plt.grid(alpha=0.4)
plt.savefig("Sem_Corte.png")
plt.show()
```



Agora faremos a definição e aplicação do corte para os mesmos eventos:

```
•[6]: # Cortes de pT e eta
      pt_cut = 20
      eta_cut = 2.4

      # Máscara
      mask = (pt > pt_cut) & (np.abs(eta) < eta_cut)

      # Selecionando apenas múons que passam os cortes
      pt = pt[mask]
      eta = eta[mask]
      phi = phi[mask]

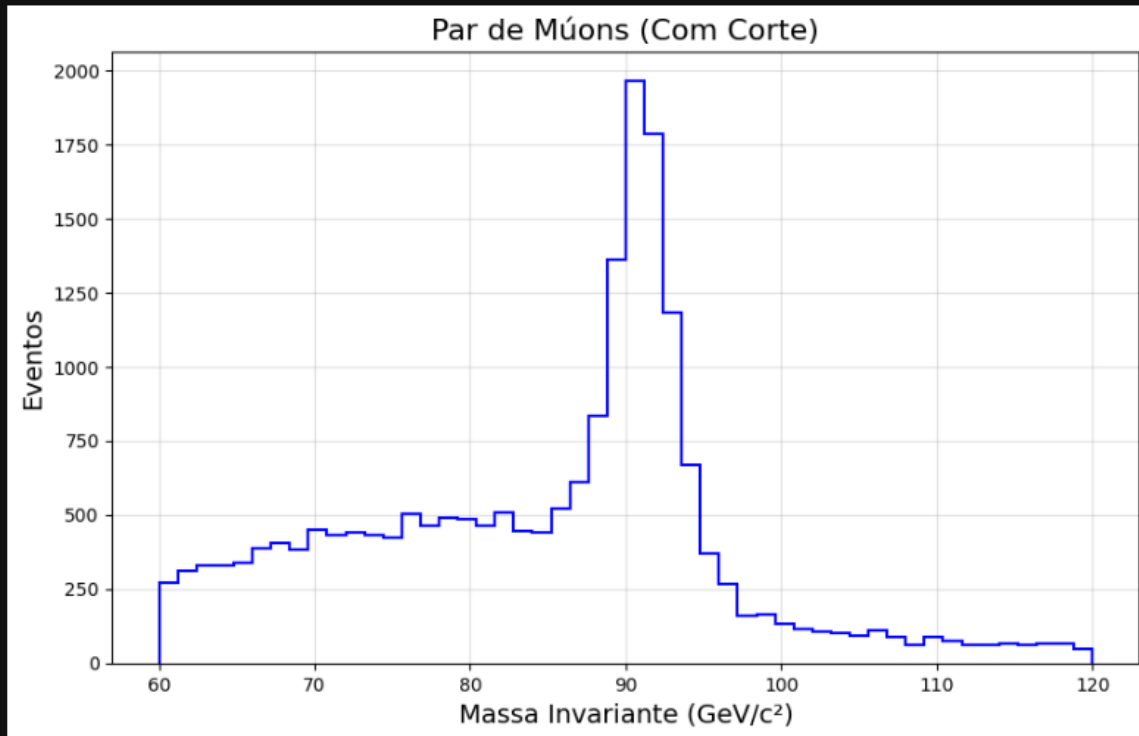
[7]: # Calculando a massa invariante para todos os pares de múons no evento
      masses2 = []
      for event_pt, event_eta, event_phi in zip(pt, eta, phi):
          # Gera todas as combinações de pares de múons no evento
          for (i, j) in combinations(range(len(event_pt)), 2):
              mass2 = invariant_mass(
                  event_pt[i], event_eta[i], event_phi[i], mass_mu,
                  event_pt[j], event_eta[j], event_phi[j], mass_mu
              )
              masses2.append(mass2)

      # Convertendo a lista de massas para um array NumPy
      masses2 = np.array(masses2)
      masses2.sum()

[7]: np.float32(2535448.2)
```

É notável a diferença de eventos após o corte. Quase metade dos eventos.

```
[8]: # Plotando o histograma
plt.figure(figsize=(10, 6))
plt.hist(masses2, bins=50, range=(60, 120), histtype='step', color='blue', linewidth=1.5)
plt.title("Par de Múons (Com Corte)", fontsize=16)
plt.xlabel("Massa Invariante (GeV/c²)", fontsize=14)
plt.ylabel("Eventos", fontsize=14)
plt.grid(alpha=0.4)
plt.savefig("Com_Corte.png")
plt.show()
```



Com Relação ao histograma COM cortes, podemos notar o pico abaixo da linha de 2000 eventos. Já no histograma SEM cortes podemos perceber o pico acima dos 2000 eventos. Portanto, visualmente a comparação dos histogramas demonstra queda no número de Eventos

O segundo programa tem como objetivo criar histogramas de comparação das variáveis P_t e η . Primeiro ele, assim como o primeiro, cria/declara tudo necessário e acessa o arquivo de dados:

```
•[11]: import uproot
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations

# Função para calcular a massa invariante
def invariant_mass(pt1, eta1, phi1, mass1, pt2, eta2, phi2, mass2):
    """Calcula a massa invariante de duas partículas."""
    px1 = pt1 * np.cos(phi1)
    py1 = pt1 * np.sin(phi1)
    pz1 = pt1 * np.sinh(eta1)
    e1 = np.sqrt(px1**2 + py1**2 + pz1**2 + mass1**2)

    px2 = pt2 * np.cos(phi2)
    py2 = pt2 * np.sin(phi2)
    pz2 = pt2 * np.sinh(eta2)
    e2 = np.sqrt(px2**2 + py2**2 + pz2**2 + mass2**2)

    # Soma os vetores momento
    e_total = e1 + e2
    px_total = px1 + px2
    py_total = py1 + py2
    pz_total = pz1 + pz2

    # Calcula a massa invariante
    mass_squared = e_total**2 - (px_total**2 + py_total**2 + pz_total**2)
    return np.sqrt(np.maximum(mass_squared, 0))

# Abrindo o arquivo ROOT
file = uproot.open("tree.root")
tree = file["Events"]

# Lendo os ramos da árvore
pt = tree["Muon_pt"].array()
eta = tree["Muon_eta"].array()
phi = tree["Muon_phi"].array()
mass_mu = 0.105 # GeV
```

Agora vamos criar, usando o Numpy, versões das variáveis que tem e não tem o corte aplicado.

```
# Definindo os cortes
pt_cut = 20
eta_cut = 2.4

# Aplicando cortes
mask = (pt > pt_cut) & (np.abs(eta) < eta_cut)

# Antes e depois dos cortes
pt_all = np.concatenate(pt) # Todos os valores de pT (antes dos cortes)
eta_all = np.concatenate(eta) # Todos os valores de eta (antes dos cortes)
pt_cut_applied = np.concatenate(pt[mask]) # pT após os cortes
eta_cut_applied = np.concatenate(eta[mask]) # eta após os cortes

# Calculando a massa invariante para todos os pares de múons
masses = []
for event_pt, event_eta, event_phi in zip(pt[mask], eta[mask], phi[mask]):
    # Gera todas as combinações de pares de múons no evento
    for (i, j) in combinations(range(len(event_pt)), 2):
        mass = invariant_mass(
            event_pt[i], event_eta[i], event_phi[i], mass_mu,
            event_pt[j], event_eta[j], event_phi[j], mass_mu
        )
        masses.append(mass)
```

E assim podemos fazer os plots de comparação:


```

# 1. Histograma de  $p_T$ 
plt.figure(figsize=(12, 6))
plt.hist(pt_all, bins=100, range=(0, 100), alpha=0.5, label="Sem cortes", color="green", density=True)
plt.hist(pt_cut_applied, bins=100, range=(0, 100), alpha=0.5, label="Com cortes", color="blue", density=True)
plt.title("Distribuição de  $p_T$  dos Múons", fontsize=16)
plt.xlabel(" $p_T$  (GeV/c)", fontsize=14)
plt.ylabel("Frequência Normalizada", fontsize=14)
plt.legend(fontsize=12)
plt.grid(alpha=0.4)
plt.savefig("pt_distribution.png")
plt.show()

# 2. Histograma de  $\eta$ 
plt.figure(figsize=(12, 6))
plt.hist(eta_all, bins=100, range=(-3.5, 3.5), alpha=0.5, label="Sem cortes", color="green", density=True)
plt.hist(eta_cut_applied, bins=100, range=(-3.5, 3.5), alpha=0.5, label="Com cortes", color="blue", density=True)
plt.title("Distribuição de  $\eta$  dos Múons", fontsize=16)
plt.xlabel(" $\eta$ ", fontsize=14)
plt.ylabel("Frequência Normalizada", fontsize=14)
plt.legend(fontsize=12)
plt.grid(alpha=0.4)
plt.savefig("eta_distribution.png")
plt.show()

```

