Beginning ROOT and Building a Gaussian Curve

Andrew Svenson

April 2, 2014

Hi, everyone! Not many people know anything about coding or ROOT, so I decided to take apart my code to help you all get started. Open source, right? Here's the main body of the code, that we'll take apart in sections.

```
#include "TH1.h"
#include "TMath.h"
#include <iostream>
#include "TRandom1.h"

using namespace std;

void gaussianRandom(){

  TRandom1 gen;
  TRandom1 descrim;
  TH1F *hist = new TH1F("hist", "Gaussian", 10, -1.0, 1.0);

  double toss;
  double decide;
  double prob;


  for(int i=0; i<100000; i++){

  toss = gen.Uniform(0,2)-1;
  prob = TMath::Gaus(toss,0,1,true);
  decide = descrim.Uniform(0,1);

  if(decide<prob){hist->Fill(toss);}

  }

  hist->Draw();
  hist->Fit("gaus");

}
```

Let's start with a skeleton. This is a small bit of code you can run through ROOT and have no errors.

```
#include "TH1.h"
#include "TMath.h"
#include <iostream>
#include "TRandom1.h"

using namespace std;

void gaussianRandom(){

........ stuff goes here ..........

}
```

All the "include" bits are tool that we'll use in our program. "TH1" is a one-dimensional histogram, "TMath" is a calculator, "iostream" is something to write to output, and "TRandom" makes a random number generator. We'll reference them later, but for now just be content that they're there.

"namespace std" is a reference to the input-output function. It's not absolutely needed since we're not writing to output, but it's useful for testing things like the number generator. If you throw in the code 'cout¡¡"Hello!"¡¡endl;' you will print the string "Hello!" and start a new line. You can replace the string, including quotes with a variable like 'cout¡¡var¡¡endl;' and print out the value of the variable.

Now we get to the meat of the program. Void means the program expects no return value. Just go with it. gaussianRandom is the name of my file (gaussianRandom.C) and ROOT expects the name of the method to be the same as the file. Again, just go with it.

Moving on, we address the programming problem. How do we make a Gaussian histogram with uniform random numbers? We need a random number generator, some logic to make a decision, and a histogram to fill. This is what the next few lines creates.

```
#include "TH1.h"
#include "TMath.h"
#include <iostream>
#include "TRandom1.h"

using namespace std;

void gaussianRandom(){

  TRandom1 gen;
  TRandom1 descrim;
  TH1F *hist = new TH1F("hist", "Gaussian", 10, -1.0, 1.0);

  double toss;
  double decide;
  double prob;

...stuff...

}
```

TRandom1 is one of four random number generators that come with ROOT. They all use different algorithms, and the internet suggested 1. You can choose others, but it makes little difference. We make two random number generators and name them "gen" and "descrim" (for discriminate. I'm a programmer, not a speller.) We call out histogram hist, because we're original. We also want to throw in some details for the histogram - the distribution goes from -1 to +1, and we want the title, "Gaussian" and we will refer to the object later as 'hist'. The rest is just syntax. The 10 in the code is at a position to choose the number of bins in our histogram. 10 means ten bins, 100 means a hundred bins, etc.

We'll also need places to store the numbers we get from our random number generators - these are toss and decide. double just means these variables accept decimal values, unlike int for integers, or bool for true/false. Prob will come into consideration later.

Now we get to logic! Let's use something called a 'for' loop, which I like to think of like a cipher, or a big wheel.

```
#include "TH1.h"
#include "TMath.h"
#include <iostream>
#include "TRandom1.h"

using namespace std;

void gaussianRandom(){

  TRandom1 gen;
  TRandom1 descrim;
  TH1F *hist = new TH1F("hist", "Gaussian", 10, -1.0, 1.0);

  double toss;
  double decide;
  double prob;


  for(int i=0; i<100000; i++){

  toss = gen.Uniform(0,2)-1;
  prob = TMath::Gaus(toss,0,1,true);
  decide = descrim.Uniform(0,1);

  if(decide<prob){hist->Fill(toss);}

  }

  hist->Draw();
  hist->Fit("gaus");

}
```

And now we have the whole program! For loop only goes from { to }. It makes a variable, called i, and starts it at zero. then this syntax will run through everything from { to } with i=0, then start over with i=1 (that's the i++ line). This will go over and over and over and over with i increasing each time, until i¡100000 is violated. Since we start at zero, we'll run until 99999 . . . 100000 turns of the 'wheel'.

Inside the loop, we run through logic for the 100000 turns. Remember our random number generator 'gen'? We ask gen to pull a number from the uniform distribution that ranges from 0-2, including decimals to a certain precision. But this is a Gaussian Distribution, and we want our values to be from -1 to +1. To do this we just subtract 1 from the range $[0, 2]$ to move it to $[-1, 1]$.

Prob is, like the name suggests, a probability. TMath opens our calculator, and gives us the probability of the number 'toss' for a gaussian distribution with mean 0, standard deviation 1, and "true" makes this a normalized distribution. Then the value of prob becomes the probability of toss being chosen from the distribution.

Now decisions! We use our second random number generator to choose a decimal number from 0 to 1, just like we got toss. Now, toss has a “prob” probability of being selected, so we say this: If decide is less than the number of prob, then we “Fill” the histogram hist with the value, and add the point to the histogram. If decide is greater than prob, then the odds have failed us and the fill statement is not executed.

Then this happens 100000 times.

After the histogram has been filled, the next statement will draw the graph, easy 1,2,3. The Fit statement fill put a line that corresponds to the Gaussian distribution ove the histogram and we can see how we’ve done.

You can code ROOT now!