# Analyse avec root

- But: évaluer la facilité d’utilisation de root pour l’analyse
- Cas d’école: analyse des jets par comparaisons reco–MC:
  - matching jets reco et jets MC
  - histos des différences d’énergie, d’angle, etc...

- Pas d'équivalent à nt/plot 1.toto.for !
- Il faut donc utiliser l'équivalent de nt/loop:
  - créer un source « modèle » par la méthode MakeClass:
  - ```
    root [1] f = new Tfile("qcd.root");
    root [2] MC_EXAM->MakeClass("myana");
    Files: myana.h and myana.C generated from
    Tree: MC_EXAM
    ```
- contrairement à PAW, il n'y a pas de commande pour faire la boucle sur les événements, c'est la classe fabriquée qui contient la boucle.
    ```

- Premier problème: le code engendré pour le ntuple de mc_exam est faux, sauf avec la dernière version de root disponible par upd (v2_23_09a), nécessaire de toute façon pour lire les ntuples de mc_exam.
- La classe engendrée contient (entre autre)
  - les variables du ntuple comme membres publics:
    - ```
      Int_t           CAL_ncal;
      Float_t         CAL_e_cal[1000];
      Int_t           CAL_ieta_cal[1000];
      Int_t           CAL_iphi_cal[1000];
      Int_t           CAL_lyr_cal[1000];
      Float_t         CAL_pt_cal[1000];
      ```
  - un constructeur qui, par défaut, ouvre le fichier de ntuple qui avec lequel on a engendré la classe
      ```

- Une méthode Loop() avec une boucle sur les événements

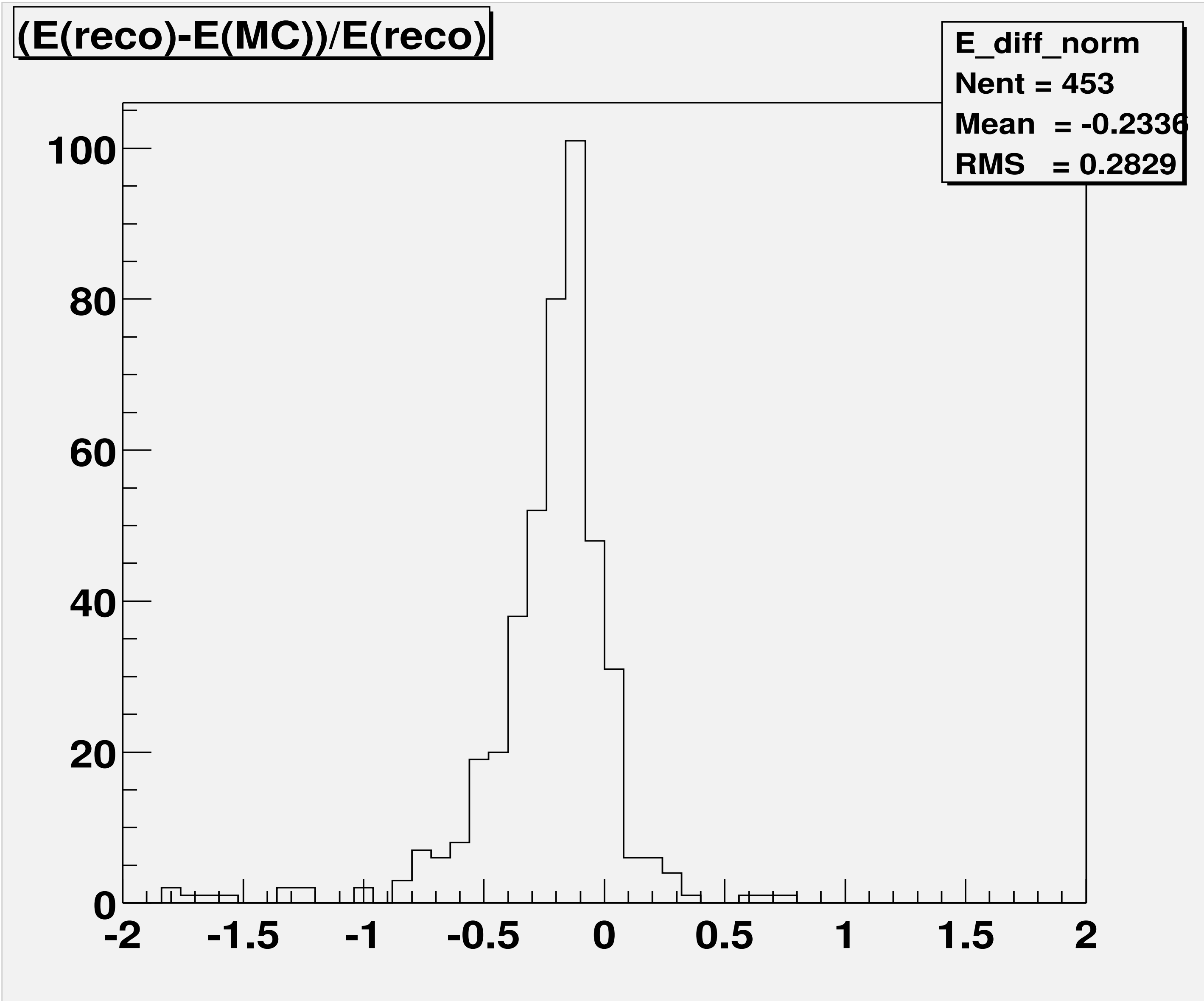
- Il faut ensuite modifier le code pour créer ses histogrammes, par ex. comme membres publiques de la classe, et les remplir dans la boucle.

- Ensuite, il faut faire connaître de root notre classe, en créer une instance, invoquer la méthode Loop(), puis afficher les histogrammes:

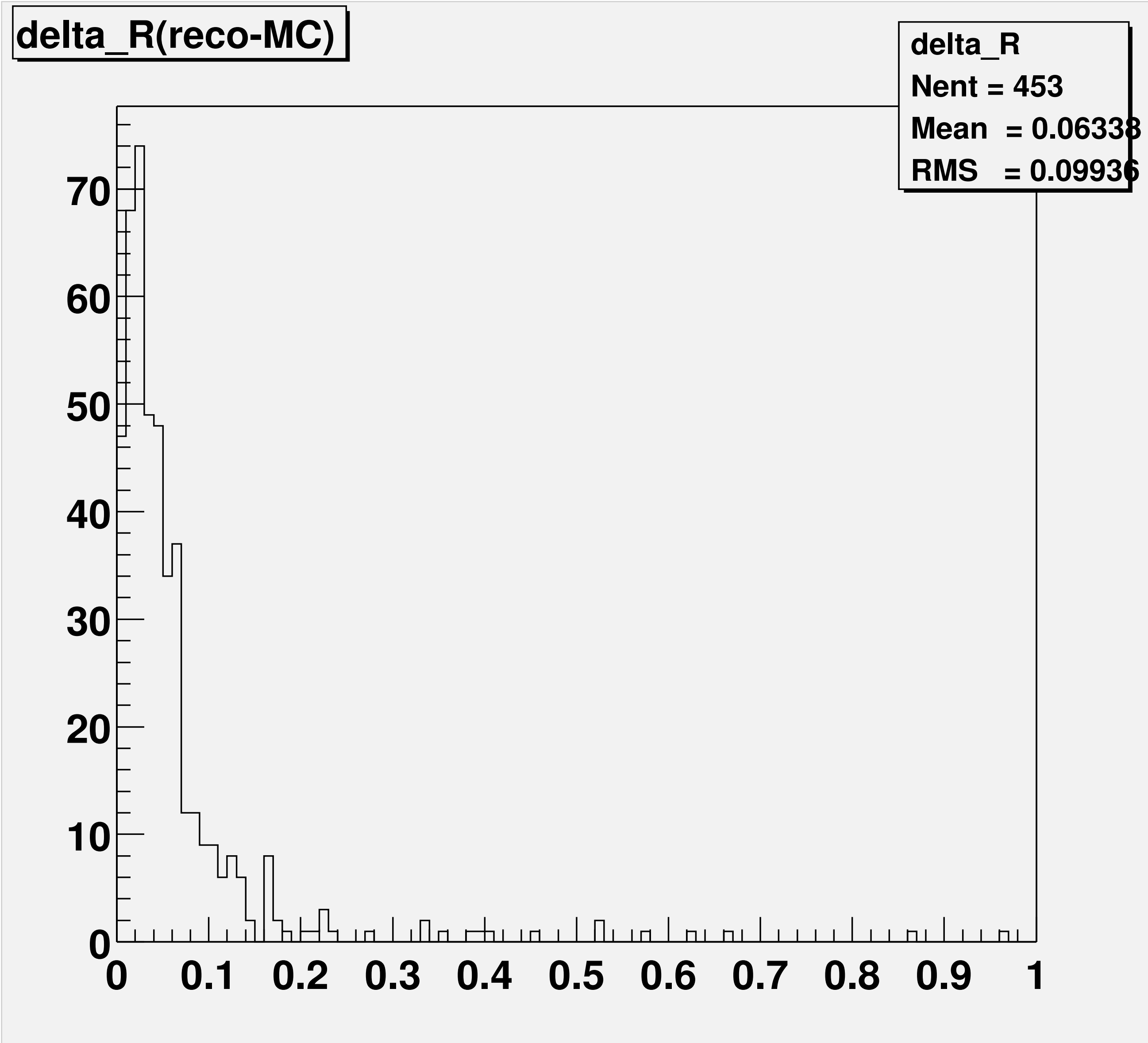
  - ```
    root [9] .L MCExam_ana.cpp
    root [10] ana = new MCExam_ana();
    root [11] ana->Loop();
    root [12] ana->H_E_diff_norm->Draw();
    ```
    ```

- Voici ma boucle sur les événements:
  - ```
    Int_t nentries = Int_t(fTree->GetEntries());
    for (Int_t i=0; i<nentries;i++)
    {
      fTree->GetEvent(i);
      int index_kt, index_ktk;
      if ( EnergyMatched(index_kt,index_ktk) )
      {
       H_E_diff->Fill(e_kt[index_kt]-e_ktk[index_ktk]);
    H_E_diff_norm->Fill((e_kt[index_kt]-e_ktk[index_ktk])/e
    _kt[index_kt]);
       float dr = sqrt(
           (eta_kt[index_kt]-eta_ktk[index_ktk])**2 +
           (phi_kt[index_kt]-phi_ktk[index_ktk])**2 );
       if ( dr > M_PI ) dr = M_2_PI - dr;
       H_delta_R->Fill(dr);
      }
     }
    ```
    ```

# QCD: (Ereco–Emc)/Ereco



# QCD: ΔR



delta_R(reco-MC)
delta_R
Nent = 453
Mean = 0.06338
RMS = 0.09936
0
10
20
30
40
50
60
70
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

# Conclusions

- Root ne permet pas d'utiliser des fonctions comme variable ou comme poids, ce qui est pourtant très utile.
- Root fournit un équivalent de nt/loop qui est (relativement) simple d'utilisation.
- Comme pour tout nouvel outil, l'apprentissage est assez long et pénible...