Example $t\bar{t}$ Root Analysis

S. Protopopescu
Sept. 14,2000

For MCC_2 a large number of $t\bar{t}$ have been generated by D0gstar pmc02 and processed by D0reco and RecoAnalyze preco03.

All root ntuple files are produced by RecoAnalyze_x preco03 are in:

`/projects/751/mc_15/reco_analyze/prec03.07.01`

The subdirectory /ttbar has all $t\bar{t}$ files.

The directory
`/prj_root/817/top2_1/preco03/ttbar/root`
has symbolic links to 10 $t\bar{t}$ files with a combined total of 5000 events.

Chaining Files

The directory
`/prj_root/817/top2_1/example`
has macros that show how to chain the files together and access the ntuple for analysis. All macros and root.csh are in this directory.

To chain files you need root v2_25 or higher. To setup the right version:

```
% setup D0RunII t01.xx.00
% source root.csh
```

To merge files

```
% root
root[] .x merge_ttb.C
```

merge_ttb.C:

```
{
TChain ch("Global");
void *i=gSystem->OpenDirectory
        ("/prj_root/817/top2_1/serban/ttbar");
const char* filename;
const char* c="t";
const char* g="/Global";
const char* add="ttbar/";
while(filename=gSystem->GetDirEntry(i)){
  if(filename[0]==c[0]){
    cout<<filename<<endl;
    char* s=new char[strlen(add)+strlen(filename)
                     +strlen(g)+1];
    s=strcat(s,add);
    s=strcat(s,filename);
    s=strcat(s,g);
    cout<<s<<endl;
    ch.Add(s);
    cout<<s<<endl;
  }
}
gSystem->FreeDirectory(i);
//  ch.Merge("merged.root");
return 0;
}
```

Simple Histograms

After chaining files everything is in ntuple named Global, you can histogram any quantity by simply

`root[] Global->Draw(``nch'')` **plot number of charged particles**

`root[] Global->Draw(``empt'',''nempart>0'')` **pt of EM particles**

`root[] Global->Draw(``empt:tpt'',''ntrk>0'')` **pt of EM particles vs track pt**

Above use default values for canvas and histograms. You can set your own values, best done with macros:

`root[] .L style.C` **load macro for canvas style**

`root[] style()` **execute style method**

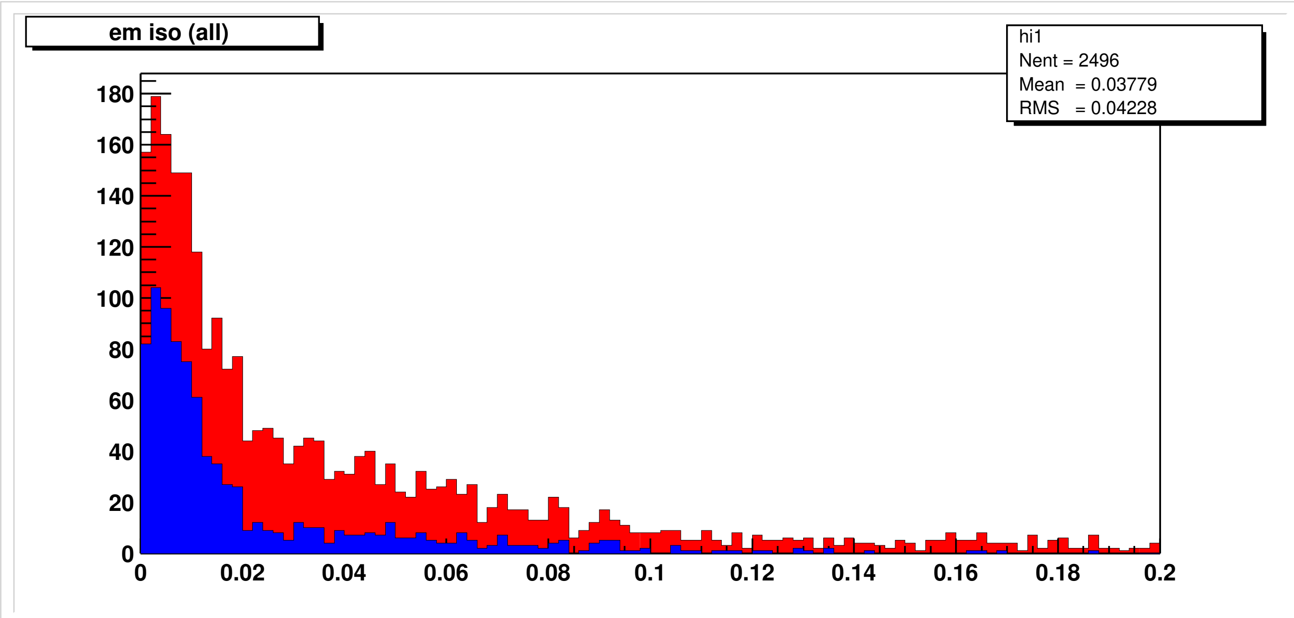
`root[] .x ttbar_iso.C` **plot e isolation histograms**

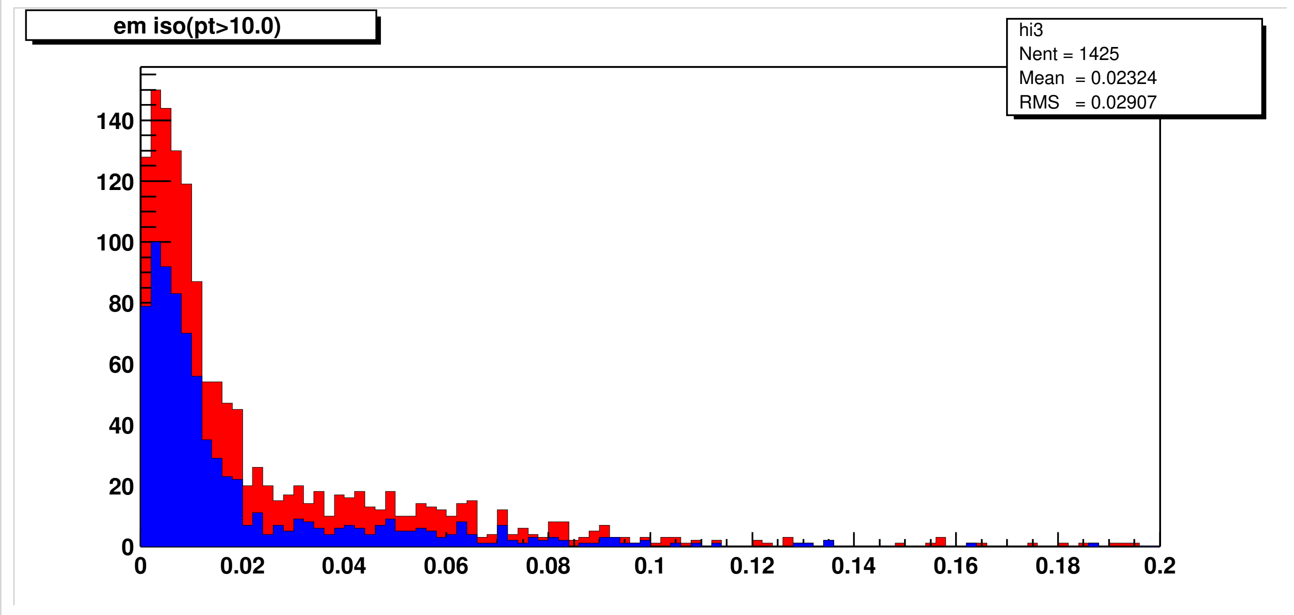
`root[] c1->Print(``e_iso.eps'')` **write eps file**

ttbar_iso.C

```
{
c1->Clear();
c1->Divide(1,2);
c1->cd(1);
TH1F* hi1=new TH1F("hi1","em iso (all)",100, 0.0,0.2);
TH1F* hi2=new TH1F("hi2","e iso (all)",100, 0.0,0.2);
TH1F* hi3=new TH1F("hi3","em iso(pt>10.0)",100, 0.0,0.2);
TH1F* hi4=new TH1F("hi4","e iso(pt>10.0)",100, 0.0,0.2);
hi1->SetFillColor(2);
Global->Draw("iso>>hi1","nempart>0");
hi2->SetFillColor(4);
Global->Draw("iso>>hi2","nempart>0 && nempart==ntrk","Same");
c1->cd(2);
hi3->SetFillColor(2);
Global->Draw("iso>>hi3","empt>10");
hi4->SetFillColor(4);
Global->Draw("iso>>hi4","empt>10 && nempart==ntrk","Same");
return 0;
}
```

ttbar_iso.C


em iso (all)
hi1
Nent = 2496
Mean = 0.03779
RMS = 0.04228
0 20 40 60 80 100 120 140 160 180
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2


em iso(pt>10.0)
hi3
Nent = 1425
Mean = 0.02324
RMS = 0.02907
0 20 40 60 80 100 120 140
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2

Handling Complex Calculations

For given ntuple you can generate a header Global.h + a template macro Global.C to manipulate the variables by

```
root[] Global->MakeClass()
```

Because the Global ntuple varies with the version of RecoAnalyze used to generate it is wise to edit and rename Global.h and Global.C before using them. The example sirectory has PGlobal.h and dilepGlobal.C.

dilepGlobal.C:

Fills histograms with pT and eta of electrons and muons after requiring that the event have at least two leptons and the muon be 0.7 away from a jet. To use it you need 3 steps:

```
root[] .L dilepGlobal.C
root[]  PGlobal pg(Global);
root[]  pg->Loop(1,10.0,10.0,15.0);
```

the last step fills histograms requiring more than 1 lepton satisfying $e\ p_T > 10.0$, $\mu\ p_T > 10.0$ and counts jets with $p_T > 15.0$.

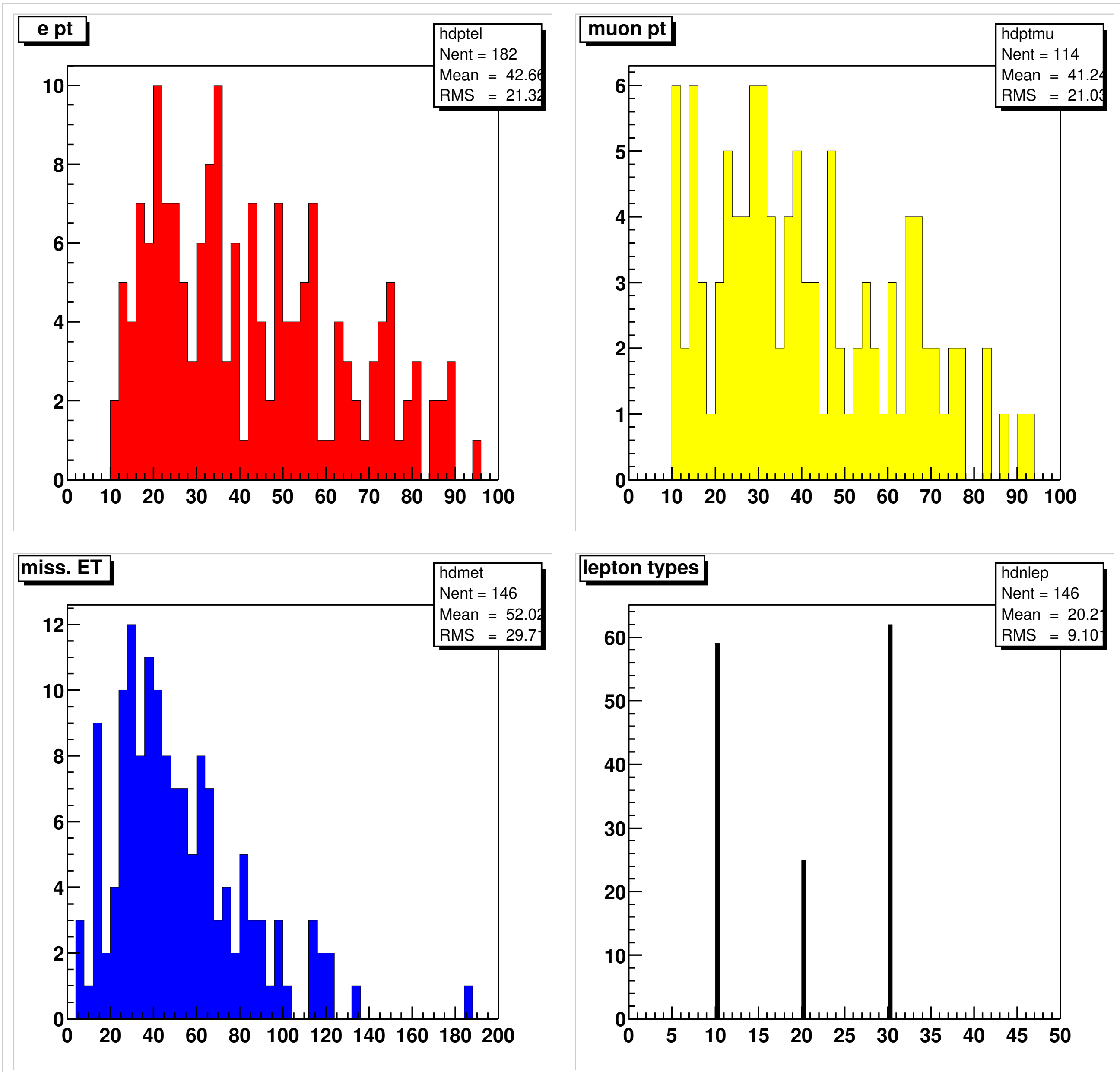
Combining Macros

All macros in the example can be run with one combined macro AllInOne.C and write out eps files of all histograms:

```
root[] .x AllInOne.C
```

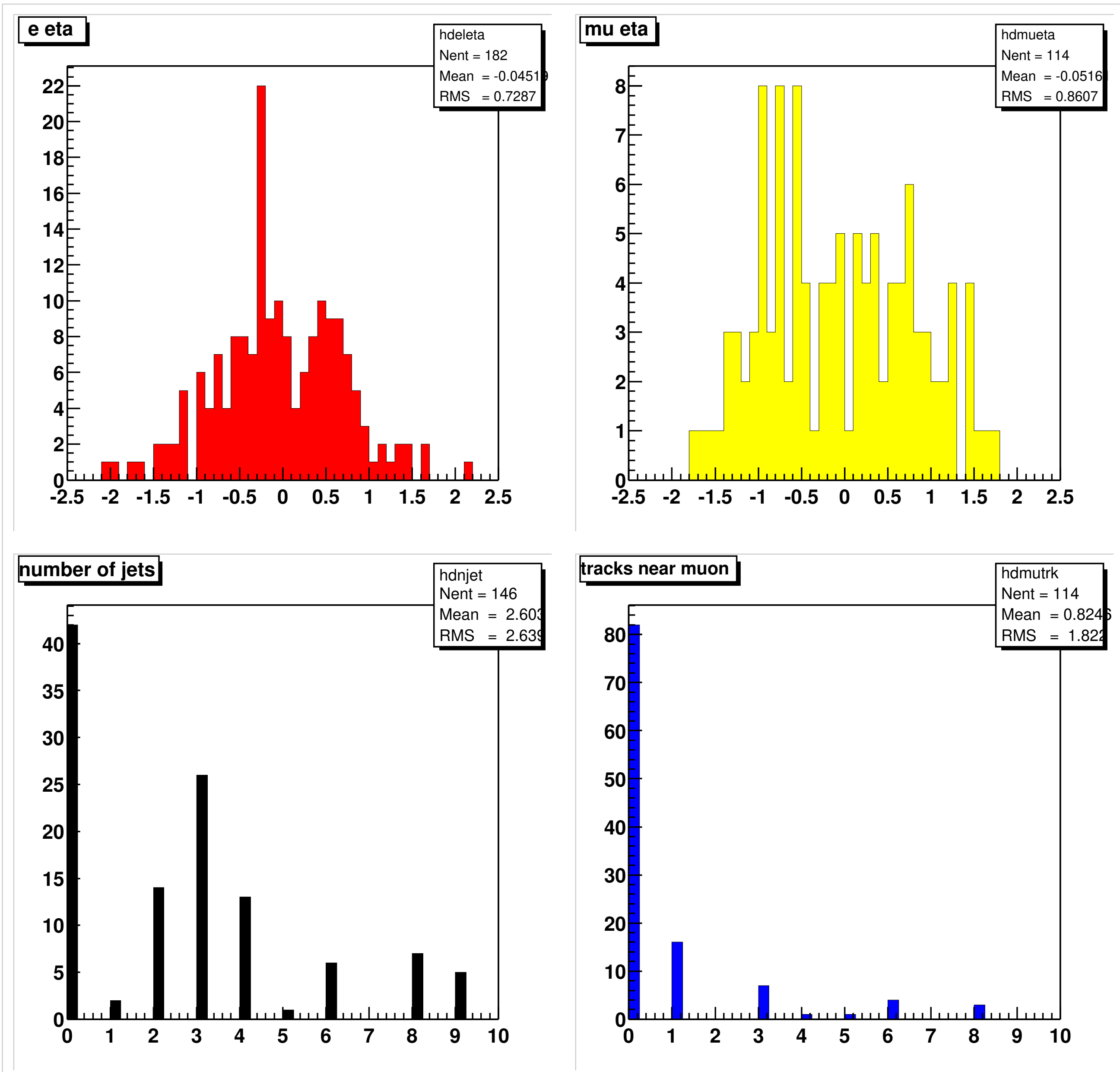
```
{
gROOT->Macro("merge_ttb.C");
gROOT->LoadMacro("style.C");
style();
gROOT->Macro("ttbar_iso.C");
c1->Print("e_iso.eps");
gROOT->LoadMacro("dilepGlobal.C");
PGlobal pg(Global);
pg->Loop(1,10.0,10.0,15.0);
gROOT->Macro("pt_hist.C");
c1->Print("pt_dilep.eps");
gROOT->Macro("eta_hist.C");
c1->Print("eta_dilep.eps");
}
```

pt_hist.C



e pt
hdptel
Nent = 182
Mean = 42.66
RMS = 21.32
muon pt
hdptmu
Nent = 114
Mean = 41.24
RMS = 21.03
miss. ET
hdmet
Nent = 146
Mean = 52.02
RMS = 29.71
lepton types
hdnlep
Nent = 146
Mean = 20.21
RMS = 9.101

eta_hist.C



e eta
hdeleta
Nent = 182
RMS = 0.7287
mu eta
hdmueta
Nent = 114
Mean = -0.0516
RMS = 0.8607
number of jets
hdnjet
Nent = 146
tracks near muon
hdmutrk
Nent = 114

PGlobal.h

```
/////////////////////////////////////////////////////////////
//   This class has been automatically generated
//     (Fri Sep 15 14:56:50 2000 by ROOT version2.25/00)
//   from TTree Global/
//   found on file: ttbar/ttbar_7.root
/////////////////////////////////////////////////////////////


#ifndef PGlobal_h
#define PGlobal_h

#include <TChain.h>
#include <TFile.h>

class PGlobal {
   public :
   TTree          *fTree;    //pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //current Tree number in a TChain
//Declaration of leaves types
   Int_t           CH_nch;
   Float_t         CH_chE[50];
   Float_t         CH_cheta[50];
      :
      :
   Int_t           CHMC_nchmc;
   Float_t         CHMC_chmcE[50];
     :
     :
   Int_t           CHVTX_nchvtx;
     :
     :
   Int_t           EMPART_nempart;
     :
     :
```

```
Int_t          GTRACK_ngtrk;
  :
  :
Int_t          JCC7_JCC7njet;
  :
  :
Int_t          JCCG_JCCGnjet;
  :
  :
Int_t          JCM7_JCM7njet;
  :
  :
Int_t          JKC1_JKC1njet;
  :
  :
Int_t          MCEM_nmcem;
  :
  :
Int_t          MCTRK_nmctrk;
  :
  :
Int_t          MCVTX_nmcvtx;
  :
  :
Float_t        MET_MET;
  :
  :
Int_t          MuoGTR_Nmgtr;
  :
  :
Int_t          Muonid_Nmuonid;
  :
  :
Int_t          PRT_nprt;
Float_t        PRT_E[1000];
  :
```

```
     :
   Int_t           PVReco_prnvtx;
     :
     :
   Int_t           PVSel_psnvtx;
     :
     :
   Int_t           SVReco_srnvtx;
     :
     :
   Int_t           TAU_ntaus;
     :
     :
   Int_t           TRK_ntrk;
     :
     :
   Int_t           VTX_nvtx;
     :
     :
//List of branches
   TBranch        *b_CH;
   TBranch        *b_CHMC;
   TBranch        *b_CHVTX;
   TBranch        *b_EMPART;
   TBranch        *b_EVT;
   TBranch        *b_EvtInfo;
   TBranch        *b_GTRACK;
   TBranch        *b_JCC7;
   TBranch        *b_JCCG;
   TBranch        *b_JCM7;
   TBranch        *b_JKC1;
   TBranch        *b_JKM1;
   TBranch        *b_MCEM;
   TBranch        *b_MCTRK;
   TBranch        *b_MCVTX;
   TBranch        *b_MET;
```

```
   TBranch        *b_MuoGTR;
   TBranch        *b_MuoMC;
   TBranch        *b_MuoSHit;
   TBranch        *b_MuoSeg;
   TBranch        *b_MuoTrack;
   TBranch        *b_MuoWHit;
   TBranch        *b_Muonid;
   TBranch        *b_PRT;
   TBranch        *b_PVReco;
   TBranch        *b_PVSel;
   TBranch        *b_SVReco;
   TBranch        *b_TAU;
   TBranch        *b_TRK;
   TBranch        *b_VTX;

   PGlobal(TTree *tree=0);
   ~PGlobal();
   Int_t Cut(Int_t entry);
   Int_t GetEntry(Int_t entry);
   Int_t LoadTree(Int_t entry);
   void  Init(TTree *tree);
   void  Loop(int nlsel, float ecut, float mucut, float jetcut);
   void  Notify();
   void  Show(Int_t entry = -1);
};

#endif

#ifdef PGlobal_cxx
PGlobal::PGlobal(TTree *tree)
{
    if(tree ==0){
gROOT->Macro("merge_ttb.C");
tree = (TTree*)gDirectory->Get("Global");
    }
```

```
   Init(tree);
}

PGlobal::~PGlobal()
{
   if (!fTree) return;
   delete fTree->GetCurrentFile();
}

Int_t PGlobal::GetEntry(Int_t entry)
{
// Read contents of entry.
   if (!fTree) return 0;
   return fTree->GetEntry(entry);
}
Int_t PGlobal::LoadTree(Int_t entry)
{
// Set the environment to read one entry
   if (!fTree) return -5;
   Int_t centry = fTree->LoadTree(entry);
   if (centry < 0) return centry;
   if (fTree->IsA() != TChain::Class()) return centry;
   TChain *chain = (TChain*)fTree;
   if (chain->GetTreeNumber() != fCurrent) {
      fCurrent = chain->GetTreeNumber();
      Notify();
   }
   return centry;
}

void PGlobal::Init(TTree *tree)
{
//   Set branch addresses
   if (tree == 0) return;
   fTree    = tree;
   fCurrent = -1;
```

```
   fTree->SetBranchAddress("CH",&CH_nch);
   fTree->SetBranchAddress("CHMC",&CHMC_nchmc);
       :
       :
}

void PGlobal::Notify()
{
//   called by LoadTree when loading a new file
//   get branch pointers
   b_CH = fTree->GetBranch("CH");
   b_CHMC = fTree->GetBranch("CHMC");
     :
     :
}

void PGlobal::Show(Int_t entry)
{
// Print contents of entry.
// If entry is not specified, print current entry
   if (!fTree) return;
   fTree->Show(entry);
}
Int_t PGlobal::Cut(Int_t entry)
{
// This function may be called from Loop.
// returns  1 if entry is accepted.
// returns -1 otherwise.
   return 1;
}
#endif // #ifdef PGlobal_cxx
```

dilepGlobal.C

```
#define PGlobal_cxx
#include "PGlobal.h"
#include "TH2.h"
#include "TStyle.h"
#include "TCanvas.h"

void PGlobal::Loop(int nlsel, float ecut,
                   float mucut, float jetcut)
{
   if (fTree == 0) return;

   hdmet=new TH1F("hdmet","miss. ET",50,0.,200.);
   hdptel=new TH1F("hdptel"," e pt",50, 0., 100.);
   hdptmu=new TH1F("hdptmu"," muon pt",50, 0., 100.);
   hdnlep=new TH1F("hdnlep","lepton types",100,0,50);
   hdnjet=new TH1F("hdnjet","number of jets",40,0,10);
   hdeleta=new TH1F("hdeleta","e eta",50,-2.5,2.5);
   hdmueta=new TH1F("hdmueta","mu eta",50,-2.5,2.5);
   hdmutrk=new TH1F("hdmutrk","tracks near muon",40,0,10);

   Int_t nentries = Int_t(fTree->GetEntries());
   cout<<"no. of entries "<<nentries<<endl;
   Int_t nbytes = 0, nb = 0;
   for (Int_t jentry=0; jentry<nentries;jentry++) {
      Int_t ientry = LoadTree(jentry);
      nb = fTree->GetEntry(jentry);   nbytes += nb;
      // if (Cut(ientry) < 0) continue;
```

```
    Int_t nlep=0;
    Int_t nel=0;
    Int_t nmuo=0;
    Int_t njet=0;
    Float_t pt_el[4];
    Float_t pt_muo[4];
    Float_t eta_el[4];
    Float_t eta_muo[4];
    Int_t ntrkdR[4];

    // find mu's

    for(int i2=0; i2<Muonid_Nmuonid; ++i2){
float deltaR2=0.5;
for(int j1=0; j1<JCC7_JCC7njet; ++j1){
    if(JCC7_JCC7PT[j1] > jetcut) ++njet;
    float dphi=abs(JCC7_JCC7Phi[j1]-Muonid_Idphi[i2]);
    if(dphi>3.1415927) dphi=6.2831853-dphi;
    float deta=JCC7_JCC7Eta[j1]-Muonid_Ideta[i2];
    float dR2=dphi*dphi+deta*deta;
    if(dR2<deltaR2) deltaR2=dR2;
}
if(Muonid_Idpt[i2]>mucut && deltaR2>0.49 && nmuo<3){
    pt_muo[nmuo]=Muonid_Idpt[i2];
    eta_muo[nmuo]=Muonid_Ideta[i2];
    ntrkdR[nmuo]=Muonid_IdntrkdR[i2];
    nmuo++;
}
    }
```

```
    //  find e's
    for(int i1=0; i1<TRK_ntrk && nel<3 ; ++i1){
if(EMPART_empt[i1]>ecut){
    pt_el[nel]=EMPART_empt[i1];
    eta_el[nel]=EMPART_emeta[i1];
    nel++;
}
    }

    nlep=nel+nmuo;
    int ityp=0;
    if(nel>1) ityp=10;
    if(nmuo > 1) ityp=20;
    if(nel>0 && nmuo>0) ityp=30;

    if(ityp>nlsel){
hdnlep->Fill(ityp);
hdmet->Fill(MET_MET);
hdnjet->Fill(njet);
for(int ilep1=0; ilep1<nel; ++ilep1){
    hdptel->Fill(pt_el[ilep1]);
    hdeleta->Fill(eta_el[ilep1]);
}
for(int ilep2=0; ilep2<nmuo; ++ilep2){
    hdptmu->Fill(pt_muo[ilep2]);
    hdmueta->Fill(eta_muo[ilep2]);
    hdmutrk->Fill(ntrkdR[ilep2]);
}
    }
```

```
    } // Loop over entries
}
```