FÍSICA DE ÍONS PESADOS E OPERAÇÕES NO CALORÍMETRO ELETROMAGNÉTICO

PROBLEMAS, DIFERENÇAS E MÉTODOS USADOS PARA A ANÁLISE DOS DADOS ABERTOS DAS COLISÕES PBPB

Thiago Rangel

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Para começar a análise vamos precisar de alguns pré-requisitos.

. Uma distribuição do LINUX ou MAC. Caso você não possua o Linux como sistema operacional "original de fábrica" pode ser feito um dual boot e há vários tutorias na internet como o seguinte: https://www.youtube.com/watch?v=6D6L9Wml1oY

. Docker que pode ser instalado através do seguinte link: https://www.docker.com/products/docker-desktop/

Feito isso vamos avançar mais um pouco, pois ainda teremos que fazer Download de alguns Containers.

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Primeiramente vamos entender o que é um container Docker. O container **é um pacote de software com todas as dependências necessárias para executar um aplicativo específico ou ambiente de desenvolvimento e a imagem é** uma representação estática do aplicativo ou do serviço e de sua configuração e dependências.

Entendo o que é um container vamos agora instalar alguns containers importantes para a nossa análise que são:

. Container ROOT e Python:

Esses containers podem ser baixados através do seguinte link: https://cms-opendata-workshop.github.io/workshop2022-lesson-docker/03-docker-for-cms-opendata/index.html

Esses containers serão uteis pois por exemplo o container ROOT será onde faremos os nossos gráficos de massa invariante e o Python será onde poderemos modificar os códigos de análise.

Outro fator é que esses containers têm 3,01GB e 2,44Gb então devemos ter um bom armazenamento livre para podermos fazer toda a análise com tranquilidade.

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Feito isso vamos agora fazer o Download do container. Vamos nos basear pela página de instruções para a análise.

https://github.com/cms-opendata-analyses/HiForestProducerTool/tree/2010

Vamos começar abrindo o terminal e digitando o seguinte código:

```
thiago@thiago-550XCJ-550XCR: $ docker run --name hi2010_od -P -p 5901:5901 -it  gitlab-registry.cern.ch/cms-cloud/cmssw-docker-opendata/cmssw_
3_9_2_patch5-slc5_amd64_gcc434:latest /bin/bash
Setting up CMSSW_3_9_2_patch5
CMSSW should now be available.
[02:21:20] cmsusr@cd7e0700e399 ~/CMSSW_3_9_2_patch5/src $
```

Após esse comando será baixado o Container junto com a imagem. Esse container tem um tamanho de 7,4GB e pode demorar para baixar dependendo da sua conexão de internet. Com isso temos nosso ambiente de análise pronto.

Podemos começar a notar algumas diferenças entre a página e o comando usado acima, já que no comando da página após os dois pontos temos 2020-11-17-e0b0b7a6 e no nosso o Latest. Esse Latest serve para que sempre façamos o Docker pull para que a versão mais atualizada seja baixada.

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Feito o Download teremos o seguinte output:

```
thiago@thiago-550XCJ-550XCR: $ docker run --name hi2010_od -P -p 5901:5901 -it  gitlab-registry.cern.ch/cms-cloud/cmssw-docker-opendata/cmssw_
3_9_2_patch5-slc5_amd64_gcc434:latest /bin/bash
Setting up CMSSW_3_9_2_patch5
CMSSW should now be available.
[02:21:20] cmsusr@cd7e8700e399 ~/CMSSW_3_9_2_patch5/src $
```

Agora dentro do container vamos executar alguns comando:

```
[02:21:20] cmsusr@cd7e8700e399 ~/CMSSW_3_9_2_patch5/src $ mkdir HiForest
[02:25:03] cmsusr@cd7e8700e399 ~/CMSSW_3_9_2_patch5/src $ cd HiForest/
[02:25:06] cmsusr@cd7e8700e399 ~/CMSSW_3_9_2_patch5/src/HiForest $
```

O primeiro comando mkdir é para gerar um diretório dentro do container, já o segundo cd HiForest é para entrar nesse diretório. Agora vamos fazer baixar os repositórios do Github através do comando:

```
alyses/HiForestProducerTool.git HiForestProducerch5/src/HiForest $ git clone -b 2010 https://github.com/cms-opendata-an
Cloning into 'HiForestProducer'...
remote: Enumerating objects: 294, done.
remote: Total 294 (delta 0), reused 0 (delta 0), pack-reused 294
Receiving objects: 100% (294/294), 6.25 MiB | 8.67 MiB/s, done.
Resolving deltas: 100% (147/147), done.
```

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Se tivéssemos feito como está na página de instrução teríamos o seguinte:

```
$ git clone -b 2010 git://github.com/cms-opendata-analyses/HiForestProducerTool.git HiForestProducer
```

E com isso teríamos o seguinte problema:

```
yses/HiForestProducerTool.git HiForestProducer$ git clone -b 2011 git://github.com/cms-opendata-anal
Cloning into 'HiForestProducer'...
fatal: unable to connect to github.com:
github.com[0: 20.201.28.151]: errno=Connection timed out
```

Essa é a última diferença entre as instruções da página e o que deve ser feito.

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Feito isso vamos entrar no diretório de HiForestProducer, que foi baixado quando fizemos o clone, e vamos compilar através do comando scram b

```
[02:35:43] cmsusr@cd7e8700e399 ~/CMSSW_3_9_2_patch5/src/HiForest $ cd HiForestProducer/
[02:43:13] cmsusr@cd7e8700e399 ~/CMSSW_3_9_2_patch5/src/HiForest/HiForestProducer $ scram b
```

Feito isso vamos ter um longo output e assim terminamos de instalar o ambiente de análise de 2010.

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Agora vamos começar de fato a análise dos dados e vamos começar modificando o código de análise. Para isso vamos editar o (hiforestanalyzer_cfg.py), que é o código de análise, e para isso usaremos o Vi o nosso editor de texto que está instalado no nosso container .

Com isso vamos visualizar o código e vamos algumas coisas e para isso vamos teclar "I" isso vai fazer com que entremos na edição do arquivo (cuidado para não escrever nada no código de forma indesejada) e vamos alterar o parâmetro de (100) para (–1), isso fará com que todos os arquivos sejam lidos. A outra coisa é comentar a linha começando por (Process.GlobalTag.connect), isso significa colocar # na frente da linha.

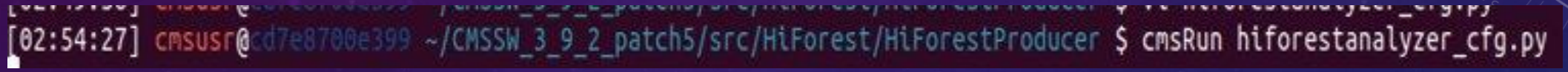
ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

No final o código deve ficar da seguinte forma:

```
#Loading necessary libraries
import FWCore.ParameterSet.Config as cms
from RecoMuon.TrackingTools.MuonServiceProxy_cff import *
import PhysicsTools.PythonAnalysis.LumiList as LumiList
import FWCore.ParameterSet.Types as CfgTypes
process = cms.Process('HiForest')
process.options = cms.untracked.PSet(SkipEvent = cms.untracked.vstring('ProductNotFound'))

#Number of events: put '-1' unless testing
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(-1) )

#HiForest script init
process.load("HiForest_cff")
process.HiForest.inputLines = cms.vstring("HiForest V3",)
version = 'no git info'
process.HiForest.HiForestVersion = cms.string(version)

goodJSON = 'Cert_150436-152957_HI7TeV_StreamExpress_Collisions10_JSON_MuonPhys_v2.txt'
myLumis = LumiList.LumiList(filename = goodJSON).getCMSSWString().split(',')
import FWCore.Utilities.FileUtils as FileUtils
files2010data = FileUtils.loadListFromFile ('CMS_HIRun2010_HIAllPhysics_ZS-v2_RECO_file_index.txt')
process.source = cms.Source("PoolSource",
    fileNames = cms.untracked.vstring(*files2010data
    )
)
process.source.lumisToProcess = CfgTypes.untracked(CfgTypes.VLuminosityBlockRange())
process.source.lumisToProcess.extend(myLumis)

#Global Tag: change the name according to the instructions
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
#process.GlobalTag.connect = cms.string('sqlite_file:/cvmfs/cms-opendata-conddb.cern.ch/GR_R_39X_V6B.db')
process.GlobalTag.globaltag = 'GR_R_39X_V6B::All'
process.load('FWCore.MessageService.MessageLogger_cfi')
process.load("Configuration.StandardSequences.MagneticField_cff")
process.HiForest.GlobalTagLabel = process.GlobalTag.globaltag
```

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Feito isso podemos rodar o script de análise



Porém ainda teremos que fazer mais algumas coisas. Provavelmente a análise pode além de demorar muito dar erro e esses erros podem ser o de não poder abrir o arquivo .ROOT que é o nosso arquivo de input que está dentro do arquivo **CMS_HIRun2010_HIAllPhysics_ZS-v2_RECO_file_index.txt** ou *segmentation violation . Para fazer com que esses erros sejam menos frequentes é reduzir o arquivo .txt em pequenos arquivos de 100 linhas preferencialmente. Embora isso seja mais demorado é o necessário para fazer a análise*

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Para reduzir o arquivo .txt vamos ter que mandar esse arquivo para fora do Docker para poder editar e mandar de volta para o container e assim rodar o script de análise. Podemos fazer isso através do comando docker copy. Selecionando apenas 100 arquivos dos 32000 e mudando o nome do arquivo para CMS_HIRun2010_1.txt, vamos mudar também o arquivo de análise o hiforestanalyzer_cfg.py de forma a ficar da seguinte forma:

```
#Loading necessary libraries
import FWCore.ParameterSet.Config as cms
from RecoMuon.TrackingTools.MuonServiceProxy_cff import *
import PhysicsTools.PythonAnalysis.LumiList as LumiList
import FWCore.ParameterSet.Types as CfgTypes
process = cms.Process('HiForest')
process.options = cms.untracked.PSet(SkipEvent = cms.untracked.vstring('ProductNotFound'))

#Number of events: put '-1' unless testing
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(-1) )

#HiForest script init
process.load("HiForest_cff")
process.HiForest.inputLines = cms.vstring("HiForest V3",)
version = 'no git info'
process.HiForest.HiForestVersion = cms.string(version)

goodJSON = 'Cert_150436-152957_HI7TeV_StreamExpress_Collisions10_JSON_MuonPhys_v2.txt'
myLumis = LumiList.LumiList(filename = goodJSON).getCMSSWString().split(',')
import FWCore.Utilities.FileUtils as FileUtils
files2010data = FileUtils.loadListFromFile ('CMS_HIRun2010_1.txt')
process.source = cms.Source("PoolSource",
    fileNames = cms.untracked.vstring(*files2010data
    )
)
process.source.lumisToProcess = CfgTypes.untracked(CfgTypes.VLuminosityBlockRange())
process.source.lumisToProcess.extend(myLumis)

#Global Tag: change the name according to the instructions
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
#process.GlobalTag.connect = cms.string('sqlite_file:/cvmfs/cms-opendata-conddb.cern.ch/GR_R_39X_V6B.db')
process.GlobalTag.globaltag = 'GR_R_39X_V6B::All'
process.load('FWCore.MessageService.MessageLogger_cfi')
process.load("Configuration.StandardSequences.MagneticField_cff")
process.HiForest.GlobalTagLabel = process.GlobalTag.globaltag
```

```
)
process.source.lumisToProcess = CfgTypes.untracked(CfgTypes.VLuminosityBlockRange())
process.source.lumisToProcess.extend(myLumis)

#Global Tag: change the name according to the instructions
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
#process.GlobalTag.connect = cms.string('sqlite_file:/cvmfs/cms-opendata-conddb.cern.ch/GR_R_39X_V6B.db')
process.GlobalTag.globaltag = 'GR_R_39X_V6B::All'
process.load('FWCore.MessageService.MessageLogger_cfi')
process.load("Configuration.StandardSequences.MagneticField_cff")
process.HiForest.GlobalTagLabel = process.GlobalTag.globaltag

#Define the output root file (change each run not to overwrite previous output)
process.TFileService = cms.Service("TFileService",
                                   fileName=cms.string("HiForestAOD_DATAtest_1.root"))

#Init Trigger Analyzer
process.hltanalysis = cms.EDAnalyzer('TriggerInfoAnalyzer',
                             processName = cms.string("HLT"),
                             triggerName = cms.string("@"),
                             datasetName = cms.string("HIAllPhysics"),  #'HICorePhysics' to look at Core Physics only
                             triggerResults = cms.InputTag("TriggerResults","","HLT"),
                             triggerEvent   = cms.InputTag("hltTriggerSummaryAOD","","HLT")
                             )

#Collect event data
process.demo = cms.EDAnalyzer('Analyzer') #present analyzer is for muons - see details in Analyzer.cc for possible modifications
process.dump=cms.EDAnalyzer('EventContentAnalyzer') #easy check of Event structure and names without using the TBrowser

process.ana_step = cms.Path(process.hltanalysis+
                            #process.dump+  #uncomment if necessary to check the name. Do not forget to change the number of events to '1'
                            process.demo+
                            process.HiForest
)
```

Devemos lembrar de sempre mudar o nome do arquivo que vai ser gerado Repetindo esse processo, vamos gerar vários arquivos .ROOT. E devemos mudar o nome e podemos fazer isso alterando a parte "fileName=CMS.string("....") devemos alterar dentro do parenteses

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

Com isso vamos gerar os arquivos .ROOT e após realizarmos um número considerável de arquivos vamos juntar esses arquivos com o comando "hadd" da seguinte forma:

```
~$ hadd 2010case.root HiForestAOD_DATAtest_1.root HiForestAOD_DATAtest_2.root ...
```

Isso dentro do container, então o hadd vai juntar os arquivos .ROOT logo depois vamos dar o nome ao arquivo que vai ser criado e depois colocamos os arquivos que queremos juntar, como mostrado acima. 2010case.root é o nome do arquivo que vai ser gerado e HiForestAOD_DATAtest_1.root o arquivo que queremos juntar. Com isso será gerado o novo arquivo .ROOT e agora temos que mandar esse arquivo para o Container do ROOT junto com outro arquivo chamado forest2dimuon.C.

ANÁLISE DOS DADOS DE 2010 DE COLISÕES PB-PB

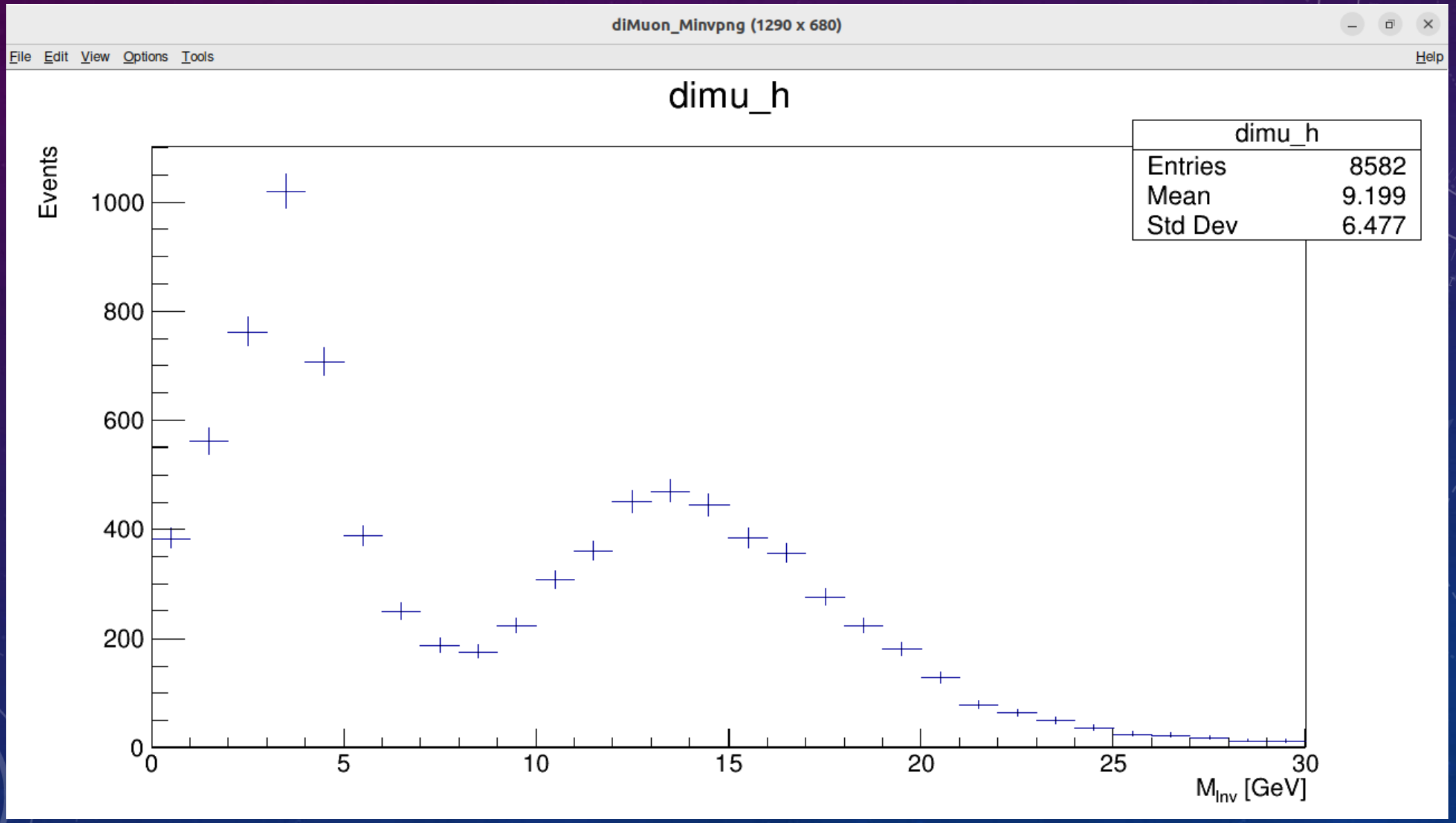
Usando o mesmo comando docker cp, vamos mandar o arquivo .ROOT para nossa área de trabalho e depois para o container ROOT. Antes de mandar o arquivo do forest2dimuon vamos ter que editar ele, pois há um pequeno erro então devemos comentar a linha 87 desse arquivo e assim podemos mandar esse arquivo para o container ROOT. Feito isso tudo podemos acessar o container do root digitando o seguinte comando:

```
thiago@thiago-550XCJ-550XCR:~$ docker start -i my_root
cmsusr@thiago-550XCJ-550XCR:/code$
```

Com isso vamos executar o comando:

```
cmsusr@thiago-550XCJ-550XCR:/code$ root -l forest2dimuon.C
```

- Vamos plottar o nosso histograma que vai variar de acordo com a quantidade de dados que foram analisados , ou seja, quantos arquivos .ROOT foram utilizados. Esse histograma ainda não está completo, pois ainda há dados que não foram analisados.



ANÁLISE DE DADOS DE 2011DE COLISÕES PB-PB

A versão de 2011 é muito similar à de 2010 e também temos uma página guia que é

https://github.com/cms-opendata-analyses/HiForestProducerTool/tree/2011

Vamos começar baixando o container e sua imagem com o seguinte comando:

```
thiago@thiago-550XCJ-550XCR:~$ docker run --name hi2011_od -it gitlab-registry.cern.ch
/cms-cloud/cmssw-docker/cmssw_4_4_7-slc5_amd64_gcc434:latest /bin/bash
```

Que também é diferente da forma mostrada no site. Feito isso vamos seguir os mesmos passos dos dados de 2010 para os de 2011.

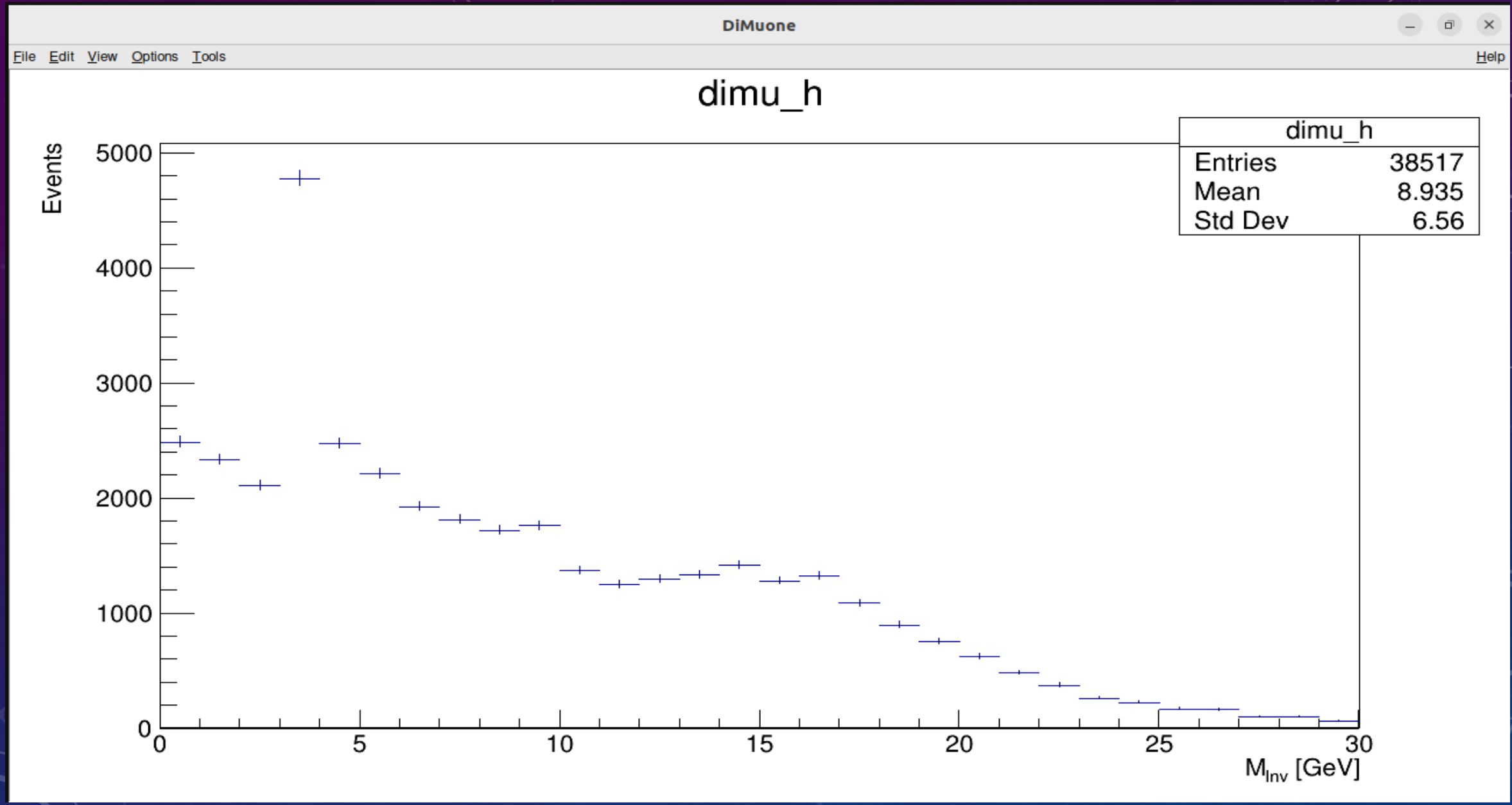
Criando um pasta HiForest e entrando vamos fazer o git clone da seguinte forma:

```
git clone -b 2011 https://github.com/cms-opendata-analyses/HiForestProducerTool.git HiForestProducer
```

Feito isso vamos entrar na pasta HiForestProducer e compilar os arquivos utilizando o comando (scram b) e repetir todo o passo que foi feito na edição do arquivo (hiforestanalyzer_cfg.py) nos dados de 2010

ANÁLISE DE DADOS DE 2011DE COLISÕES PB-PB

- Feito isso vamos repetir o processo que fizemos para 2010. Vamos fazer mandar o arquivo .txt para fora, mandar de 100 em 100 e produzir os arquivos .ROOT e juntar todos com o comando hadd. Feito isso devemos mandar o arquivo final para o container do ROOT assim como o arquivo forest2dimuon.C (esse arquivo NÃO é o mesmo arquivo de análise de 2010 e portanto não tem o mesmo problema, então não vamos editar NADA). Mandando os arquivos para o container ROOT podemos fazer o plot do histograma dos dados de 2011 que é o seguinte:


DiMuone
File Edit View Options Tools
Help
dimu_h
dimu_h
Entries 38517
Mean 8.935
Std Dev 6.56
Events
5000
4000
3000
2000
1000
0
0
5
10
15
20
25
30
M_{Inv} [GeV]