$TZ \rightarrow 3\ell$

First quick plots

Lorenzo Basso

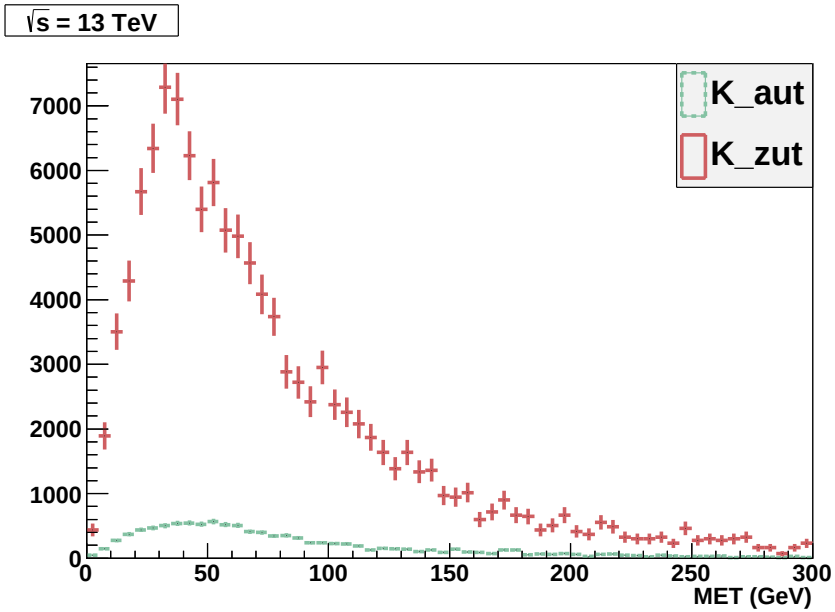
Electrons - preselection

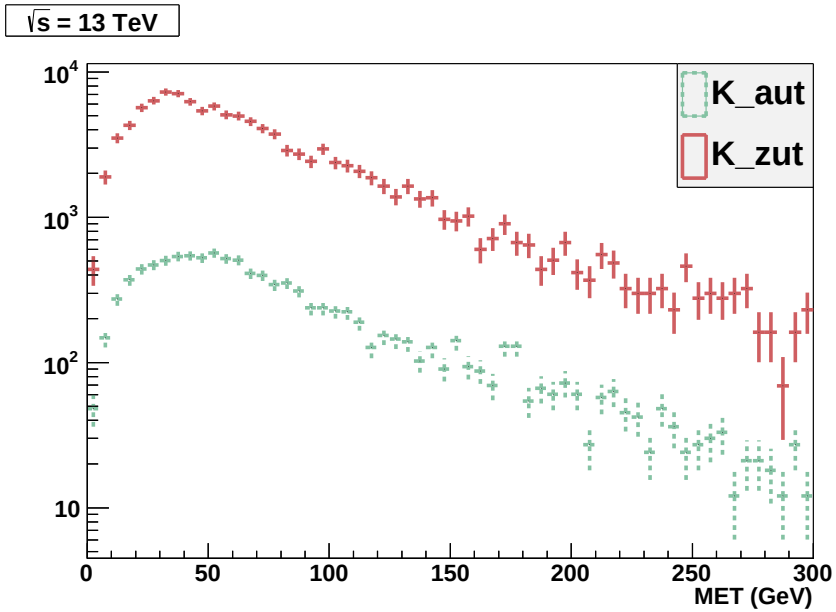
```
// Filling electron containers
// Difere t isolation and pt requirements -> loose and signal electrons
// Object overlap removal with the muons
for(unsigned int i=0; i<event.rec()->electrons().size(); i++)
{
  const RecLeptonFormat *CurrentElectron = &(event.rec()->electrons()[i]);
  double abseta = fabs(CurrentElectron->eta());
  double pt = CurrentElectron->pt();
  if(abseta<2.4 && !(abseta>1.442 && abseta<1.566))
  {
    if(pt>8.) AllE.push_back(CurrentElectron);
    bool IsolatedFromMuons=true;
    for(unsigned int k=0; k<Muons.size();k++)
    // the requirement is DELTAR > .1
      if(CurrentElectron->dr(Muons[k])<=.1)
      {
        IsolatedFromMuons=false;
        break;
      }
    for(unsigned int k=0; k<Antimuons.size();k++)
      if(CurrentElectron->dr(Antimuons[k])<=.1)
      {
        IsolatedFromMuons=false;
        break;
      }
    if(!IsolatedFromMuons) continue;
    for(unsigned int j=0; j<CurrentElectron->isolCones().size(); j++)
    {
      if(fabs(CurrentElectron->isolCones()[j].deltaR() - 0.3) < 0.001)
      {
        double sumpt = CurrentElectron->isolCones()[j].sumPT();
        double sumet = CurrentElectron->isolCones()[j].sumET();
        if(pt>20. && (sumpt+sumet)/pt<0.1)
        {
          if(CurrentElectron->charge()>0) Electrons.push_back(CurrentElectron);
          else                            Positrons.push_back(CurrentElectron);
        }
        if(pt>5. && (sumpt+sumet)/pt<0.2)
        {
          if(CurrentElectron->charge()>0) LooseElectrons.push_back(CurrentElectron);
          else                            LoosePositrons.push_back(CurrentElectron);
        }
        break;
      }
    }
  }
}
 SORTER->sort(AllE);
```

Masses - selection

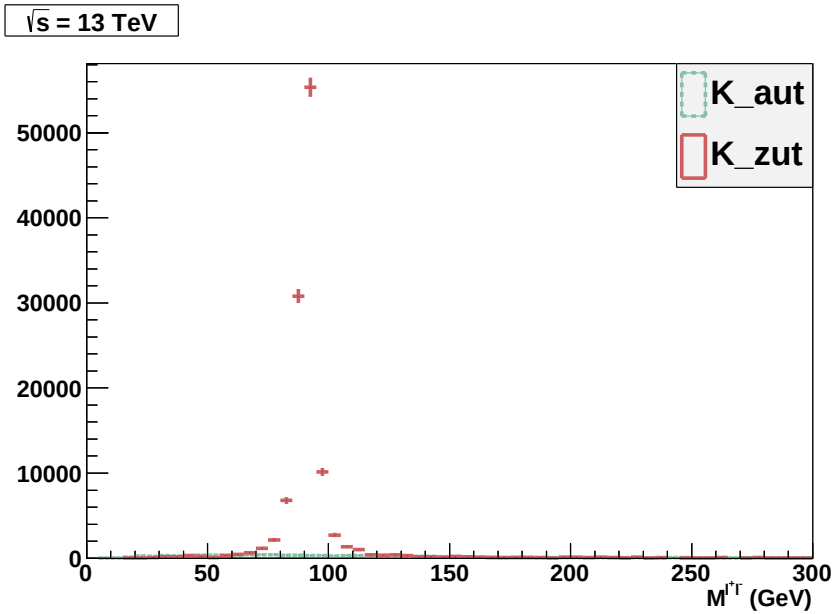
```
// Z->e+e- and top reco with (whichever) 3rd lepton:


        TLorentzVector MET = event.rec()->MET().momentum();
        METpt->Fill(MET.Pt(),weight);

        if (Electrons.size() + Positrons.size() > 1){
        double MZee = 1000.;
        double MT2ee = 1000.;
        double MT3ee = 1000.;

        for (unsigned  int i=0;i<Electrons.size();i++){
         for (unsigned  int j=0;j<Positrons.size();j++){
        TLorentzVector qee = Electrons[i]->momentum();
        qee +=Positrons[j]->momentum();
        double Minv2ee = qee.M();
           if( abs(Minv2ee - 91.12) < abs(MZee - 91.12)) {
          MZee = Minv2ee;
          for (unsigned  int k=0;k<AllLeptons.size();k++){
          if (AllLeptons[k]!=Electrons[i] && AllLeptons[k]!=Positrons[j]) {
          double MT2eein = AllLeptons[k]->mt_met(event.rec()->MET().momentum());
        TLorentzVector qtee = AllLeptons[k]->momentum();
        qtee += myBjets[0]->momentum();
//      cout << "AllLeptons[k]->e() = " << k << " , " << AllLeptons[k]->e() << endl;
//      cout << "AllE[i]->pt() = " << i << " , " << AllE[i]->pt() << endl;
//      cout << "AllE[j]->pt() = " << j << " , " << AllE[j]->pt() << endl;
//      cout << "myBjets[0]->e() = " << myBjets[0]->e() << endl;
//      cout << "qtee->e() = " << qtee.E() << endl;
          double ETsumee = pow(qtee.M()*qtee.M()+qtee.Pt()*qtee.Pt(),0.5)+MET.Pt();
        TLorentzVector qsumee = qtee + MET;
          double valueee = ETsumee*ETsumee - qsumee.Pt()*qsumee.Pt();
          double MT3eein = sqrt(valueee);
           if( abs(MT2eein - 80) < abs(MT2ee - 80) && abs(MT3eein - 170) < abs(MT3ee - 170)) {MT2ee = MT2eein; MT3ee = MT3eein;}}}
        }}}
        MZ->Fill(MZee,weight);
        MW->Fill(MT2ee,weight);
        Mtop->Fill(MT3ee,weight);
        }
```

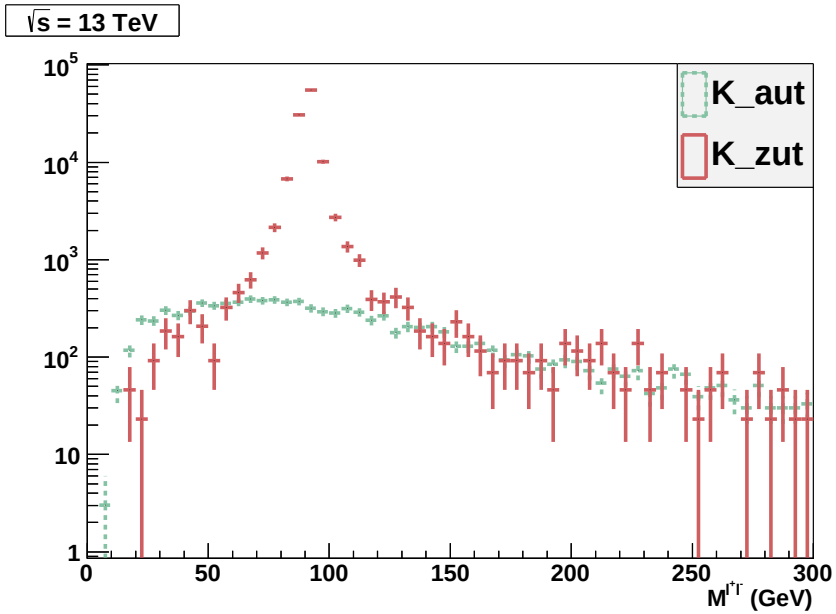
MET - selection



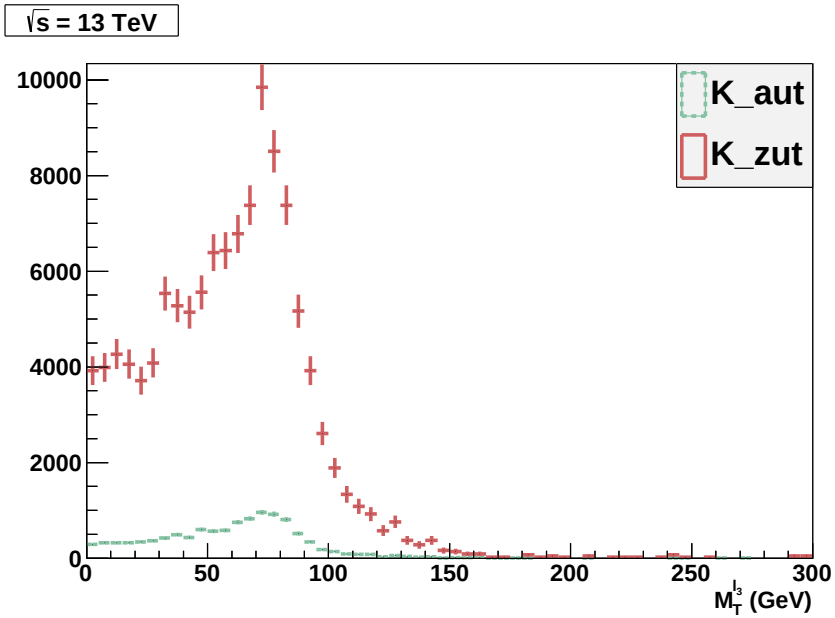


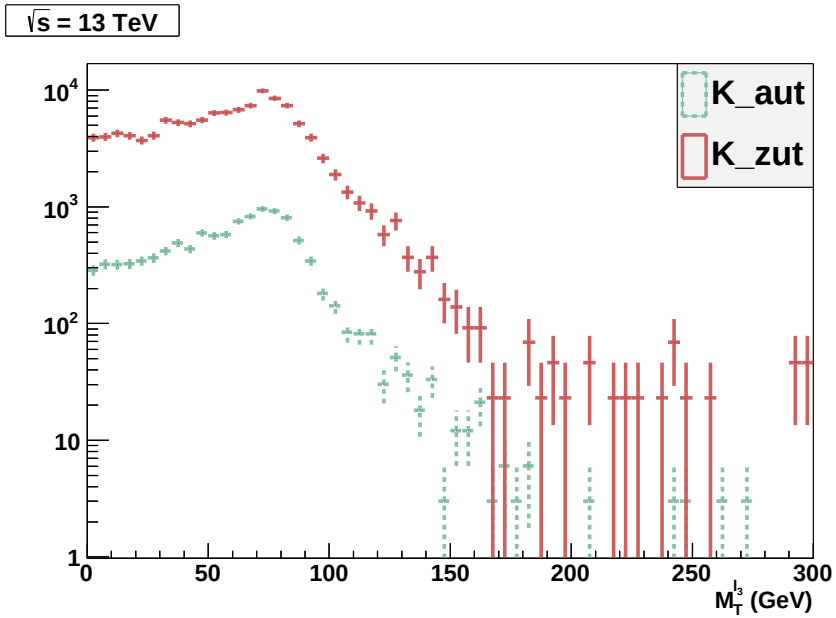
MZ - selection





MW - selection





Mtop - selection