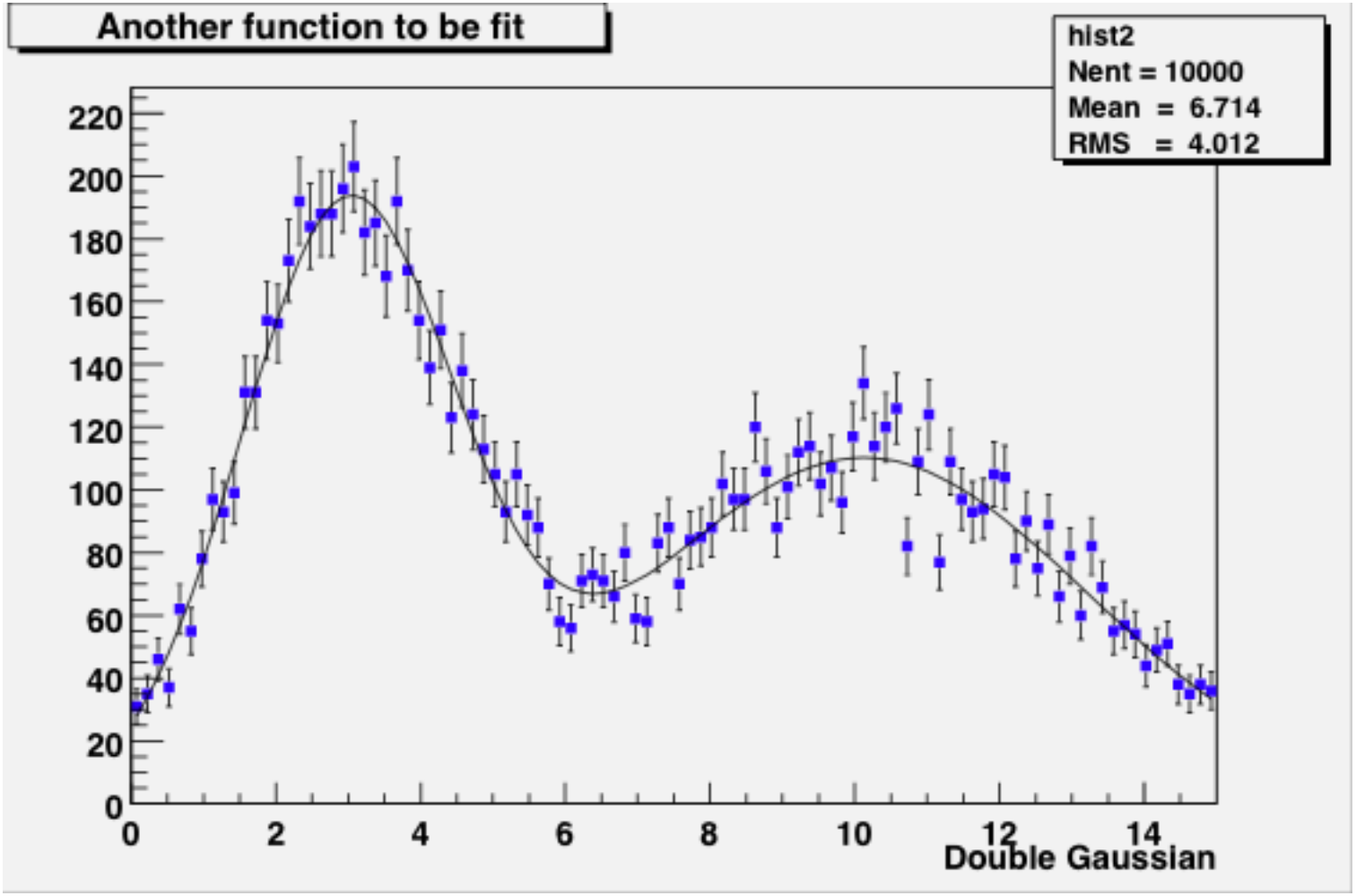
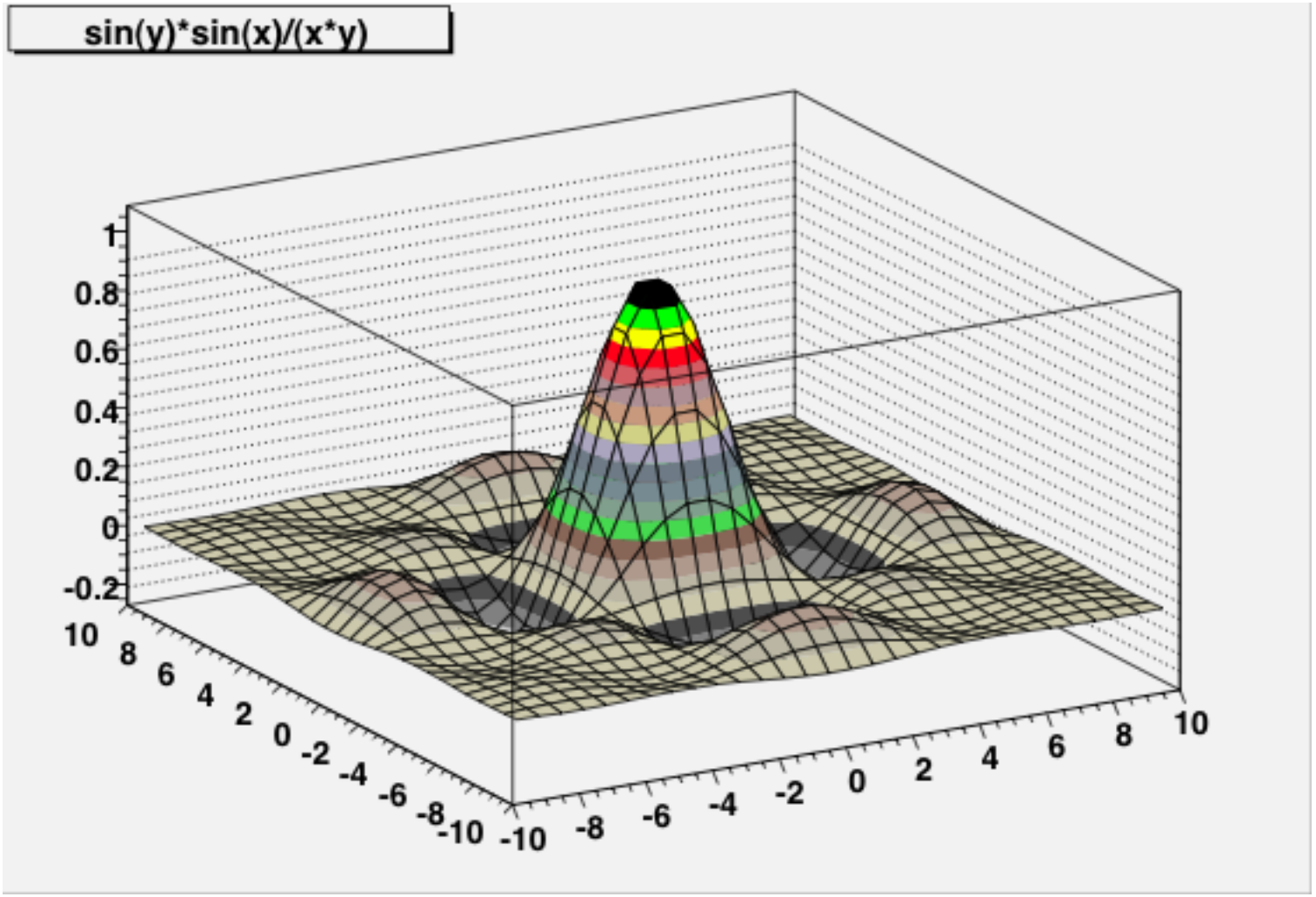
What is ROOT?
Why do we use it?

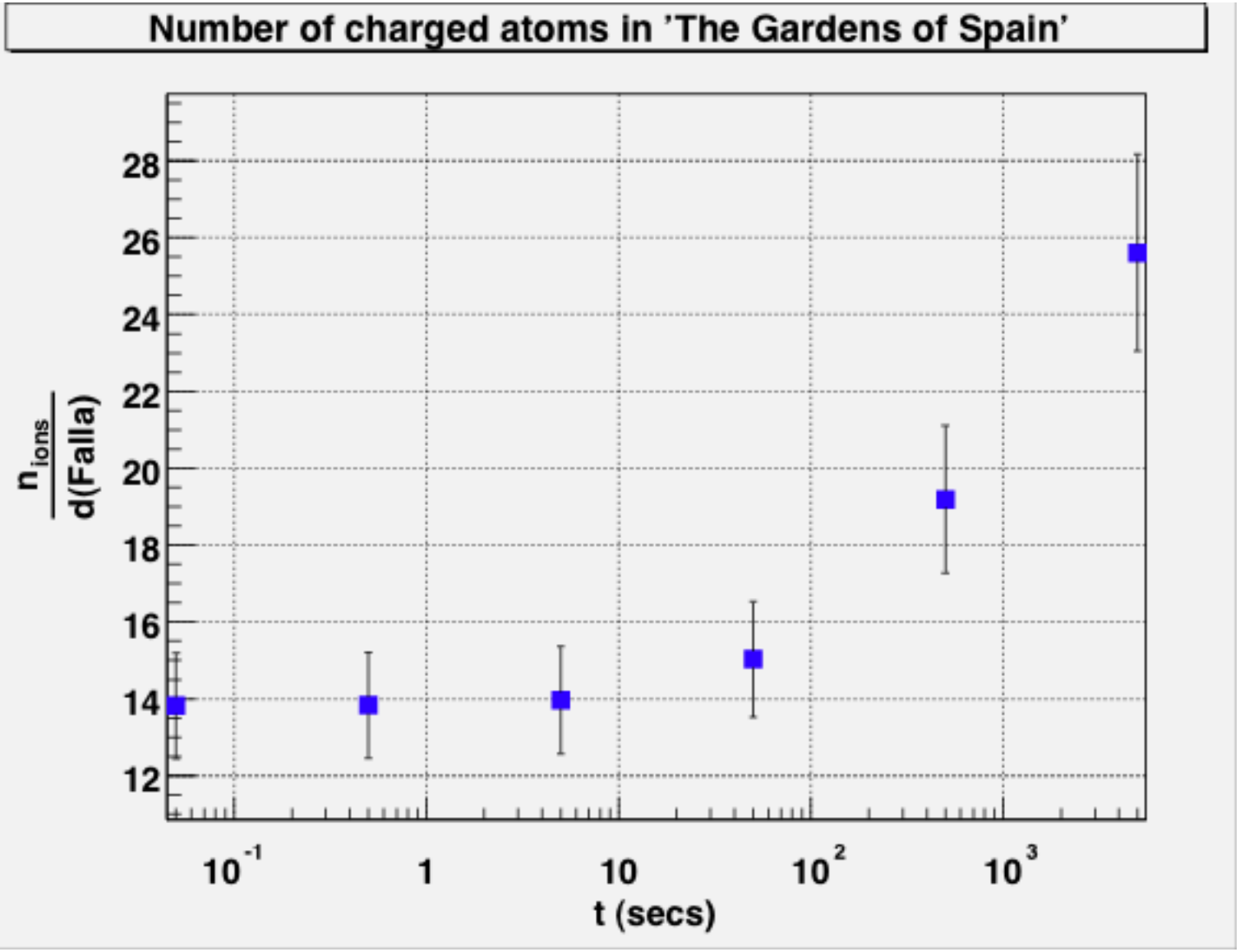
Answer:

ROOT does what physicists do:

It makes plots.


Another function to be fit
hist2
Nent = 10000
Mean = 6.714
RMS = 4.012
220
200
180
160
140
120
100
80
60
40
20
0
0
2
4
6
8
10
12
14
Double Gaussian


sin(y)*sin(x)/(x*y)
1
0.8
0.6
0.4
0.2
0
-0.2
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10



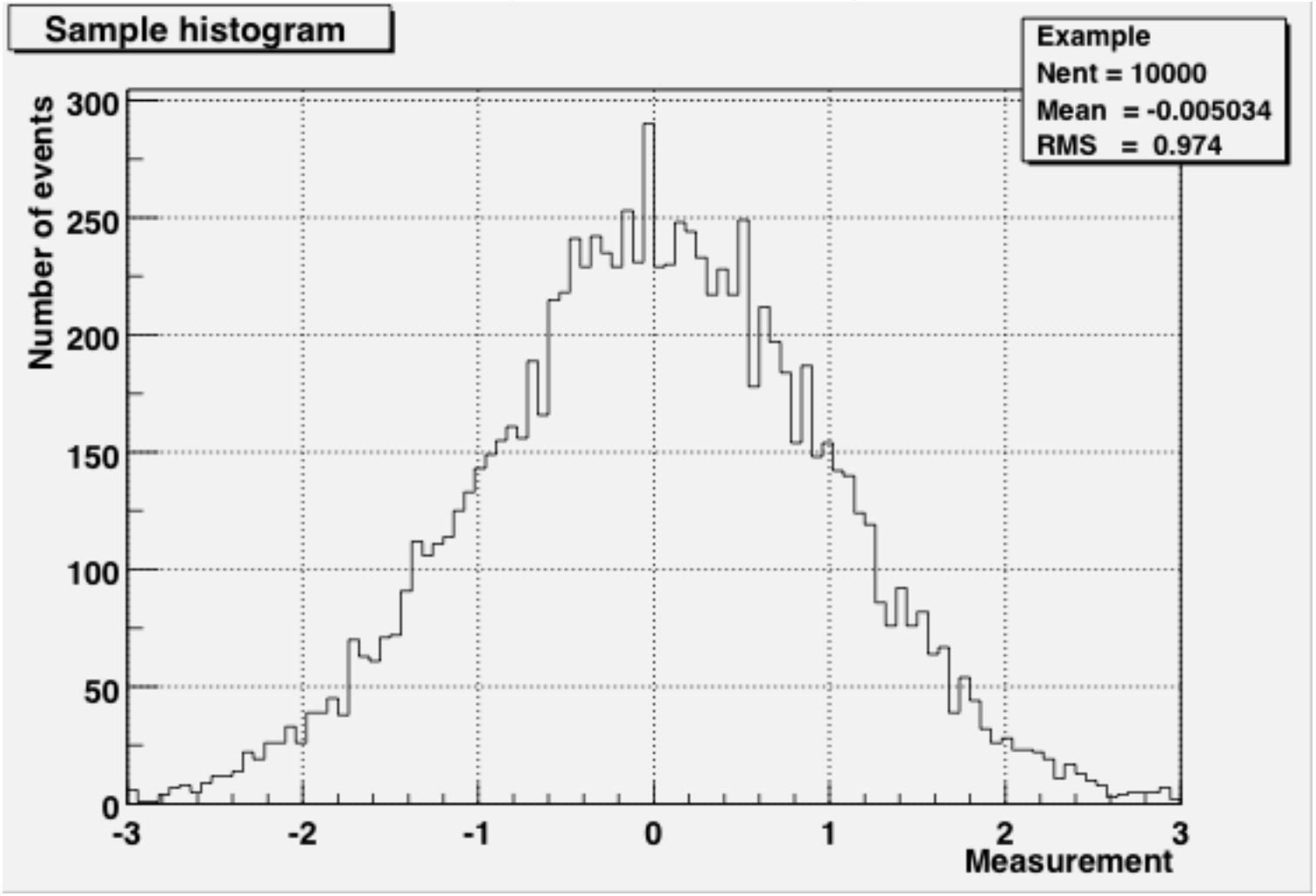
Can you spot the pun in this plot?

The typical analysis task that you will be asked to do:

Take variables in an n-tuple, perform some computations, and make histograms.

So what is a histogram, what is an n-tuple, and how do we perform the computations?

Anatomy of a histogram

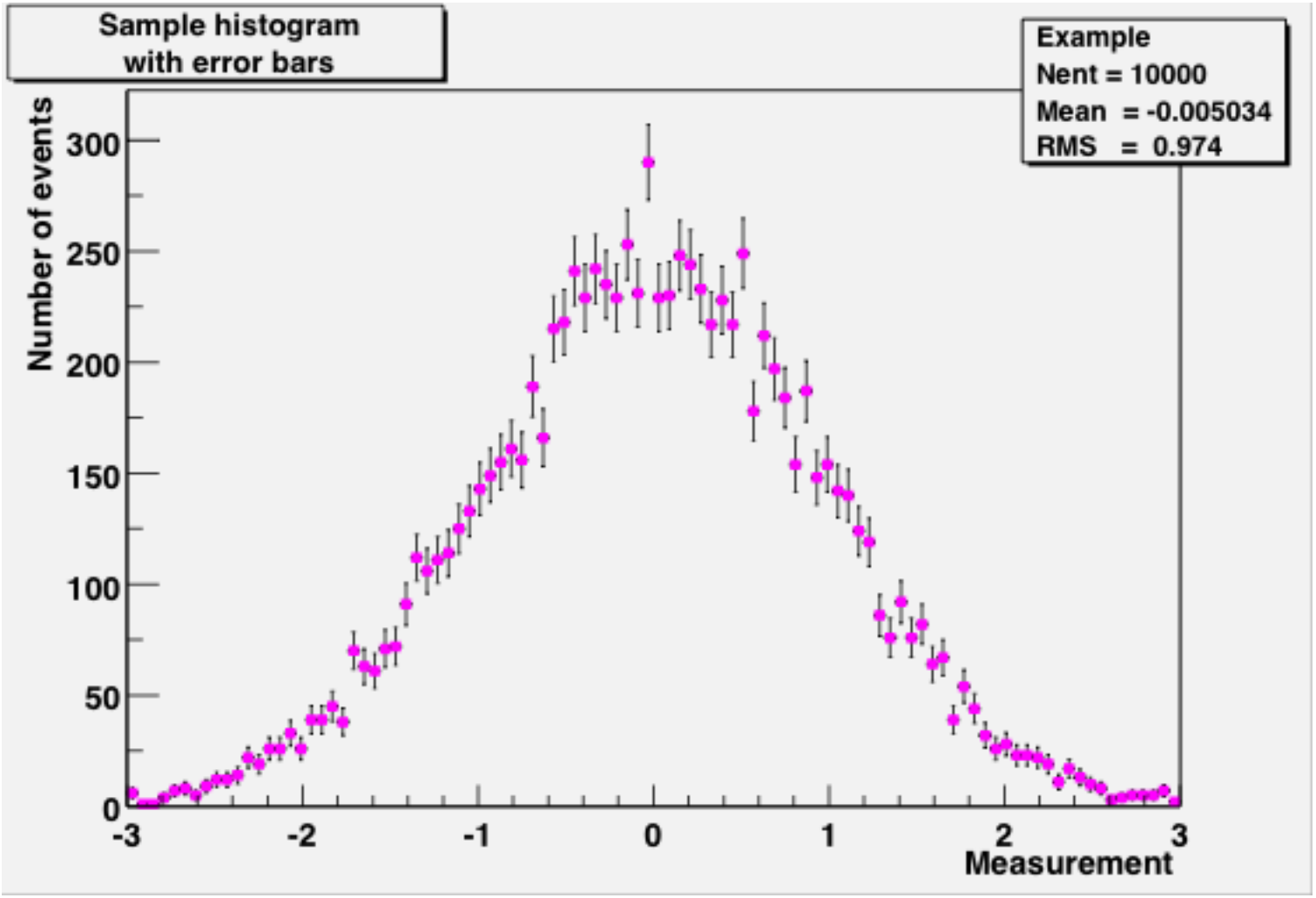


Properties of a histogram

- Name or Identifier
- Title (to be displayed on plot)
- Number of bins
- Lower bin limit
- Upper bin limit

A ROOT command that might be used to create this histogram:

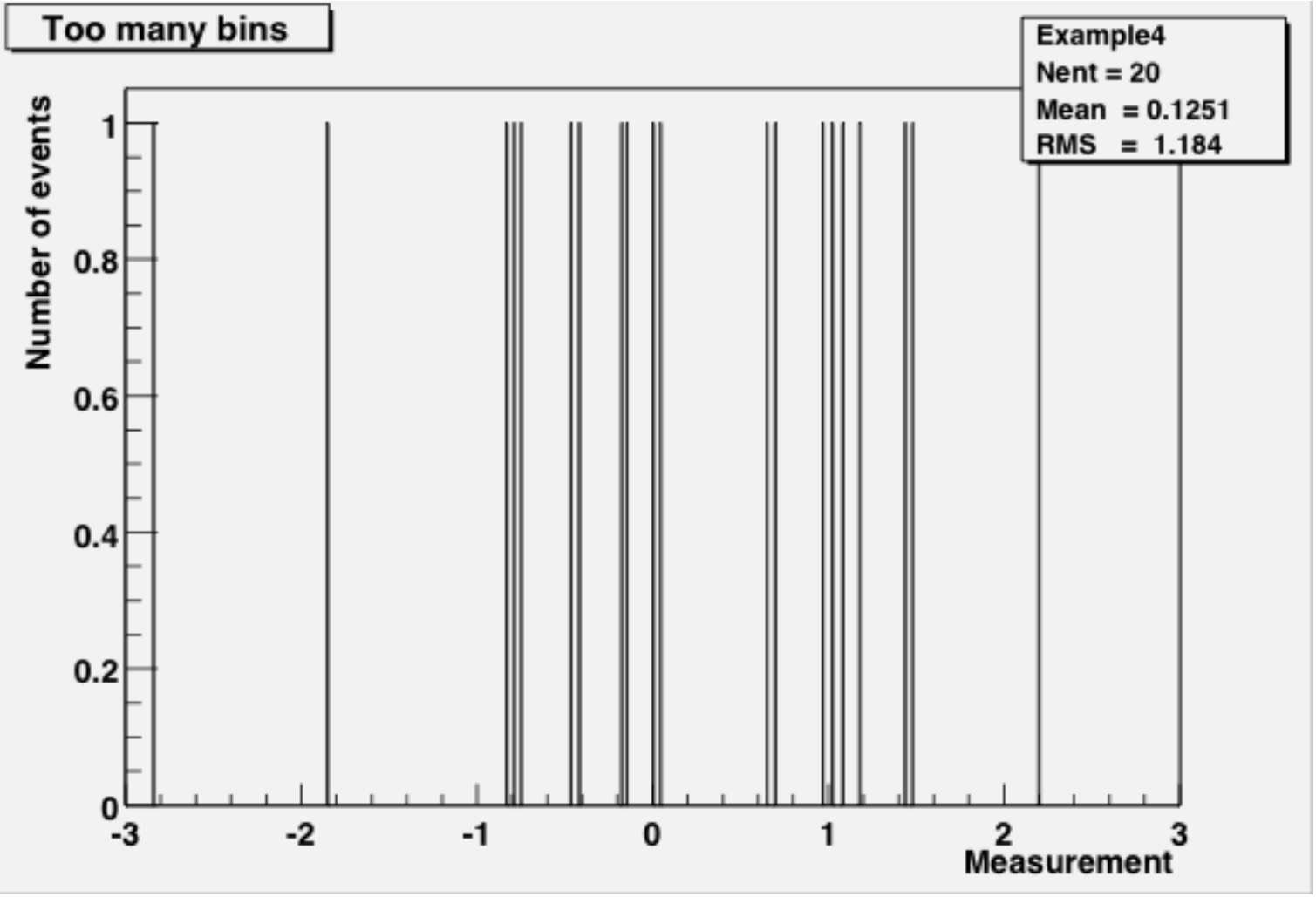
```
TH1F myPlot("Example","Sample histogram",100,-3,3)
```

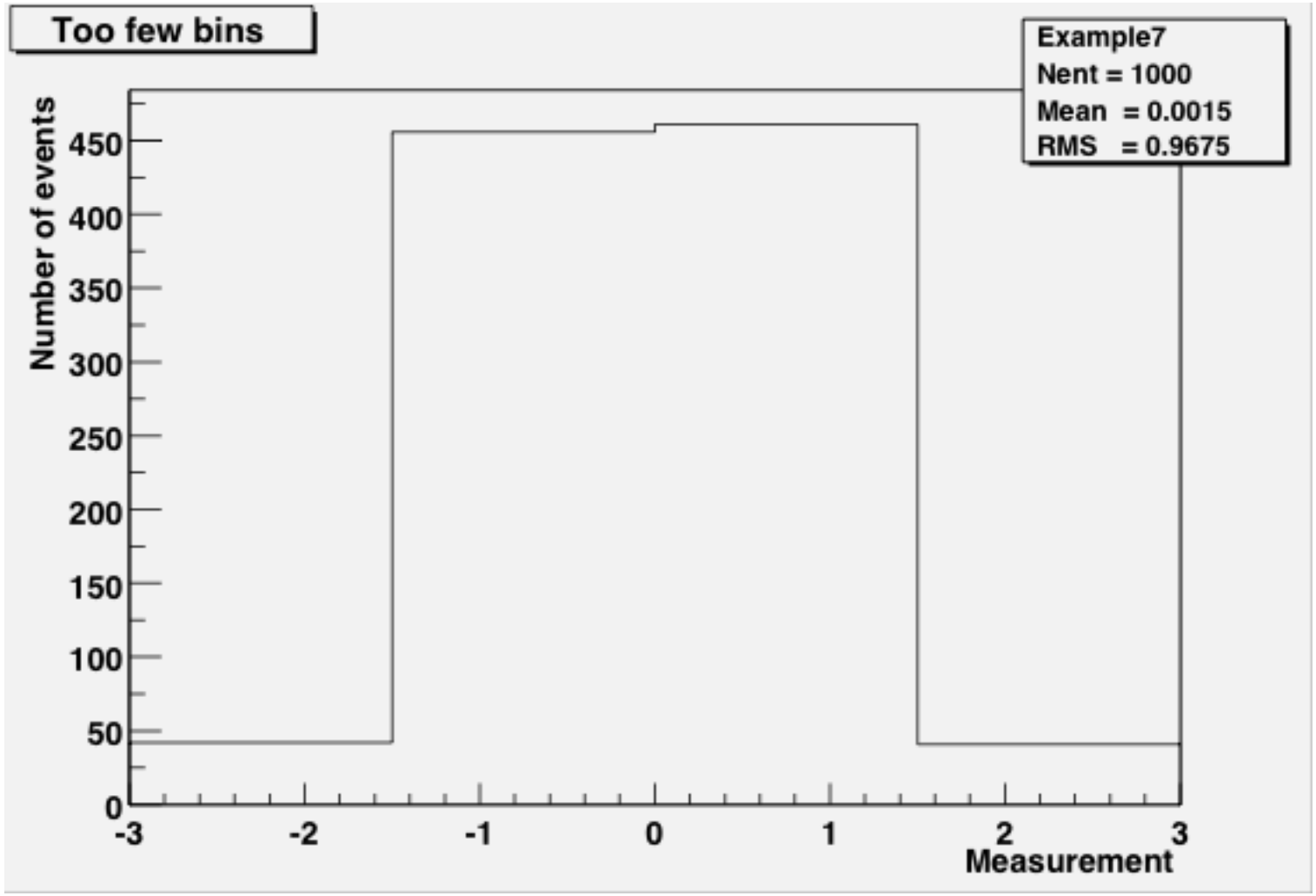


Don't forget the errors!

For simple histograms, the error in one bin is the square root of the number of events in that bin.

There's an art to histogram design...




Too few bins
Example7
Nent = 1000
Mean = 0.0015
RMS = 0.9675
Number of events
0
50
100
150
200
250
300
350
400
450
-3
-2
-1
0
1
2
3
Measurement

Anatomy of an n-tuple (a simple form of a ROOT Tree)

Row	event	ebeam	px	py	pz
0	0	150.14	14.33	-4.02	143.54
1	1	149.79	0.05	-1.37	148.60
2	2	150.16	4.01	3.89	145.69
3	3	150.14	1.46	4.66	146.71
4	4	149.94	-10.34	11.07	148.33
5	5	150.18	17.08	-12.14	143.10
6	6	150.02	5.19	7.79	148.59
7	7	150.05	7.55	-7.43	144.45
8	8	150.07	0.23	-0.02	147.78
9	9	149.96	1.21	7.27	146.99
10	10	149.92	5.35	3.98	140.70
11	11	149.88	-4.63	-0.08	147.91

An n-tuple is an ordered list of numbers.

A ROOT Tree can be an ordered list of any collections of C++ objects.

Probably you'll only be asked to work with n-tuples this summer, but in the advanced tutorial you can see what it's like to work with a ROOT Tree.

Why ROOT?

- It knows about n-tuples and histograms
  (and 4-vectors and object persistency and schema evolution
  and detector geometry and Feynmann diagrams
  and linear algebra and function-fitting and…)
- It can handle large volumes of data
  (millions of physics events; files of gigabytes->terabytes in size).
- Multi-platform (Windows, Mac, many UNIX flavors)
- It's free.

But...

- It's open-source, with a complicated design history.
- User-interface issues and documentation are often neglected.
  ROOT is not easy to use.
- You have to know some C++ in order to use ROOT effectively,
  in order to perform computations.
- What does C++ look like? Well...

```
#define Analyze_cxx
#include "Analyze.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

void Analyze::Loop()
{
//   In a Root session, you can do:
//      Root > .L Analyze.C
//      Root > Analyze t
//      Root > t.GetEntry(12); // Fill t data members with entry number 12
//      Root > t.Show();       // Show values of entry 12
//      Root > t.Show(16);     // Read and show values of entry 16
//      Root > t.Loop();       // Loop on all entries
//

//     This is the loop skeleton
//       To read only selected branches, Insert statements like:
// METHOD1:
//    fChain->SetBranchStatus("*",0);  // disable all branches
//    fChain->SetBranchStatus("branchname",1);  // activate branchname
// METHOD2: replace line
//    fChain->GetEntry(i);  // read all branches
//by  b_branchname->GetEntry(i); //read only this branch
   if (fChain == 0) return;

   Long64_t nentries = fChain->GetEntries();

   Long64_t nbytes = 0, nb = 0;
   for (Long64_t jentry=0; jentry<nentries;jentry++) {
      Long64_t ientry = LoadTree(jentry);
      nb = fChain->GetEntry(jentry);   nbytes += nb;
      // if (Cut(ientry) < 0) continue;
   }
}
```

Web Links

(the only part you should bother to write down)

All the documents you've seen (and will see) during the class today can be found at:

http://www.nevis.columbia.edu/~seligman/root-class/

ROOT and C++ links, including links to reference books on C++ and statistics, can be found at:

http://www.nevis.columbia.edu/~seligman/root-class/links.html

The Hands-on Course:

Basic Data Analysis using ROOT

ROOT basics

Over the next two days, you will learn how to:

- look up ROOT command references
- plot a function
- histogram a variable
- fit a histogram
- create C++ code for an n-tuple
- get a variable from an n-tuple
- apply cuts

-- but not necessarily in this order!

The advanced tutorial (which you may get to) includes sets of additional exercises to help turn you into a ROOT expert:

- Creating an x-y plot
- Working with large numbers of histograms
- Extracting your own n-tuples

Try to go over as much of it as you can.

A Brief ROOT Demonstration