# Exercise/Hands-on #3

# Generation & Interpolation with an UML fit with RooFit

## Statistical Data Analysis for HEP

Prof. Alexis Pompili (University of Bari Aldo Moro)*

Erasmus+ Teaching Mobility Program / 16-20 October 2023 @ Sofia Physics Faculty

* alexis.pompili@ba.infn.it (or alexis.pompili@cern.ch )

**Purpose of the exercise: learn to generate a distribution & get familiar with a fitting task**

**A very interesting feature of `RooFit` is the possibility to generate pseudo-experiments (called also MC toys), namely, sampling distributions according to a certain given model (p.d.f.).**

**The macro `RooConvolutionExpNew.C` ...**

- **generates a distribution (it is an unbinned dataset: `RooDataSet` in `RooFit`)**
- **executes an Unbinned Maximum Likelihood (UML) fit of this distribution earlier generated**
- **gives as output both a plot [**written in ./plots/**]**

  **... and a txt file [**written in ./txt_files/**]** (that can be used externally for other purposes)

**Note: the # of bins is settled only for representation purposes (the fit is still *unbinned* !)**

Let’s inspect now the code in **RooConvolutionExpNew.C**

```
#include "RooPolynomial.h"
#include "RooRealVar.h"
#include "RooBreitWigner.h"
#include "RooNumConvPdf.h"
//#include "RooVoigtian.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 "RooRandom.h" // needed for Randomizer
//
using namespace RooFit; //----Working in RooFit//
//
/////////////////////////////////////////////////////////////////////////////////////
//
// root [0] .L RooConvolutionExpNew.C
// root [1]  RooConvolutionExpNew("10000", 1010, 80)
//
//--- providing #events (as a string), seed# and #bins
//
// Try at least 10K, 100K, 500K, 1M (it takes from about 2 t 20 minutes!).
//
// Choose the most relevant parameters and show for each of them how the best estimate
// given by the fit gets closer to the true (generated) value
//
/////////////////////////////////////////////////////////////////////////////////////
```

INSTRUCTIONS HOW-TO_RUN

```
////////////////////////////////////////////////////////////////////////////////
//
void RooConvolutionExpNew(TString argv, int seed=1000, int bins=200) {
  //
  //-- note: external values override those dummy initializing values!
  //         (check that changhing seed you get a different distribution)
  //
  int events = atoi(argv.Data());  // it converts string "number" into an integer
  TString name = argv;
  //
  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 // perform a numerical convolution
  RooNumConvPdf convolution("convolution", "BW (X) gauss", xvar, signal , resol);
  //
  //-- note that alternatvely you can try a Voigtian
  //
  // TotalPdf = Gaussian + Bkg //
  RooRealVar sigfrac("sig1frac","fraction of component 1 in signal",0.5,0.,1.) ;
  RooAddPdf total("totalPDF", "totalPDF", RooArgList(convolution, bkg),sigfrac);
  //
  cout <<"\n------Generating " << name << " events\n" << endl ;
  cout <<"\n------Remember: initial values for fitting step are the generated (true) values in generation ---" << endl;
  //
```

interface

**Preparation step: define model**

**pdf modelling**

- signal as convolution of a Breit-Wigner and a gaussian resolution function

- exponential background

## 1st step: generation

```
cout <<"\n------Generating " << name << " events\n" << endl ;
cout <<"\n------Remember: initial values for fitting step are the generated (true) values in generation ---" << endl;
//
//////////////////////////////////////////////////////////////////////////
// Generating data
//////////////////////////////////////////////////////////////////////////
//
cout << "\n-----Remember to change seed every time to get different distributions------" << endl;
//
RooRandom::randomGenerator()->SetSeed(seed);
//
RooDataSet* data = total.generate(xvar,events);
//
sprintf(bufferstring,"./txt_files/%d_events.txt",events);
data->write(bufferstring);
//
```

In execution, we get **after generation step**:

```
[pompili@vm-pompili Esercitazione-4]$ root -l
root [0] .L RooConvolutionExpNew.C
root [1] RooConvolutionExpNew("10000", 1010, 80)

------Generating 10000 events


------Remember: initial values for fitting step are the generated (true) values in generation ---

-----Remember to change seed every time to get different distributions------
[#1] INFO:NumericIntegration -- RooRealIntegral::init(convolution_Int[xvar]) using numeric integrator RooIntegrator1D to calculate Int(xvar)
[#1] INFO:NumericIntegration -- RooRealIntegral::init(convolution_Int[xvar]) using numeric integrator RooIntegrator1D to calculate Int(xvar)
[#1] INFO:DataHandling -- RooDataSet::write(totalPDFData) writing ASCII file ./txt_files/10000_events.txt
```

```
cout <<"\nFitting " << name << " events\n" << endl ;
//
//////////////////////////////////////////////////////////////////////////
// Fitting data
//////////////////////////////////////////////////////////////////////////
//
RooAbsReal* nll = total.createNLL(*data);  // neg-log-likelihood
RooMinuit min(*nll);
//
min.migrad();  // execute MIGRAD fit
cout << "\n--------------------minimization done; now recalculating the uncertainties----------" << endl;
//
min.hesse();   // calculate the uncertainties in the parabolic approximation
cout << "\n--------------------fit done; check best estimates for the model parameters---------" << endl;
//
////////////////////////////////////////////////////////////////////////
// Fit result and data representation
//////////////////////////////////////////////////////////////////////////
//
TCanvas *myC = new TCanvas("RooCanvas","Roofit Canvas", 1000, 750);
//
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();
myC->SaveAs("plots/RooConvGen_"+name+".png");
//
////////////////////
//
if (myC)
  {
   myC->Close();
   delete myC;
  }
//
}
```

**2nd step: fitting**

**Fitting sequence: MIGRAD + HESSE**

**3rd step: plotting**

In execution, we get **@ fitting step (MIGRAD)**:

```
Fitting 10000 events

[#1] INFO:NumericIntegration -- RooRealIntegral::init(convolution_Int[xvar]) using numeric integrator RooIntegrator1D to calculate Int(xvar)
 **********
 **    13 **MIGRAD        2500           1
 **********
 FIRST CALL TO USER FUNCTION AT NEW START POINT, WITH IFLAG=4.
 START MIGRAD MINIMIZATION.  STRATEGY  1.  CONVERGENCE WHEN EDM .LT. 1.00e-03
 FCN=28493.8 FROM MIGRAD    STATUS=INITIATE       20 CALLS          21 TOTAL
                     EDM= unknown      STRATEGY= 1      NO ERROR MATRIX
  EXT PARAMETER               CURRENT GUESS       STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  #Gamma       2.00000e+00   4.90000e-01   2.06953e-01  -2.77994e+01
   2  #alpha      -5.00000e-02   2.50000e-02   8.28427e-02   3.92141e+01
   3  #sigma       1.50000e+00   4.90000e-01   2.24553e-01  -2.27174e+01
   4  m            2.00000e-01   2.00000e-01   2.05758e-01   3.73329e+01
   5  sig1frac     5.00000e-01   1.00000e-01   2.01358e-01   8.00078e+01
                               ERR DEF= 0.5
 MIGRAD MINIMIZATION HAS CONVERGED.
 MIGRAD WILL VERIFY CONVERGENCE AND ERROR MATRIX.
 COVARIANCE MATRIX CALCULATED SUCCESSFULLY
 FCN=28489 FROM MIGRAD    STATUS=CONVERGED     166 CALLS         167 TOTAL
                     EDM=4.32044e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  #Gamma       3.40452e+00   5.25290e-01   5.98428e-03   2.11982e-02
   2  #alpha      -5.44013e-02   4.53968e-03   1.10219e-03   8.90567e-03
   3  #sigma       9.73194e-01   2.41004e-01   4.83234e-03   3.41460e-02
   4  m            1.08019e-01   5.01055e-02   5.56296e-03   2.28328e-02
   5  sig1frac     5.42053e-01   2.62713e-02   2.33086e-03  -3.69678e-02
                               ERR DEF= 0.5
 EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  5    ERR DEF=0.5
  2.809e-01 -1.514e-03 -1.189e-01 -1.053e-03  1.234e-02
 -1.514e-03  2.061e-05  5.744e-04 -4.527e-05 -8.325e-05
 -1.189e-01  5.744e-04  5.841e-02  3.025e-04 -4.698e-03
 -1.053e-03 -4.527e-05  3.025e-04  2.513e-03 -5.369e-05
  1.234e-02 -8.325e-05 -4.698e-03 -5.369e-05  6.908e-04
 PARAMETER  CORRELATION COEFFICIENTS
       NO.  GLOBAL      1      2      3      4      5
        1  0.97483   1.000 -0.629 -0.929 -0.040  0.886
        2  0.73477  -0.629  1.000  0.524 -0.199 -0.698
        3  0.94605  -0.929  0.524  1.000  0.025 -0.740
        4  0.32276  -0.040 -0.199  0.025  1.000 -0.041
        5  0.92525   0.886 -0.698 -0.740 -0.041  1.000
```

In execution, we get **@ fitting step (HESSE)**:

```
--------------------minimization done; now recalculating the uncertainties----------
**********
**   18 **HESSE        2500
**********
COVARIANCE MATRIX CALCULATED SUCCESSFULLY
FCN=28489 FROM HESSE      STATUS=OK            33 CALLS         200 TOTAL
                    EDM=4.58768e-06    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                              INTERNAL      INTERNAL
 NO.   NAME        VALUE          ERROR       STEP SIZE       VALUE
  1  #Gamma       3.40452e+00   6.09271e-01   2.39371e-04   3.56273e-01
  2  #alpha      -5.44013e-02   4.87576e-03   4.40875e-05   1.23943e+00
  3  #sigma       9.73194e-01   2.77711e-01   9.66467e-04  -6.99185e-01
  4  m            1.08019e-01   5.01294e-02   1.11259e-03   1.08230e-01
  5  sig1frac     5.42053e-01   2.97534e-02   4.66172e-04   8.42046e-02
                               ERR DEF= 0.5
EXTERNAL ERROR MATRIX.    NDIM=  25    NPAR=  5    ERR DEF=0.5
 3.803e-01 -2.072e-03 -1.628e-01 -1.392e-03  1.675e-02
-2.072e-03  2.377e-05  8.199e-04 -4.362e-05 -1.081e-04
-1.628e-01  8.199e-04  7.769e-02  4.539e-04 -6.643e-03
-1.392e-03 -4.362e-05  4.539e-04  2.515e-03 -6.751e-05
 1.675e-02 -1.081e-04 -6.643e-03 -6.751e-05  8.863e-04
PARAMETER  CORRELATION COEFFICIENTS
      NO.  GLOBAL      1      2      3      4      5
       1  0.98147   1.000 -0.689 -0.947 -0.045  0.912
       2  0.77533  -0.689  1.000  0.603 -0.178 -0.745
       3  0.95974  -0.947  0.603  1.000  0.032 -0.800
       4  0.32402  -0.045 -0.178  0.032  1.000 -0.045
       5  0.94225   0.912 -0.745 -0.800 -0.045  1.000
```

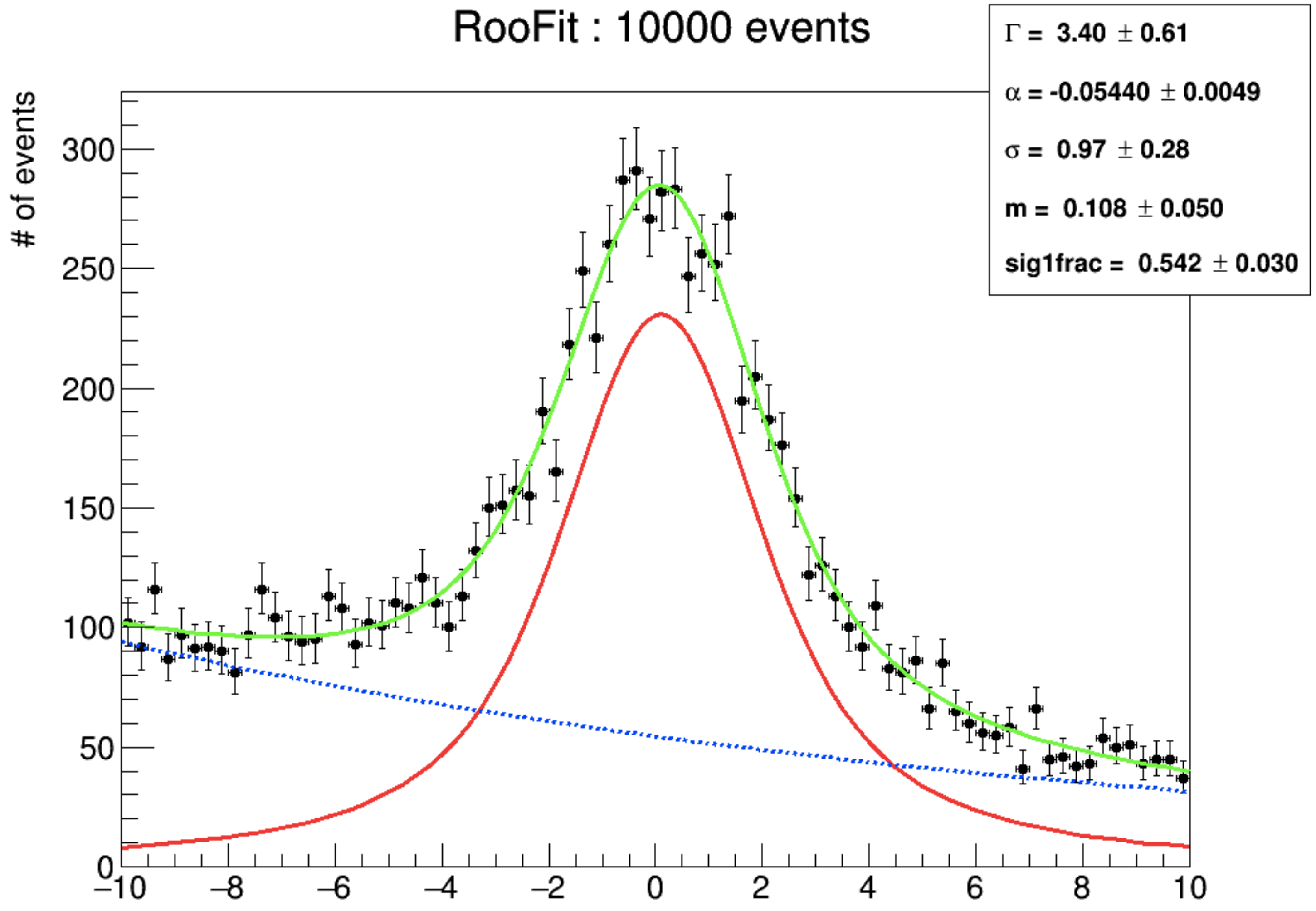
Same central values:
only uncertaities are **recalculated**!

In execution, we get **@ plotting step** :

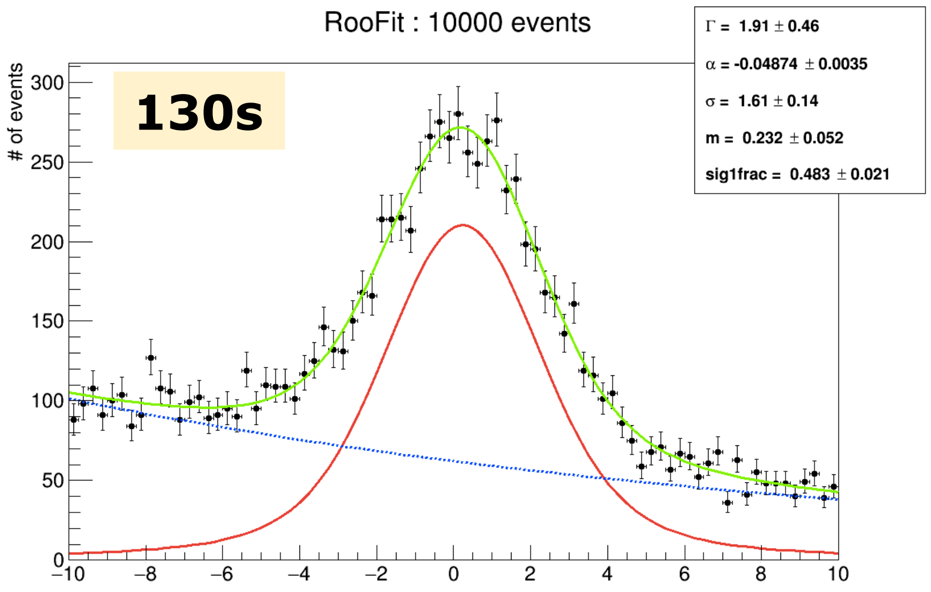
```
--------------------fit done; check best estimates for the model parameters---------
[#1] INFO:NumericIntegration -- RooRealIntegral::init(convolution_Int[xvar]) using numeric integrator RooIntegrator1D to calculate Int(xvar)
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) directly selected PDF components: (convolution)
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) indirectly selected PDF components: (BW,resol)
[#1] INFO:NumericIntegration -- RooRealIntegral::init(convolution_Int[xvar]) using numeric integrator RooIntegrator1D to calculate Int(xvar)
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) directly selected PDF components: (Bkg)
[#1] INFO:Plotting -- RooAbsPdf::plotOn(totalPDF) indirectly selected PDF components: ()
[#1] INFO:NumericIntegration -- RooRealIntegral::init(convolution_Int[xvar]) using numeric integrator RooIntegrator1D to calculate Int(xvar)
Info in <TCanvas::Print>: file plots/RooConvGen_10000.png has been created
```

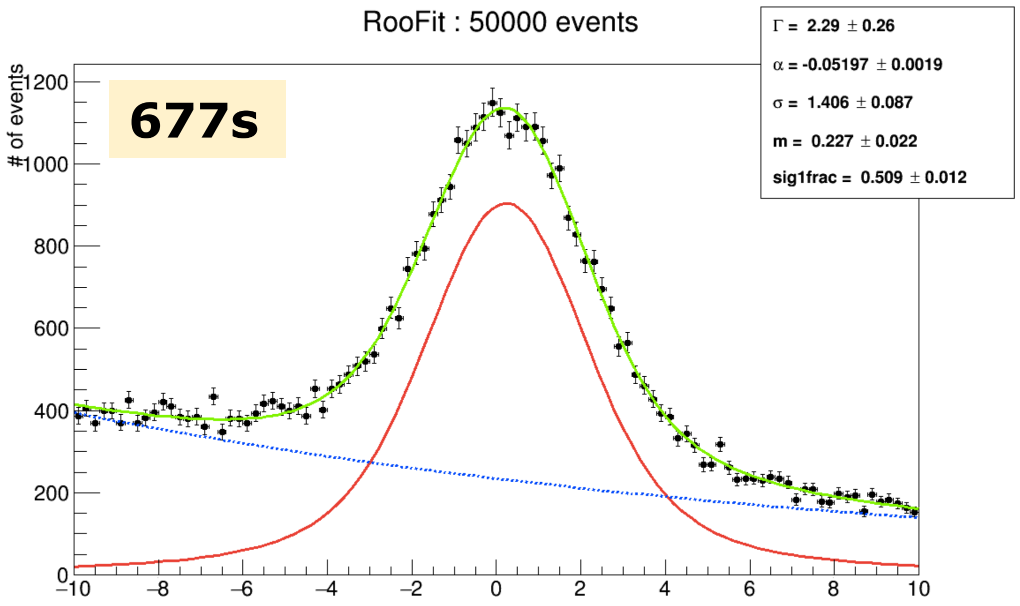
The obtained plot:

```
[[pompili@vm-pompili Esercitazione-4]$ display plots/RooConvGen_10000.png
```



**Example of plots obtained in these 2 cases** (timing info obtained from a past exercise)**:**



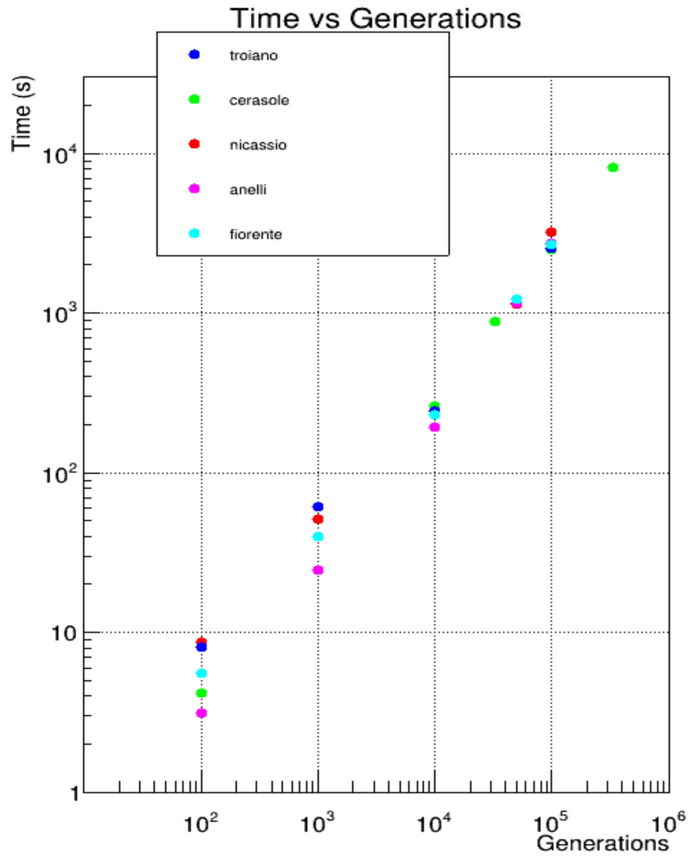




Since the generation time is negligible with respect to the fitting time,
the needed time refers to the UML fitting task (on some server @ ReCas, long time ago):

| #events | CPU-timing (sec) |
|---|---|
| 10K | 130 |
| 50K | 677 |
| 100K | 1250 |

scaling behaviour seems to be approx. **linear**
(see also next slide)
[by extrapolation 1M would require ~210min]

- In the course of academic year 2021-22 the students - as an exercise - measured these CPU-timings :




- To shorten the CPU-timing we can exploit the **parallel computing capabilities** of Roofit; just add the following option:

```
RooAbsReal*  nll = total.createNLL(*data, NumCPU(4));
```

To check how many core the VM of the course has, give the command `lscpu` or alternatively do `cat /proc/cpuinfo`

# Considerations about the seed to provide to the random generator

To be sure to generate two different distributions we need to provide two different seeds with the method `SetSeed`:

```
RooRandom::randomGenerator()->SetSeed( )
```

put here an `int` or `long int` number! (*)

Note : it can be checked (from the output txt file) that:

- if you use the same seed twice the generated values are the same (and thus the two distributions)

- if you use the same seed and generate two different sequences/lists of values (A and B), with #A < #B, the first #A generated values in the list B are the same as those in list A!

(*) Ideally, the seed can be chosen as the time of the system at the start of the execution of the macro.
In this way the seed is automatically different every time you run the task.

# Exercise for homework

It's a good practice to **compare the result of the fit** (best estimates of the parameters) **w.r.t. the values used at generation!**

**You can verify that the agreement enhances when the number of generated events increases**

**(as expected from the consistency property of a maximum likelihood estimator).**

## Insight info

**1) The Monte Carlo method is well and compactly explained in chapter 3 of Cowan's textbook !**

Note that **RooFit uses the *acceptance-rejection method*** (paragraph 3.3)

**2) Being able to generate distributions according to some model can be rather useful in order to use the so called *MC toys technique.***

Have a look for instance at:
http://roofit.sourceforge.net/docs/tutorial/fitgen/roofit_tutorial_fitgen.pdf

A specific application of the MC toys is set up when one needs to estimate the ***p-value*** of a distribution to determine the **statistical significance of a physical signal.**

See for instance slides 5-6 of A.P. talk @ Conference ACAT2016:
https://indico.cern.ch/event/397113/contributions/1837858/attachments/1213108/1770056/pompili_acat16_final.pdf