

**GENERAZIONE ed
INTERPOLAZIONE *Unbinned Maximim Likelihood*
con *ROOT***

LEZIONE PRATICA per il Corso di Lab. Analisi Dati

Docente: A.Pompili – 21 dicembre 2015

Questa macro di RooFit esegue il fit UML sulla stessa distribuzione preliminarmente generata [genera anche un file di testo nella sottodir `txt_files` che puo' essere dato "in pasto" ad una macro esterna, p.e.s di GooFit].

Questa macro puo' essere eseguita sul server del corso, cioe' sulla macchina virtuale 212.189.205.223 !

E' richiesta l'accortezza di modificare il file di login da eseguire all'inizio in modo tale da contenere:

```
export ROOTSYS=/afs/cern.ch/sw/lcg/app/releases/ROOT/5.34.26/x86_64-slc5-gcc47-opt/root/  
export PATH=$PATH:$ROOTSYS/bin:.  
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib:.  
source /afs/cern.ch/sw/lcg/app/releases/ROOT/5.34.26/x86_64-slc5-gcc47-opt/root/bin/thisroot.sh
```

Qualora cio' non fosse sufficiente, ho predisposto un file da eseguire :
rootsource.sh

Dopodiche' si lancia ROOT e si esegue la macro di RooFit:

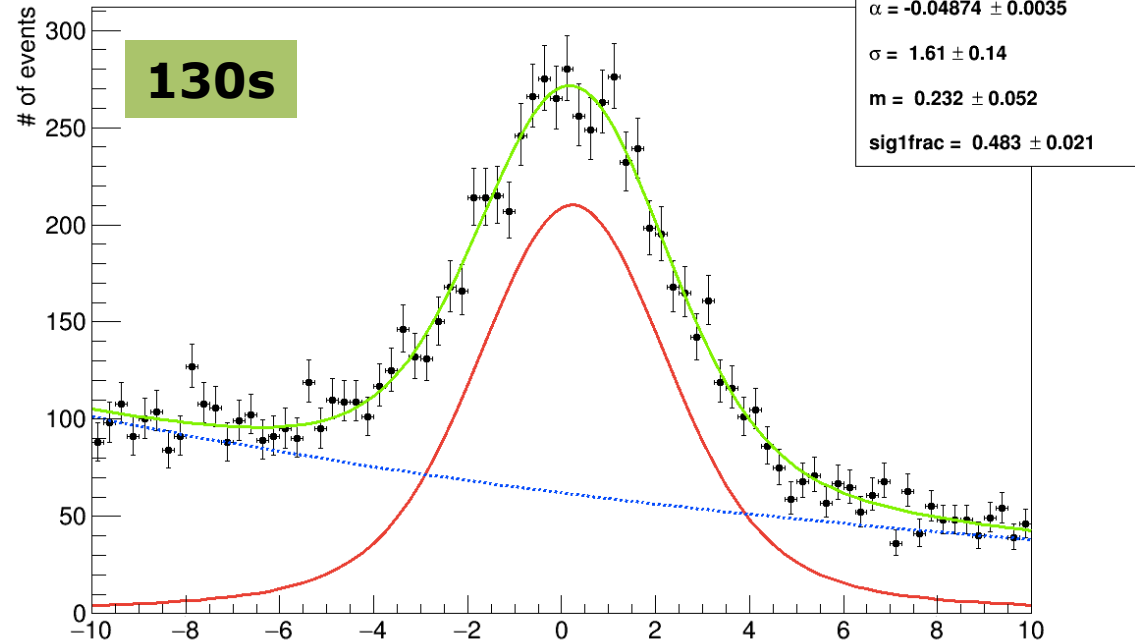
```
Root [0] .L RooConvolutionExp.C+  
Root [1] RooConvolutionExp("#events",#bins)
```

```
P.es. #events=10000 e #bins=80 (ci impiega 130s)  
oppure #events=100000 e #bins=120 (ci impiega 1250s )
```

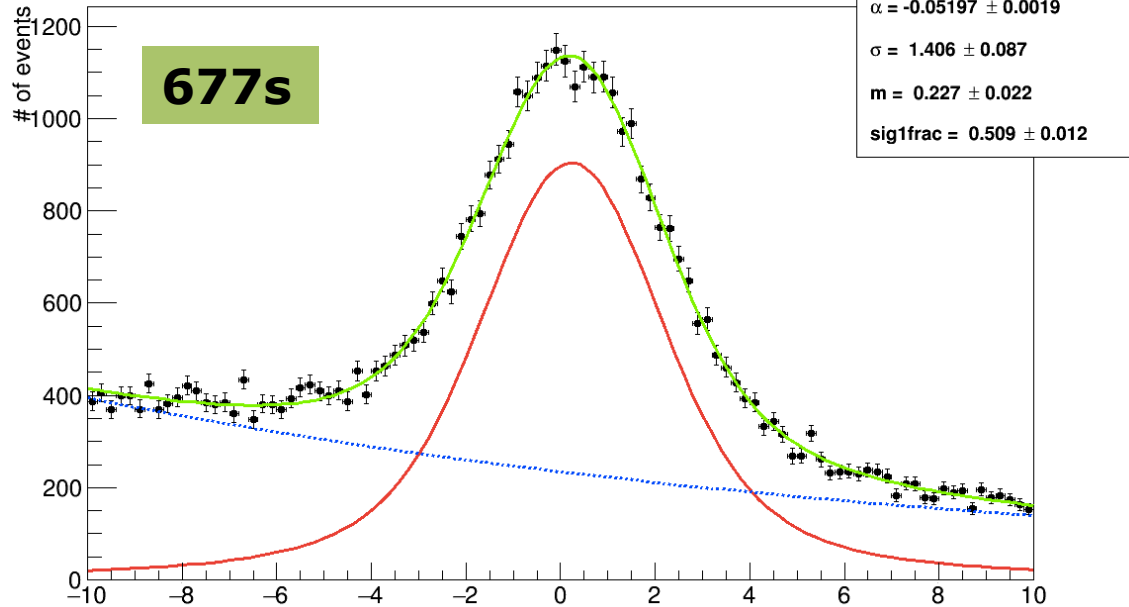
N.B.: Il #bins viene fissato a soli fini di rappresentazione (il fit e' UML !)

**Si ottiene p.es.
nei 2 casi seguenti:**

RooFit : 10000 events



RooFit : 50000 events



E' buona pratica confrontare il risultato dell'interpolazione con i valori dei parametri che sono stati messi in generazione. Si puo' verificare come l'accordo aumenti all'aumentare del # di eventi generati!

**Una tabellina dei tempi per valutare le prestazioni;
il tempo impiegato si riferisce al solo fit (ma in ogni caso
il tempo di generazione e' trascurabile rispetto a quello di fit):**

#eventi	RooFit
10K	130s
100K	677s
1M	1250s

Ispezioniamo il codice della macro di RooFit (**RooConvolutionExp.C**)

```
#include "RooPolynomial.h"
#include "RooRealVar.h"
#include "RooBreitWigner.h"
#include "RooNumConvPdf.h"
#include "RooGaussian.h"
#include "RooExponential.h"
#include "RooDataSet.h"
#include "RooDataHist.h"
#include "RooAbsData.h"
#include "RooMinuit.h"
#include "RooPlot.h"
#include "RooChebychev.h"
#include "RooAddPdf.h"
#include "RooArgList.h"
#include "TH1F.h"
#include <vector>
#include "TCanvas.h"

#include <sys/time.h>
#include <sys/times.h>

using namespace RooFit; //Working in RooFit//

timeval startTime, stopTime, totalTime;
timeval startTimeRead, stopTimeRead, totalTimeRead;
clock_t startCPU, stopCPU;
clock_t startCPURead, stopCPURead;
tms startProc, stopProc; //Struct time intervals in clock ticks//
tms startProcRead, stopProcRead;

void RooConvolutionExp(TString argv, int bins=200) {
    int events = atoi(argv.Data()); //converte stringa "numero" in numero --
    TString name = "";
    switch (events)
    {
        case 100: name = "100";
            break;
        case 1000: name = "1k";
            break;
        case 10000: name = "10k";
            break;
        case 100000: name = "100k";
            break;
        case 500000: name = "500k";
            break;
        case 1000000: name = "1M";
            break;
        case 5000000: name = "5M";
            break;
        case 10000000: name = "10M";
            break;
        case 50000000: name = "50M";
            break;
        case 100000000: name = "100M";
            break;
        //
        default: name = argv;
            break;
    }

    char bufferstring[256];
```

```

char bufferstring[256];

RooRealVar xvar("xvar", "", -10, 10);
xvar.setBins(bins);

// Breit Wigner Signal //
RooRealVar mean("m", "mean", 0.2, -1, 1); //Breit Wigner mean//
RooRealVar gamma("#Gamma", "gamma", 2, 0.1, 5); //Breit Wigner width//
RooBreitWigner signal("BW", "BW signal", xvar, mean, gamma); //Breit Wigner pdf//

// Gaussian Resolution Function //
RooRealVar zero("zero", "Gaussian resolution mean", 0.); // offset from mean
RooRealVar sigma("#sigma", "sigma", 1.5, 0.1, 5); //Gaussian sigma//
RooGaussian resol("resol", "Gaussian resolution", xvar, zero, sigma); //Gaussian pdf//

// Background //
RooRealVar alpha("#alpha", "Exponential Parameter", -0.05, -2.0, 0.0);
RooExponential bkg("Bkg", "Bkg", xvar, alpha);

// Gaussian + BW convolution //
RooNumConvPdf convolution("convolution", "BW (X) gauss", xvar, signal , resol);

// TotalPdf = Gaussian + Bkg //
RooRealVar sigfrac("sigfrac", "fraction of component 1 in signal", 0.5, 0., 1.) ;
RooAddPdf total("totalPDF", "totalPDF", RooArgList(convolution, bkg), sigfrac);

cout << "\nGenerating " << name << " events\n" << endl ;

////////////////////////////////////
// Generating data
////////////////////////////////////

RooDataSet* data = total.generate(xvar, events);
//
sprintf(bufferstring, "./txt_files/%d_events.txt", events);
data->write(bufferstring);

cout << "\nFitting " << name << " events\n" << endl ;

```

**Generazione secondo
il modello (pdf) *total***

**Scrive la massa generata
evento-per-evento nel
file .txt esterno**

```

cout << "\nFitting " << name << " events\n" << endl ;

////////////////////////////////////
// Fitting data
////////////////////////////////////

RooAbsReal* nll = total.createNLL(*data);

//Declare null (pointer) and assign -log(Likelihood) to it, Likelihood -> convolution and *data//
RooMinuit min(*nll);
gettimeofday(&startTime, NULL);
startCPU = times(&startProc);
//Migrad Fit
min.migrad();

stopCPU = times(&stopProc);
gettimeofday(&stopTime, NULL);

////////////////////////////////////
// Fit result and data representation
////////////////////////////////////

TCanvas *foo = new TCanvas("RooCanvas","Roofit Canvas", 1200, 800);
RooPlot *frame = xvar.frame("");
sprintf(bufferstring, " RooFit : %d events", events);
frame->SetTitle(bufferstring);
frame->SetYTitle("# of events");
data->plotOn(frame);
total.plotOn(frame, LineColor(kGreen));
total.plotOn(frame, Components(RooArgSet(convolution)), LineColor(kRed));
total.plotOn(frame, Components(RooArgSet(bkg)), LineColor(kBlue), LineStyle(kDashed));
total.paramOn(frame, Layout(0.75, 0.99, 0.99));
frame->getAttText()->SetTextSize(0.028);

frame->Draw();
foo->SaveAs("plots/RooConvGen_"+name+".eps");
foo->SaveAs("plots/RooConvGen_"+name+".png");

// Print total fitting time
cout << "\n-----" << endl ;
double myCPUc = (stopCPU - startCPU)*10000;
cout << "Total CPU time: " << (myCPUc / CLOCKS_PER_SEC);
cout << "\n-----" << endl ;
cout << endl ;

```

PDF

dati generati
(unbinned!!)

UML FIT

file esterni con il plot

}