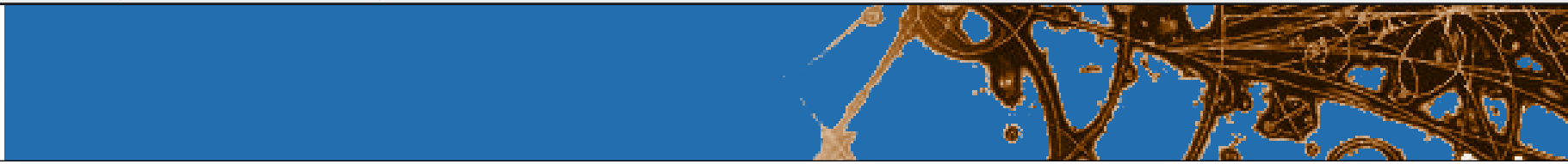


Introduzione ai tool di analisi

I. ROOT & RooFit

Simone Stracka[1,2]

[1]INFN Milano

[2]Dipartimento di Fisica
Università degli Studi di Milano

Metodologie di Analisi dei Dati
20 Novembre 2009

Outline

Outline

Perchè ROOT? Perchè RooFit?

Cosa sono:

- ROOT e RooFit sono tool per l'analisi statistica dei dati e la loro visualizzazione.
- Sono stati sviluppati in C++ all'interno di grandi collaborazioni di Fisica delle Alte Energie.
- Sono utilizzati prevalentemente in questo ambito.

Perchè li usiamo:

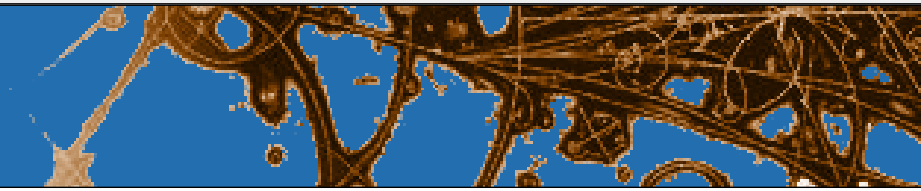
- C++!
- É software libero, distribuito sotto GPL, che potete installare a casa senza dover acquistare costose licenze.
- Avete già avuto a che fare con ROOT nelle esercitazioni di Laboratorio di Calcolo II (Andreazza, Carminati, Galli).
- Li so usare! (Se potete accettare il principio antropico potete accettare anche questa ...)

ROOT e RooFit non vi soddisfano?

R Project:

- http://www.r-project.org/
- Interprete da linea di comando diffuso tra statistici universitari.
- Open source.
- Provatelo, ma a casa!

Generalità



Scopo di questo tutorial è consentirvi di affrontare in serenità le esercitazioni con Pietro.

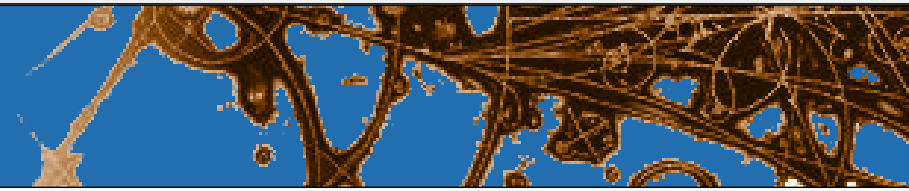
Dopo queste lezioni sarete in grado di:

- Manipolare alla bell'e meglio un dataset
- Definire una PDF parametrica o non parametrica
- Determinare i parametri di una PDF mediante un fit χ^2 o ML
- Generare campioni Monte Carlo secondo una data PDF

Prerequisiti

- Rudimenti di C/C++ (I/O, utilizzo di classi in un programma)
- Infarinatura di ROOT (macro, istogrammi, grafici)

Documentazione



Link utili

- Alfio's ROOT Lectures
  http://www.mi.infn.it/~lazzaro/ROOT/index.html
- ROOT/RooFit User's Guide
  http://root.cern.ch/drupal/content/users-guide
- ROOT/RooFit Reference Guide
  http://root.cern.ch/drupal/content/reference-guide
- RooFit Tutorial
  http://roofit.sourceforge.net/docs/tutorial/index.html
- ROOT HowTo's
  http://root.cern.ch/root/HowTo.html
- Root Talk Forum Index -> ROOT Support
  http://root.cern.ch/phpBB2/

Installazione di ROOT

Versione 5.22

Esercizio a casa: installare ROOT 5.22 sul vostro PC o laptop e verificare il funzionamento di RooFit e TMVA.

Prima di iniziare: Ricordatevi di salvare tutto il codice che scriverete (esercizi diversi = macro diverse) e tutti i grafici plottati. Zippateli e inviateceli.

Configurazione dell'account in laboratorio

Versione 5.xx

- Prendete visione di quanto riportato qui.
- Modificate il file `~/.zshrc` aggiungendo le seguenti righe:
  `export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH`
- Create nella vostra home un file `~/rootlogon.C` (attenzione: i nomi sono case-sensitive) così composto:

```
{
  gSystem->Load("libRooFit.so");
  // ...  altre istruzioni eseguite all'avvio di ROOT
}
```

- Create nella vostra home un file `~/.rootrc` contenente la linea
  `Rint.Logon:  ~/rootlogon.C`

Utilizzare RooFit in un programma esterno

Dovete innanzitutto modificare il nome della funzione principale in `main()` e inserire nel codice tutti gli `#include` statements relativi alle classi usate nel programma. Per compilare sarà quindi sufficiente includere la libreria di RooFit:

Example

```
g++ -O2 -Wall -Wno-deprecated `root-config --cflags` \
-L$ROOTSYS/lib `root-config --libs` -lRooFit macro.cc \
-o macro.x
```

Per includere la libreria di TMVA il comando è analogo:

Example

```
g++ -O2 -Wall -Wno-deprecated `root-config --cflags` \
-L$ROOTSYS/lib `root-config --libs` -lTMVA -lMLP \
-lTreePlayer -lMinuit macro.cc -o macro.x
```

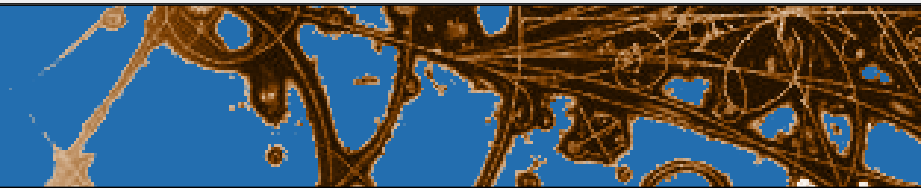
Una macro semplice semplice ...

Prototipo

```
#include <iostream>
#include "RooRealVar.h"

using namespace std;

int macro();

int main(){
  return macro();
}

int macro(){
  cout << "La prima macro!" << endl;
  return 0;
}
```

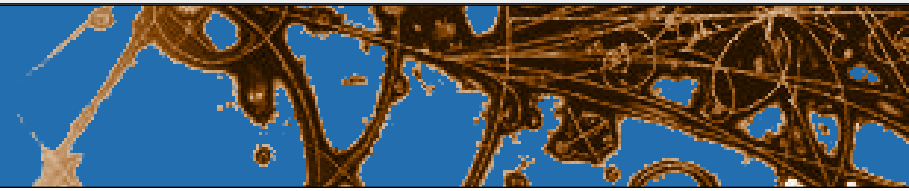
Esercizio: Compilate questa macro, includendo le librerie di RooFit.

Outline



Aprire un TTree

Cos'e'? Come si dichiara?



- È un oggetto di ROOT che contiene i dati per un dato set di variabili
- Noi lo utilizzeremo in un solo modo, associando a ogni ramo (TBranch) una e una sola variabile.
- Crea un nuovo tree:

```
TTree* t = new TTree("t","nuovo tree");
```

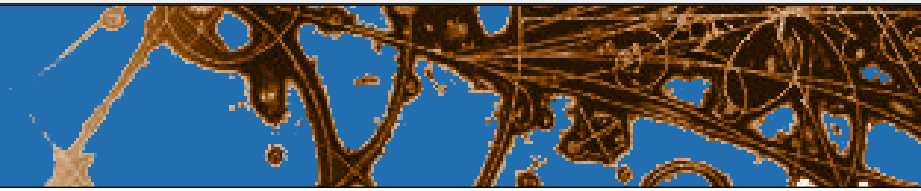
- Aggiungi una variabile di tipo `double`:

```
double var1;
TBranch* bvar1 = t->Branch("var1",&var1,"var1/D");
```

- Aggiunge un'entrata al TTree, con var1=3:

```
var1=3.;
t->Fill();
```

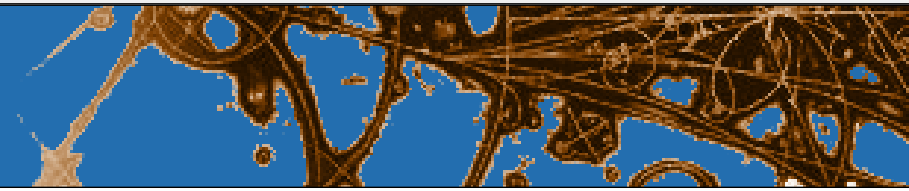
Aprire un TTree



Example

```
TTree* t = new TTree("t","nuovo tree");

double var1, var2;
t->Branch("var1",&var1,"var1/D");
t->Branch("var2",&var2,"var2/D");

var1 = 5.0;
var2 = 1.2;
t->Fill();

var1 = 4.5;
var2 = 1.8;
t->Fill();
```

Aprire un TTree



Example

```
root [0] .ls
 OBJ: TTree   t        nuovo tree : 0 at: 0x86e8e08

root [1] t.Scan()
************************************
*    Row   *      var1 *      var2 *
************************************
*        0 *         5 *       1.2 *
*        1 *       4.5 *       1.8 *
************************************
(Long64_t)2
```

Inciso

Come sapere come utilizzare una data classe nel programma

- ROOT in modalità interattiva consente di conoscere gli argomenti richiesti da un metodo di una data classe (compresa la dichiarazione):

```
root [0] TFile* f = new TFile([Tab] // [Tab] = premere Tab
root [0] TFile f([Tab]
TFile TFile()
TFile TFile(const char* fname, Option_t* option = "",
const char* ftitle = "", Int_t compress = 1)
```

- Potete anche ottenere una lista di metodi associati a un oggetto precedentemente dichiarato:

```
root [0] f->[Tab]
```

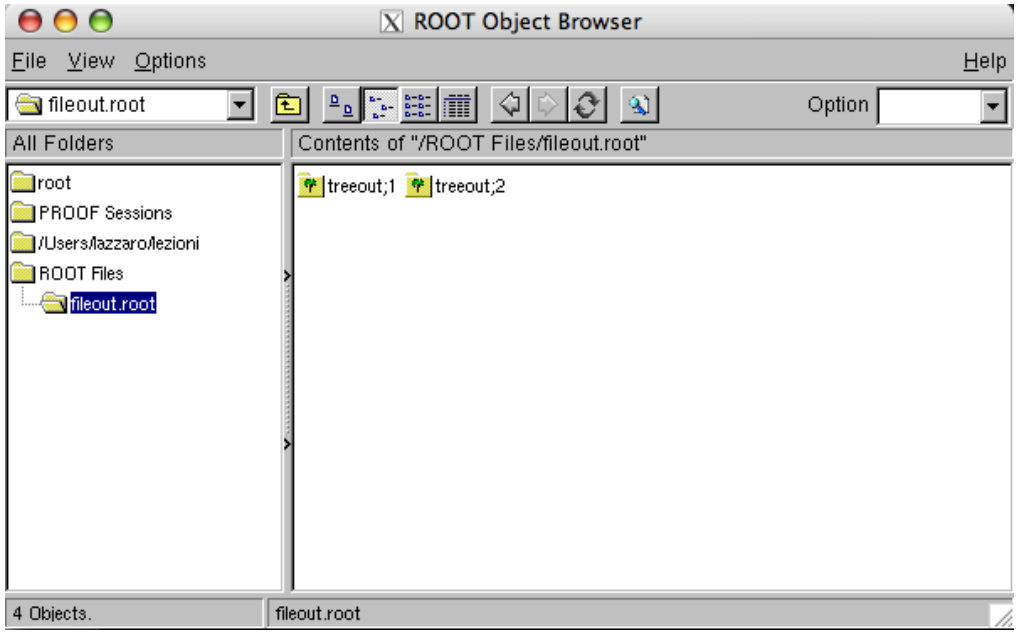
Esercizio: Create un nuovo file `.root` sovrascrivendo eventuali file esistenti con lo stesso nome (`option = "RECREATE"`)
Esercizio: Aprite un file `.root` in modalità di lettura (`option = "READ"`)

Apertura di un file `.root`

nome_del_file.root

- Un file `.root` esistente può essere aperto in modalità interattiva all'apertura di ROOT
- Questo è particolarmente utile per visualizzare il contenuto del file con il comando `.ls` o aprendo un TBrowser e navigando

```
> root -l nome_del_file.root
[0] .ls
[1] new TBrowser()
```

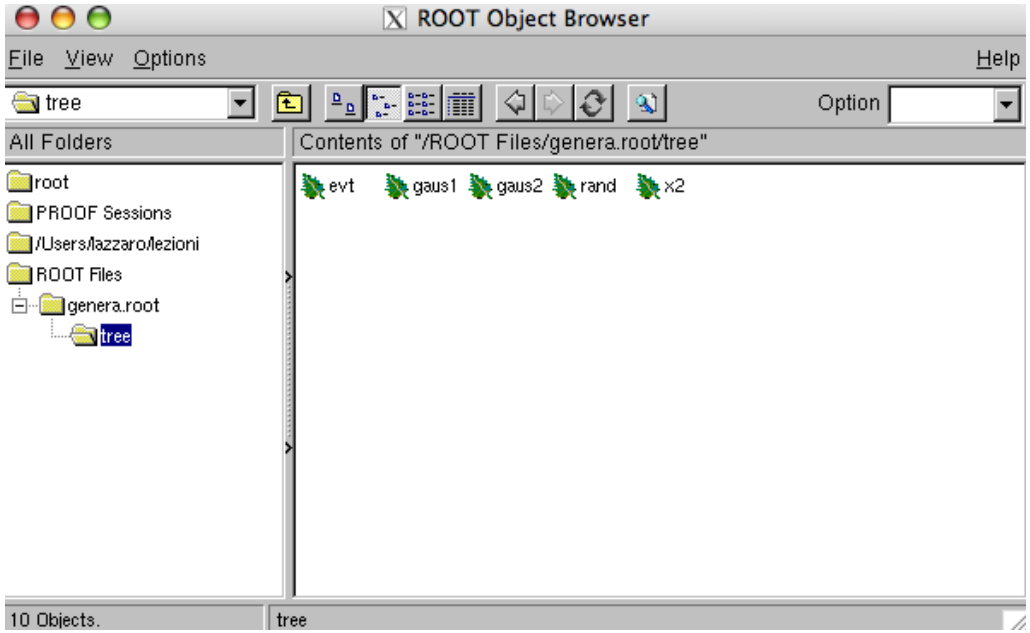


Apertura di un file .root

nome_del_file.root

- Il contenuto di un albero contenuto nel file può essere verificato con:

```
> root -l nome_del_file.root
[0] .ls
[1] nome_albero->Scan()
[2] nome_albero->Show()
[3] nome_albero->StartViewer()
```



Scrittura di un TTree in un file

Esercizio: Aprite in scrittura un file (con `RECREATE`), create un TTree con due variabili (var1 e var2), riempitelo con 2 entry, e scrivetelo nel file con:

```
f->cd(); // si sposta all'interno del file
t->Write(); // scrive il TTree nel file
f->Close(); // chiude il file
```

Lettura di un TTree da un file

```
TFile* f = new TFile("myfile.root");
TTree* t = (TTree*)f->Get("mytree");
gROOT->cd();

if (!t) return 1; // verifica che il TTree esista

// associa i TBranch alle variabili del programma
double varA, varB;
t->SetBranchAddress("var1",&varA);
t->SetBranchAddress("var2",&varB);

// legge le entry
for (int i=0; i<t->GetEntries(); i++){
  t->GetEntry(i);
  // ...  altre operazioni con la entry ...
}
```

Creazione di un TH1 a partire da un TTree

Può essere preferibile rappresentare i dati sotto forma di istogramma. Per questo si utilizza un oggetto TH1.

```
TH1D* h = new TH1D("h","istogramma",20,0.,5.)
```

Un modo per riempire l'istogramma a partire da un TTree consiste nell'inserire nel ciclo di lettura della entry l'istruzione

```
for(int i=0; i<t->GetEntries(); i++){
  h->GetEntry(i); // recupero la entry i
  h->Fill(var1); // riempio l'istogramma
}
```

Dichiarare oggetti matematici in RooFit

In RooFit a ogni oggetto matematico è associata una classe:

- Variabili (osservabili e parametri)

```
RooRealVar t("t","t",1.5,0.,2.,"ps");
RooRealVar x("x","x",1.); // questa è considerata costante
RooRealVar y("y","y",0.,2.);
cout << y.getVal() << endl;
```

- Liste ordinate

```
RooArgList list(x,y,"list");
// per accedere agli elementi uso:
for (Int_t i=0; i<list.getSize(); i++){
  par=(RooRealVar*)list.at(i);
  par->Print();
}
```

Dichiarare oggetti matematici in RooFit

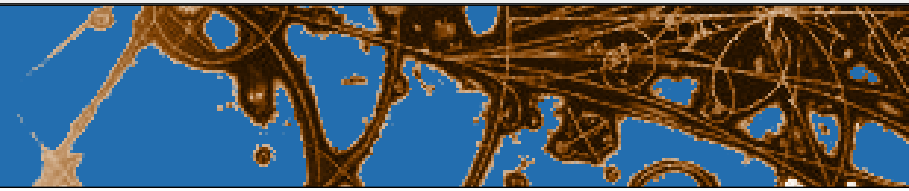
- Liste non ordinate

```
RooArgSet coll(x,y,"coll");
// per accedere agli elementi uso:
TIterator* iter = coll.createIterator();
RooRealVar* par;
while (par=(RooRealVar*)iter->Next()){
  par->Print();
}
```

- Formule

```
RooFormulaVar l("l","l","@0+@1",RooArgList(x,y));
cout << l.getVal() << endl;
```

Dataset in RooFit



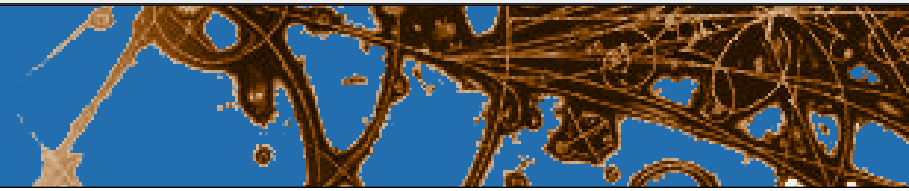
Un dataset in RooFit può essere dichiarato a partire da un TTree (RooDataSet) o da un istogramma TH1 (RooDataHist):

Example

```
// dichiara le variabili (con stesso nome del TTree)
RooRealVar var1("var1","var1",1.5,0.,2.,"ps");
```

1. `RooDataSet data("data","dataset",t,RooArgSet(var1));`
   `// dichiarato a partire da un TTree`
2. `RooDataHist dh("dh","hist",RooArgSet(var1),data);`
   `// dichiarato a partire da un RooDataSet`
3. `RooDataHist dh("dh","hist",RooArgList(var1),h);`
   `// dichiarato a partire da un TH1`

Dataset in RooFit

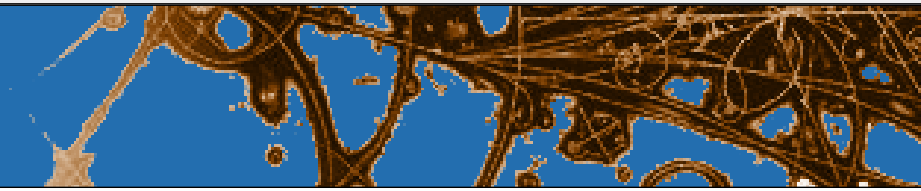


Il principale oggetto grafico in RooFit è il RooPlot, che costituisce l'interfaccia per la visualizzazione degli istogrammi associati a una variabile (frame) nonchè il contenitore di un set arbitrario di oggetti (istogrammi, curve, ...). In questo esempio si mostra come visualizzare un dataset:

Example

```
using namespace RooFit;
...
TCanvas* canvas = new TCanvas("canvas","new canvas");
...
RooPlot* var1frame = var1.frame();
data.plotOn(var1frame,Name("myHist1"));
canvas->cd();
var1frame->Draw();
```

Outline



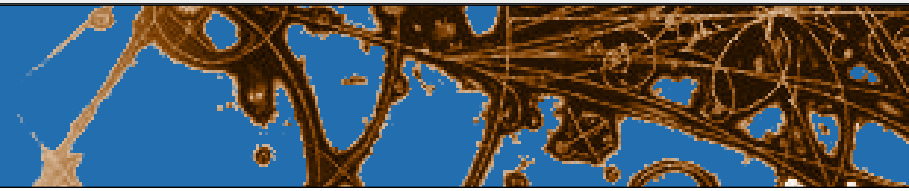
Dichiarare una PDF

Il codice che segue dichiara e visualizza una RooAbsPdf, cioè un oggetto che implementa le funzionalità di una PDF (es: normalizzazione):

```
// Build Gaussian PDF
RooRealVar x("x","x",-10,10);
RooRealVar mean("mean","mean of gaussian",0,-10,10);
RooRealVar sigma("sigma","width of gaussian",3);
RooGaussian gauss("gauss","gaussian PDF",x,mean,sigma);

// Plot PDF
RooPlot* xframe = x.frame();
gauss.plotOn(xframe,Name("myCurve1"));
xframe->Draw();
```

- La classe RooRealVar viene utilizzata per osservabili e parametri.
- La distinzione avviene in associazione a un dataset. Le variabili che compaiono nel dataset sono osservabili, le altre parametri.

Altre PDF utili



1. Gaussiana

```
RooGaussian gauss("gauss","",x,mean,sigma);
```

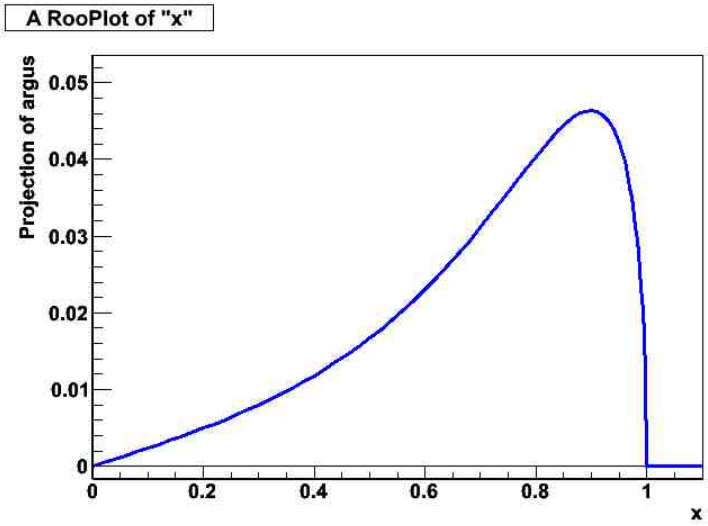
2. Polinomio Chebychev (la lunghezza della lista dei coefficienti determina il grado del polinomio)

```
RooChebychev cheby("cheby","",x,RooArgList(c1,c2,c3));
```

3. Funzione a soglia Argus ($c < 0$)

```
RooArgusBG argus("argus","",x,x0,c);
```

Dichiarare una PDF



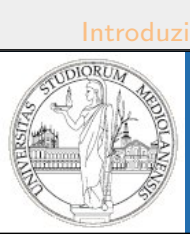
Esercizio: Dichiarate e disegnate una funzione Argus

Determinare i parametri di una PDF dai dati

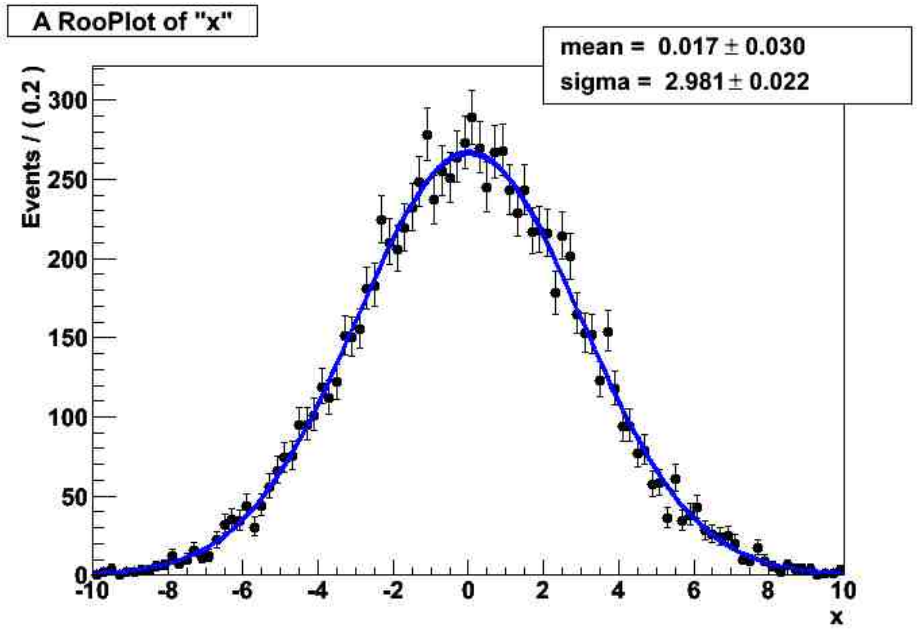
fitgauss.root

Un problema comune consiste nel determinare i parametri di una distribuzione che meglio riproducono un dato campione di dati (che possono essere anche dati Monte Carlo). In RooFit questo viene fatto richiamando il metodo `fitTo` di una PDF.

```
// fitta la PDF ai dati
gauss.fitTo(data);

// plotta la pdf sovrapposta ai dati
data.plotOn(xframe);
gauss.plotOn(xframe); // normalizzata al numero di eventi
gauss.paramOn(xframe); // mostra i valori fittati
xframe.Draw();
```



Determinare i parametri di una PDF dai dati



Esercizio: Utilizzate quanto imparato finora per aprire il file, leggere il TTree, caricarlo in un dataset, dichiarare la gaussiana, fittarla e mostrarla.
Segnalazione: Il campione utilizzato per questo eserczio è stato generato utilizzando il metodo `generate`:

```
RooDataSet* data = gauss.generate(x,10000);
```

Qualità del fit da test del χ^2

Avete avuto cura di definire un nome per l'istogramma ottenuto plottando il dataset e la curva corrispondente al modello fittato?

Example

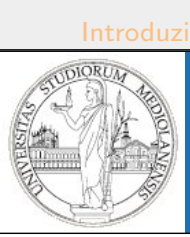
```
data.plotOn(var1frame,Name("myHist1"));
gauss.plotOn(xframe,Name("myCurve1"));
```

Se sì, questo codice sarà sufficiente per ricavare il χ^2 associato:

Example

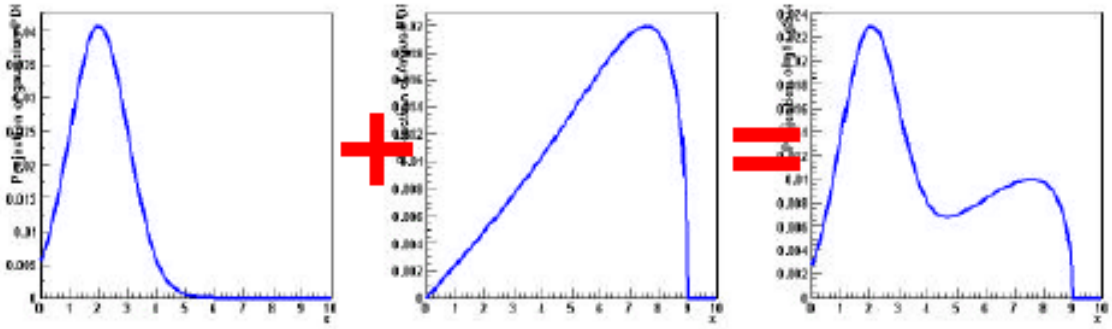
```
Double_t chi2_1 = xframe->chiSquare("myCurve1","myHist1");
```

N.B.: il fit non è istogrammato, ma il test sì.

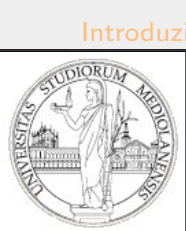


Composizione di PDF

Somma

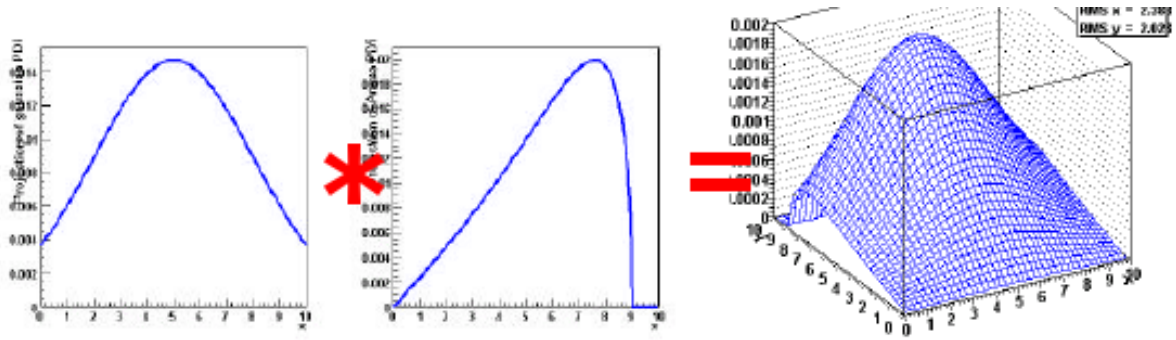


```
RooRealVar x("x","x",1.5,-1.,1.);

RooRealVar gxmean("gxmean","gxmean",0.,-0.5,0.5);
RooRealVar gxsigma("gxsigma","gxsigma",0.1,0.001,0.2);
RooGaussian gauss("gauss","gaussian",x,sgxmean,sgxsigma);

RooPolynomial flat("flat","flat",x,RooArgList(),0);

RooRealVar f("f","frac",0.5,0.,1.);
RooAddPdf s("s","",RooArgList(gauss,flat),RooArgList(f));
```



Composizione di PDF

Prodotto



```
RooRealVar x("x","x",1.5,-1.,1.);
RooRealVar y("y","y",0.,-1.,1.);

// ...

RooGaussian gaussx("gaussx","gaussian",x,meanx,sigmax);
RooGaussian gaussy("gaussy","gaussian",y,meany,sigmay);

RooProdPdf p("p","",RooArgSet(gaussx,gaussy));
```

Composizione di PDF

Esercizio 1: Generate un campione bidimensionale di 10000 eventi distribuito gaussianamente in entrambe le variabili, e plottatelo.
Esercizio 2: Generate un campione monodimensionale di 10000 eventi con distribuzione data dalla somma di due distribuzioni a piacere.
Suggerimento: per salvare il RooDataSet in un file usare

```
data->tree()->SetName("t1");
data->tree()->Write();
```

Outline

Conclusione

I commenti di chi l'ha provato

"Quando ho davanti un vero campione , anch'io uso RooFit!"