Likelihood

- Likelihood function, is a function of parameters of a statistical model given data.
- Plays a key role in methods of estimating a parameter from a set of statistics.

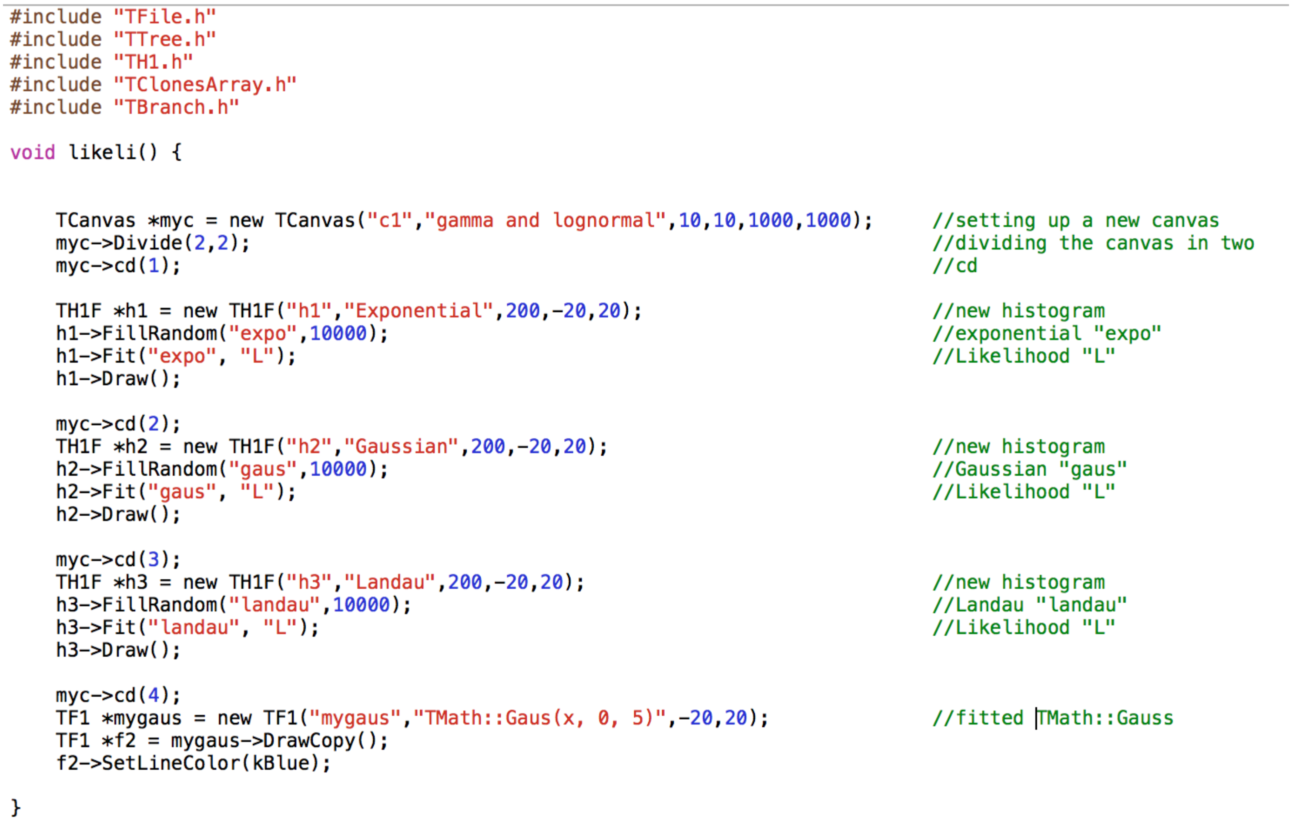
The Difference Between Likelihood & Probability

Proability

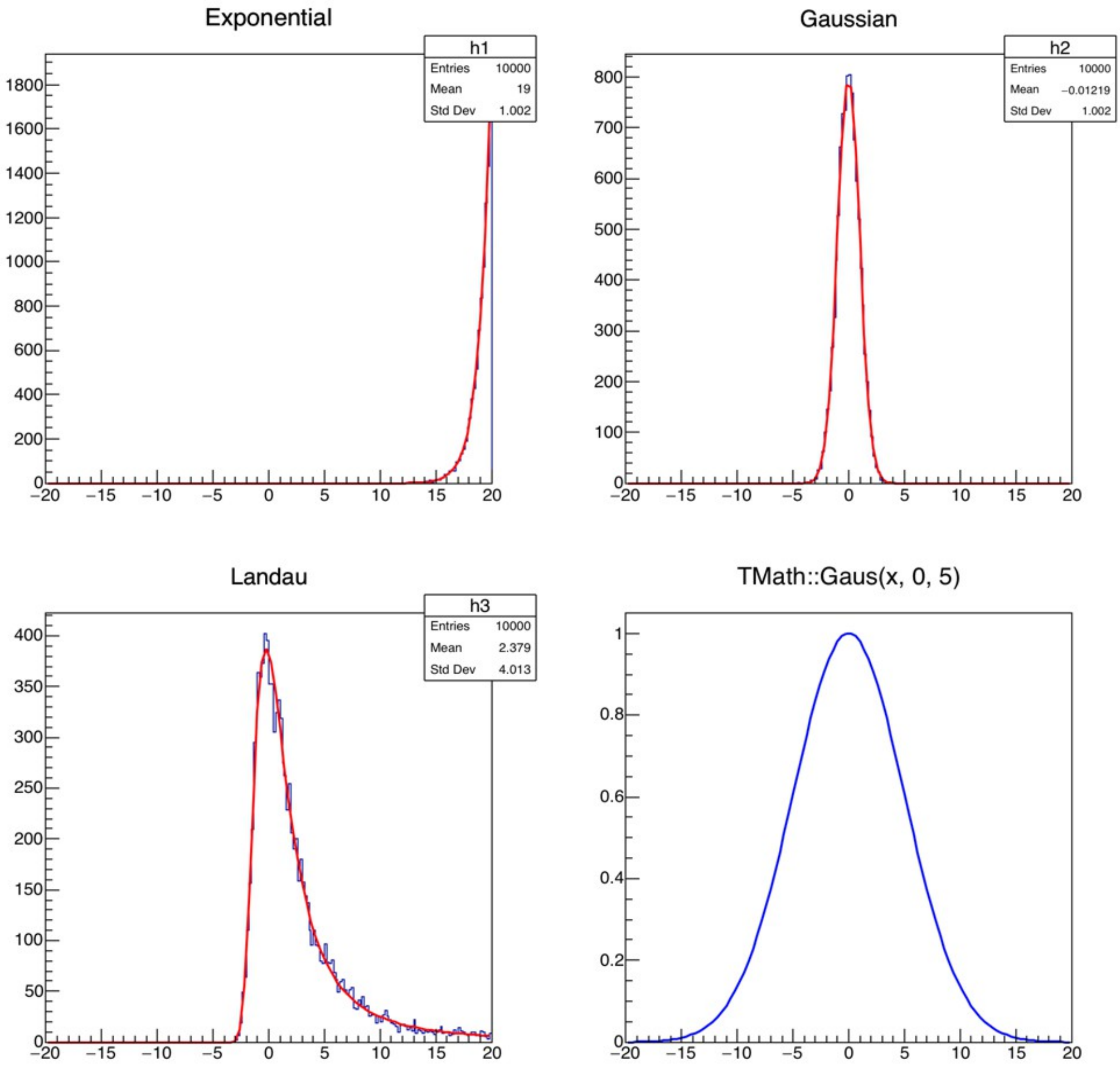
- The probability of observing a particular set of outcomes can be calculated by making suitable assumptions, like the coin tosses being independent, or the prob. Of heads or tails being ½.
- By this definition, O is the “observed outcomes” and “θ” is the set of parameters.
- P(O|θ) : given the parameters, the probability that we would observe the outcomes.

Likelihood

- In real life, we often don't know the parameters θ completely.
- We can observe O, and then estimate θ.
- L(θ|O): choosing a θ that would maximize the probability that we would observe O.



```
#include "TFile.h"
#include "TTree.h"
#include "TH1.h"
#include "TClonesArray.h"
#include "TBranch.h"

void likeli() {


    TCanvas *myc = new TCanvas("c1","gamma and lognormal",10,10,1000,1000);     //setting up a new canvas
    myc->Divide(2,2);                                                           //dividing the canvas in two
    myc->cd(1);                                                                 //cd

    TH1F *h1 = new TH1F("h1","Exponential",200,-20,20);                         //new histogram
    h1->FillRandom("expo",10000);                                               //exponential "expo"
    h1->Fit("expo", "L");                                                       //Likelihood "L"
    h1->Draw();

    myc->cd(2);
    TH1F *h2 = new TH1F("h2","Gaussian",200,-20,20);                            //new histogram
    h2->FillRandom("gaus",10000);                                               //Gaussian "gaus"
    h2->Fit("gaus", "L");                                                       //Likelihood "L"
    h2->Draw();

    myc->cd(3);
    TH1F *h3 = new TH1F("h3","Landau",200,-20,20);                              //new histogram
    h3->FillRandom("landau",10000);                                             //Landau "landau"
    h3->Fit("landau", "L");                                                     //Likelihood "L"
    h3->Draw();

    myc->cd(4);
    TF1 *mygaus = new TF1("mygaus","TMath::Gaus(x, 0, 5)",-20,20);              //fitted TMath::Gauss
    TF1 *f2 = mygaus->DrawCopy();
    f2->SetLineColor(kBlue);

}
```

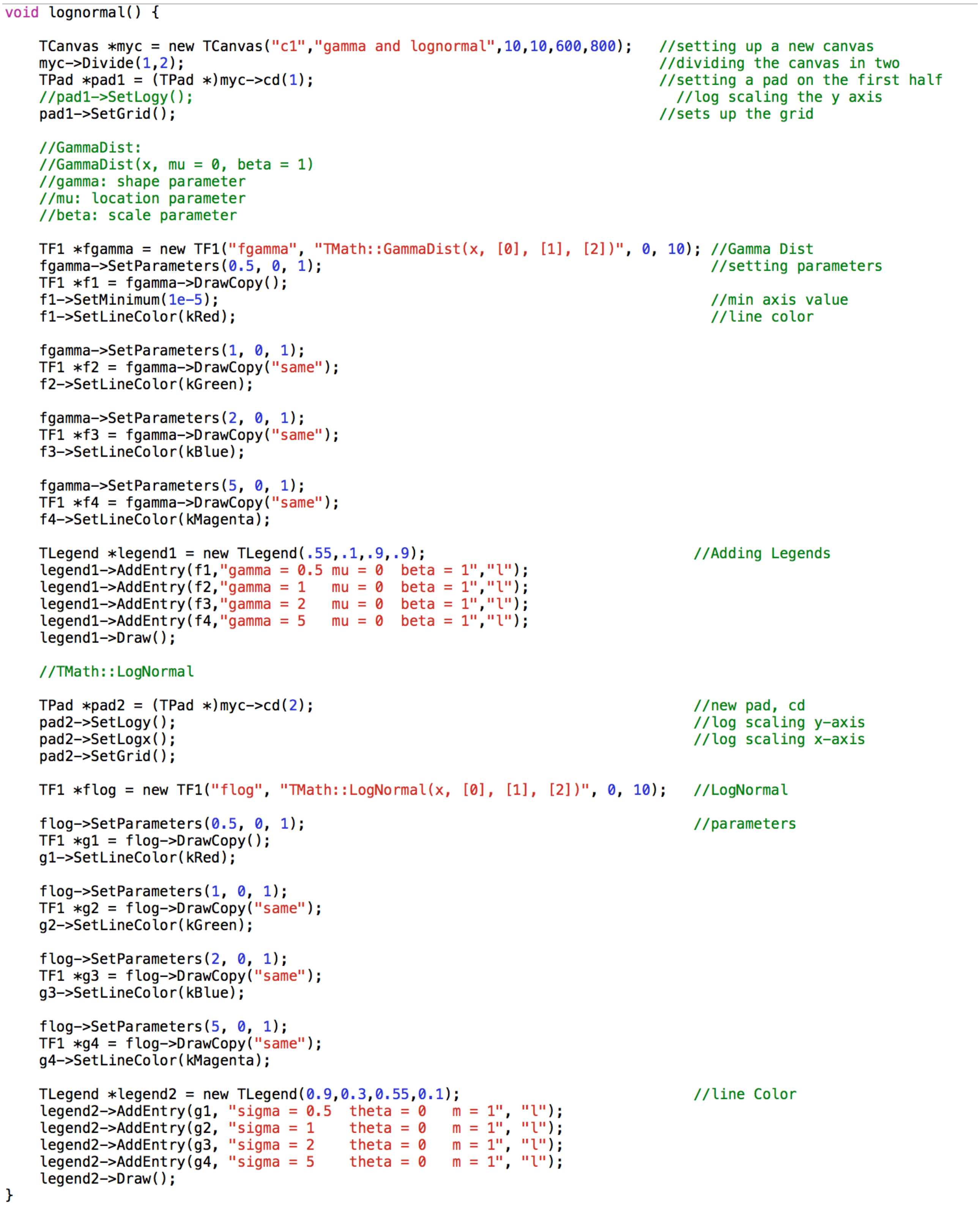

Exponential
h1
Entries 10000
Mean 19
Std Dev 1.002
0
200
400
600
800
1000
1200
1400
1600
1800
−20
−15
−10
−5
0
5
10
15
20
Gaussian
h2
Entries 10000
Mean −0.01219
Std Dev 1.002
0
100
200
300
400
500
600
700
800
Landau
h3
Entries 10000
Mean 2.379
Std Dev 4.013
0
50
100
150
200
250
300
350
400
TMath::Gaus(x, 0, 5)
0
0.2
0.4
0.6
0.8
1

Lognormal

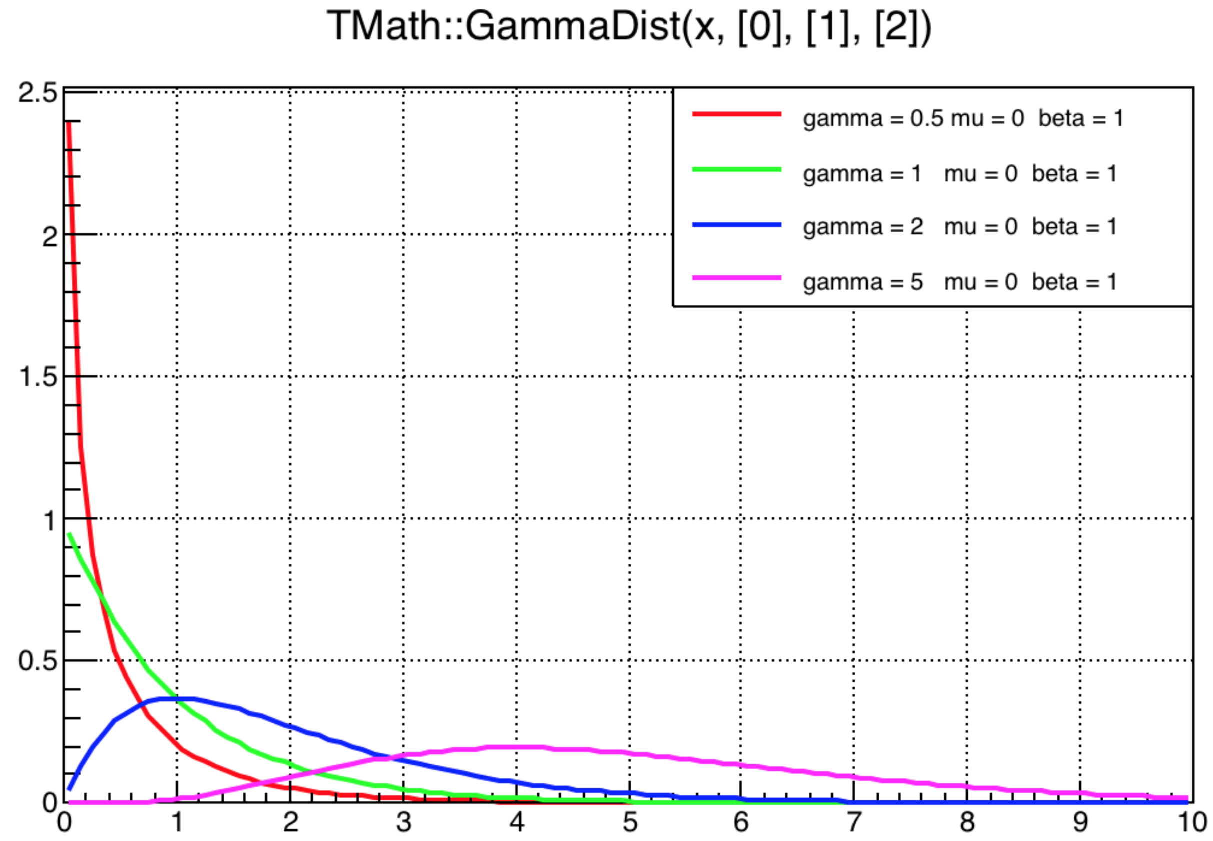
It is the continuous probability distribution of a random variable, whose logarithm is a normal (Gaussian) distribution.

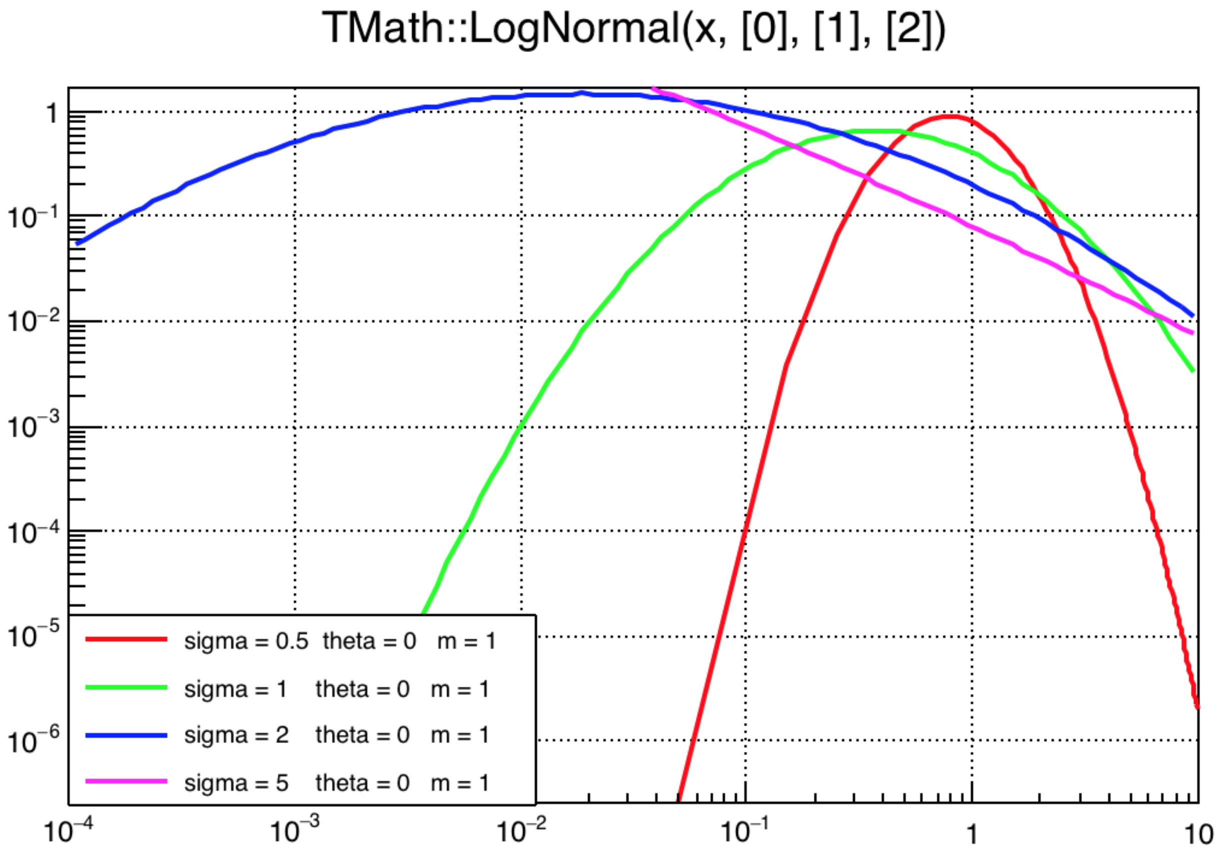
Applications:

- Particle size distributions (colloidal chemistry)
- Particle size distributions (technology)
- Economics (income of +95% distributed log-normally)
- Hydrology (ex: rainfall values)



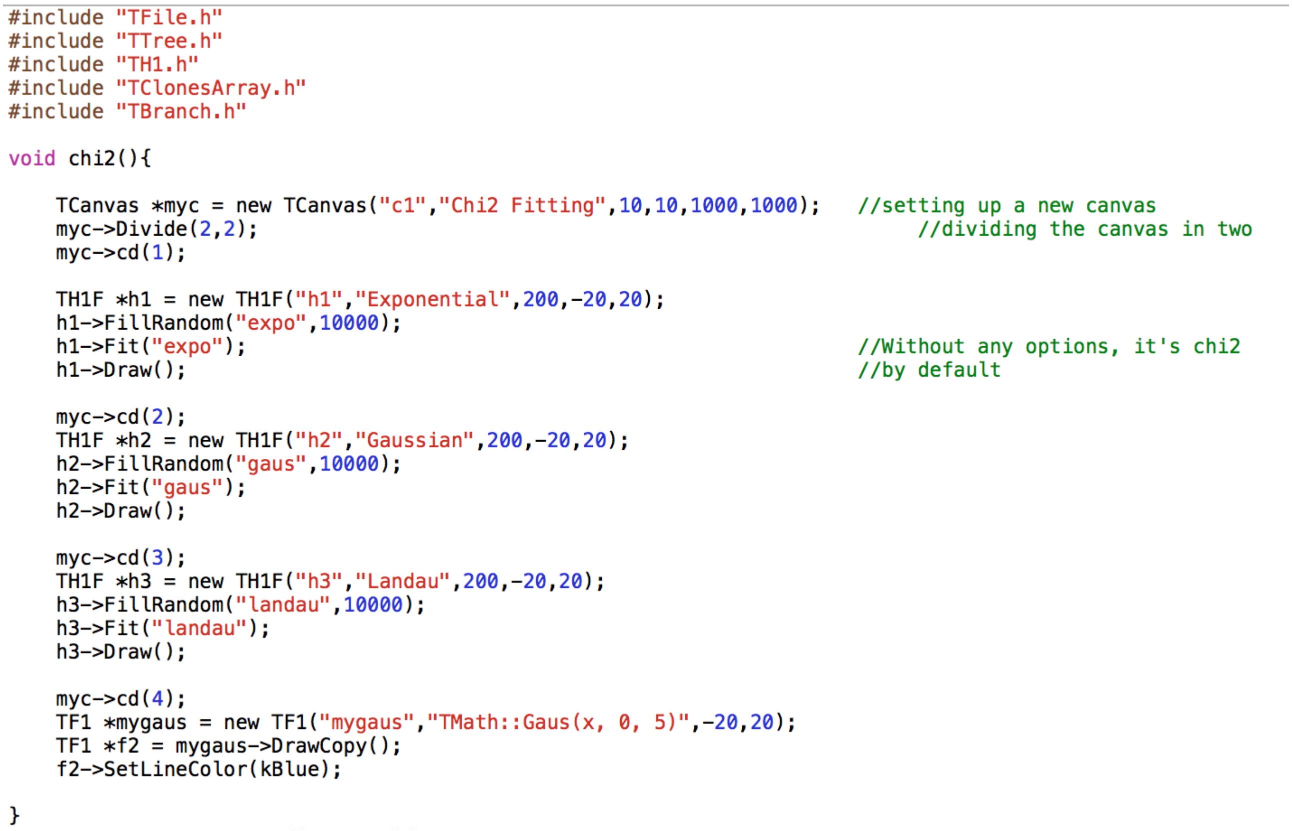
```
void lognormal() {

    TCanvas *myc = new TCanvas("c1","gamma and lognormal",10,10,600,800);   //setting up a new canvas
    myc->Divide(1,2);                                                        //dividing the canvas in two
    TPad *pad1 = (TPad *)myc->cd(1);                                         //setting a pad on the first half
    //pad1->SetLogy();                                                         //log scaling the y axis
    pad1->SetGrid();                                                         //sets up the grid

    //GammaDist:
    //GammaDist(x, mu = 0, beta = 1)
    //gamma: shape parameter
    //mu: location parameter
    //beta: scale parameter

    TF1 *fgamma = new TF1("fgamma", "TMath::GammaDist(x, [0], [1], [2])", 0, 10); //Gamma Dist
    fgamma->SetParameters(0.5, 0, 1);                                              //setting parameters
    TF1 *f1 = fgamma->DrawCopy();
    f1->SetMinimum(1e-5);                                                          //min axis value
    f1->SetLineColor(kRed);                                                        //line color

    fgamma->SetParameters(1, 0, 1);
    TF1 *f2 = fgamma->DrawCopy("same");
    f2->SetLineColor(kGreen);

    fgamma->SetParameters(2, 0, 1);
    TF1 *f3 = fgamma->DrawCopy("same");
    f3->SetLineColor(kBlue);

    fgamma->SetParameters(5, 0, 1);
    TF1 *f4 = fgamma->DrawCopy("same");
    f4->SetLineColor(kMagenta);

    TLegend *legend1 = new TLegend(.55,.1,.9,.9);                                //Adding Legends
    legend1->AddEntry(f1,"gamma = 0.5 mu = 0  beta = 1","l");
    legend1->AddEntry(f2,"gamma = 1   mu = 0  beta = 1","l");
    legend1->AddEntry(f3,"gamma = 2   mu = 0  beta = 1","l");
    legend1->AddEntry(f4,"gamma = 5   mu = 0  beta = 1","l");
    legend1->Draw();

    //TMath::LogNormal

    TPad *pad2 = (TPad *)myc->cd(2);                                             //new pad, cd
    pad2->SetLogy();                                                             //log scaling y-axis
    pad2->SetLogx();                                                             //log scaling x-axis
    pad2->SetGrid();

    TF1 *flog = new TF1("flog", "TMath::LogNormal(x, [0], [1], [2])", 0, 10);   //LogNormal

    flog->SetParameters(0.5, 0, 1);                                              //parameters
    TF1 *g1 = flog->DrawCopy();
    g1->SetLineColor(kRed);

    flog->SetParameters(1, 0, 1);
    TF1 *g2 = flog->DrawCopy("same");
    g2->SetLineColor(kGreen);

    flog->SetParameters(2, 0, 1);
    TF1 *g3 = flog->DrawCopy("same");
    g3->SetLineColor(kBlue);

    flog->SetParameters(5, 0, 1);
    TF1 *g4 = flog->DrawCopy("same");
    g4->SetLineColor(kMagenta);

    TLegend *legend2 = new TLegend(0.9,0.3,0.55,0.1);                            //line Color
    legend2->AddEntry(g1, "sigma = 0.5  theta = 0   m = 1", "l");
    legend2->AddEntry(g2, "sigma = 1    theta = 0   m = 1", "l");
    legend2->AddEntry(g3, "sigma = 2    theta = 0   m = 1", "l");
    legend2->AddEntry(g4, "sigma = 5    theta = 0   m = 1", "l");
    legend2->Draw();
}
```


TMath::GammaDist(x, [0], [1], [2])
gamma = 0.5 mu = 0 beta = 1
gamma = 1   mu = 0 beta = 1
gamma = 2   mu = 0 beta = 1
gamma = 5   mu = 0 beta = 1
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10

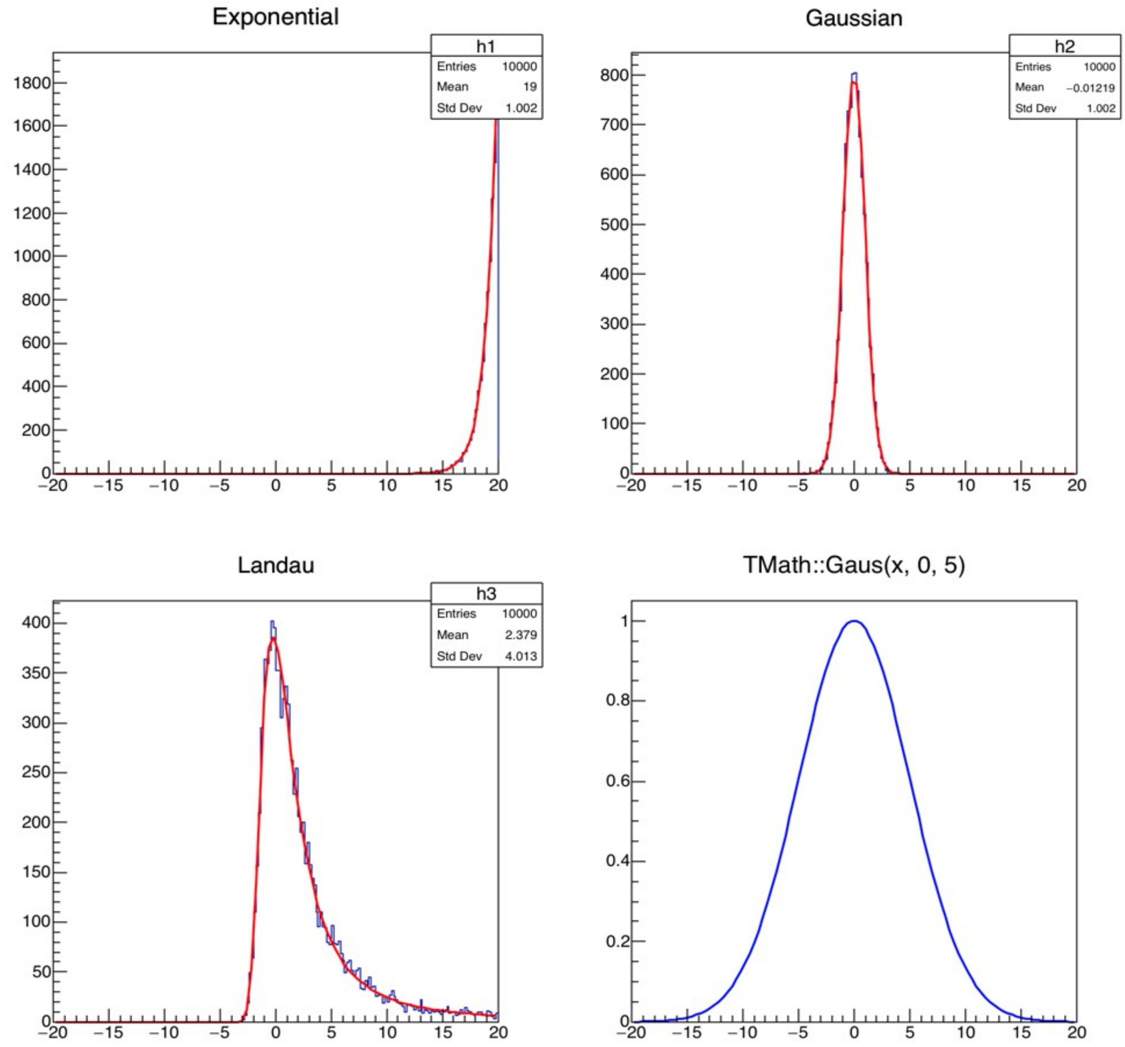

TMath::LogNormal(x, [0], [1], [2])
sigma = 0.5 theta = 0   m = 1
sigma = 1   theta = 0   m = 1
sigma = 2   theta = 0   m = 1
sigma = 5   theta = 0   m = 1
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10

Chi-Squared

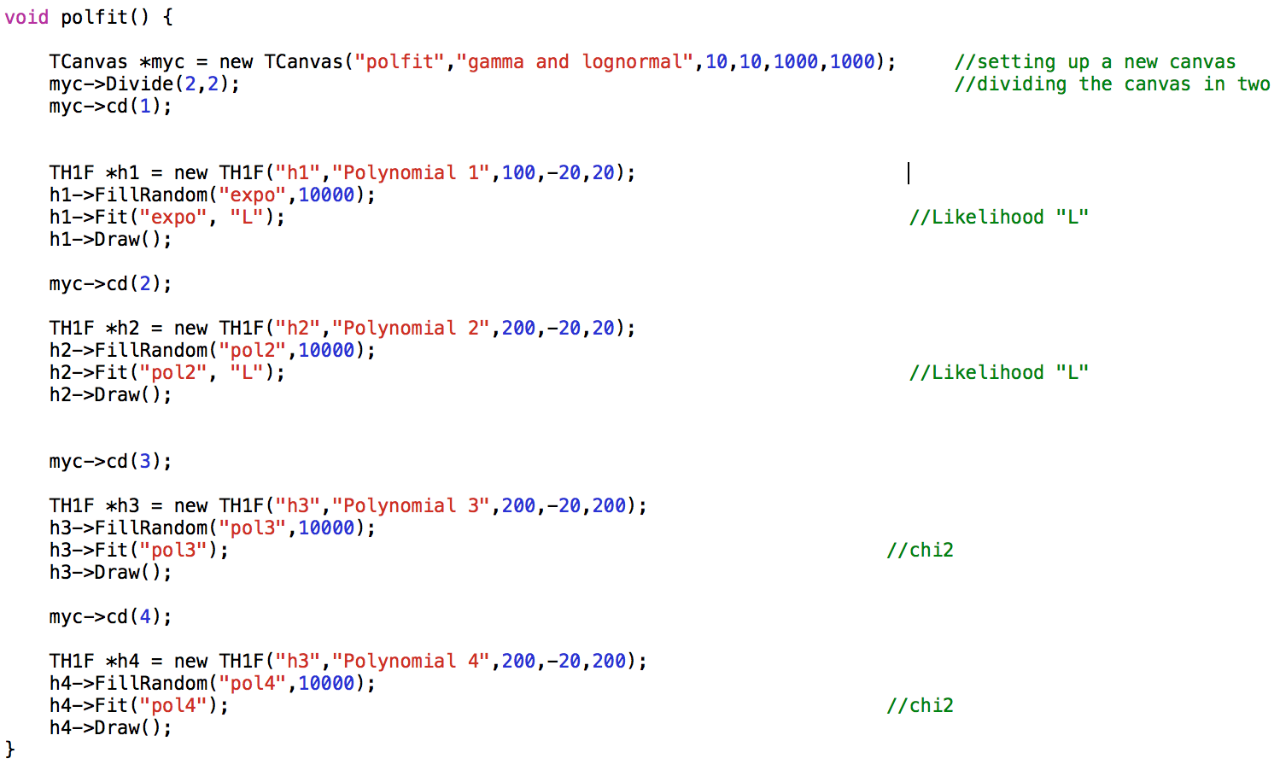
- Distribution of a sum of the squares of N independent standard normal random variables.
- Chi2 test, tests the goodness of a fit.



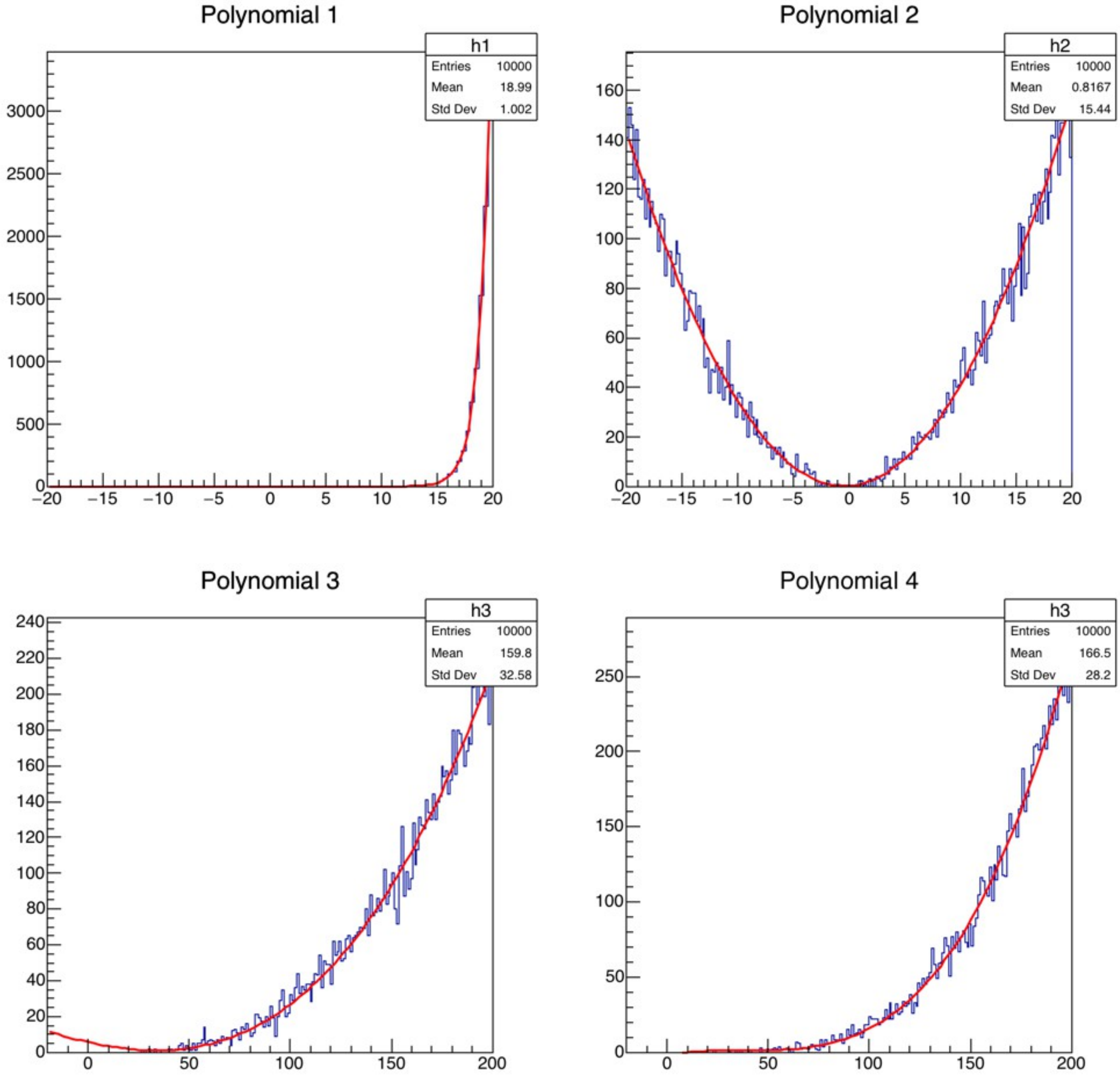
```
#include "TFile.h"
#include "TTree.h"
#include "TH1.h"
#include "TClonesArray.h"
#include "TBranch.h"

void chi2(){

    TCanvas *myc = new TCanvas("c1","Chi2 Fitting",10,10,1000,1000);   //setting up a new canvas
    myc->Divide(2,2);                                                       //dividing the canvas in two
    myc->cd(1);

    TH1F *h1 = new TH1F("h1","Exponential",200,-20,20);
    h1->FillRandom("expo",10000);
    h1->Fit("expo");                                                    //Without any options, it's chi2
    h1->Draw();                                                         //by default

    myc->cd(2);
    TH1F *h2 = new TH1F("h2","Gaussian",200,-20,20);
    h2->FillRandom("gaus",10000);
    h2->Fit("gaus");
    h2->Draw();

    myc->cd(3);
    TH1F *h3 = new TH1F("h3","Landau",200,-20,20);
    h3->FillRandom("landau",10000);
    h3->Fit("landau");
    h3->Draw();

    myc->cd(4);
    TF1 *mygaus = new TF1("mygaus","TMath::Gaus(x, 0, 5)",-20,20);
    TF1 *f2 = mygaus->DrawCopy();
    f2->SetLineColor(kBlue);

}
```



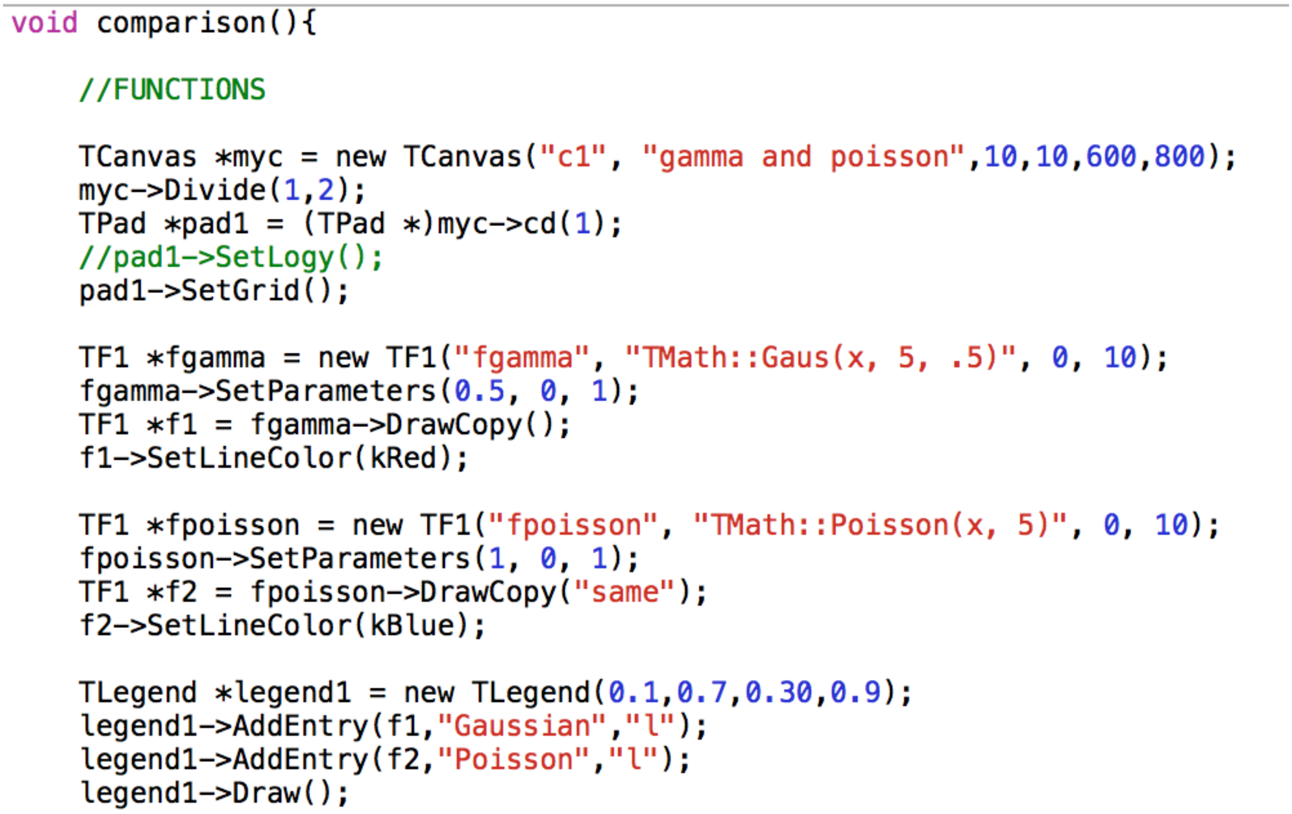
Predefined Polynomial Fitting



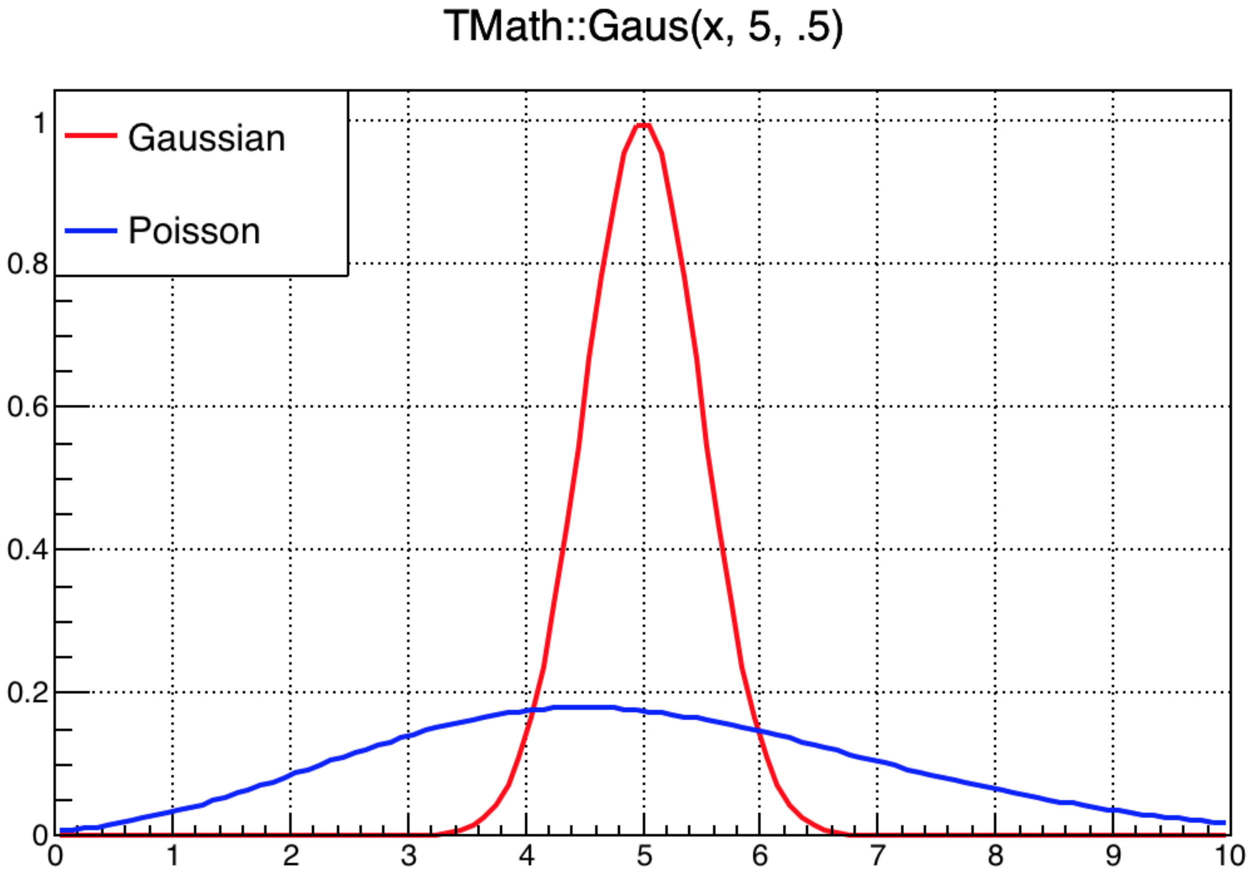
```
void polfit() {

    TCanvas *myc = new TCanvas("polfit","gamma and lognormal",10,10,1000,1000);     //setting up a new canvas
    myc->Divide(2,2);                                                               //dividing the canvas in two
    myc->cd(1);


    TH1F *h1 = new TH1F("h1","Polynomial 1",100,-20,20);
    h1->FillRandom("expo",10000);
    h1->Fit("expo", "L");                                                       //Likelihood "L"
    h1->Draw();

    myc->cd(2);

    TH1F *h2 = new TH1F("h2","Polynomial 2",200,-20,20);
    h2->FillRandom("pol2",10000);
    h2->Fit("pol2", "L");                                                       //Likelihood "L"
    h2->Draw();


    myc->cd(3);

    TH1F *h3 = new TH1F("h3","Polynomial 3",200,-20,200);
    h3->FillRandom("pol3",10000);
    h3->Fit("pol3");                                                          //chi2
    h3->Draw();

    myc->cd(4);

    TH1F *h4 = new TH1F("h3","Polynomial 4",200,-20,200);
    h4->FillRandom("pol4",10000);
    h4->Fit("pol4");                                                          //chi2
    h4->Draw();
}
```



TMath::Poisson, TMath::Gaus



```
void comparison(){

    //FUNCTIONS

    TCanvas *myc = new TCanvas("c1", "gamma and poisson",10,10,600,800);
    myc->Divide(1,2);
    TPad *pad1 = (TPad *)myc->cd(1);
    //pad1->SetLogy();
    pad1->SetGrid();

    TF1 *fgamma = new TF1("fgamma", "TMath::Gaus(x, 5, .5)", 0, 10);
    fgamma->SetParameters(0.5, 0, 1);
    TF1 *f1 = fgamma->DrawCopy();
    f1->SetLineColor(kRed);

    TF1 *fpoisson = new TF1("fpoisson", "TMath::Poisson(x, 5)", 0, 10);
    fpoisson->SetParameters(1, 0, 1);
    TF1 *f2 = fpoisson->DrawCopy("same");
    f2->SetLineColor(kBlue);

    TLegend *legend1 = new TLegend(0.1,0.7,0.30,0.9);
    legend1->AddEntry(f1,"Gaussian","l");
    legend1->AddEntry(f2,"Poisson","l");
    legend1->Draw();
```


TMath::Gaus(x, 5, .5)
Gaussian
Poisson