ROOT Tutorial

Nilanga Wickramaarachchi

06/17/2021

Introduction to ROOT

- **ROOT is a software framework for data processing and analysis**
- **Started development in 1994 at CERN**
- **Object-oriented program and library written in C++**
- **Lot of in-built classes to simplify data analysis**
- **Used widely in high energy physics experiments**
    - **Store large chunks of data collected**
    - **Visualization of data (histograms, graphs)**
    - **Include standard mathematical functions for modeling data (fitting etc.)**

Installing ROOT

Method 1 - use pre-compiled binary distribution

- This is the quickest way to install ROOT !

Steps:

(1) Download binary distribution of ROOT that matches your operating system

Select a version of ROOT from https://root.cern/install/all_releases/

Binary distributions

Platform	Files	Size
CentOS Cern 7 gcc4.8	root_v6.14.06.Linux-centos7-x86_64-gcc4.8.tar.gz	141M
Linux fedora27 gcc7.3	root_v6.14.06.Linux-fedora27-x86_64-gcc7.3.tar.gz	132M
Linux fedora28 gcc8.2	root_v6.14.06.Linux-fedora28-x86_64-gcc8.2.tar.gz	131M
Ubuntu 14 gcc4.8	root_v6.14.06.Linux-ubuntu14-x86_64-gcc4.8.tar.gz	141M
Ubuntu 16 gcc5.4	root_v6.14.06.Linux-ubuntu16-x86_64-gcc5.4.tar.gz	142M
Ubuntu 18 gcc7.3	root_v6.14.06.Linux-ubuntu18-x86_64-gcc7.3.tar.gz	140M
OsX 10.12 clang90	root_v6.14.06.macosx64-10.12-clang90.dmg	126M
OsX 10.12 clang90	root_v6.14.06.macosx64-10.12-clang90.tar.gz	125M
OsX 10.13 clang100	root_v6.14.06.macosx64-10.13-clang100.dmg	128M
OsX 10.13 clang100	root_v6.14.06.macosx64-10.13-clang100.tar.gz	127M
OsX 10.14 clang100	root_v6.14.06.macosx64-10.14-clang100.dmg	128M
OsX 10.14 clang100	root_v6.14.06.macosx64-10.14-clang100.tar.gz	127M

(2) Unpack to the destination folder

```
Nilangas-MacBook-Air:~ niw$ cd Desktop/
Nilangas-MacBook-Air:Desktop niw$ pwd
/Users/niw/Desktop
Nilangas-MacBook-Air:Desktop niw$ mkdir ROOT_binary_dist
Nilangas-MacBook-Air:Desktop niw$ cd ROOT_binary_dist/
Nilangas-MacBook-Air:ROOT_binary_dist niw$ tar -xzvf ../../Downloads/root_v6.14.
06.macosx64-10.12-clang90.tar
```

```
Nilangas-MacBook-Air:ROOT_binary_dist niw$ ls
root
Nilangas-MacBook-Air:ROOT_binary_dist niw$ ls root/bin/
g2root               root-config           rootprint
genreflex            root.exe              rootrm
h2root               rootbrowse            roots
hadd                 rootcint              roots.exe
hist2workspace       rootcling             rootslimtree
memprobe             rootcp                setenvwrap.csh
pq2                  rootd                 setxrd.csh
prepareHistFactory   rootdrawtree          setxrd.sh
proofd               rooteventselector     ssh2rpd
proofexecv           rootls                thisroot.csh
proofserv            rootmkdir             thisroot.sh
proofserv.exe        rootmv                xpdtest
rmkdepend            rootn.exe
root                 rootnb.exe
```

Look for thisroot.sh (csh) file inside root/bin folder

(3) Source thisroot.sh file

(4) Run “root”

```
Nilangas-MacBook-Air:ROOT_binary_dist niw$ source root/bin/thisroot.sh
Nilangas-MacBook-Air:ROOT_binary_dist niw$ root
   ------------------------------------------------------------
  | Welcome to ROOT 6.14/06                http://root.cern.ch |
  |                               (c) 1995-2018, The ROOT Team |
  | Built for macosx64                                         |
  | From tags/v6-14-06@v6-14-06, Nov 05 2018, 10:35:04         |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
   ------------------------------------------------------------

root [0] .q
```

type “.q” to quit

Now ready to use ROOT !!

Method 2 - Build from source

(https://root.cern/install/build_from_source/)

(1) Download the source for the version of ROOT required

Source distribution

Platform	Files	Size
source	root_v6.18.02.source.tar.gz	158M

(2) Run following commands from terminal

```
$ mkdir <builddir> <installdir>
$ cd <builddir>
$ cmake -DCMAKE_INSTALL_PREFIX=<installdir> <sourcedir>
$ cmake --build .
$ source <installdir>/bin/thisroot.sh
```

Using ROOT as a calculator

Use the ROOT interactive shell to do some math !

```
root [0] 2+3
(int) 5
root [1] sqrt(4)
(double) 2.0000000
root [2] TMath::Pi()
(double) 3.1415927
root [3] int a = 4
(int) 4
root [4] double b = TMath::Pi()
(double) 3.1415927
root [5] a*b
(double) 12.566371

root [6] TMath::Cos(60)
(double) -0.95241298
root [7] TMath::Cos(60*TMath::DegToRad())
(double) 0.50000000
```

All ROOT classes start with "T"

many mathematical functions available in ROOT "TMath" namespace

Plotting functions with TF1 class

TF1 (const char ***name**, const char *formula, **Double_t xmin**, **Double_t xmax**, **Option_t** *option)

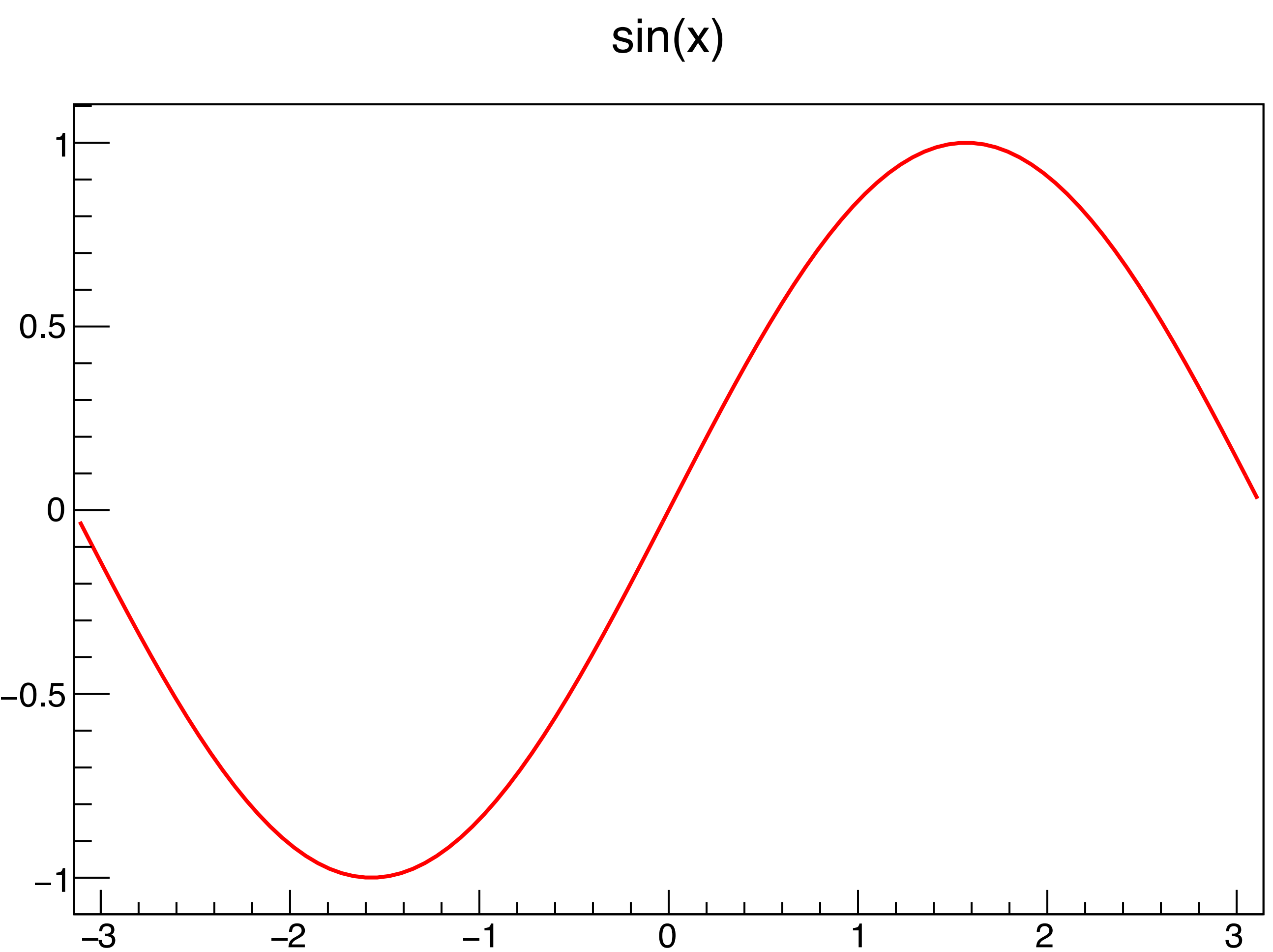
```
TF1 f1("f1","sin(x)",-TMath::Pi(),TMath::Pi());
f1.Draw();
```

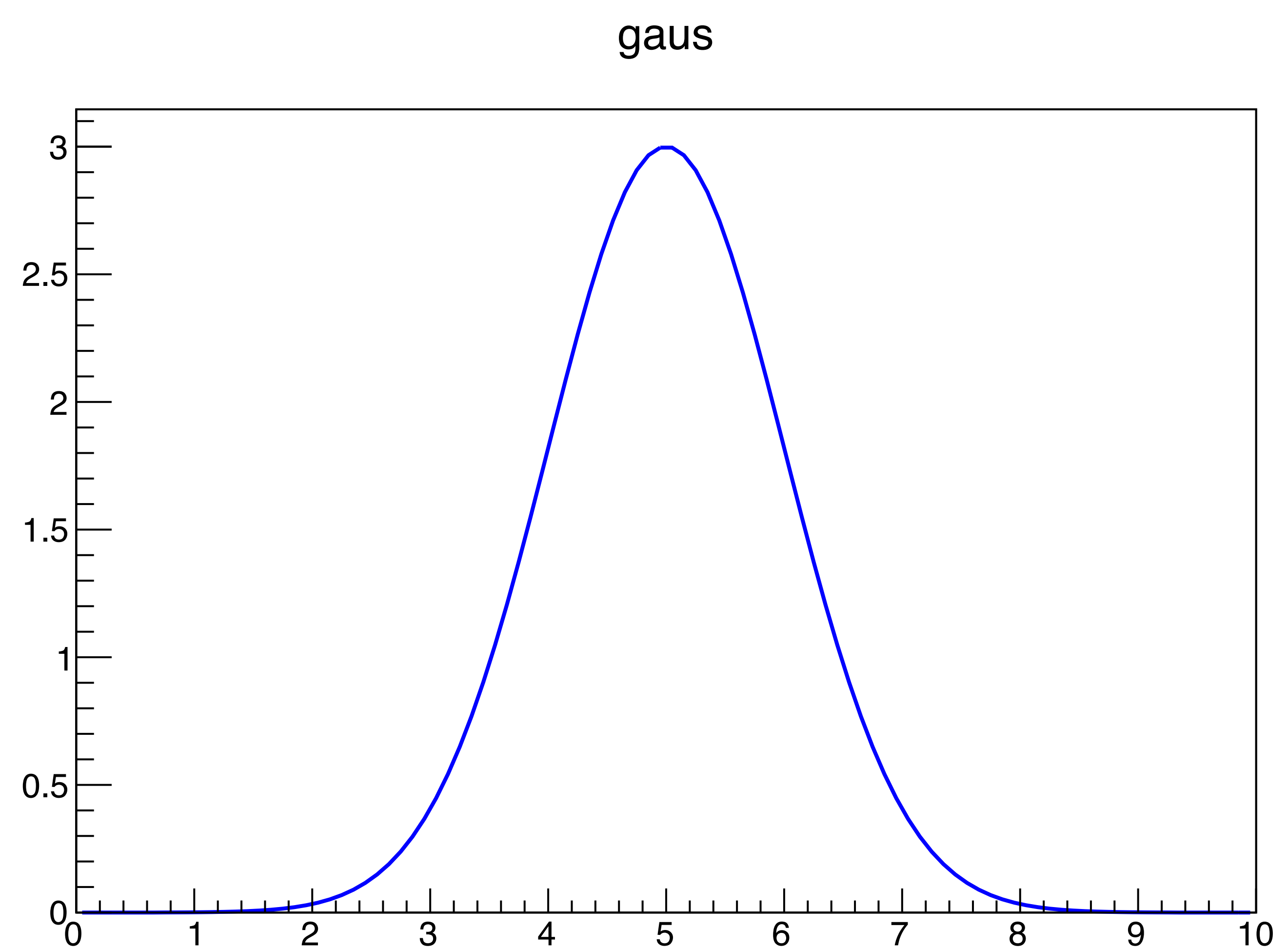
```
TF1 f2("f2","gaus",0,10);
f2.SetParameters(3,5,1);
f2.SetLineColor(4);
f2.Draw();
```

gaus function

```
[0]*exp(-0.5*((x-[1])/[2])**2)
```

parameters





C function with parameters

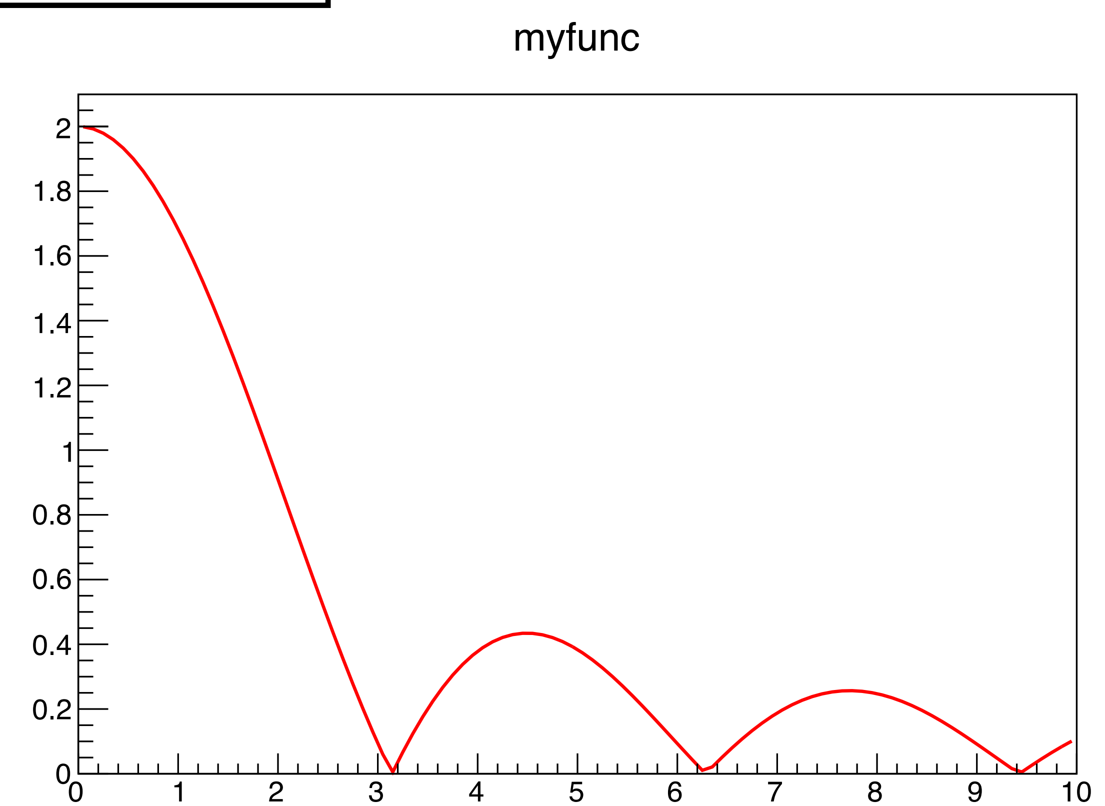
```
Double_t myfunction(Double_t *x, Double_t *par)
{
  Float_t xx =x[0];
  Double_t f = TMath::Abs(par[0]*sin(par[1]*xx)/
xx);
  return f;
}

void myfunc()
{
  TF1 *f1 = new TF1("myfunc",myfunction,0,10,2);
  f1->SetParameters(2,1);
  f1->SetParNames("constant","coefficient");
  f1->Draw();
}
```

number of parameters

macro myfunc.C

Load the macro in ROOT

```
root [0] .L myfunc.C
root [1] myfunc()
```



Histograms

- **Histogram represents how often a measurement value occurs**
- **TH1 class is the base class for histograms**
- **TH1I, TH1D, TH1F are for different memory usage (int, double, float)**
- **Similarly there are TH2*, TH3* classes for 2D and 3D histograms**

```
TH1D (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup)
```

key **title** **number of bins** **x range**

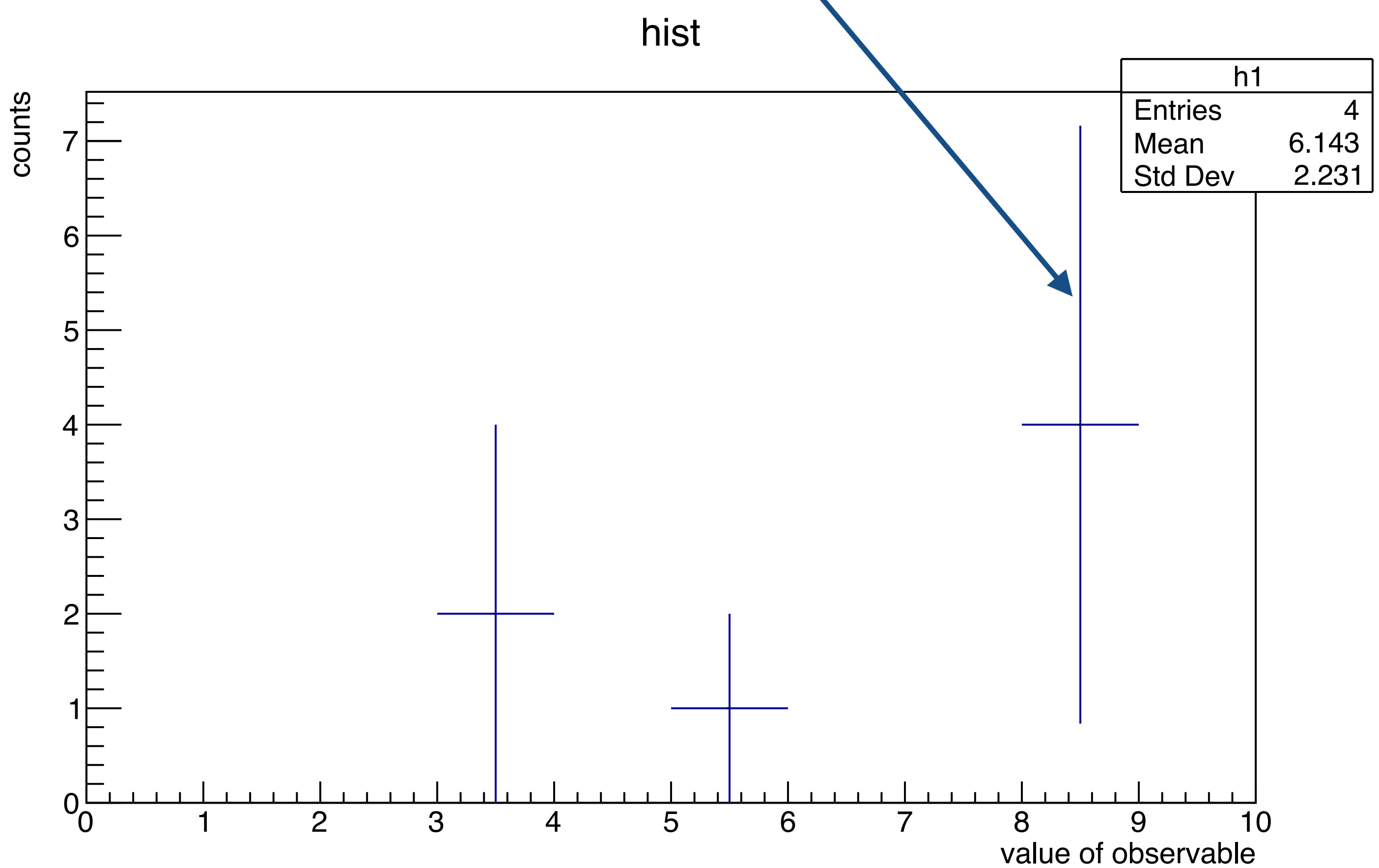
```
TH1D* h1 = new TH1D("h1","hist",10,0,10);
h1->Fill(5)
h1->Fill(3,2)
h1->Fill(8,3)
h1->Fill(8,1)
h1->GetYaxis()->SetTitle("counts");
h1->GetXaxis()->SetTitle("value of observable”);
h1->Draw();
```

value to fill

fill with a weight

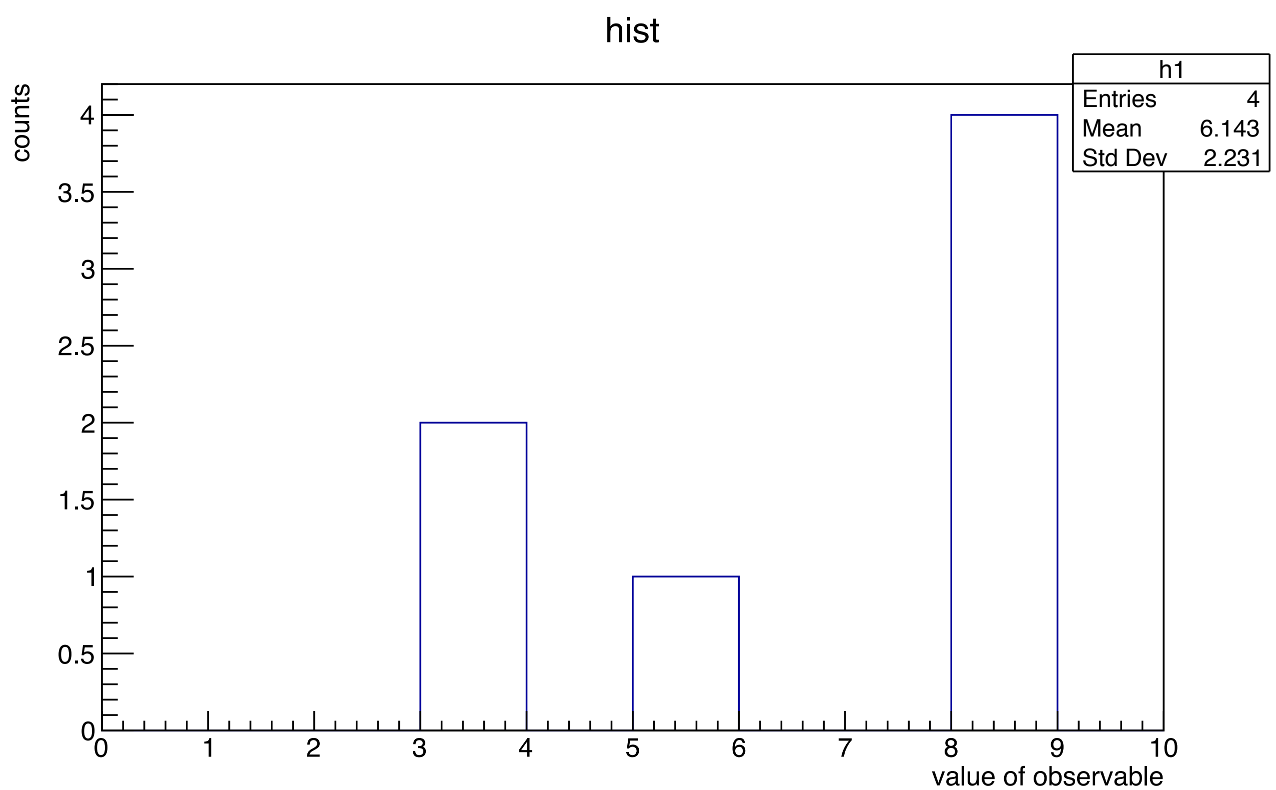
error bar is the square root of sum of weights squared

(e.g. $\Rightarrow$ error = $\sqrt{(3^2 + 1^2)}$)



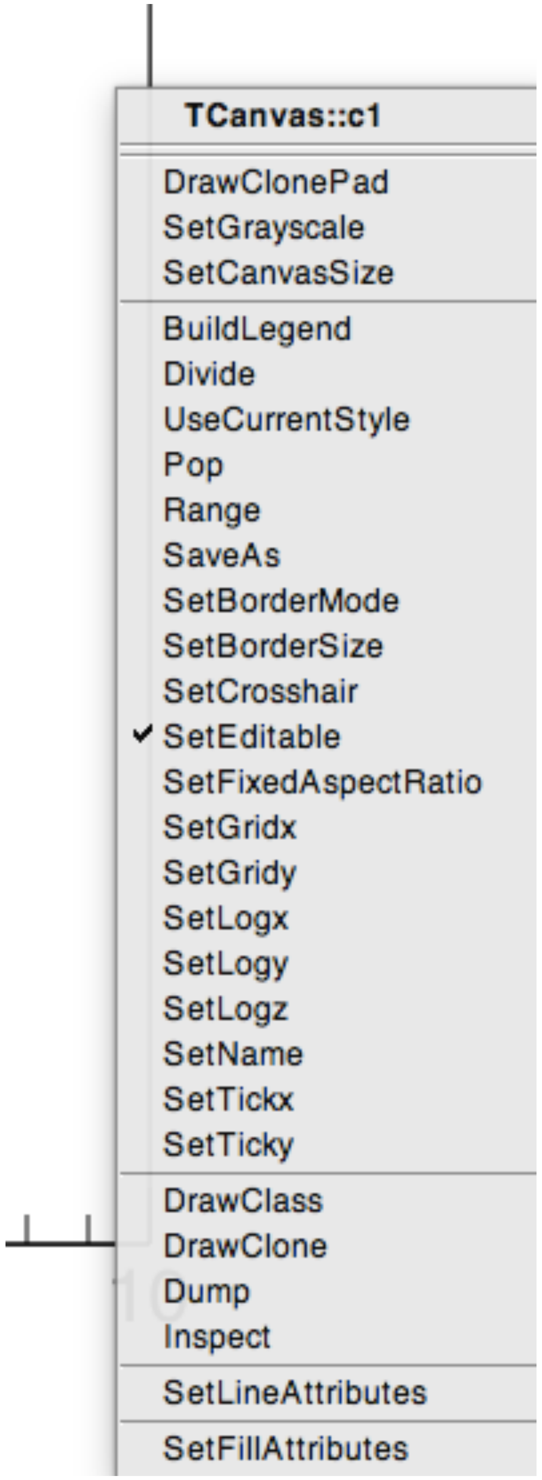
draw without error bars

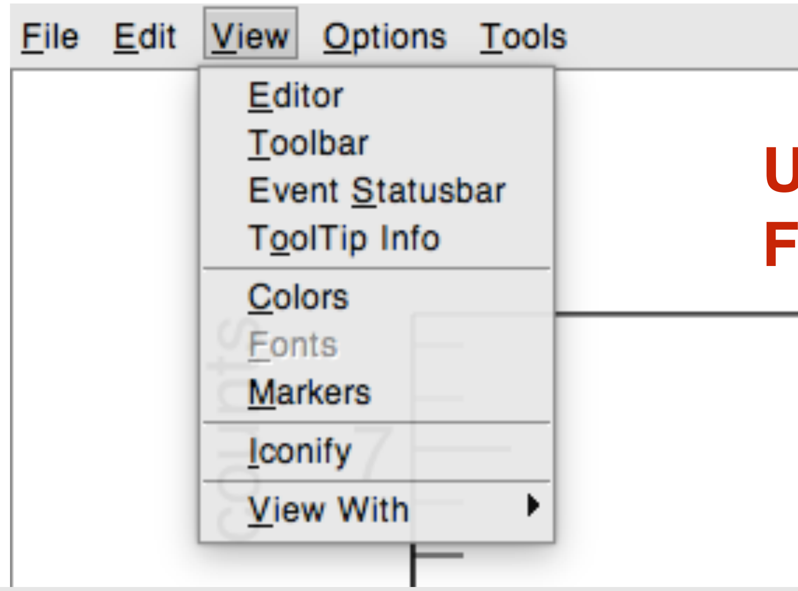
```
h1->Draw("hist");
```



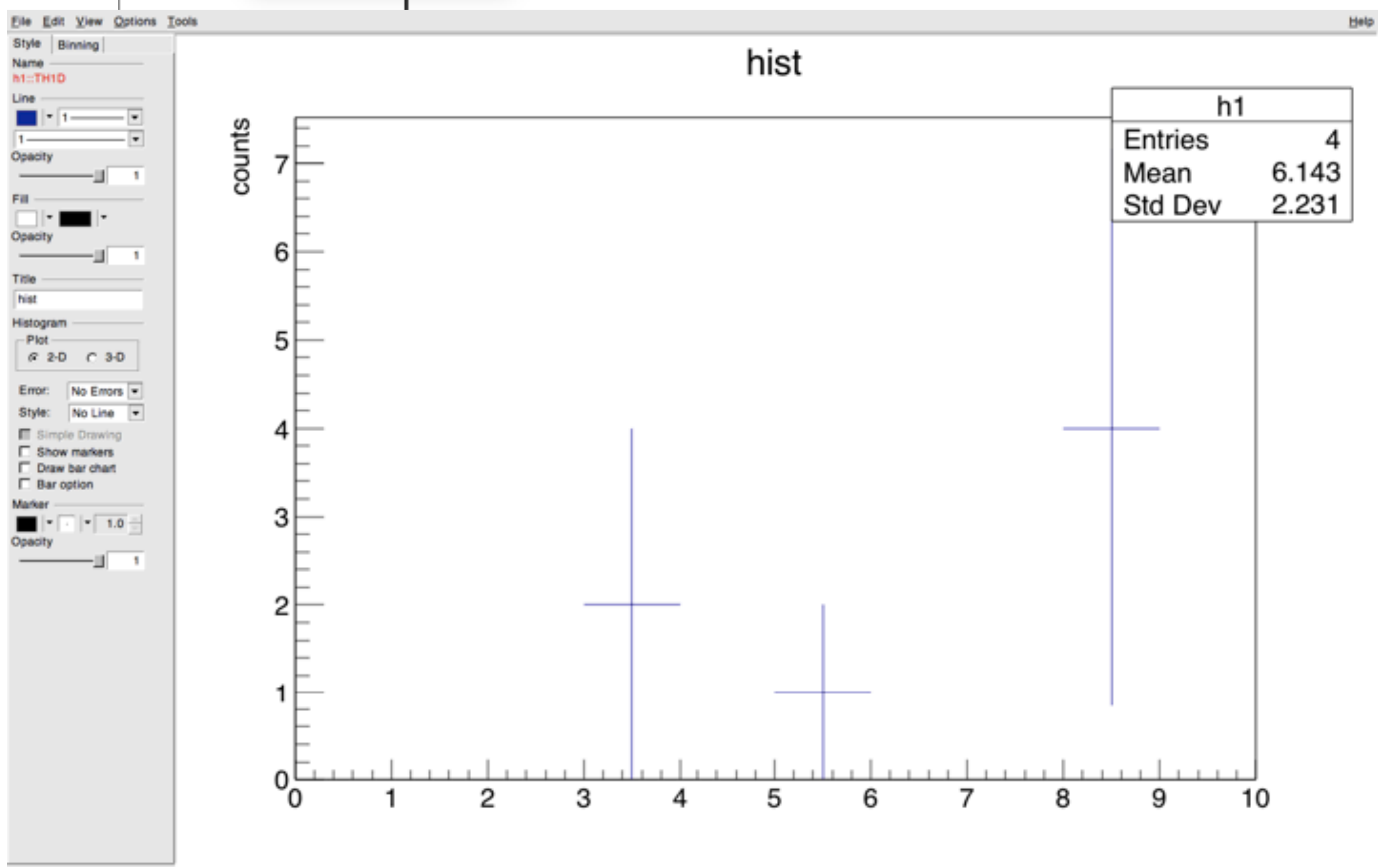
Using ROOT Interactively

Right-click on the canvas
for apply settings interactively





Use View->Editor
For many visualization settings



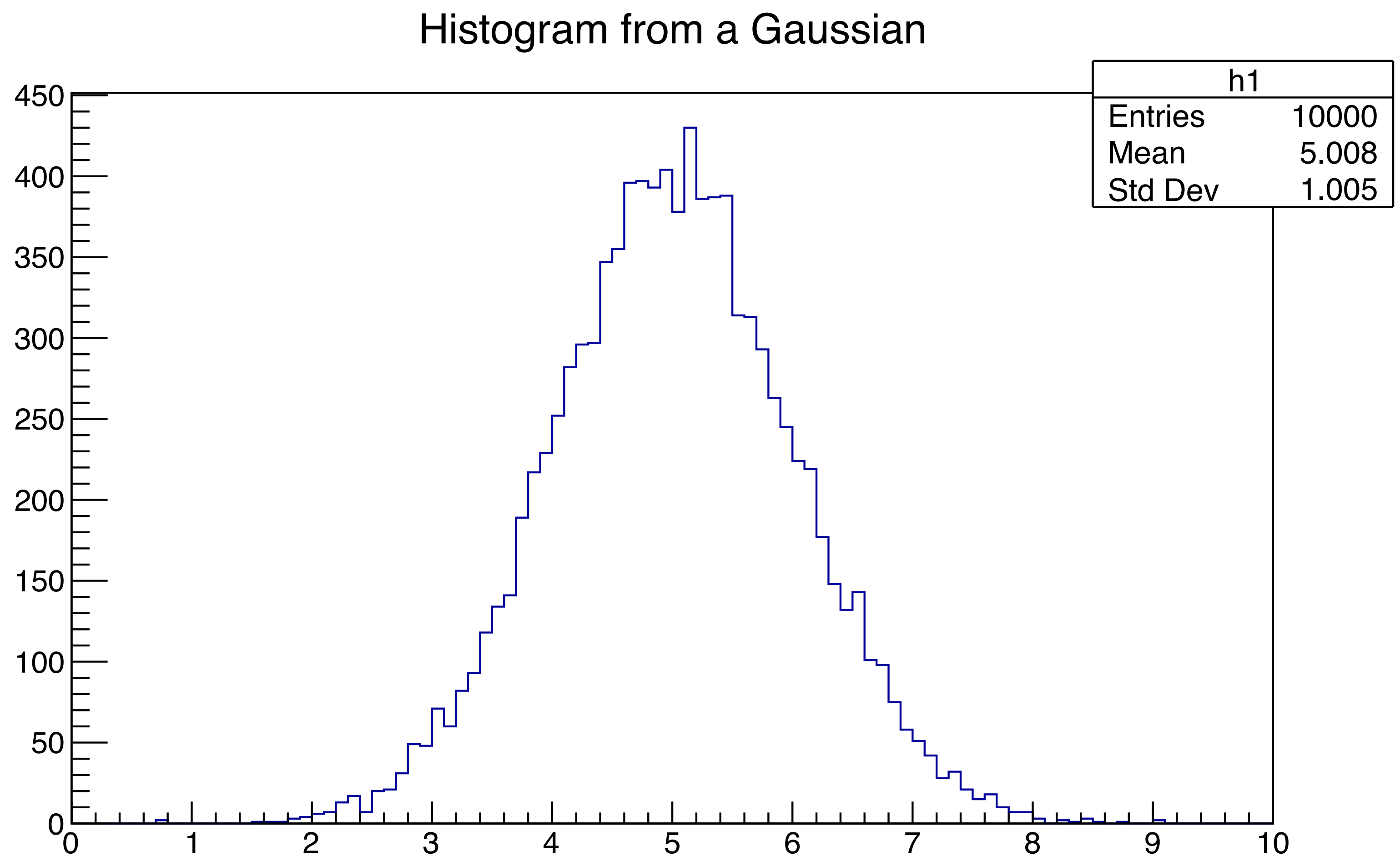
Fitting a histogram

This example fits a histogram filled randomly with 10000 entries using a Gaussian (a simplified case)

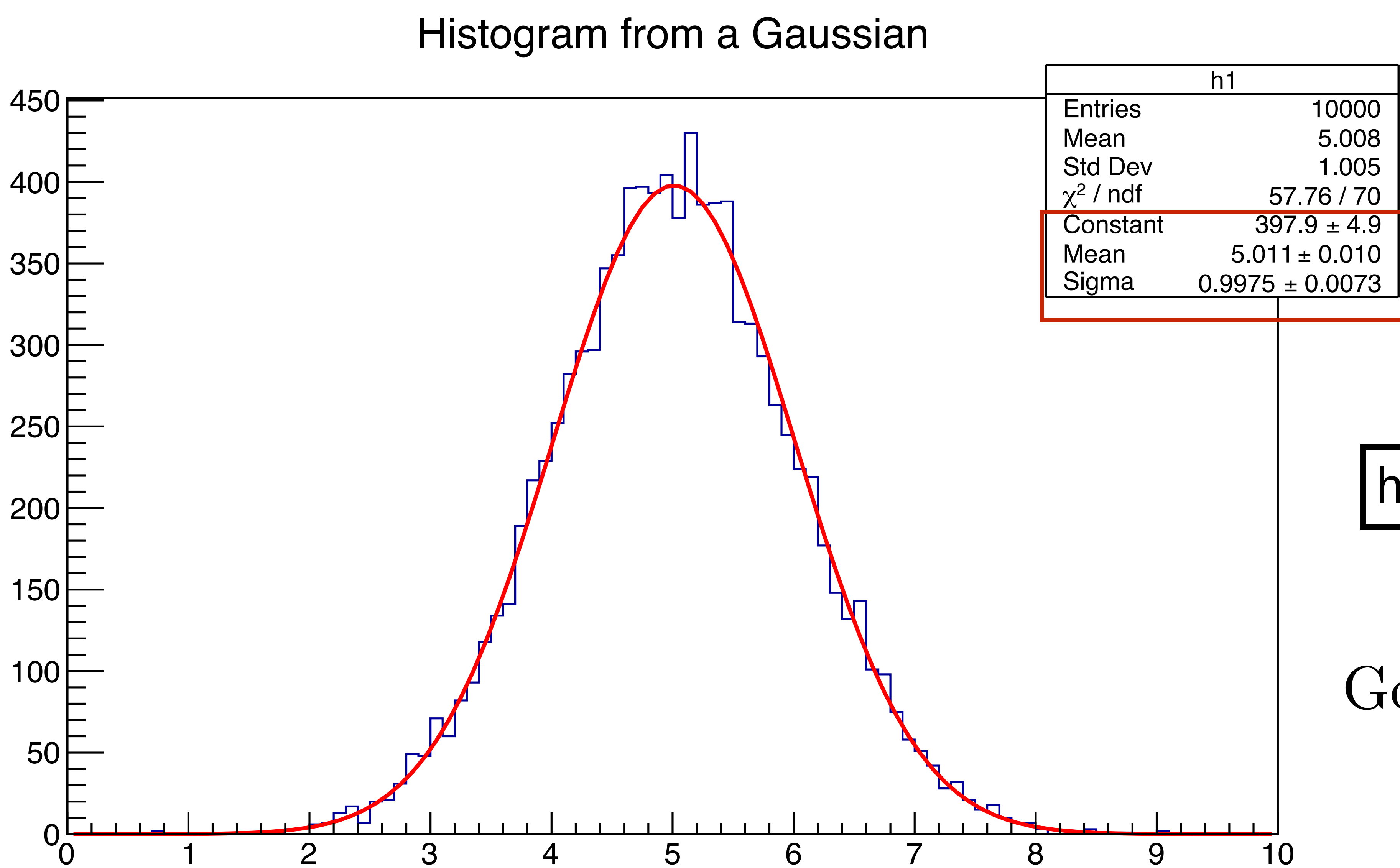
Use TH1::FillRandom() method

```
TH1D h1("h1","Histogram from a Gaussian",100,0,10);
h1.FillRandom("f2",10000);
```

Gaussian we defined before



```
root [5] h1.Fit("f2");
 FCN=57.7629 FROM MIGRAD    STATUS=CONVERGED      62 CALLS          63 TOTAL
                     EDM=3.29105e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     3.97891e+02   4.93046e+00   1.49364e-02   3.64654e-06
   2  Mean         5.01139e+00   1.00492e-02   3.75730e-05   7.94527e-03
   3  Sigma        9.97455e-01   7.30906e-03   7.36429e-06   3.97905e-03
```



fit parameters

```
h1.Fit("f2");
```

Good fit $\implies \chi^2/ndf \sim 1$

Homework

1. Install ROOT on your computer using a binary distribution or source
2. Produce a histogram randomly filled with 50000 entries with a Gaussian function of mean = 1.0 GeV and sigma = 50 MeV and 1000 bins from 0.5-1.5 GeV
3. Label the axes assuming you're plotting a mass spectrum of a resonance
4. Try two coarser binning options for histogram from the editor and compare the fit results (use the Gaussian for the fit)
5. Obtain the integral of the histogram for (mean +- 2 sigma) mass range