Table of Contents

2018 CMS Data Analysis School short tau Exercise

Notes

The color scheme of the Exercise is as follows:

- Commands will be embedded in a grey box, e.g.

  ```
  cmsRun testRun_cfg.py
  ```
- Output and screen printouts will be embedded in a green box, e.g.

  ```
  OUTPUT
  ```
- Code to paste will be embedded in a pink box, e.g.

  ```
  PASTE THIS CODE
  ```
- Formatted descriptions will be embedded in a blue box e.g.

  ```
  FORMATTED DESCRIPTION OR EXPLANATION
  ```
- Code to edit will be embedded in a purple box e.g.

  ```
  CODE TO EDIT
  ```

Introduction

The goal of this exercise is to measure the jet->tau fake rake and the efficiency for several tau identification working points (IDs).

1. Measure jet to tau fake rate from MC or Data.
2. Measure tau ID efficiency from MC.
3. Make a Roc curve with 3 different tau IDs.

In case you have time you can compare the jet to tau fake rate between data and MC. All you need is to repeat Step 1 on Single Muon dataset and compare it with fake rate from W+Jet MC Simulation samples.

First Steps

Login on cmslpc-sl6

In case you overlooked the first pre-exercise

Show ▸ Hide ▾

```
go to
http://computing.fnal.gov/authentication/krb5conf/

and download the corresponding Fermilab Kerberos Configuration File for
Linux, OSX or Windows

the do:
sudo cp krb5.conf /etc/krb5.conf


then in:
~/.ssh/config
```

```
add the lines:
GSSAPIAuthentication yes
GSSAPIDelegateCredentials yes

then initialise:
kinit -f username@FNAL.GOV

as always: check, type in your shell:
klist

you should see something like:
Credentials cache: API:92B37754-DE6A-4B63-AF2E-6F717C63F95B
        Principal: perieanu@FNAL.GOV

  Issued                Expires               Principal
Jan  8 10:28:44 2016  Jan  9 12:28:42 2016  krbtgt/FNAL.GOV@FNAL.GOV

then this should work:
                SL6: ssh -K username@cmslpc-sl6.fnal.gov
```

Set Up Area

To set up your directory area at the LPC

```
#For the LPC at FNAL
source /cvmfs/cms.cern.ch/cmsset_default.csh
setenv SCRAM_ARCH slc6_amd64_gcc491

mkdir TauExercise
cd TauExercise
cmsrel CMSSW_7_4_14
cd CMSSW_7_4_14/src
cmsenv
```

Get Necessary Files From GitHub Using wget

In this area please run the command

```
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/TreeReader.h
```

The output of this command is

Show ▸ Hide ▾

```
--2015-12-23 16:02:58--  https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/TreeRead
Resolving raw.githubusercontent.com... 23.235.44.133
Connecting to raw.githubusercontent.com|23.235.44.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28442 (28K) [text/plain]
Saving to:  TreeReader.h

100%[======================================>] 28,442      --.-K/s   in 0s

2015-12-23 16:02:59 (325 MB/s) -  TreeReader.h  saved [28442/28442]
```

Now run the command

```
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/Make.sh
chmod 700 Make.sh
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/plotEfficiency.py
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/plotFakeRate.py
```

Get Necessary Root Files

```
xrdfsls ls -u /store/user/abdollah/CMSDAS2017Short
```

The output should resemble

```
root://131.225.207.127:1094//store/user/abdollah/CMSDAS2017Short/DYJetsToLL_M-50_Inc_ShortEx.root
root://131.225.207.127:1094//store/user/abdollah/CMSDAS2017Short/WJetsToLNu_Inc_ShortEx.root
```

For this exercise we will use `/eos/uscms/store/user/abdollah/CMSDAS2017Short/WJetsToLNu_Inc_ShortEx.root` which is how you can access the file at LPC.

These files are called "ggNtuples", since they are flat trees that come from running MiniAOD with `ggNtuplizer`. They are now centrally produced and available. The code that produces the ggNtuples is here. This exercise could be done from MiniAOD ntuples, but would take much longer to run over the full statistics.

Exercise 1: Jet to Tau Fake Rate

The jet->tau fake rate can be easily measured both on Data (in this case using a Single Muon dataset) and on Monte Carlo Simulation (a W+Jet Sample)

This exercise aims to answer the question "How likely is a jet to be mis-identified as a tau lepton?" for different tau identification working points (IDs). The general process is as follows:

1. Select W events (W->mu nu events). To do so find a good muon in event and check the Transverse Mass of muon and MET to be compatible with W Transverse Mass (M_T).
2. Find a tau jet which pass decay mode finding and anti muon and anti electron discriminator.
3. Find how often this object can pass different working points of the Tau Isolation.

While doing this exercise, please consider the following:

1. Try all eta ranges on the taus. Remember to always make an eta cut of 2.1 for your muons and 2.3 for your taus! Why is this eta range important? Answer:
   Show ▸ Hide ▾
   The tracker, with a pseudorapidity : eta < | 2.5 |, is required to reconstruct the trajectories of the charged particles from the tau decay. The method used in this exercise requires events selected with the single muon trigger which has an eta window of eta < | 2.1 |. In order to ensure that the tau cone is within the tracker coverage a window of eta < | 2.3 | is chosen.
2. One can parametrize this fake rate v.s. Tau Pt, closest Jet pt to Tau or Tau eta, etc.
3. Compare tau fake rate between data and MC
4. See the fake rate of different working points! Add in other tauIDs from the TreeReader.h
5. After measuring tau ID from ZTT MC sample, make the ROC curve!

Instructions for Exercise

Create a new file named `jetToTauFakeRate.cc` and paste in the file

Show ▸ Hide ▾
```
/////////////////////////////////////////////////////////////////////////////////////////////
//   Compiling the code:   ./Make.sh jetToTauFakeRate.cc
//   Running the code:     ./jetToTauFakeRate.exe OutPut.root   Input.root
/////////////////////////////////////////////////////////////////////////////////////////////

#include "TreeReader.h"
```

```
#include <string > //remove spaces or the file will not be made!
#include  <ostream > //remove spaces or the file will not be made!

int main(int argc, char** argv) {
    using namespace std;

    std::string out = *(argv + 1);

    cout  <<"\n\n\n OUTPUT NAME IS:    "  <<out  <<endl;     //PRINTING THE OUTPUT FILE NAME
    TFile *fout = TFile::Open(out.c_str(), "RECREATE");

    std::string input = *(argv + 2);
    cout  <<"\n\n\n INPUT NAME IS:    "  <<input  <<endl;     //PRINTING THE INPUT FILE NAME
    TFile * myFile = TFile::Open(input.c_str());

    //add the histrograms of All gen-matched taus, and all gen-matched taus passing numerator

    TH1F *    histoDenominator = new TH1F ("histoDenominator","histoDenominator", 300, 0, 300);
    TH1F *    histoNumerator = new TH1F ("histoNumerator","histoNumerator", 300, 0, 300);


    TTree *Run_Tree = (TTree*) myFile->Get("EventTree");
    cout.setf(ios::fixed, ios::floatfield);


    //close the file
    fout->cd();
    histoNumerator->Write();
    histoDenominator->Write();
    fout->Close();
}
```

Please edit these two lines

```
#include <string > //remove spaces or the file will not be made!
#include  <ostream > //remove spaces or the file will not be made!
```

After `cout.setf(ios::fixed, ios::floatfield);` add the following lines to access the branches you want. The names added here should match those found in `TreeReader.h`.

Show ▸ Hide ▾

```
    Run_Tree->SetBranchAddress("nTau", &nTau);
    Run_Tree->SetBranchAddress("tauPt"  ,&tauPt);
    Run_Tree->SetBranchAddress("tauEta"  ,&tauEta);
    Run_Tree->SetBranchAddress("tauPhi"  ,&tauPhi);
    Run_Tree->SetBranchAddress("tauMass"  ,&tauMass);
    Run_Tree->SetBranchAddress("tauDxy",&tauDxy);
    Run_Tree->SetBranchAddress("tauByTightMuonRejection3", &tauByTightMuonRRejection3);
    Run_Tree->SetBranchAddress("tauByMVA6LooseElectronRejection", &tauByMVA6LooseEElectronRejecti
    Run_Tree->SetBranchAddress("tauByLooseCombinedIsolationDeltaBetaCorr3Hits",&tauByLooseCombiin

    Run_Tree->SetBranchAddress("nMu", &nMu);
    Run_Tree->SetBranchAddress("muPt"  ,&muPt);
    Run_Tree->SetBranchAddress("muEta"  ,&muEta);
    Run_Tree->SetBranchAddress("muPhi"  ,&muPhi);
    Run_Tree->SetBranchAddress("muIsMediumID",&muIsMediumID);
    Run_Tree->SetBranchAddress("muPFChIso", &muPFChIso);
    Run_Tree->SetBranchAddress("muPFPhoIso", &muPFPhoIso);
    Run_Tree->SetBranchAddress("muPFNeuIso", &muPFNeuIso);
    Run_Tree->SetBranchAddress("muPFPUIso", &muPFPUIso);
    Run_Tree->SetBranchAddress("muD0",&muD0);
    Run_Tree->SetBranchAddress("muDz",&muDz);

    Run_Tree->SetBranchAddress("pfMET",&pfMET);
    Run_Tree->SetBranchAddress("pfMETPhi",&pfMETPhi);
    Run_Tree->SetBranchAddress("HLTEleMuX", &HLTEleMuX);


    float MuMass= 0.10565837;
    float eleMass= 0.000511;
```

In order to look at all events in the file, a loop over all events must be added. Paste the following lines after initializing your branches and declaring the muon and electron mass:

Show ▸ Hide ▾

```
    Int_t nentries_wtn = (Int_t) Run_Tree->GetEntries();
    cout <<"nentries_wtn===="  <<nentries_wtn  <<"\n";
    for (Int_t i = 0; i  <nentries_wtn; i++) {
        Run_Tree->GetEntry(i);
        if (i % 1000 == 0) fprintf(stdout, "\r  Processed events: %8d of %8d ", i, nentries_wtn);
        fflush(stdout);

        TLorentzVector Mu4Momentum, Tau4Momentum;

       ///////////////////////////////////////////////
       //Important Analysis Loop Will Happen Here!!!//
       ///////////////////////////////////////////////


   } //End Processing all entries
```

Within the *Important Analysis Loop*, we want to select good W-boson events. We want events with a well identified muon coming from the W-boson, so we can be sure all taus are fakes. There should only be at most one good lepton per event, in this case we will require a good muon. First we make a loop over muons for our *Important Analysis Loop*.

```
        for  (int imu=0 ; imu <nMu; imu++){


        }
```

We will start by (1) looping over muons in that event and (2) selecting a *good* , *isolated* muon. with

```
 pt > 30 GeV && |eta| < 2.1
```

```
if ( muPt->at(imu) <30 ||  fabs(muEta->at(imu)) > 2.1 ) continue;
```

The medium working point of the muon Id can be applied as the following:

```
if  (muIsMediumID->at(imu)  <0.5) continue;
```

The muon isolation can be appiled as the following:

```
float IsoMu=muPFChIso->at(imu)/muPt->at(imu);
if ( (muPFNeuIso->at(imu) + muPFPhoIso->at(imu) - 0.5* muPFPUIso->at(imu) )  > 0.0)
IsoMu= ( muPFChIso->at(imu)/muPt->at(imu) + muPFNeuIso->at(imu) + muPFPhoIso->at(imu) - 0.5* muPF

if  (IsoMu > 0.1) continue;
```

To be sure that the muon is from the W-boson, require the Transverse Mass between muon and MET to be greater than 50 GeV.

The TMass_F method is defined in TreeReader.h as:

```
float TMass_F(float pt3lep, float px3lep, float py3lep, float met, float metPhi) {
    return sqrt(pow(pt3lep + met, 2) - pow(px3lep + met * cos(metPhi), 2) - pow(py3lep + met * si
}



Mu4Momentum.SetPtEtaPhiM(muPt->at(imu),muEta->at(imu),muPhi->at(imu),MuMass);
float MuMetTranverseMass= TMass_F(muPt->at(imu), muPt->at(imu)*cos(muPhi->at(imu)),muPt->at(imu)*

if (MuMetTranverseMass <50) continue;

            ///////////////////////////////////////////////
            //Tau Loop will happen here only if good muon is found//
            ///////////////////////////////////////////////
```

Add a loop over the tau collection where indicated, requiring

```
 Tau Pt > 30 GeV and |eta| < 2.3
```

and that the distance between the muon and the tau is `dR>0.5`. We can add this tau loop in the code.

```
            for  (int itau=0 ; itau <nTau; itau++){

                if ( tauPt->at(itau)  <30 ||  fabs(tauEta->at(itau)) > 2.3) continue;

                if (tauByTightMuonRejection3->at(itau)  <0.5 || tauByMVA6LooseElectronRejection->

                if (Mu4Momentum.DeltaR(Tau4Momentum)  <0.5 ) continue;


                Tau4Momentum.SetPtEtaPhiM(tauPt->at(itau),tauEta->at(itau),tauPhi->at(itau),tauMa


                // Fill Denominator
                histoDenominator->Fill(tauPt->at(itau));
                // Fill Numerator
                if (tauByLooseCombinedIsolationDeltaBetaCorr3Hits->at(itau) > 0.5)
                histoNumerator->Fill(tauPt->at(itau));

            } //End Tau Loop
```

The denominator is all fake taus. The numerator is all fake taus that pass the specific ID.

Now your code is all set up. You can

```
./Make.sh jetToTauFakeRate.cc
```

The successful output of this should be

Show ▸ Hide ▾

```
[user@cmslpc37 src]$ ./Make.sh jetToTauFakeRate.cc
====> Here are the libraries: \n -pthread -std=c++11 -Wno-deprecated-declarations -m64 -I/cvmfs/c

====> Created exe file :
-rwxr-xr-x 1 ldodd us_cms 79311 Jan  7 11:21 jetToTauFakeRate.exe
====> Done.
=================================================================
```

The executable can be run

```
./jetToTauFakeRate.exe outputFR.root root://cmseos.fnal.gov://store/user/abdollah/CMSDAS2017Short
```

The output of this should resemble this output

Show ▸ Hide ▾

```
 [user@cmslpc37 src]$ ./test.exe outputFR.root root://cmseos.fnal.gov://store/user/abdollah/CMSDA



 OUTPUT NAME IS:    outputFR.root



 InPUT NAME IS:    root://cmseos.fnal.gov://store/user/abdollah/CMSDAS2017Short/WJetsToLNu_Inc_Sh
nentries_wtn====89828
  Processed events:    89000 of    89828
```

Use the `plotFakeRate.py` macro to plot the jet to tau fake rate. Open the file and edit the input file name as necessary

```
OutFile=TFile("outputFR.root")
```

Now you can run the plotting script.

```
python plotFakeRate.py
```

The output should be

```
[user@cmslpc38 src]$ python plotFakeRate.py
loaded
Info in
```

Exercise 2: Tau ID Efficiency

This is a MC only exercise because we are accessing the Generator-Level information.

This aims to answer the question "How likely is it that a real tau passes a specific tau ID?".

The general flow of section is as follows:

1. Find real reconstructed taus that pass and do-not pass ID.
2. Save histograms of generator-level taus and of reconstructed taus.
3. The efficiency is the ratio of real taus that pass an ID over all real taus.

Ideas to consider while working through the exercise:

1. Try all eta ranges on the taus. Remember to always make an eta cut of 2.1 for your muons and 2.3 for your taus! Why is this eta range important? Answer:
   Show ▸ Hide ▾
   The tracker, with a pseudorapidity : eta < | 2.5 |, is required to reconstruct the trajectories of the charged particles from the tau decay. The method used in this exercise requires events selected with the single muon trigger which has an eta window of eta < | 2.1 |. In order to ensure that the tau cone is within the tracker coverage a window of eta < | 2.3 | is chosen.
2. Try a Pt cut for the generator level taus, this could be added to the TauPreSelection for example! The gen-Pt of the tau will be higher than the reconstructed pt of the tau, because a hadronic tau-decay will have a neutrino in the decay products. Try plotting the efficiency as a function of the genPt of the tau.
3. See the efficiency of different working points! Add in other tauIDs from the `TreeReader.h`
4. The choice of tau Id should be based on both fake rate and efficiency.

Instructions for Exercise

Now create a new file named `tauEfficiency.cc`. At the top of this file, paste

Show Hide

```
/////////////////////////////////////////////////////////////////////////////////////////////////
//   Compiling the code:   ./Make.sh tauEfficiency.cc
//   Running the code:     ./tauEfficiency.exe OutPut.root   Input.root
/////////////////////////////////////////////////////////////////////////////////////////////////
#include "TreeReader.h"
#include  <string > //remove spaces or the file will not be made!
#include   <ostream > //remove spaces or the file will not be made!


int main(int argc, char** argv) {
    using namespace std;

    std::string out = *(argv + 1);

    cout << "\n\n\n OUTPUT NAME IS:    " << out << endl;     //PRINTING THE OUTPUT FILE NAME
    TFile *fout = TFile::Open(out.c_str(), "RECREATE");

    std::string input = *(argv + 2);
    cout << "\n\n\n INPUT NAME IS:    " << input << endl;     //PRINTING THE INPUT FILE NAME
    TFile * myFile = TFile::Open(input.c_str());

    //add the histrograms of All gen-matched taus, and all gen-matched taus passing numerator
    TH1F *    histoDenominator = new TH1F ("histoDenominator","histoDenominator", 300, 0, 300);
    TH1F *    histoNumerator = new TH1F ("histoNumerator","histoNumerator", 300, 0, 300);


    TTree *Run_Tree = (TTree*) myFile->Get("EventTree");
    cout.setf(ios::fixed, ios::floatfield);


    //close the file at the end
    fout->cd();
    histoNumerator->Write();
    histoDenominator->Write();
    fout->Close();
}
```

Remember to edit

```
#include  <string > //remove spaces or the file will not be made!
#include   <ostream > //remove spaces or the file will not be made!
```

You want to read specific branches from the provided Ntuples, so add the following lines to access the branches you want after opening the file, i.e. after `cout.setf(ios::fixed, ios::floatfield);`

Show Hide

```
    Run_Tree->SetBranchAddress("nMC", &nMC);
    Run_Tree->SetBranchAddress("mcPID", &mcPID);
    Run_Tree->SetBranchAddress("mcPt", &mcPt);
    Run_Tree->SetBranchAddress("mcMass", &mcMass);
    Run_Tree->SetBranchAddress("mcEta", &mcEta);
    Run_Tree->SetBranchAddress("mcPhi", &mcPhi);

    Run_Tree->SetBranchAddress("nTau", &nTau);
    Run_Tree->SetBranchAddress("tauPt"  ,&tauPt);
    Run_Tree->SetBranchAddress("tauEta"  ,&tauEta);
    Run_Tree->SetBranchAddress("tauPhi"  ,&tauPhi);
    Run_Tree->SetBranchAddress("tauMass"  ,&tauMass);
    Run_Tree->SetBranchAddress("tauDxy",&tauDxy);
    Run_Tree->SetBranchAddress("tauByTightMuonRejection3", &tauByTightMuonRRejection3);
    Run_Tree->SetBranchAddress("tauByMVA6LooseElectronRejection", &tauByMVA6LooseEElectronRejecti
    Run_Tree->SetBranchAddress("tauByLooseCombinedIsolationDeltaBetaCorr3Hits",&tauByLooseCombiin
```

Now add a loop to process all events after you've declared the branches.

Show ▸ Hide ▾

```
    Int_t nentries_wtn = (Int_t) Run_Tree->GetEntries();
    cout<<"nentries_wtn===="<<nentries_wtn<<"\n";
    for (Int_t i = 0; i < nentries_wtn; i++) {
        Run_Tree->GetEntry(i);
        if (i % 1000 == 0) fprintf(stdout, "\r  Processed events: %8d of %8d ", i, nentries_wtn);
        fflush(stdout);

       ////////////////////////////////////////////////
       //Important Analysis Loop Will Happen Here!!!//
       ////////////////////////////////////////////////


   } //End Processing all entries, end of loop
```

In the *Important Analysis Loop* add a loop over reconstructed taus, with selections, e.g. `Tau Pt > 30  GeV and |eta| < 2.3`

The pre-selection selects only tau leptons which decay into charged particles close to the interaction point: `fabs(tauDxy->at(itau)) < 0.05`. Displaced charged tracks from B-mesons which could fake a tau lepton are discarded.

Paste the following into the *Important Analysis Loop*:

Show ▸ Hide ▾

```
   TLorentzVector MC4Momentum, Tau4Momentum;


   for  (int itau=0 ; itau < nTau; itau++){
      Tau4Momentum.SetPtEtaPhiM(tauPt->at(itau),tauEta->at(itau),tauPhi->at(itau),tauMass->at(ita

      bool TauPtCut = tauPt->at(itau) > 30  && fabs(tauEta->at(itau)) < 2.3 ;
      bool TauPreSelection = fabs(tauDxy->at(itau)) < 0.05 ;

      //Loop Over Generator-Level Tau events

      ///////////////////////////////////////////////
      //Loop Over Generator-Level Taus///////////////
      ///////////////////////////////////////////////
   }//end reconstructed tau loop
```

Now insert the *Loop Over Generator-Level Taus*. This will fill two histograms for the numerator and denominator.

Show ▸ Hide ▾

```
      for  (int imc=0 ; imc < nMC; imc++){

         MC4Momentum.SetPtEtaPhiM(mcPt->at(imc),mcEta->at(imc),mcPhi->at(imc),mcMass->at(imc));

         bool Select_GenTau= abs(mcPID->at(imc))==15&&MC4Momentum.DeltaR(Tau4Momentum) < 0.2;

         if (!Select_GenTau) continue;

         if (TauPtCut && TauPreSelection)
            histoDenominator->Fill(tauPt->at(itau));
         if (TauPtCut && TauPreSelection && tauByLooseCombinedIsolationDeltaBetaCorr3Hits->at(ita
            histoNumerator->Fill(tauPt->at(itau));

         break; //Exit the tau loop, a match was found!
      }
```

Now your code is all set up. You can

```
./Make.sh tauEfficiency.cc
```

and then

```
./tauEfficiency.exe outputEfficiency.root root://cmseos.fnal.gov://store/user/abdollah/CMSDAS2017
```

When you have `outputEfficiency.root` you can run

```
python plotEfficiency.py
```

to create `tauEfficiency.pdf`. The plot will show efficiency as a function of reconstructed tau Pt. This can also be looked at as a function of Generator-Level Pt of the tau.

Exercise 3: ROC Curve

With `outputEfficiency.root` and `outputFR.root` in hand with 4 histograms in each (1 denominator and 3 numerator histograms), a ROC curve can be plotted with the 3 IDs.

If you would like to check the the contents of your .cc files, you can run the following commands to compare to another version.

```
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/tauEfficiency.cc  -O tauEffi
diff tauEfficiency.cc tauEfficiencyGithubVersion.cc
```

Show Hide
If you are having trouble diffing the files, you can reindent both files to have the same indent. In `vim` the command is

```
 gg=G
```

```
The output of this may show differences in spacing and in names but otherwise should be the same.

<//   Running the code:     ./tauEfficiency.exe OutPut.root   Input.root
---
> //   Running the code:     ./tauEfficiency.exe OutPut.root Input.root
6,8c6,7
 <#include
```

Likewise the same commands can be run to check `jetToTauFakeRate.cc`

Show Hide

```
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/jetToTauFakeRate.cc -O jetTo
diff jetToTauFakeRate.cc jetToTauFakeRateGithubVersion.cc
```

Instructions for adding extra histograms to the files

Now that you have produced two plots of fake rate and efficiency. You can now add extra histograms looking at other IDs to your `jetToTauFakeRate.cc` and `tauEfficiency.cc`.

Copy `jetToTauFakeRate.cc` into `jetToTauFakeRate3ids.cc`.

```
cp jetToTauFakeRate.cc jetToTauFakeRate3ids.cc
cp tauEfficiency.cc tauEfficiency3ids.cc
```

Open `jetToTauFakeRate3ids.cc` and replace

```
 TH1F *    histoNumerator = new TH1F ("histoNumerator","histoNumerator", 300, 0, 300);
```

with the following lines

```
    TH1F *    histoNumeratorLoose = new TH1F ("histoNumeratorLoose","histoNumeratorLoose", 300, 0
    TH1F *    histoNumeratorMedium = new TH1F ("histoNumeratorMedium","histoNumeratorMedium", 300
    TH1F *    histoNumeratorTight = new TH1F ("histoNumeratorTight","histoNumeratorTight", 300, 0
```

After

```
Run_Tree->SetBranchAddress("tauByLooseCombinedIsolationDeltaBetaCorr3Hits",&tauByLooseCombinedIso
```

add new branches to access

```
    Run_Tree->SetBranchAddress("tauByMediumCombinedIsolationDeltaBetaCorr3Hits",&tauByMediumCombb
    Run_Tree->SetBranchAddress("tauByTightCombinedIsolationDeltaBetaCorr3Hits",&tauByTightCombiin
```

You can edit the main analysis loop now to use the 2 new tau IDs you've added. Where you have

```
                //Fill Numerator
                if (Mu4Momentum.DeltaR(Tau4Momentum) > 0.5 && TauPtCut && TauPreSelection && tauB
                histoNumerator->Fill(tauPt->at(itau));
```

replace with

```
                // Fill Numerator
                if (Mu4Momentum.DeltaR(Tau4Momentum) > 0.5 && TauPtCut && TauPreSelection && tauB
                histoNumeratorLoose->Fill(tauPt->at(itau));
                if (Mu4Momentum.DeltaR(Tau4Momentum) > 0.5 && TauPtCut && TauPreSelection && tauB
                histoNumeratorMedium->Fill(tauPt->at(itau));
                if (Mu4Momentum.DeltaR(Tau4Momentum) > 0.5 && TauPtCut && TauPreSelection && tauB
                histoNumeratorTight->Fill(tauPt->at(itau));
```

At the end of the code fix

```
    fout->cd();
    histoNumerator->Write();
    histoDenominator->Write();
    fout->Close();
```

to be

```
    fout->cd();
    histoNumeratorLoose->Write();
    histoNumeratorMedium->Write();
    histoNumeratorTight->Write();
    histoDenominator->Write();
    fout->Close();
```

After you've saved your file run

```
./Make.sh jetToTauFakeRate3ids.cc
```

Repeat this process for the `tauEfficiency.cc` so that you now have `tauEfficiency3ids.cc` fully edited (save for changing `Mu4Momentum` to `MC4Momentum`) and run

```
./Make.sh tauEfficiency3ids.cc
```

If time runs short, one can download

```
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/jetToTauFakeRate3ids.cc
./Make.sh jetToTauFakeRate3ids.cc
wget https://raw.githubusercontent.com/abdollah110/Tau-CMSDAS/master/tauEfficiency3ids.cc
./Make.sh tauEfficiency3ids.cc
```

The executable can be run

```
./jetToTauFakeRate3ids.exe outputFR.root     root://cmseos.fnal.gov://store/user/abdollah/CMSDAS2
./tauEfficiency3ids.exe outputEfficiency.root    root://cmseos.fnal.gov://store/user/abdollah/CMS
```

and produce the new outputFR.root and outputEfficiency.root files.

Instructions for plotting scripts

Copy the following into a new file named `plotRocCurve_def.py`. This file will contain the definitions for plotting the roc curve.

Show ▸ Hide ▾

```
'''
Definitions for plotRocCurve.py so that man file can stay cleaner.
2016 CMSDAS Tau Exercise
'''

from subprocess import Popen
from sys import argv, exit, stdout, stderr

import ROOT
import numpy #necessary to pass doubles in python
from array import array

# So things don't look like crap.
ROOT.gROOT.SetStyle("Plain")
ROOT.gROOT.SetBatch(True) #run in batch mode
ROOT.gStyle.SetOptStat(0) #don't plot statistics

####################################
canvas = ROOT.TCanvas("asdf", "adsf", 800, 800)

def get_histo(ntuple_file, histname=''):
    hist =ntuple_file.Get(histname)
    return hist

def get_ratio(histoId, ntuple):
    ''' Get the effi given one ntuple and two histograms in the ntuple '''
    num = get_histo(ntuple,histoId)
    denom = get_histo(ntuple,"histoDenominator")
    effi = num.Integral() / denom.Integral()
    return effi

def produce_tgraph(histoId, ntupleEff,ntupleFR,color,N,MarkerStyle=20):
    ''' Create a TGraph with one point(TGraph) '''
    id_Effi = get_ratio(histoId, ntupleEff)
    eff = numpy.array([id_Effi],dtype=float)
    id_FakeRate = get_ratio(histoId, ntupleFR)
    fr = numpy.array([id_FakeRate],dtype=float)
    n = numpy.array([N],dtype=int)
    tgraph = ROOT.TGraph(N,eff,fr)
    tgraph.SetMarkerStyle(MarkerStyle)
    tgraph.SetMarkerColor(color)
    return tgraph

def produce_pair(histoId, ntupleEff,ntupleFR,N,tgline):
    ''' Add a point to  a roc curve (TGraph) '''
    id_Effi = get_ratio(histoId, ntupleEff)
    eff = numpy.array([id_Effi],dtype=float)
    id_FakeRate = get_ratio(histoId, ntupleFR)
    fr = numpy.array([id_FakeRate],dtype=float)
    tgline.SetPoint(N,eff,fr)

def produce_roc_curve(ntupleEff,ntupleFR,histoId1, legend1, histoId2, legend2, histoId3, legend3,
    ''' Create the Roc Curve Plot, including styles, canvas and saving. '''
    frame = ROOT.TMultiGraph()
    frame.SetTitle(title)
    #Create a TGraph to draw a line behind the other points
    tgline = produce_tgraph(histoId1, ntupleEff, ntupleFR, ROOT.kBlack, 3,1)
    produce_pair(histoId2, ntupleEff, ntupleFR, 1, tgline)
    produce_pair(histoId3,ntupleEff, ntupleFR, 2, tgline)
    #Create TGraphs to add to the TMultiGraph
    tg1 = produce_tgraph(histoId1, ntupleEff, ntupleFR, ROOT.kBlue-9, 1)
    tg2 = produce_tgraph(histoId2, ntupleEff, ntupleFR, ROOT.kRed-9, 1)
    tg3 = produce_tgraph(histoId3, ntupleEff, ntupleFR, ROOT.kOrange-2, 1)
    #Add the TGraphs to the TMultigraph
    frame.Add(tgline)
    frame.Add(tg1)
    frame.Add(tg2)
    frame.Add(tg3)
    #Draw Axis,Line,Points
    frame.Draw("ALP")
    frame.GetXaxis().SetLimits(0.,1.)
    frame.GetYaxis().SetRangeUser(0.,1.)
    #Add Legend for the IDs
    legend = ROOT.TLegend(0.7, 0.7, 0.89, 0.8, "", "brNDC")
    legend.SetFillColor(ROOT.kWhite)
    legend.SetBorderSize(1)
    legend.AddEntry(tg1,legend1,"pe")
    legend.AddEntry(tg2,legend2,"pe")
    legend.AddEntry(tg3,legend3,"pe")
    legend.Draw()
    #Save with a specific file name
    saveas = label+'.png'
    print saveas
    canvas.SaveAs(saveas)
```

Copy the following into a new file named `plotRocCurve.py`. This will contain the commands for creating the ROC curve plot.

```
'''
Usage:python plotRocCurve.py
Script to plot a Roc curve based on 3 IDs from 2 root files with historgrams.
2016 CMSDAS Tau Exercise
'''

from subprocess import Popen
from sys import argv, exit, stdout, stderr

import ROOT
import numpy
from array import array

import plotRocCurve_def
```

```
# So things don't look like crap.
ROOT.gROOT.SetStyle("Plain")
ROOT.gROOT.SetBatch(True) #run in batch mode
ROOT.gStyle.SetOptStat(0) #don't plot statistics

######## Load Files #########
ntuple_fileeff = ROOT.TFile("outputEfficiency.root") #file efficiency
ntuple_filefr = ROOT.TFile("outputFR.root") #file fake rate

#####################################
plotRocCurve_def.produce_roc_curve(
                         ntuple_fileeff,ntuple_filefr, #ntuple files with histograms
                         'histoNumeratorLoose','byCombinedLoose', # 'histogramName No. 1', ' lege
                         'histoNumeratorMedium','byCombinedMedium', #  'histogramName No. 2', ' l
                         'histoNumeratorTight','byCombinedTight',  #  'histogramName No. 3', ' le
                         'ROC; tau Efficiency;tau Fake Rate', #  'Top Title of Plot; X-Axis Title
                         'roc' #name of saved plot
)
```

Now to create the plot you can run

```
python plotRocCurve.py
```

By adding many histograms into your original file you can make many comparisons in this file. For example, by adding another function at the end e.g.

```
plotRocCurve_def.produce_roc_curve(
            ntuple_fileeff,ntuple_filefr, #ntuple files with histograms
            'histoNumerator1','ID WorkingPoint 1', # 'histogramName No. 1', ' legendTitle No. 1'
            'histoNumerator2','ID WorkingPoint 2', #  'histogramName No. 2', ' legendTitle No. 2'
            'histoNumerator3','ID WorkingPoint 3',  #  'histogramName No. 3', ' legendTitle No. 3
            'ROC; tau Efficiency; tau Fake Rate', #  'Top Title of Plot; X-Axis Title; Y-Axis Tit
            'roc_123' #name of saved plot
)
```

This way you can make many different plot combinations without having to edit the definitions to do so.

Responsible: AbdollahMohammadi

-- AbdollahMohammadi - 2017-11-21

This topic: CMS > SWGuideCMSDataAnalysisSchoolLPC2018TauShortExercise
Topic revision: r2 - 2018-01-08 - AbdollahMohammadi