Bitácora

Pablo Jibrán Pomares Valdés

20 de enero de 2025

Filtro

Parámetros considerados.
Se busca que:

- Existe conservación de carga
- Se tengan 4 muones
- No haya jets con btag, rudeza media (DeepbTag $<$ 0,5847). ref
- Todos los muones sean globales.
- MET_significance $<$ 0,1

Código

```
void filter(){
  TFile *f_old = TFile::Open("root://eospublic.cern.ch//eos/opendata/cms/Run2016H/DoubleMuon/NANOAOD/UL2
  TTree* t_old;
  f_old->GetObject("Events", t_old);

  const int nentries = t_old->GetEntries();

  // Deactivate all branches
  t_old->SetBranchStatus("*", 0);

  // Activate desired branches
  for (auto actBranchName : {"run", "event", "Muon_charge", "Muon_dxy", "Muon_dxyErr", "Muon_isGlobal",
                             "Muon_isTracker", "Muon_pt", "Muon_phi", "Muon_eta", "nMuon", "MET_phi",
                             "MET_pt", "MET_significance", "nJet", "Jet_btagCSVV2", "Jet_btagDeepB"}){
    t_old->SetBranchStatus(actBranchName, 1);
  };

  // Entry selection
  UInt_t nMuon, nJet;
  Float_t Jet_btagDeepB[20];
  Bool_t Muon_isGlobal[10];
  Int_t Muon_charge[10];
  t_old->SetBranchAddress("nMuon", &nMuon);
  t_old->SetBranchAddress("nJet", &nJet);
  t_old->SetBranchAddress("Jet_btagDeepB", &Jet_btagDeepB);
  t_old->SetBranchAddress("Muon_isGlobal", &Muon_isGlobal);
  t_old->SetBranchAddress("Muon_charge", &Muon_charge);

  TFile newfile("prueba1.root", "recreate");
  auto t_new = t_old->CloneTree(0);
```

Código

```
for (int i=0; i<nentries; i++){
  t_old->GetEntry(i);

  int sumCharge = 0;
  for (int j=0; j<nMuon; j++){
    sumCharge += Muon_charge[j];
  };
  bool chargeViolation = sumCharge;

  bool passbTag = true;
  for (int j=0; j<nJet; j++){
    if (Jet_btagDeepB[j] > 0.5847){
      passbTag = false;
    };
  };

  bool passnMuon = false;
  if (nMuon == 4){
    passnMuon = true;
  };

  bool passAllGlobal = true;
  for (int j=0; j<nMuon; j++){
    if (Muon_isGlobal[j] == 0){
      passAllGlobal = false;
    };
  };
  if (passAllGlobal && passbTag && passnMuon && !chargeViolation) {
    t_new->Fill();
  };

};

t_new->Print();
newfile.Write();
}
```

filter_all.sh

12/01/2025

Con un bash script y modificaciones menores a filter.C se puede automatizar y filtrar todos los archivos de root que se mencionan en el índice del Open data.

```
#!/bin/bash

# Checks for an index file
if [ "$#" -ne 1 ]; then
  echo "Error: No index file specified"
  echo "Usage: $0 <path/to/index_file.txt>"
  exit 1
fi

# Creates the output file name
IFS='/' read -ra INPUT <<<"$1"
IFS='_' read -ra NAME <<<${INPUT[-1]}
OUTPUT_NAME="${NAME[0]}_${NAME[1]}_${NAME[2]}_${NAME[6]}.root"

# Checks if directory exists. If not, creates it
DIR="rootfiles_${NAME[0]}_${NAME[1]}_${NAME[2]}_${NAME[6]}"
if [ ! -d $DIR ]; then
  echo "Creating output directory as $DIR"
  mkdir $DIR
  cd $DIR
else
  echo "Output directory already exists"
  cd $DIR
fi

i=0
while IFS= read -r line; do
  OUTPUT_NAME_PARTIAL="${NAME[0]}_${NAME[1]}_${NAME[2]}_${NAME[6]}_p$i.root"
  macro_command="../filter.C( \"${line}\", \"${OUTPUT_NAME_PARTIAL}\")"
  root -l -q "${macro_command}"
  ((i = i + 1))
done <"../${INPUT[-1]}"

hadd "../$OUTPUT_NAME" *.root
```

Buscador de bosones Z

Parámetros considerados

Consideré las variables:

- Muon_pt
- Muon_charge
- Muon_eta
- Muon_phi

Masa invariante

Sabemos que la expresión para la masa invariante:

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

```
Double_t inv_mass(Float_t pt1, Float_t pt2, Float_t phi1, Float_t phi2, Float_t eta1, Float_t eta2) {

  Double_t eta_diff = eta1 - eta2;
  Double_t phi_diff = phi1 - phi2;
  Double_t pt_prod = 2*pt1*pt2;

  Double_t m2 = pt_prod*(TMath::CosH(eta_diff) - TMath::Cos(phi_diff));
  Double_t m = TMath::Sqrt(m2);

  return m;
}
```

z_finder I

En general, se comparan todos los muones para buscar la generación de bosones Z. Por conservación de la carga, los muones que comparemos tiene que ser opuestos. Si esa condición se cumple, se busca que la masa invariante se encuentre en el rango $|m - m_Z| \leq 10$ GeV.
En caso de que se encuentren dos candidatos que compartan un leptón, se selecciona el que tenga un mayor ángulo entre los muones.
Por último, se regresa un tuple que contenga el número de bosones Z en el determinado evento y una masa.

z_finder II

```
// Find the number of Z bosons on a event and its mass.
// If >1 it returns (for now) a single mass. However, this is unimportant because event is disca
class Z_finder{
  public:
    UInt_t num_z;
    Double_t masses[2];
    std::array<int, 2> z_muon_index;

    Z_finder(Float_t muon_pt[4], Float_t muon_phi[4], Float_t muon_eta[4], Int_t muon_charge[4])
      : num_z(0), masses{0.0, 0.0}, z_muon_index{0, 0}
    {
      for (int i=0; i<4; i++){
        int local_z = 0; // Number of Z candidates for that share a muon
        int z_local_index[3] {-1, -1, -1};
        Double_t local_masses[2] = {0., 0.};

        for (int j=i+1; j<4; j++){
          // Muon properties
          Float_t pt1 = muon_pt[i];
          Float_t pt2 = muon_pt[j];
          Float_t phi1 = muon_phi[i];
          Float_t phi2 = muon_phi[j];
          Float_t eta1 = muon_eta[i];
          Float_t eta2 = muon_eta[j];
          Int_t q1 = muon_charge[i];
          Int_t q2 = muon_charge[j];

          // Check for different charge
          bool same_charge = q1 + q2;
          z_local_index[0] = i;
          z_muon_index[0] = i;

          if (!same_charge) {
            Double_t m = inv_mass(pt1, pt2, phi1, phi2, eta1, eta2);
            if (m > 81.2 && m < 101.2){
              local_masses[local_z] = m;
              z_local_index[local_z+1] = j;
              local_z++;
            };
          };
        };
```

z_finder III

```
        // If more there is more than one local Z choose the one with greater
        // phi difference
        if (local_z == 2){
          float phi1 = TMath::Abs(muon_phi[z_local_index[0]]);
          float phi2 = TMath::Abs(muon_phi[z_local_index[1]]);
          float phi3 = TMath::Abs(muon_phi[z_local_index[2]]);

          float diff1_2 = phi1 - phi2;
          float diff1_3 = phi1 - phi3;

          if (diff1_2 > diff1_3){
            masses[num_z] = local_masses[0];
            z_muon_index[1] = z_local_index[1];
          }
          else {
            masses[num_z] = local_masses[1];
            z_muon_index[1] = z_local_index[2];
          };
        };

        if (local_z) {num_z++;};
      }
    }
};
```

z_finder results

```
Se tienen 2121 eventos con 1 Z.
Se tienen 28 eventos con 2 Z.
Se tienen 96221 eventos con ningún Z.
```

Algoritmo usado

Por el momento estoy utilizando el algoritmo que se encuentra en [3] para calcular el m_{T2}, ya que es el más rápido y preciso que he encontrado. Aunque me gustaría tratar de implementar el mío (aunque sea peor) en algún momento.
Lo que tuve que hacer fue descomponer el momento transversal en sus componentes x y y tanto como de los muones como de la energía perdida. De ahí el archivo "lester_mt2_bisct.h"se encarga de lo demás.

WWZ_finder

06/01/2025

Previamente presenté resultados pero puede ser que se encuentren equivocados. La energía de todos los leading muons es demasiado grande, alrededor de 110 GeV/c^2 lo cual lo veo improbable. Tengo que checar mis unidades en el código. Además, otra cosa que encuentro sospechosa es que de 110 eventos con un Z, 108 tengan dos W. En [4] se nos indica que la mayoría del background proviene de producción de ZZ, por lo que encuentro improbable que haya tenido mucha suerte. Seguramente también sea problemas de unidades.

WWZ_finder

En el archivo WWZ_finder.C está mis pocos avances para mi algoritmo de búsqueda mientras que en test.C está actualmente implementado el algoritmo de [3].
Cosas que planeo hacer:

- Corregir el error en el que ningun dato cumple las condiciones.
- Usar datos que me mandó Jeremy para guiarme en lo que tengo que observar. Por mi cuenta haré algo similar pero con muones en lugar de electrones, para que se asemeje lo más posible a mi caso.
- Automatizar

WWZ_finder

13/01/2025
El error con las unidades fue corregido y ahora los resultados son más realistas. Además, logré arreglar el error de las inconsistencias al momento de buscar la producción de bosones Z. La solución fue cambiar a un objeto Z_finder() en lugar de usarlo como función, esto tuvo la ventaja adicional de que el código es más legible.
También convertí en objeto la función vec_comp() (ahora Vec_comp()).
Aún no hago la distribución de mT para el muón.

WWZ_finder

20/01/2025
Cambios menores a filter.C: quité MET_significance y agrugué que existiera una MET_pt mayor a 20. Como ya tengo MET, no hay posibilidad de que tenga ZZ, por lo que también removí esa opción en wwz_finder.C. Si existen dos candidatos a Z en el mismo evento, tomo aquél cuyos muones estén más opuestos (asumo que no hay boost y que el vértice primario no está fuera del centro).
En ningún artículo sobre WW mencionan al mT2. Lo que me llamó la atención bastante es el rango bastante grande en el que pueden existir los WW.

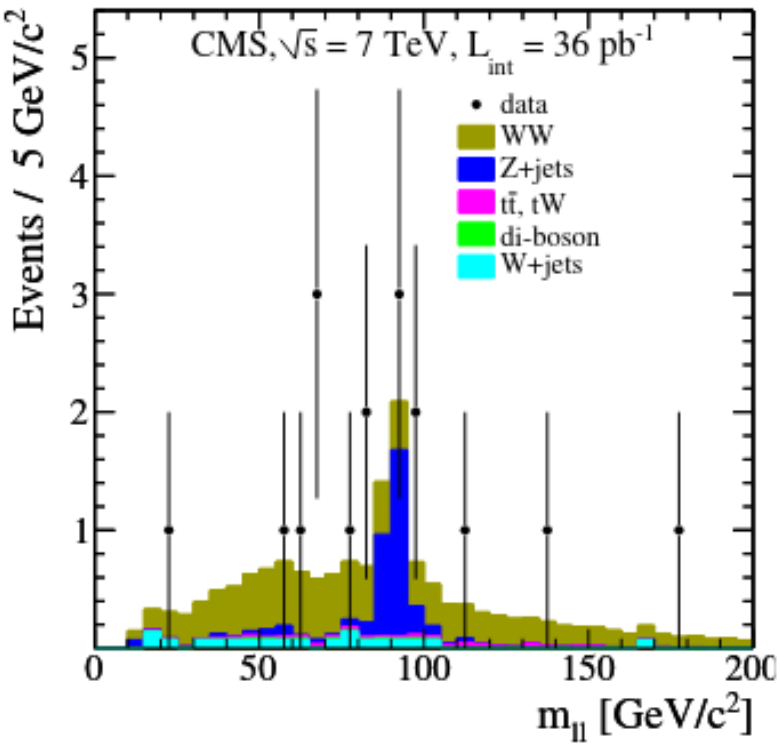

Figura 1: Gráfico de [1]

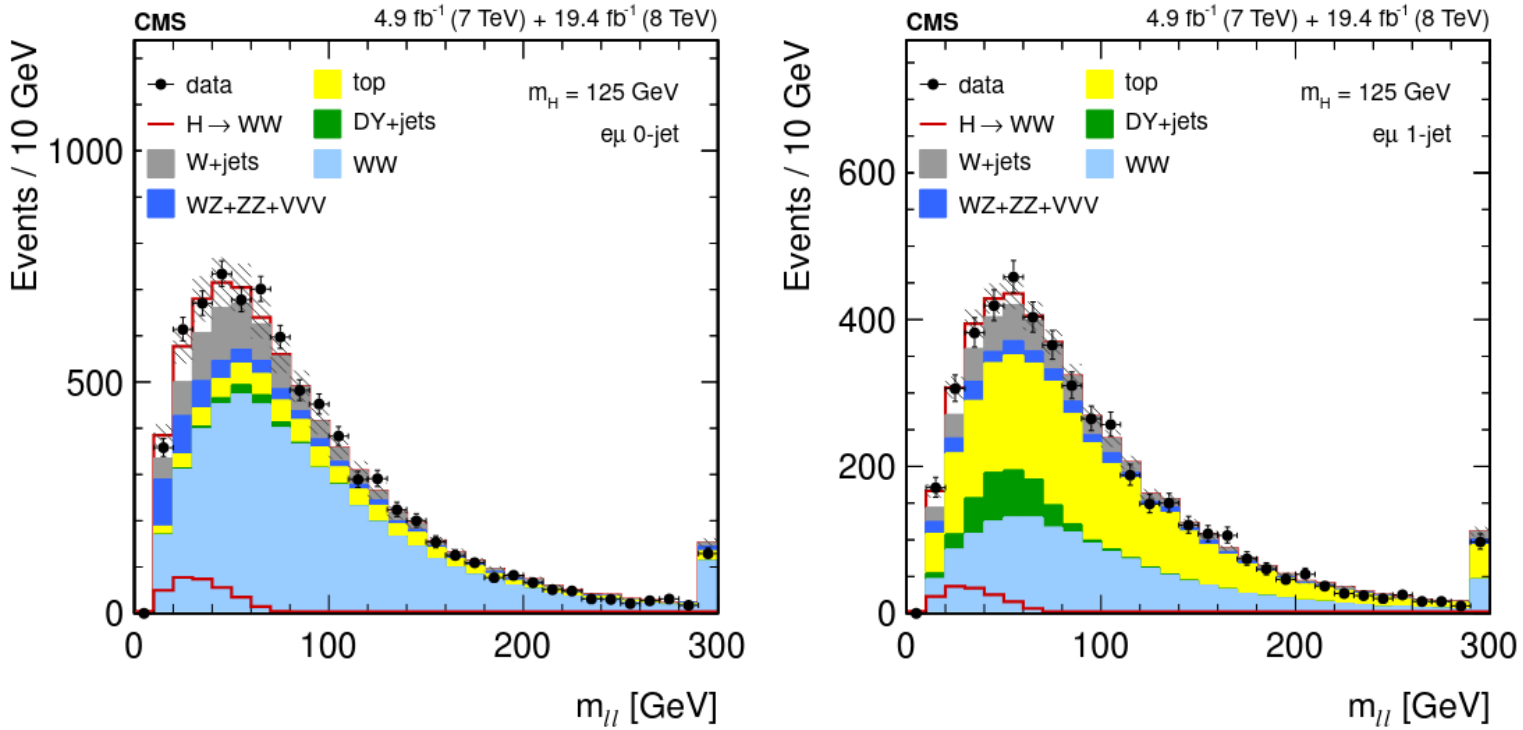

Figura 2: Gráfico de [2]

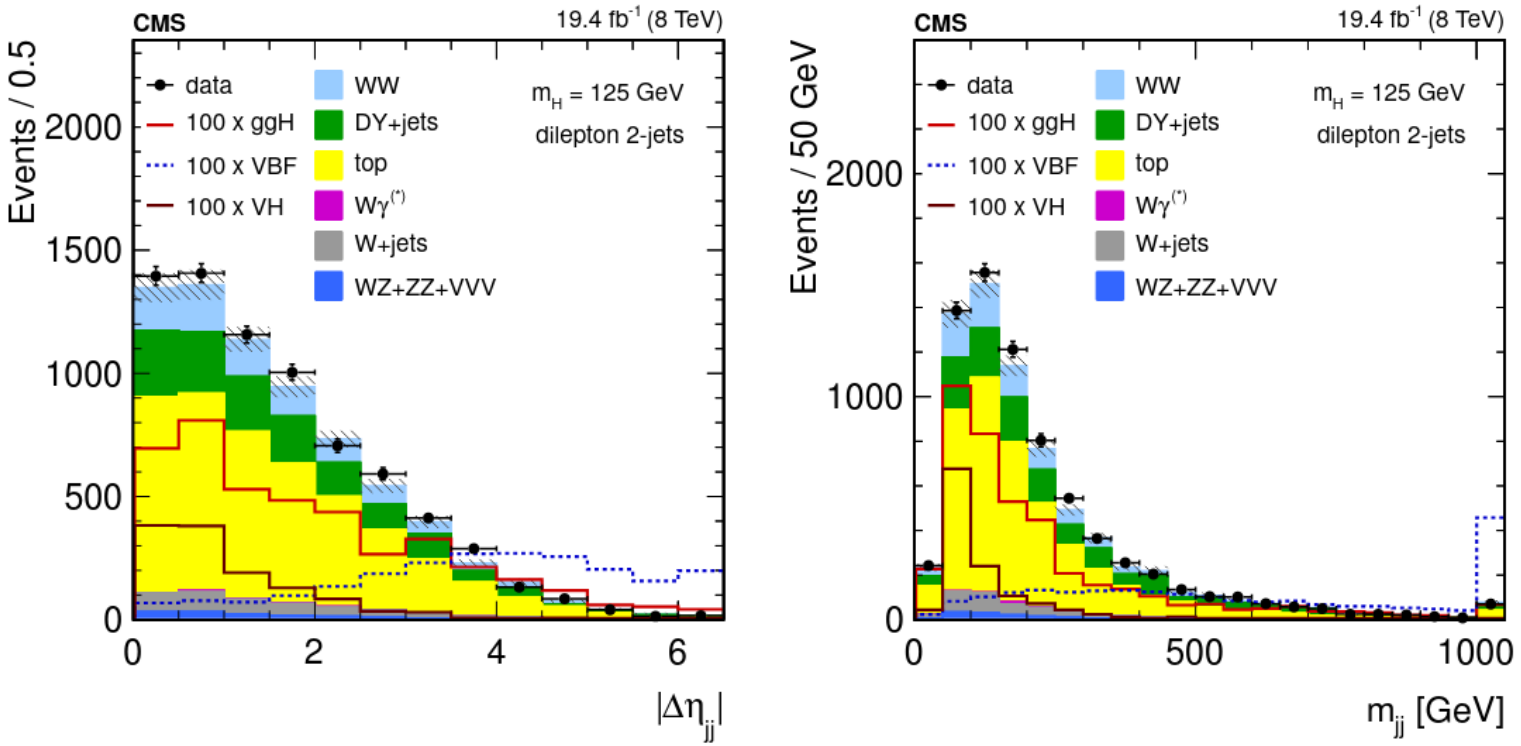

Figura 3: Gráfico de [2]