

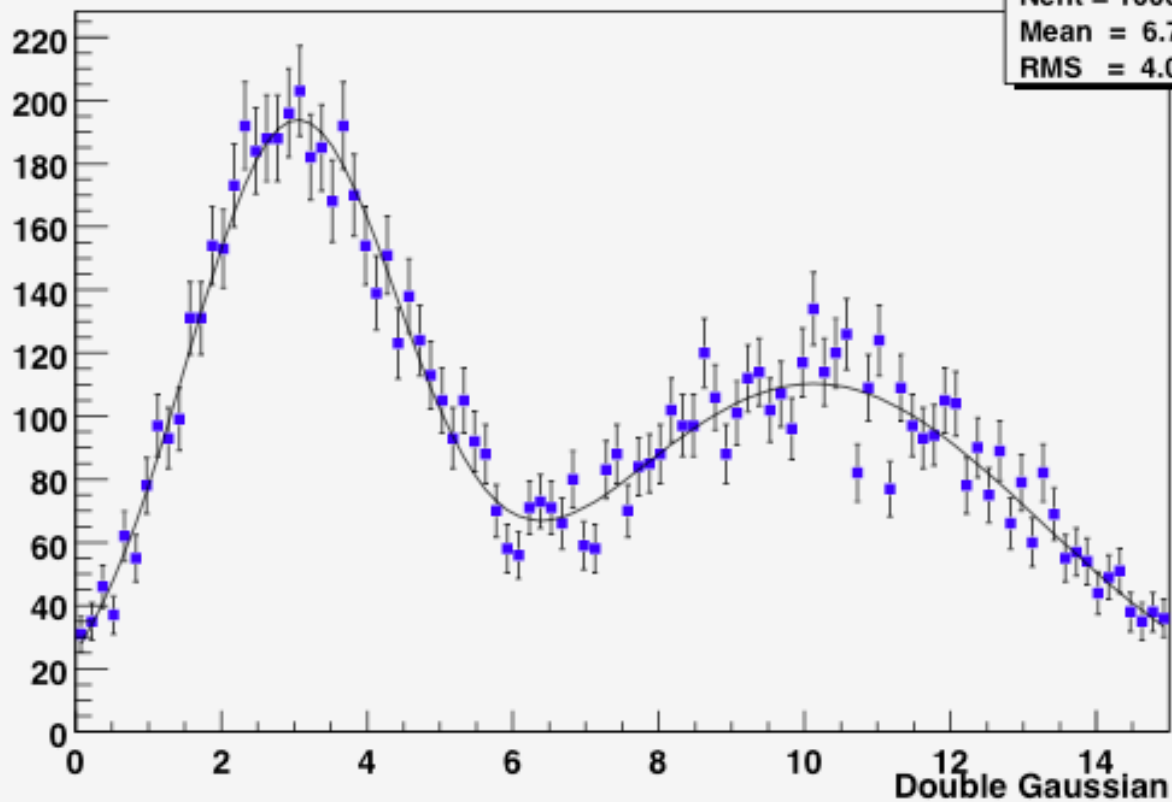
What is ROOT?  
Why do we use it?

Answer:

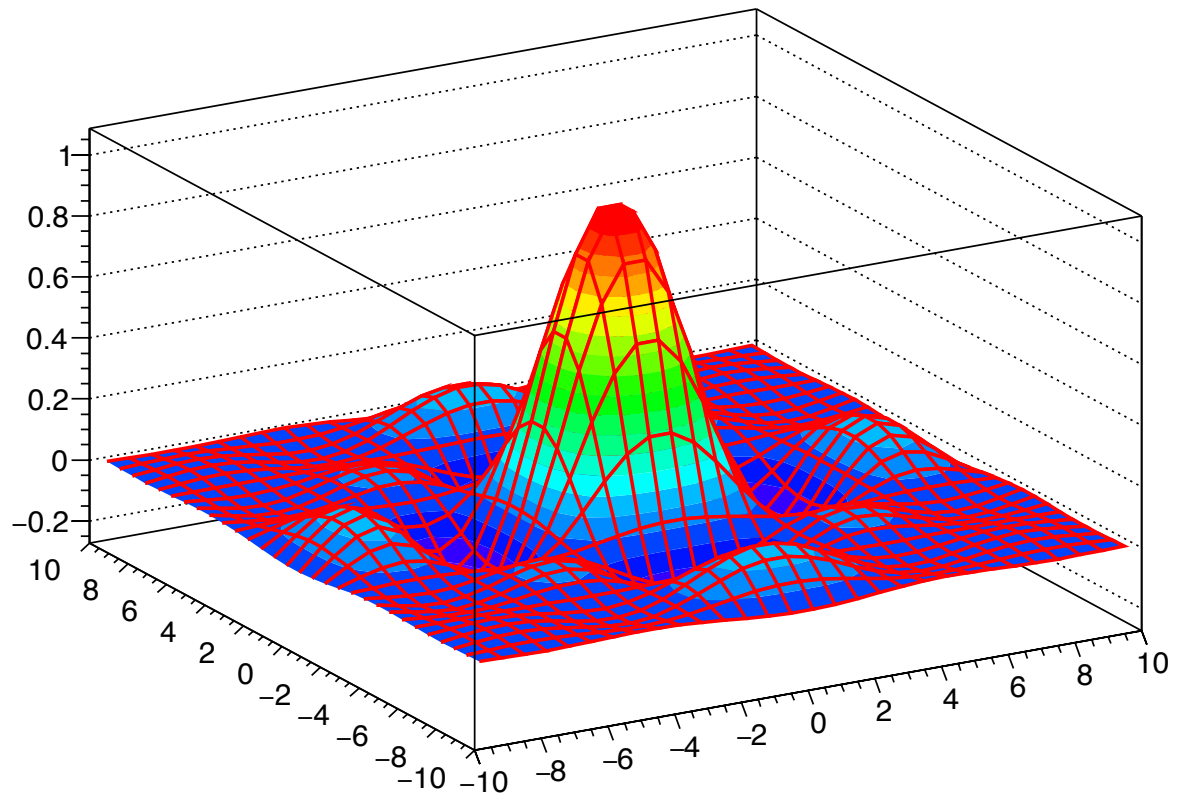
ROOT does what physicists do:

It makes plots.

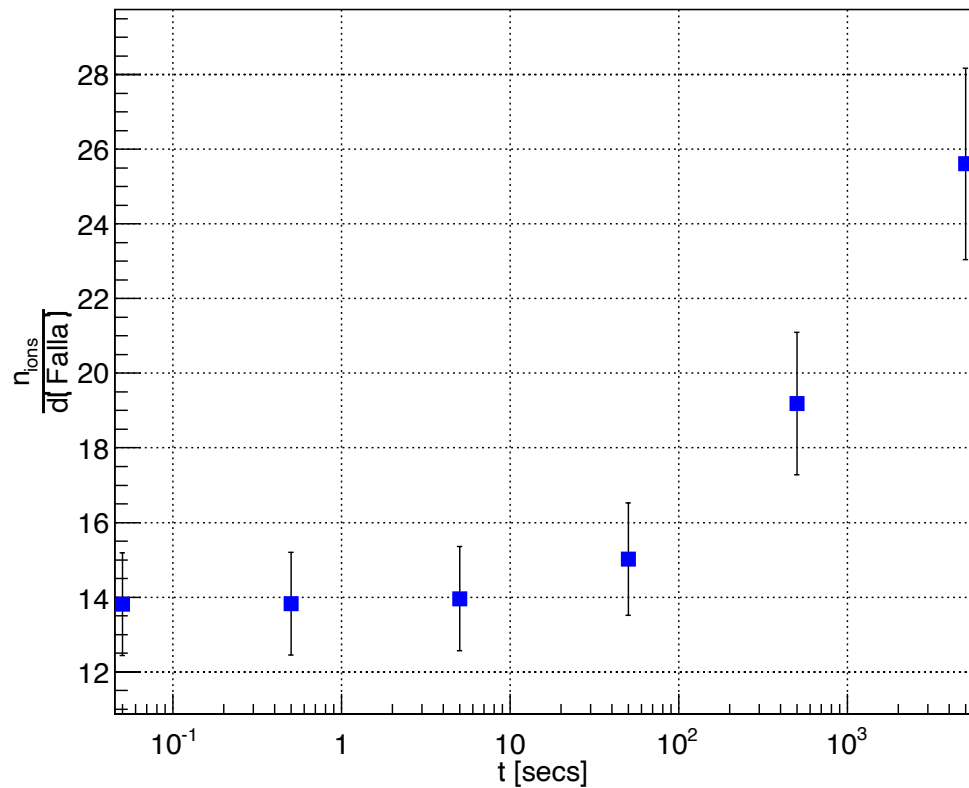
## Another function to be fit



$$\sin(x) \cdot \sin(y) / (x \cdot y)$$



Number of charged atoms in 'Nights in the Gardens of Spain'



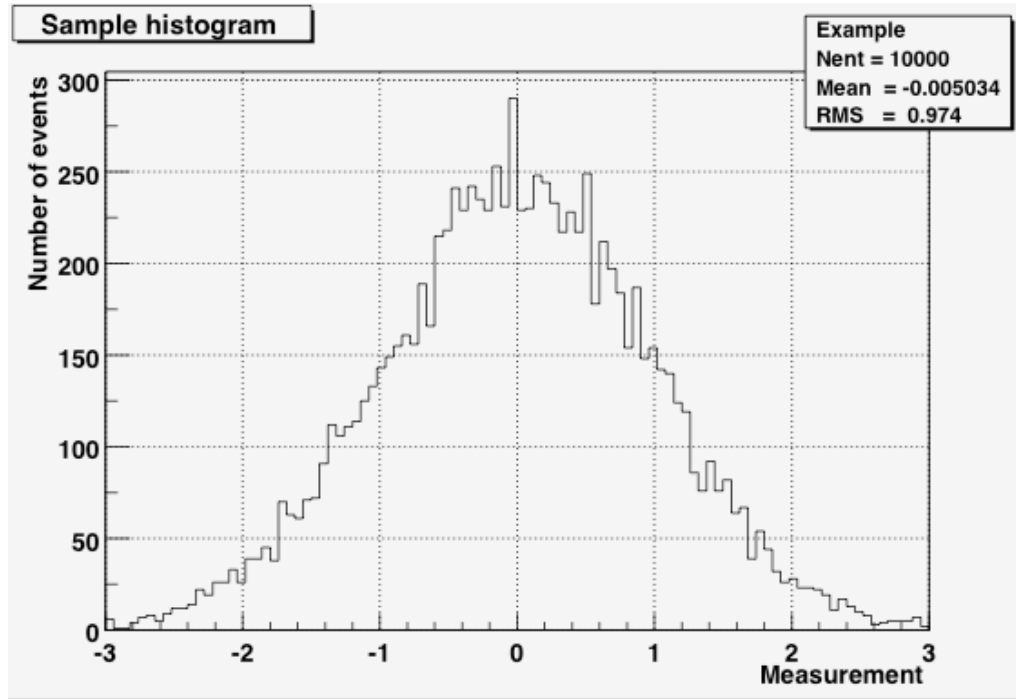
Can you spot the pun in this plot?

A basic analysis task that you could 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

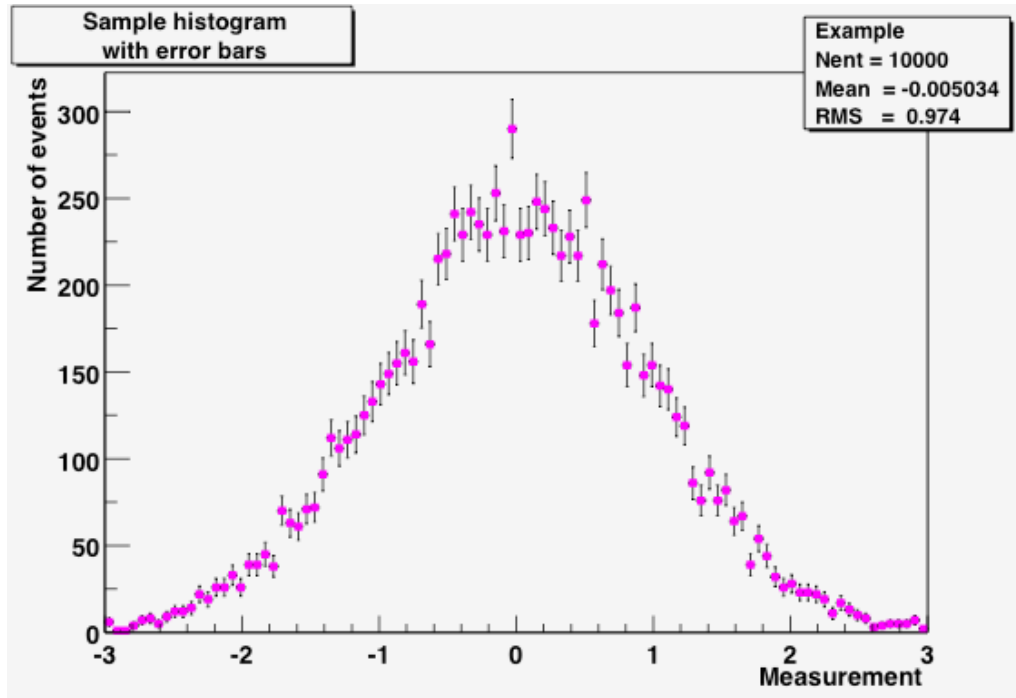


## Properties of a histogram

- 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 define 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.

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

Branches (columns) →

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

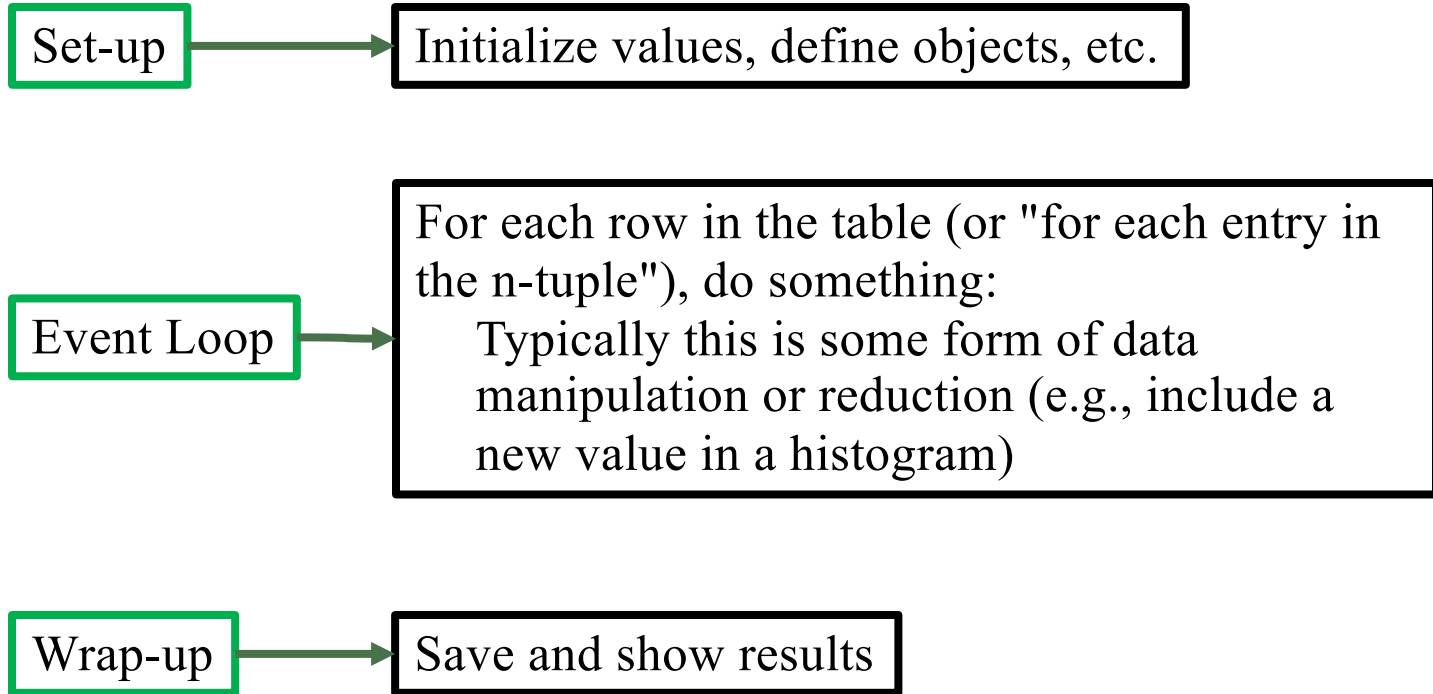
← Entries (rows)

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 Expert section of the tutorial (and in the appendix) you can see what it's like to work with a ROOT Tree.

**Computation:** A typical physics analysis task consists of:



TreeViewer, RDataFrame, and C++ macros are among the ROOT tools that can automate this process.

# Why ROOT?

- The **histogram** is a fundamental data reduction tool in high-statistics physics analyses.
- In physics, the **n-tuple** is a fundamental data format that's the result of, and the source for, histograms.
- The event loop (for example, "filling a histogram") is a fundamental **computation** in the final steps of a physics analysis.
- **ROOT** provides tools to support and automate these steps.

Examples:

- histogram arithmetic (with correct propagation of errors);
- merging, chaining, and updating n-tuples;
- interactive C++ interfaces and functions to perform complex computations;
- Python bindings to enable C++ functionality within the interpreted environment.

What is ROOT?  
Why do we use it?

Answer:

ROOT does what physicists do:

It makes plots **in a way that  
seems natural to them.**

## Wait... there's more!

- ROOT contains a wide range of tools for data analysis  
such as [4-vectors](#) and [object persistency](#) and [schema evolution](#) and [detector geometry](#) and [Feynmann diagrams](#) and [linear algebra](#) and [function-fitting](#) and [multi-variant analysis](#) and...
- It can handle large volumes of data  
billions → trillions of physics events; files of terabytes → petabytes in size; complex file structures; multi-threaded and batch processing
- Multi-platform (Windows, Mac, most UNIX flavours)
- It's free.
- Some python-based alternatives to ROOT exist (e.g., [uproot](#), [coffea](#)), but most such alternatives are wrappers around ROOT.
- According to database experts, HDF5 is better for multi-dimensional arrays used in HPC applications. For other data structures associated with physical sciences, ROOT's file format is generally more efficient.

# But...

- ROOT is open-source, with a complicated design history.
- User-interface issues and documentation are often neglected.
- It's not a pre-packaged “app.” ROOT is not easy to install.
- “ROOT is not your friend.”
- ROOT is pretty much only used in high-energy physics.
- You have to know some C++ in order to use ROOT effectively.
- What does C++ look like? Well...

```
void AnalyzeReader() {

    //***** Definition section *****

    //***** Initialization section *****

    // Open the file containing the tree.
    TFile* myFile = TFile::Open("experiment.root");

    // Create a TTreeReader for the TTree "tree1" in 'myFile'.
    TTreeReader myReader("tree1", myFile);

    // Tell the TTreeReader which branches (or columns) we'll read.
    // Because TTreeReaderValue behaves like a pointer, to access the
    // contents of the variable 'ebeam' I have to use pointer syntax;
    // e.g., "auto energy = *ebeam;".

    TTreeReaderValue<Float_t> ebeam(myReader, "ebeam");

    // Loop over all the entries of the TTree.
    while (myReader.Next()) {

        //***** Loop section *****

    }

    //***** Wrap-up section *****
}
```

# If you prefer Python, there's pyroot

```
import ROOT

# Open the file.
myfile = ROOT.TFile( 'experiment.root' )

# Retrieve the n-tuple of interest.
mychain = ROOT.gDirectory.Get( 'tree1' )
entries = mychain.GetEntriesFast()

# Create a 2D histogram
myHist = ROOT.TH2D("hist2D","chi2 vs ebeam",100,0,20,100,149,151)
myHist.GetXaxis().SetTitle("chi2")
myHist.GetYaxis().SetTitle("ebeam [GeV]")

for jentry in xrange( entries ):

    # Copy next entry into memory and verify.
    nb = mychain.GetEntry( jentry )
    if nb <= 0:
        continue

    # Fetch the variables from the entry and fill the histogram.
    chi2 = mychain.chi2
    ebeam = mychain.ebeam
    myHist.Fill(chi2,ebeam)

# Display the scatterplot.
myHist.Draw()
```

# The Hands-on Course

## Basic Data Analysis using ROOT

Over the next 2-3 days, you will learn how to:

- look up ROOT command references
  - plot a function
  - histogram a variable
  - fit a histogram
  - get a variable from an n-tuple
  - apply cuts
  - do a quick study using TreeViewer (optional)
  - use the Jupyter notebook server for quick coding
  - work with an n-tuple using RDataFrame, Python, or C++
- but not necessarily in this order!

# Wait... there's more!

The written tutorial includes intermediate topics, advanced topics, and appendices. They're there if you have the time during the tutorial sessions, or for reference later on as you work in physics.

Intermediate topics include:

- x-y plots
- Advanced histogram techniques
- Combining n-tuples across files (chains and friends)
- How to install ROOT on your laptop (if you must)

Advanced ROOT, and becoming a ROOT expert:

- Working with large numbers of histograms
- Extracting your own n-tuples

The appendices include:

- Statistics jargon that physicists use
- Programming tips
- Batch systems
- ROOT dictionaries
- git

These are expansions of additional talks, past and present, that I gave in-between ROOT tutorial sessions.

# What I won't teach you

- C++
- Python

I'm well-aware that some of the PIs are under the impression that I can teach an entire programming language in just 2-3 days.

At most you'll learn a bit about syntax; e.g.,:

- defining a variable
- writing a loop
- **if** statements

If programming is completely new to you, or you know that your research group does not use ROOT, take the RDataFrame path. You'll learn some techniques that will hopefully be useful for the analysis tools in your field of study.

# Web Links

- All the documents you've seen (and will see) during these tutorial sessions can be found here.
- This is the *only* link you need to write down from my lectures)

<https://www.nevis.columbia.edu/~seligman/root-class/>

Links to references on ROOT, C++, and statistics, can be found at:

<https://www.nevis.columbia.edu/~seligman/root-class/links.html>

# A Brief ROOT Demonstration

- Using the command line
- Using the notebook server

A couple of anecdotal points

