Introduction à Root

root.cern.ch

Root

root.cern.ch

•Root est un programme spécialement concu pour **l'analyse de données en physique des particules**. Il a été créé dans les années 90 au CERN.
•Le language utilisé est le C++.
•La plupart des personnes (ou toutes les personnes!) qui travaillent en physique des particules utilisent ROOT.

•ROOT est essentiellement une collection de classes et de librairies qui permettent de lire des données (format .txt par exemple) de remplir des histogrammes/des graphiques afin d'analyser ces données.
•ROOT à également une puissante librairie graphique qui permet la simulation de detecteurs complexes en physique des particules.

ROOT est 100% gratuit. Pour telecharger la derniere version (5.28):

http://root.cern.ch/drupal/content/production-version-528

S'installe sur Linux, Windows, Mac OS.

Premiere session

Root peut fonctionner en interpreteur. C'est à dire qu'il peut lire directement des lignes de code sans avoir besoin de compiler.
C'est ce que nous allons faire pour les TP.
pour demarer ROOT taper dans un terminal:
> root
la session root est ouverte.
Ecrire les lignes de codes et pour quitter taper:
> .q

Voici un exmple (trivial):

```
xx169:~ sebastienmurphy$ root
   ****************************************
   *                                      *
   *       W E L C O M E  to  R O O T     *
   *                                      *
   *   Version   5.24/00      29 June 2009  *
   *                                      *
   *  You are welcome to visit our Web site  *
   *          http://root.cern.ch         *
   *                                      *
   ****************************************

ROOT 5.24/00 (trunk@29257, Jun 30 2009, 09:23:51 on macosx)

CINT/ROOT C/C++ Interpreter version 5.17.00, Dec 21, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] double a=2
root [1] double b=3
root [2] cout<<" a="<<a<<" b="<<b<<" a+b="<<a+b<<endl
 a=2 b=3 a+b=5
(class ostream)-1609235200
root [3] .q
xx169:~ sebastienmurphy$
```

Macro root

En principe nous aurons des nombreuses lignes de code à ecrire. Il sera beaucoup trop fastidieux et compliquer de devoir tout rentrer à la main comme precedemment.
=> on crée une ce qu'on appelle une macro ROOT: C'est un fichier .C qui contient les commandes que l'on veut executer.

(macro1.C)

tout écrire entre accolades

```
{
   double a=2;
   double b=3;
   cout << "Voici ma premiere macro ROOT!" << endl;
   cout<<" a="<<a<<" b="<<b<<" a+b="<<a+b<<endl;
}
```

```
xx169:~ sebastienmurphy$ root
   ******************************************
   *                                        *
   *       W E L C O M E  to  R O O T       *
   *                                        *
   *   Version   5.24/00      29 June 2009   *
   *                                        *
   *  You are welcome to visit our Web site *
   *          http://root.cern.ch           *
   *                                        *
   ******************************************

ROOT 5.24/00 (trunk@29257, Jun 30 2009, 09:23:51 on macosx)

CINT/ROOT C/C++ Interpreter version 5.17.00, Dec 21, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] .x macro1.C
Voici ma premiere macro ROOT!
 a=2 b=3 a+b=5
root [1]
```

pour executer la macro taper .x <nom_de_la_macro>

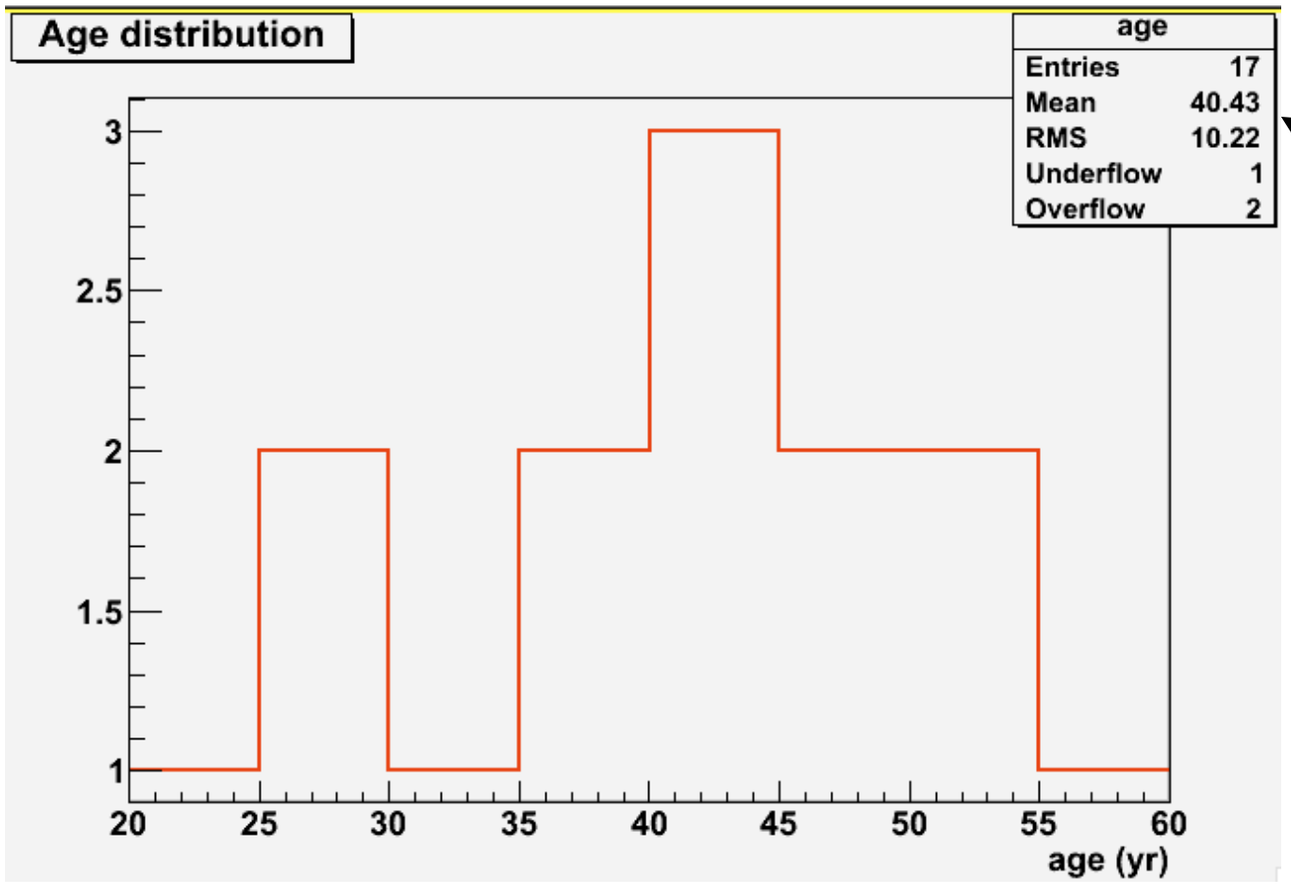
Creer un histogramme (TH1)

On crée un objet de la classe TH1F (ici un pointeur que l'on appelle h) de la facon suivante:

(macro2.C)

```
TH1F *h= new TH1F ("name","title",nbins,1stbin,lastbin)
```

```
TH1F* hage = new TH1F("age","Age distribution;age (yr)",8,20,60);
root [] float
ages[17]={47,35,42,19,56,54,40,39,29,27,30,60,62,23,53,49,42};
root [] for (int i=0; i<17; ++i) hage->Fill(ages[i]);
root [] hage->Draw();
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [] hage->SetLineWidth(2);
root [] hage->SetLineColor(2);
root [] hage->Draw();
```



la boite de statitistique s'affiche par default et donne le nombre d'entrées la valeur moyenne et le RMS.

Lire un fichier de données et les rentrer dans un histo

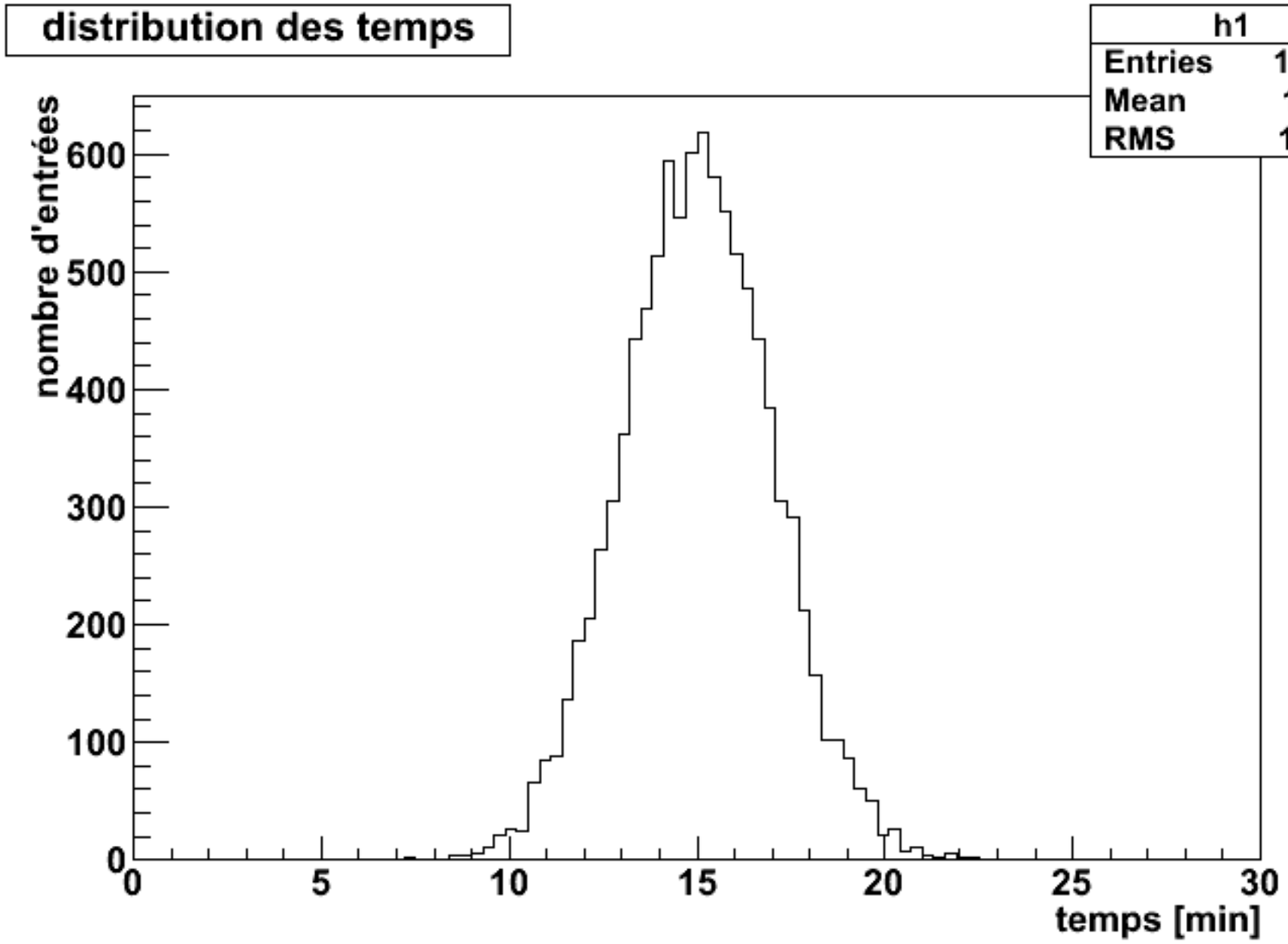
(macro3.C)

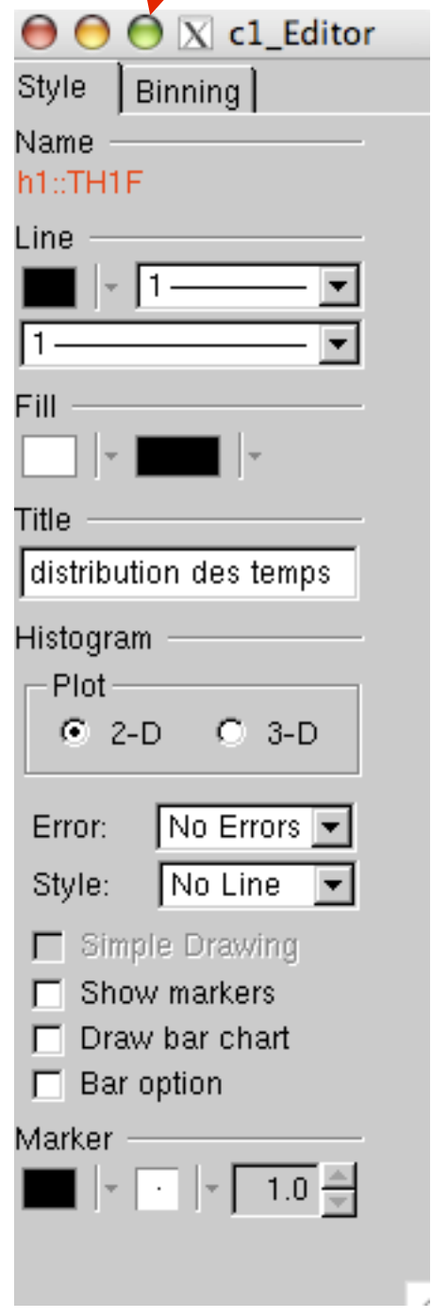
Dans cet exemple on veut lire un fichier time.txt qui contient 10'000 mesures (par exemple on répète une epreuve 10'000 fois et à chaque fois on chronometre le temps mis pour la realiser).
On veut lire ces données et les mettre dans un histogramme pour voir comment ces données sont distribuées.

```
{
    gROOT->SetStyle("Plain");//cree un evironement graphique plus joli
    h1= new TH1F("h1","distribution des temps; temps [s];nombre d'entrées",100,0,30);/* je declare l'histo,
                                                                                       avec les titres pour les axes y*/
    fstream ftime("time.txt",ios::in);//j'ouvre le fichier time.txt
    double t;
    while(!f.eof())//tant que l'on a pas atteint la fin du fichier
    {
        ftime>>t;//je lis l'entrée dans le fichier et je la stoke dans t
        h1->Fill(t);// je remplis l'histogramme avec la valeur
    }
    ftime.close();// on peut fermer la fichier maintenant
    h1->Draw();
}
```

distribution obtenue à partir des valeurs du fichier

en faisant un clic droit sur l'histo on peut selectionner le "draw panel".

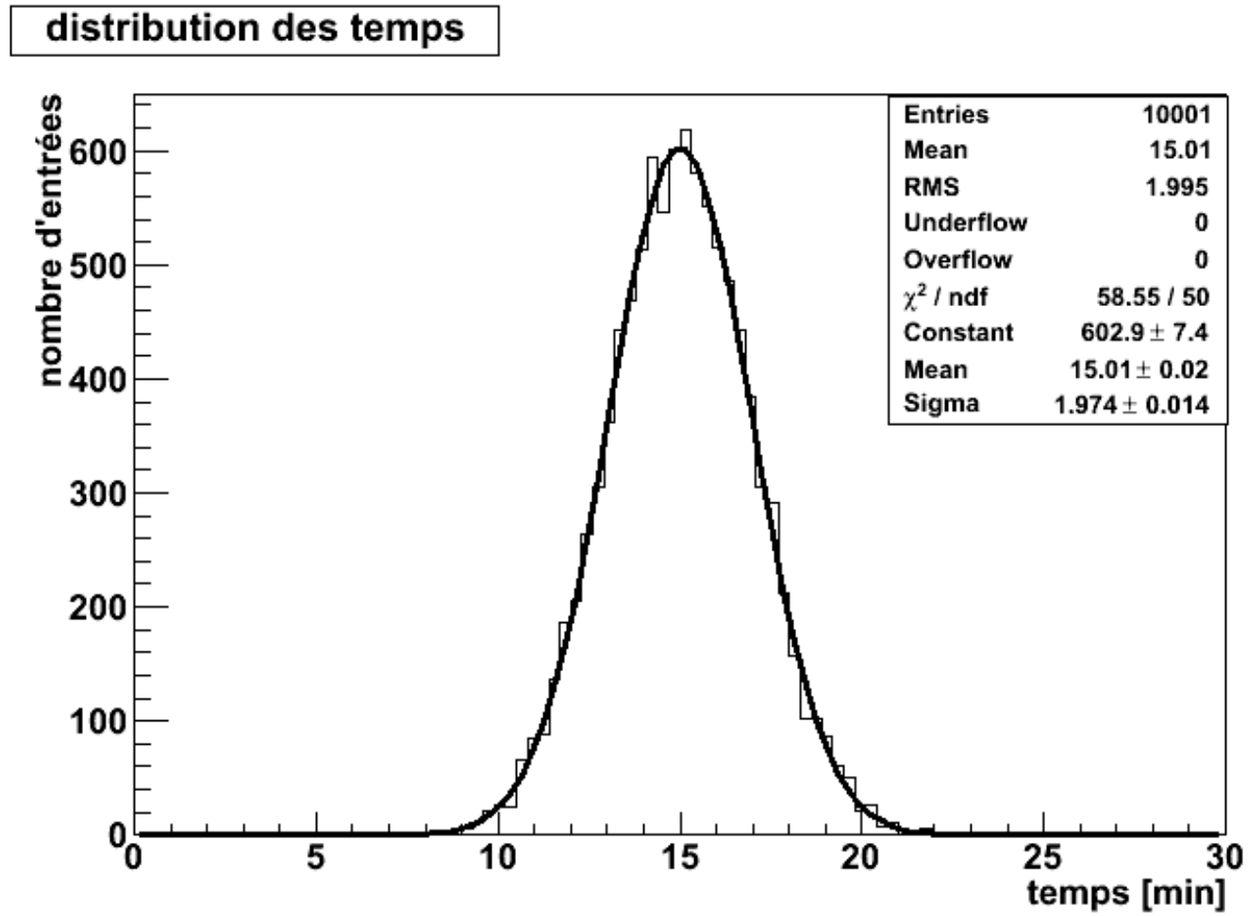




Fiter le données

(macro4.C)

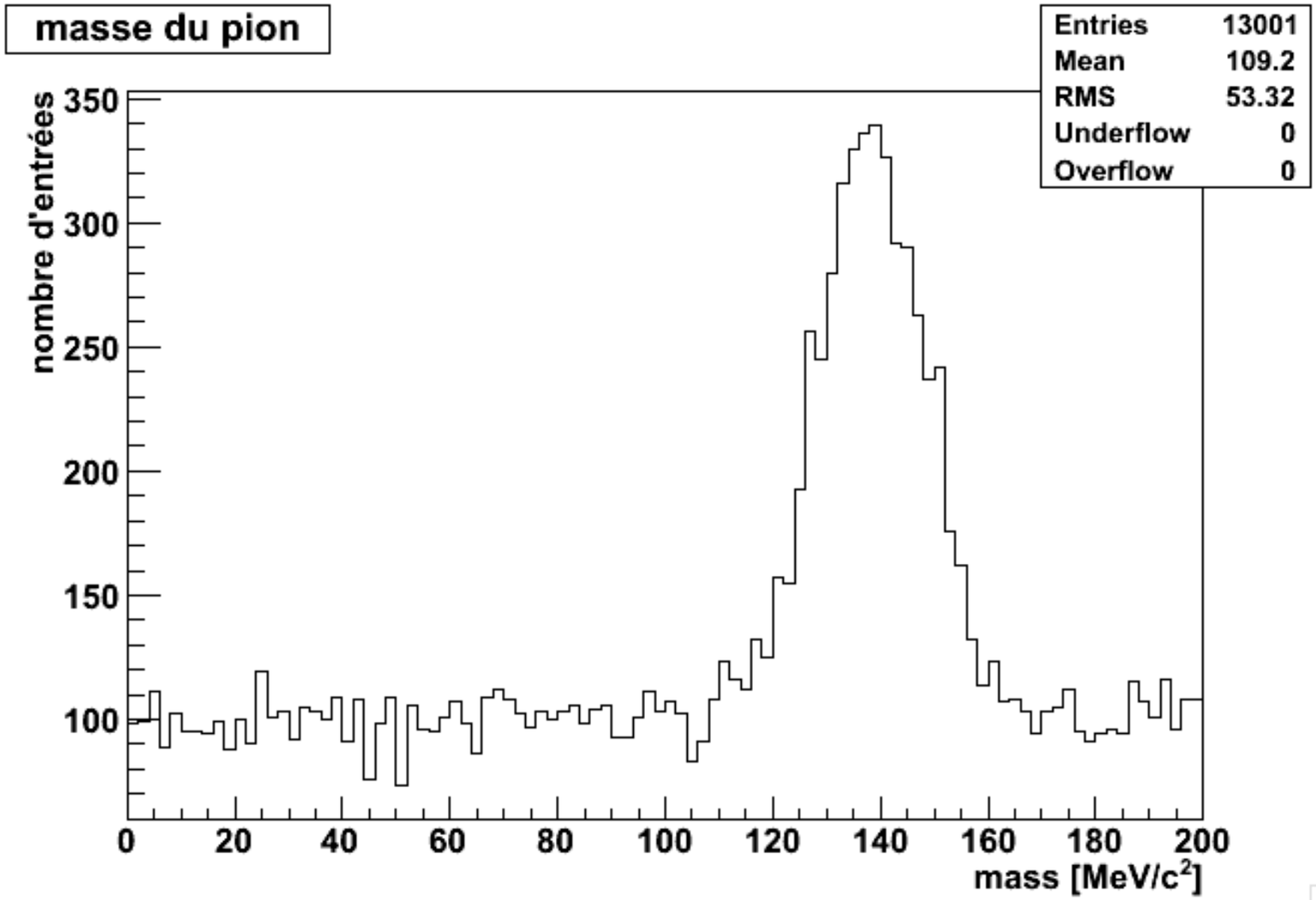
```
gStyle->SetOptStat("remuo");/* par default le nom, le rms, la valueur moyenne et
                               le nombre d'entrée sont affichés
                               ici on veut aussi afficher les "underflow" et
                               "overflow" i.e le nombre eventuels d'entrées
                               au-dessus ou en dessous de l'interval de l'histo
                             */
gStyle->SetOptFit(111);/* pour afficher des infos du fit dans
                          la boite de stat chi2, valeurs des parametres fité etc..*/
  TF1 *f_gauss= new TF1("f_gauss","gaus",);
h1->Fit("gaus");// je fit mon histo avec une gaussienne (root a des fonctions types predefinies)
```



un autre exemple

on lit le fichier pion.txt qui contient 13'000 mesures de la masse du pion.

(macro5.C)



on veut fitter cet histogramme. A priori il s'agit d'une gaussienne avec un bruit de fond constant.

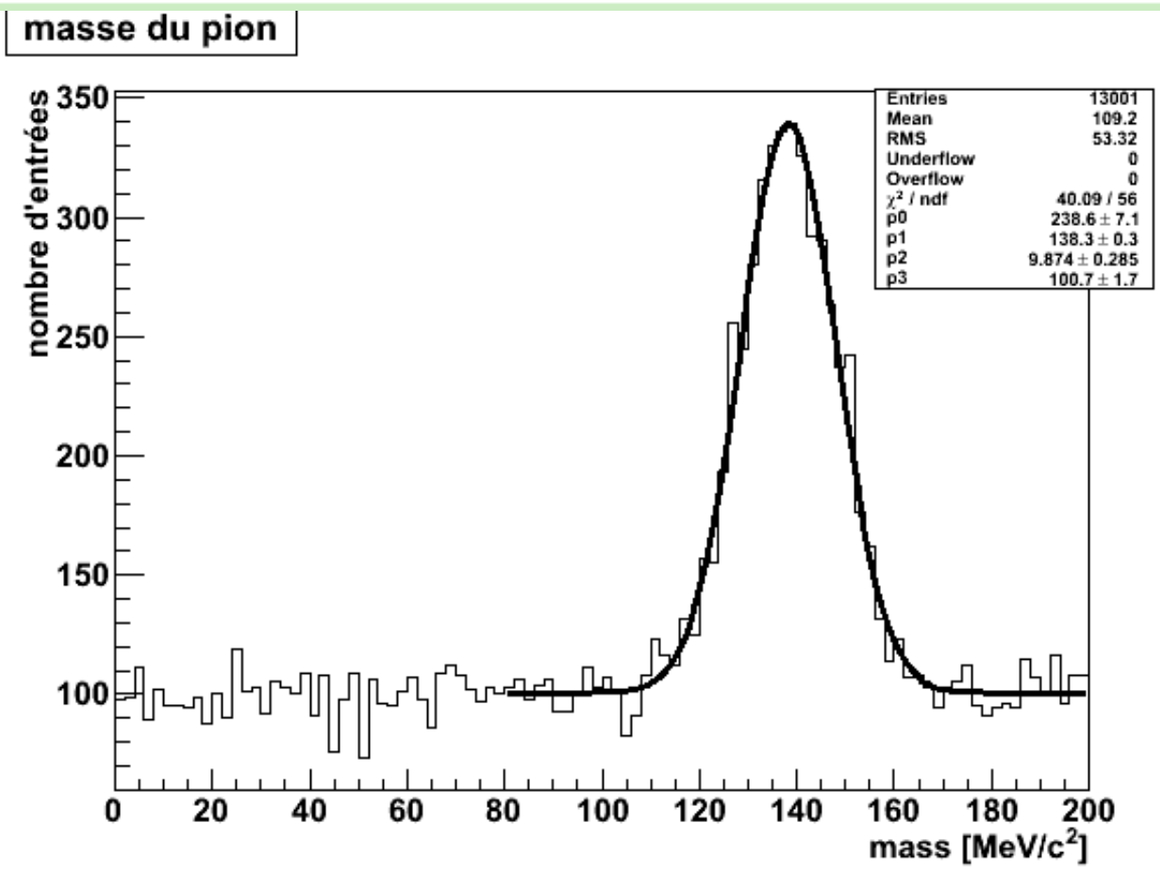
exemple 3

Ici on doit nous même créer la fonction (pas de fonction comme celle ci predefinie dans root). Dans root la classe pour les fonctions s'appelle TF1:

(macro6.C)

```
TF1 *f= new TF1 ("name","formulae",xmin,xmax)
```

```
   TF1 *myfunc= new TF1("myfunc","[0]*exp(-.5*((x-[1])/[2])^2)+[3]",80,200);
/*je definis une fonction (classe TF1) entre 80 et 100, les chiffres entre crochets sont les parametres du fit
  (i.e des valeurs qu'on itinitalize à la main et qui vont varier pendant le fit)
  ce sont des indices d'un tableau*/

   myfunc->SetParameters(350,140,10,100);/*j'initialize les parametres du fit:
                                          l'amplitude à 350, la valeur moyenne à 140 MeV/c^2 ( la masse du pion),
                                          le sigma à 10 et le bruit de fond à 100*/
   /* je peux aussi fixer certains parametres par exemple si je veux fixer la masse du pions:
      myfunc->FixParameter(1,140);
    */
   /* je peux aussi fixer des limites sur les parametres par ex:
      myfunc->FixParameter(2,8,12);// j'autorize le sigma à varier entre 8 et 12 pdt le fit
    */
```



amelioration

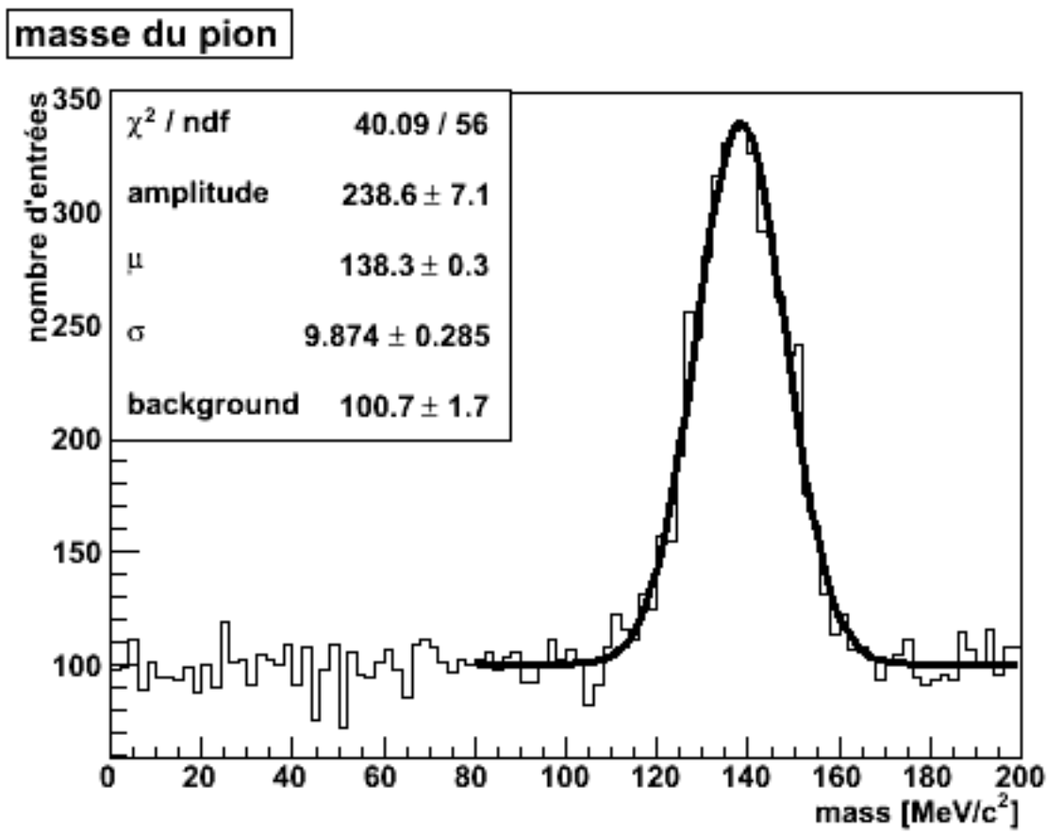
```
TCanvas *c2= new TCanvas();
gStyle->SetOptStat(0);
myfunc->SetParName(0,"amplitude");
myfunc->SetParName(1,"#mu");
myfunc->SetParName(2,"#sigma");
myfunc->SetParName(3,"background");
c2->Divide(2,2);
c2->cd(1);
h1->Draw();
h1->Fit("myfunc","R");// je fit mon histo avec ma fonction qui est une gausienne + un bruit de fond constant
c2->cd(2);
gPad->SetLogy();
myfunc->SetLineColor(3);

h1->Draw("e");
myfunc->Draw("same");
c2->SaveAs("myfit2.eps");// je sauve l'image en eps
```

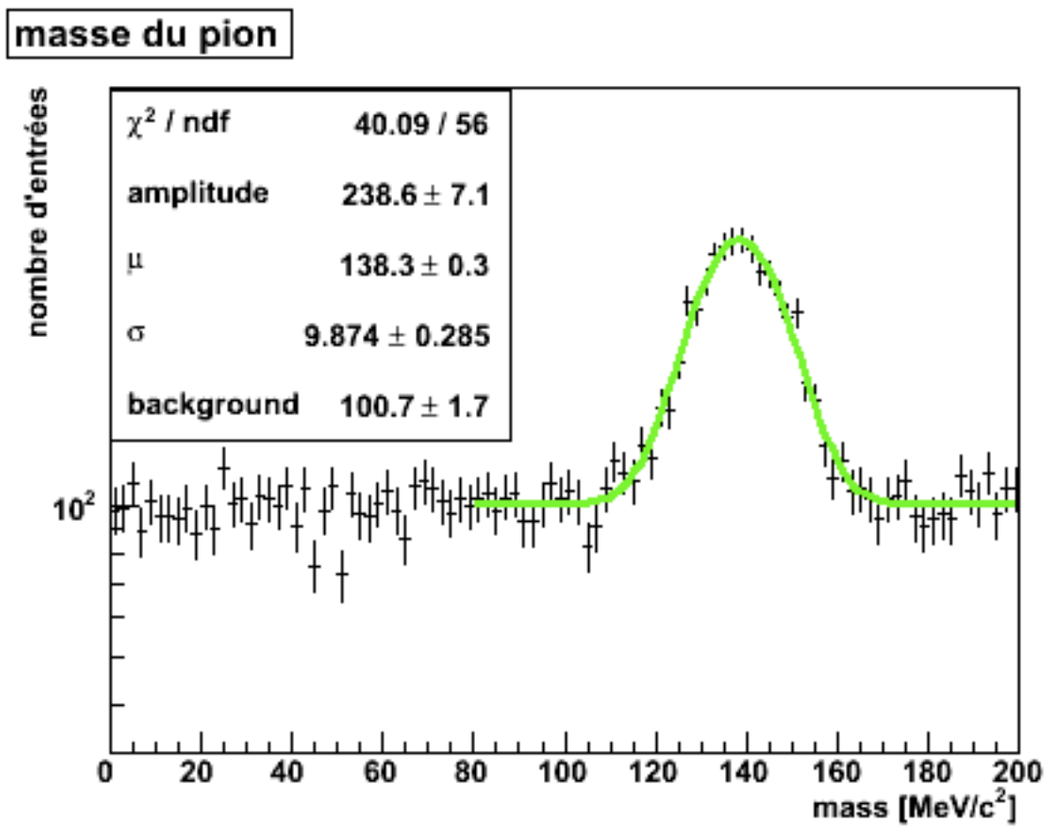
j'utilise la classe TCanvas pour afficher un fenêtre graphique

pour afficher en caractères grecs mettre un # devant

je divise ma fenêtre en 2

(macro7.C)





Liens utiles

- **homepage**: root.cern.ch.
- **user guide** (guide en pdf par themes avec des exemples etc..) : http://root.cern.ch/drupal/content/users-guide
- **reference guide** (A à Z de toutes les classes et de leur membres): http://root.cern.ch/root/html528/ClassIndex.html
- **roottalk** (forum/questions-résponses, souvent trés utile): http://root.cern.ch/phpBB3/

- Les classes les plus utilisées:
- **TStyle** (classe qui gouverne l'environement graphique, le display etc..) : http://root.cern.ch/root/html/TStyle.html
  - des qu'on demarre root un pointeur vers la classe TStyle est créé il s'appelle *gStyle*.
- **TH1** (histogramme): http://root.cern.ch/root/html/TH1.html
- **TF1** (fonction): http://root.cern.ch/root/html/TF1.html
- **TCanvas** (creer une fenêtre graphique):http://root.cern.ch/root/html/TCanvas.html
- **TLegend** (pour creer des legendes): http://root.cern.ch/root/html/TLegend.html