Computational Physics

Introduction to ROOT package

Korea University
Eunil Won

Introduction

ROOT web page

http://root.cern.ch

You will (should) visit this web site often - to survive through this class!

What is ROOT?

- What is ROOT?

The ROOT system provides a set of Object-Oriented frameworks with all the functionality needed to handle and analyze large amounts of data in a very efficient way (http://root.cern.ch).

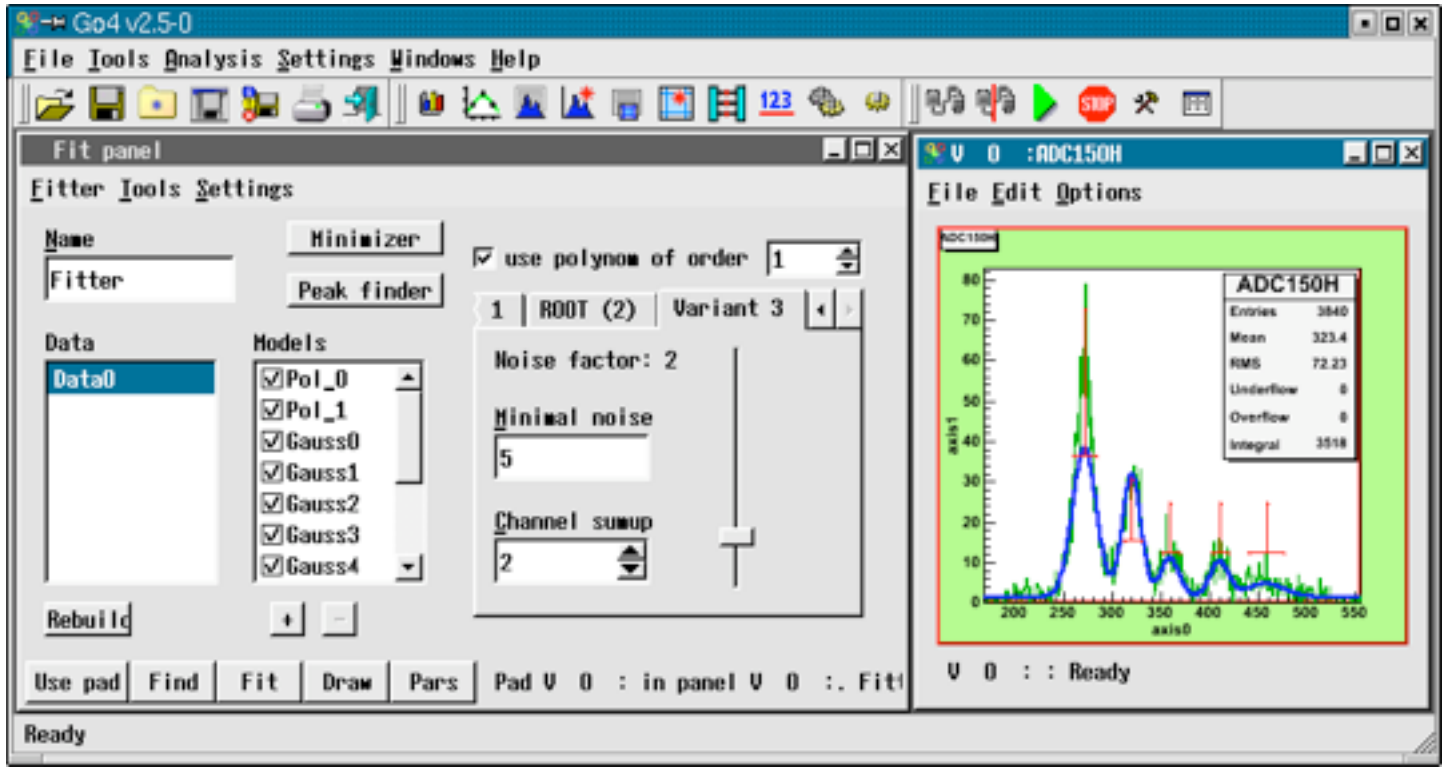
- What do we do with ROOT in this class?

We use ROOT for

- C++ interpreter (in principle one can compile your source codes with ROOT library)
- Main tool for computation
- graphical user interface

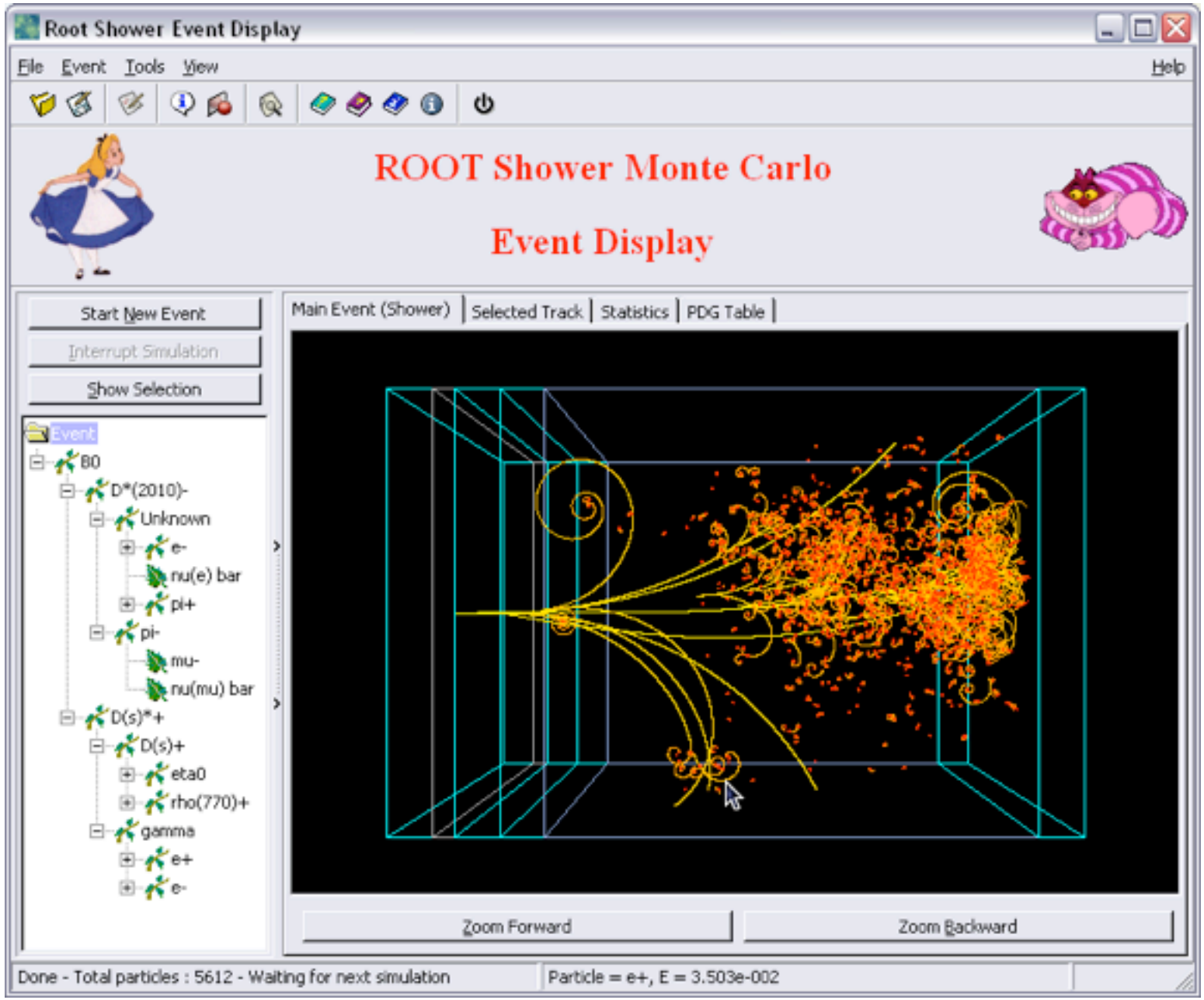
- Now, let us have a look on what ROOT can do

What can ROOT do?

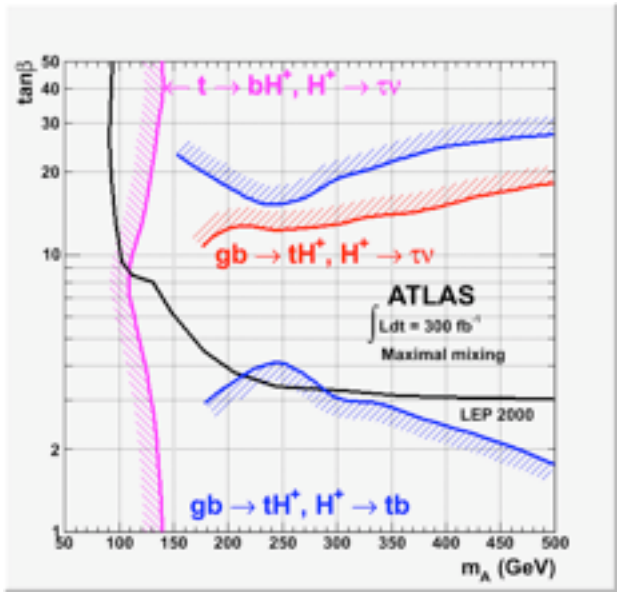


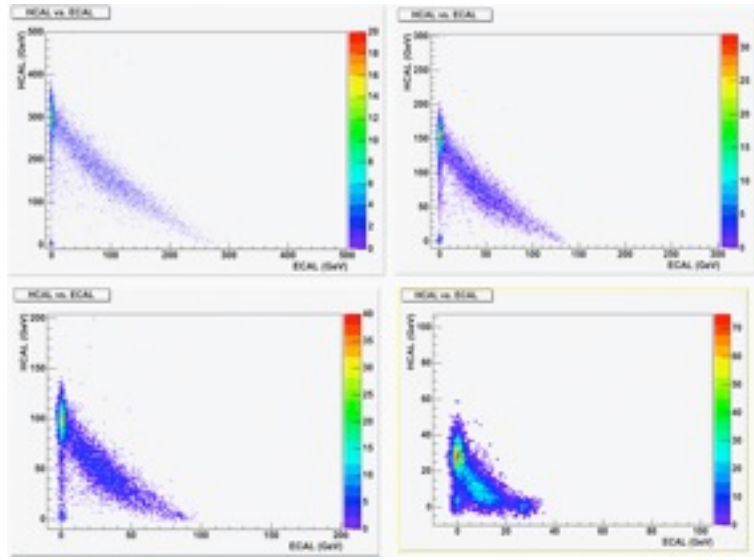
- A complex GUI environment with plots (too advanced for us)

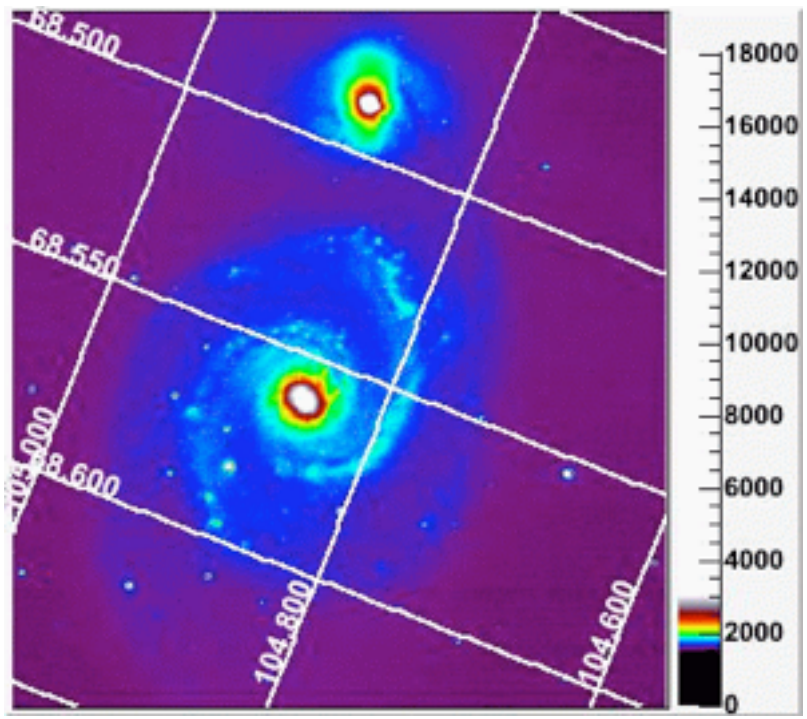
What can ROOT do?

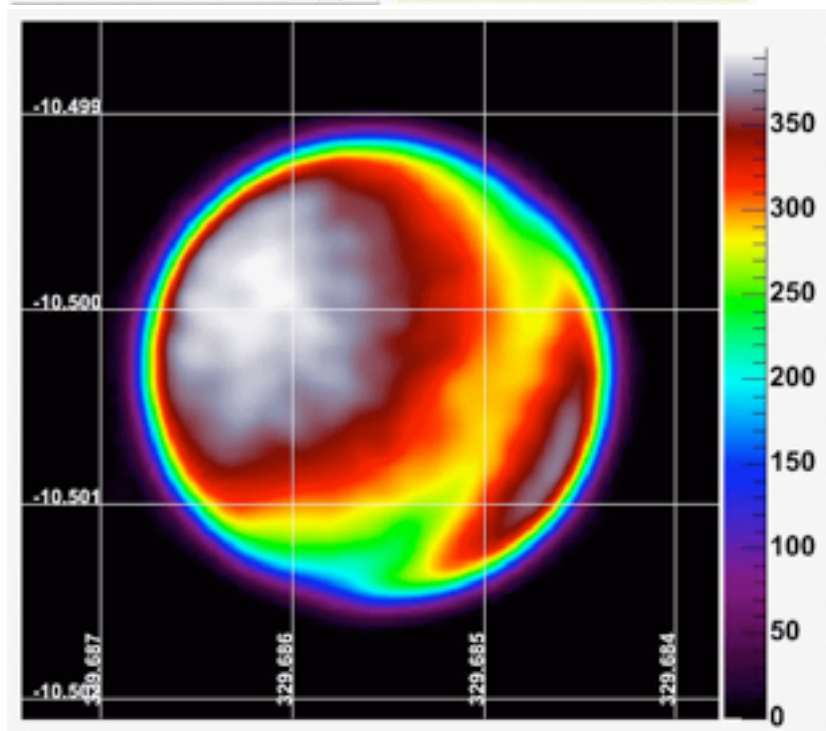


What can ROOT do?

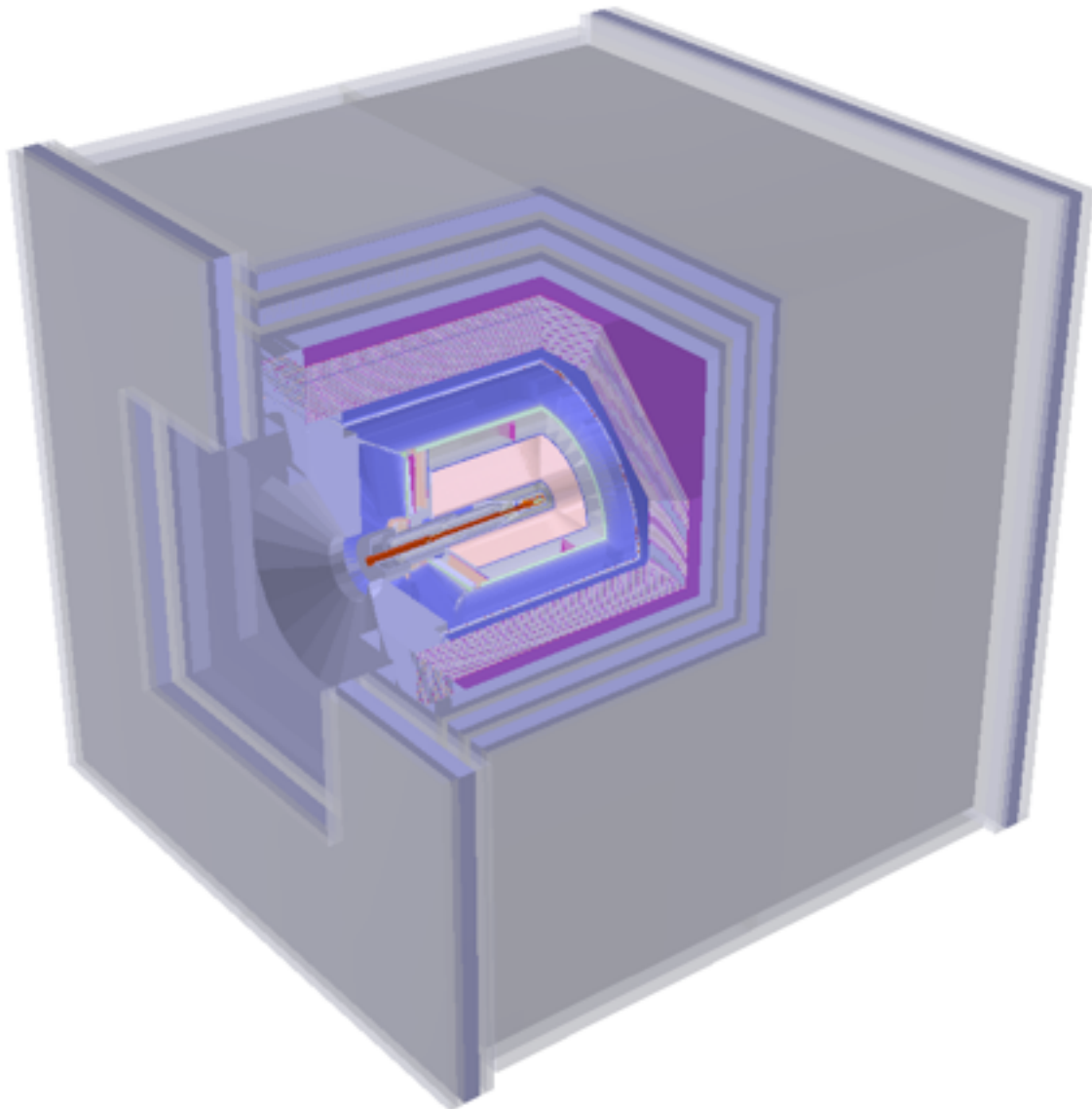








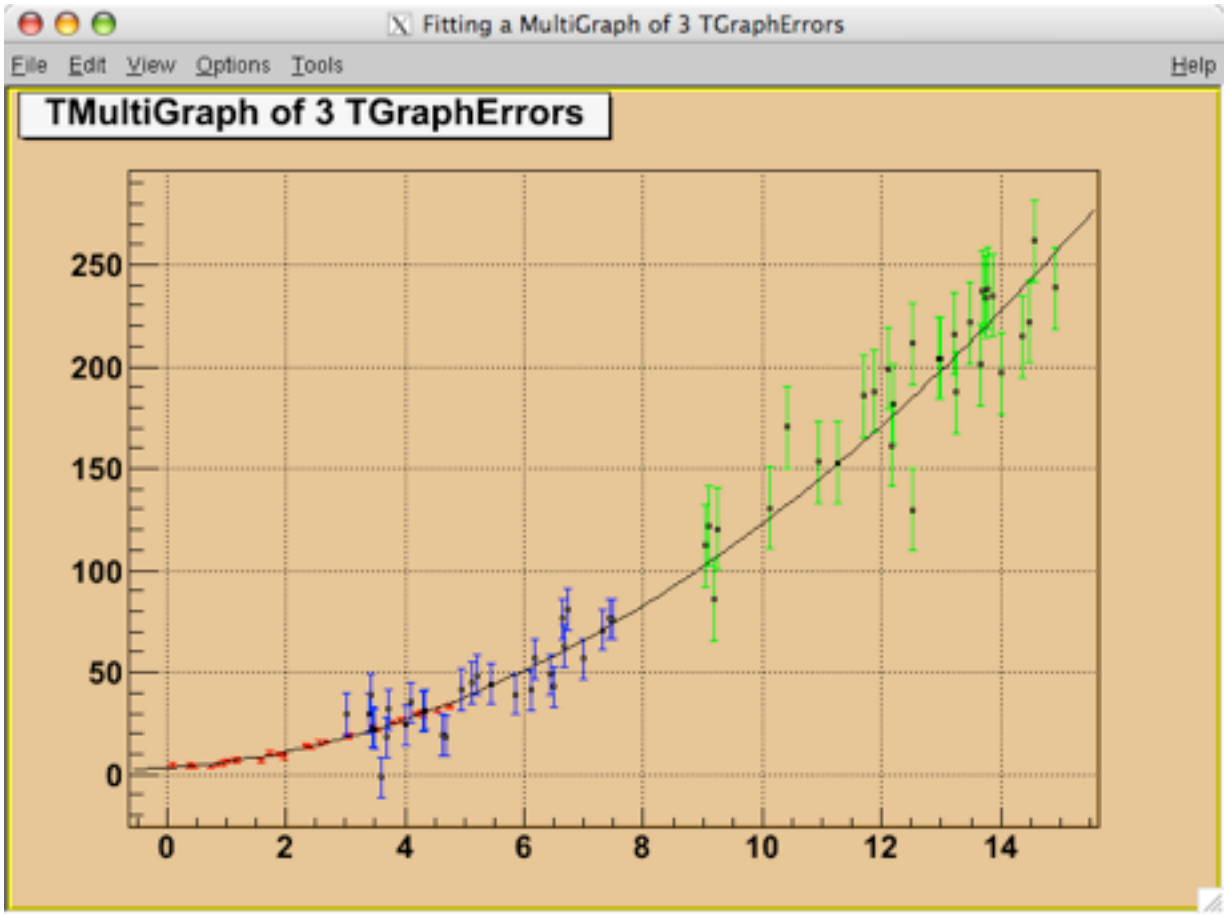
What can ROOT do?



This is the Belle detector that is located in Tsukuba, Japan

What can ROOT do?

- There are wide-range of applications that ROOT can do
  - but we stick to simple ones during our class (not relax yet!)



This is one of examples we will often see during our class

Installation

Installation (to where?)

- You should first choose your operating system to use
  - Linux, Windows, osx (Macintosh) are most common choices

- I cannot force you to use any particular one (may be legal issues?)
  - my recommendation
    If you can deal with linux : choose linux
    If you are more comfortable with M$ product, go for Windows
    If you own Mac, you can try osx (will be very much similar to linux)

Which OS I want to use?

In the computer room, all PCs can be booted with "CERN Scientific linux" - so you can use it

In the computer room, all PCs can also be booted with Windows 7

If you do not own Mac, then it is difficult. There is a way to install osx in PC (Google hackintosh...)

If you own a laptop and want to install linux, you need to repartition your disk etc - TA will help you(?)

Installation (to linux/osx)

- Download the latest stable version of ROOT package from http://root.cern.ch
  - As of 2010 Jan., it is **ROOT 5.26/00**
  - I have `root_v5.26.00.macosx105-i386-gcc-4.0.tar.gz`

- Type in

```
$ tar zxvf root_v5.26.00.macosx105-i386-gcc-4.0.tar.gz
$ mv root $HOME/root
```

- Edit your $HOME/.bashrc (or $HOME/.cshrc, $HOME/.tcshrc depending on your shell) and add following lines into it

```
export ROOTSYS=$HOME/root
export PATH=$PATH:$ROOTSYS/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
```

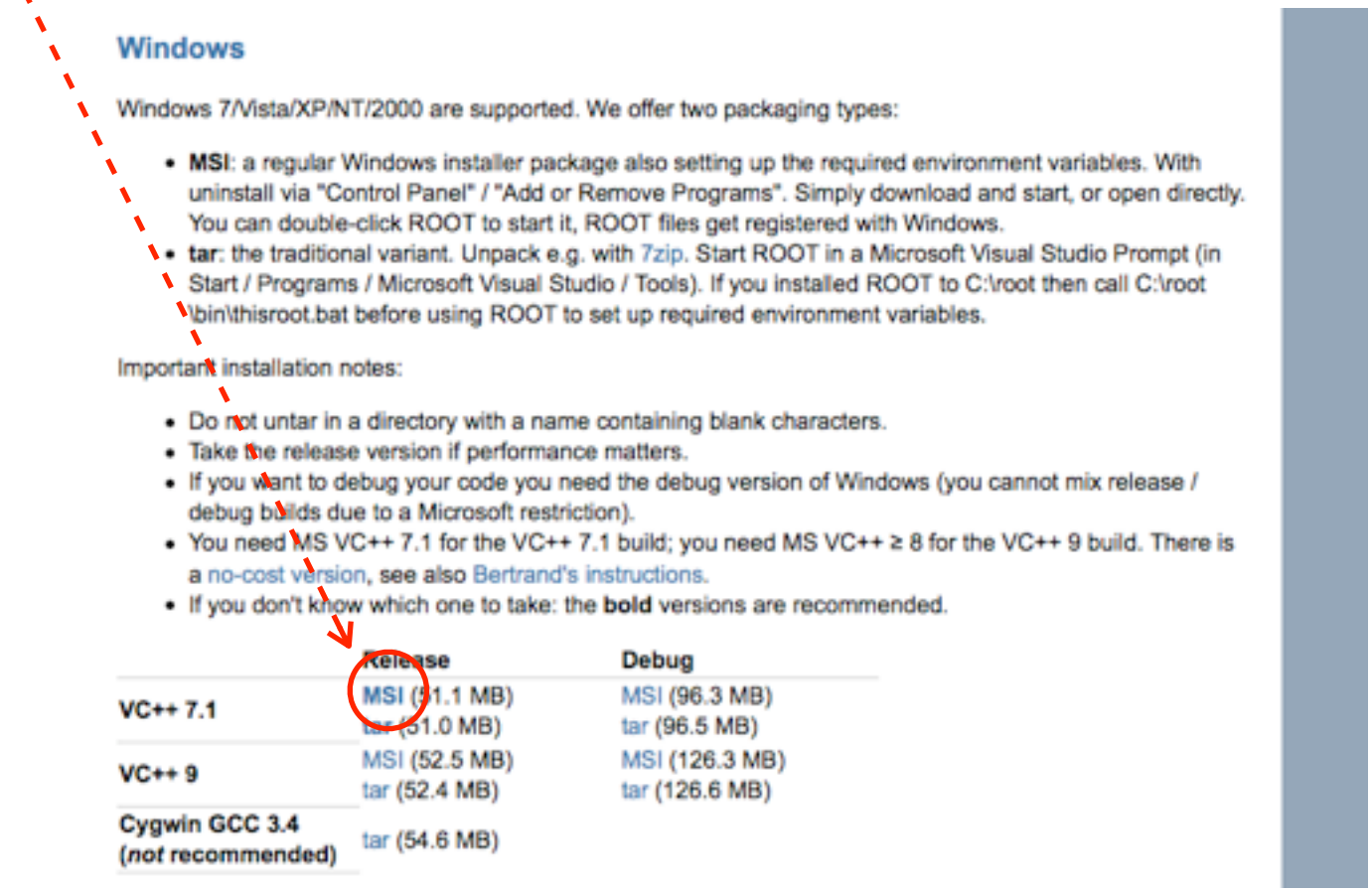
- If everything is OK, then you would get

```
$ which root
/Users/eunil/root/bin/root
```

or something similar to it

Installation (to M$ Windows)

- Goto http://root.cern.ch
  - Click "Download", "Pro" and scroll down to "Windows" block
  - Click on MSI of VC++ 7.1



Windows

Windows 7/Vista/XP/NT/2000 are supported. We offer two packaging types:

- **MSI**: a regular Windows installer package also setting up the required environment variables. With uninstall via "Control Panel" / "Add or Remove Programs". Simply download and start, or open directly. You can double-click ROOT to start it, ROOT files get registered with Windows.
- **tar**: the traditional variant. Unpack e.g. with 7zip. Start ROOT in a Microsoft Visual Studio Prompt (in Start / Programs / Microsoft Visual Studio / Tools). If you installed ROOT to C:\root then call C:\root \bin\thisroot.bat before using ROOT to set up required environment variables.

Important installation notes:

- Do not untar in a directory with a name containing blank characters.
- Take the release version if performance matters.
- If you want to debug your code you need the debug version of Windows (you cannot mix release / debug builds due to a Microsoft restriction).
- You need MS VC++ 7.1 for the VC++ 7.1 build; you need MS VC++ ≥ 8 for the VC++ 9 build. There is a no-cost version, see also Bertrand's instructions.
- If you don't know which one to take: the **bold** versions are recommended.

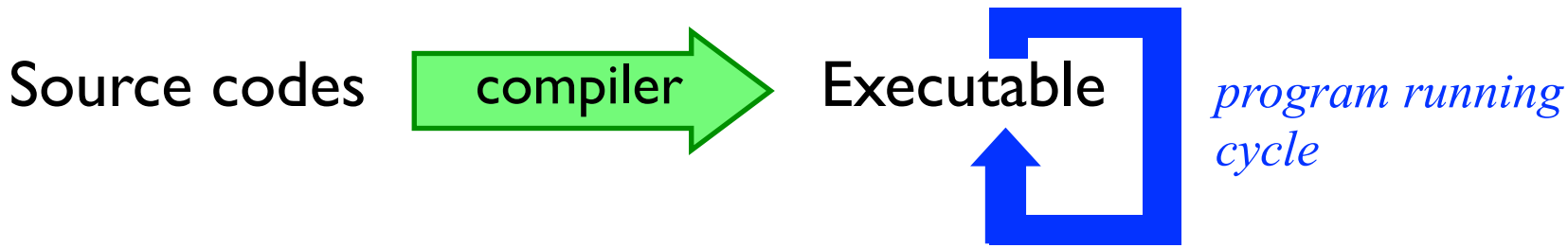
	Release	Debug
VC++ 7.1	**MSI** (51.1 MB) tar (51.0 MB)	MSI (96.3 MB) tar (96.5 MB)
VC++ 9	MSI (52.5 MB) tar (52.4 MB)	MSI (126.3 MB) tar (126.6 MB)
Cygwin GCC 3.4 (*not* recommended)	tar (54.6 MB)	

Running the ROOT

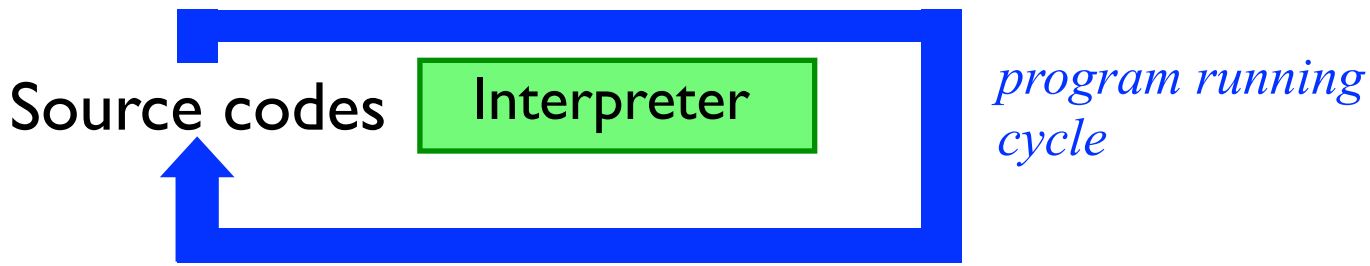
ROOT CINT

- Compiler vs Interpreter

Compilers: Translate a source (human-writable) program to an executable (machine-readable) program



Interpreters: Convert a source program and execute it at the same time.



ROOT can be used in both ways but we use "interpreter" during our class : so we use ROOT CINT (if your program gets slower, we can compile it for faster running)

Running the ROOT

- Starting the ROOT program

```
eunil$ root
  ****************************************
  *                                      *
  *        W E L C O M E  to  R O O T    *
  *                                      *
  *   Version   5.26/00  14 December 2009   *
  *                                      *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch         *
  *                                      *
  ****************************************

ROOT 5.26/00 (trunk@31882, Dec 14 2009, 20:18:36 on macosx)

CINT/ROOT C/C++ Interpreter version 5.17.00, Dec 21, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
root [0] .q
eunil$
```

How to get out of ROOT prompt

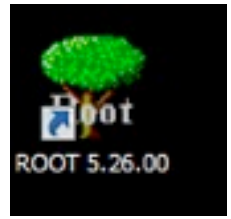
- Which version you are running with?

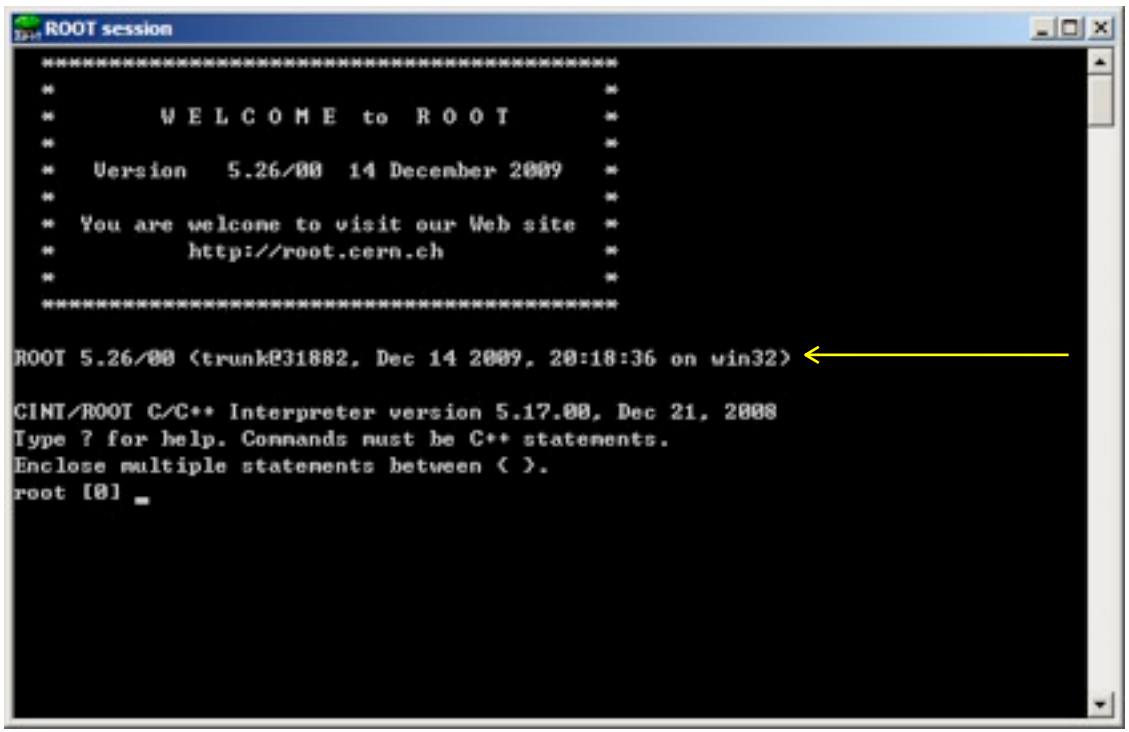
```
root [1] gROOT->GetVersion()
(const char* 0x170c240)"5.26/00"
```

What is gROOT->GetVersion()?

Running the ROOT under Windows xx

- Double-click ROOT icon





This tells you that you are running on M$ Windows xx (mine is Windows 7)

- Which version you are running with?

```
root [1] gROOT->GetVersion()
(const char* 0x170c240)"5.26/00"
```

What is gROOT->GetVersion()?

Running the ROOT

- hello world program with ROOT

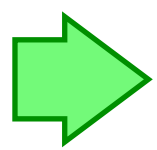
```
root -l hello.cc
root [0]
Processing hello.cc...
Error: Function hello() is not defined in current scope  :0:
*** Interpreter error recovered ***
root [1]
```

It makes error and stop execution

We are invoking ROOT CINT interpreter when we run ROOT

- So, the "`main()`" function is already in ROOT CINT
- All you have to do is the following

```
#include <iostream>
main()
{
  //
  // c++ version of remark
  //
  std::cout << "Hello, world" << std::endl;
}
```



```
#include <iostream>
hello()
{
  //
  // c++ version of remark
  //
  std::cout << "Hello, world" << std::endl;
}
```

Now it works!

```
eunil$ root -l hello.cc
root [0]
Processing hello.cc...
Hello, world
(int)(-1602173696)
root [1]
```

What is this? Can you figure it out?

Examples

- There are tons of built-in examples

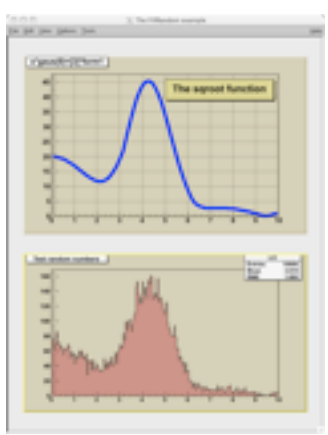
```
ls /Users/eunil/root/tutorials/
MyTasks.cxx       fft             graphics        image       physics          roofit          ruby        unuran
README            fit             graphs          io          proof            roostats        spectrum    xml
benchmarks.C      foam            gui             math        pyroot           rootalias.C     splot
cont              gallery.root    hist            matrix      pythia           rootenv.C       sql
demos.C           geant3tasks.C   hsimple.C       mc          quadp            rootlogoff.C    tasks.C
demoshelp.C       geom            html            mlp         regexp.C         rootlogon.C     thread
eve               gl              htmlex.C        net         regexp_pme.C     rootmarks.C     tree
```

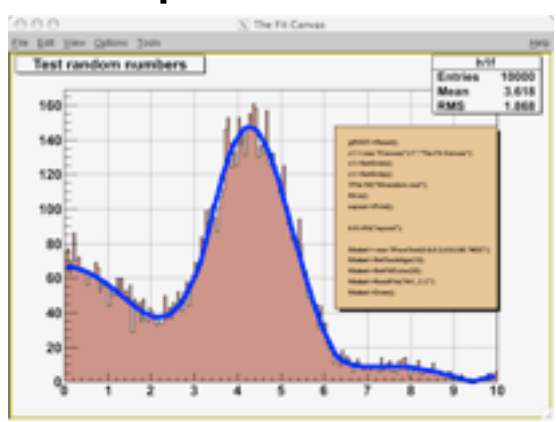
If you type in

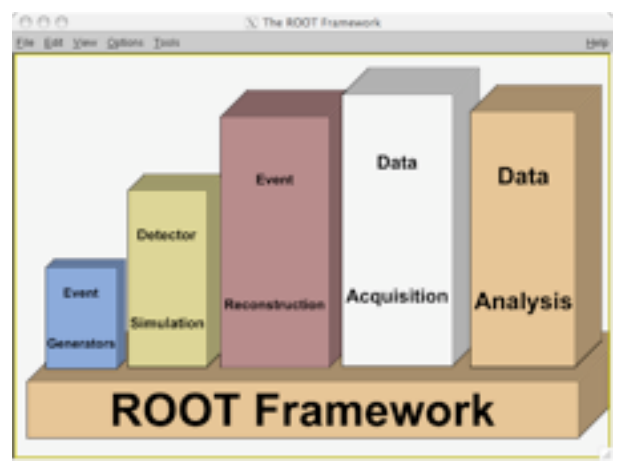
```
eunil$ root -l /Users/eunil/root/tutorials/demos.C
root [0]
Processing /Users/eunil/root/tutorials/demos.C...
root [1]
```

Go through one by one and try to understand the source codes

a window like this will pop-up and by clicking on each, you execute individual examples







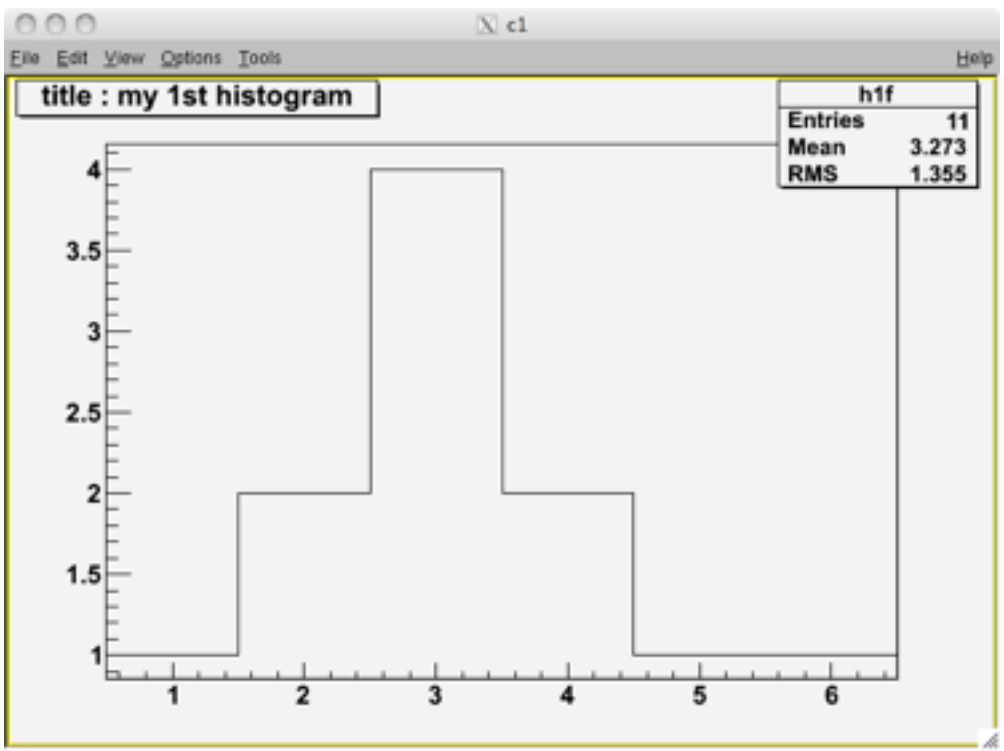
Histogram

Histogram

Histogram is just occurrence counting, i.e. how often they appear

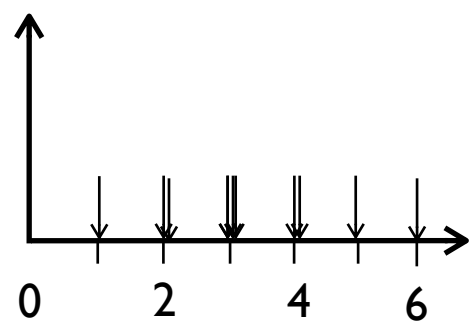
Suppose I have a set of integer data {1,3,2,6,2,3,4,3,4,3,5}

- If I want to see how frequently particular value is repeated (= I want to see "distribution of my data") in each region



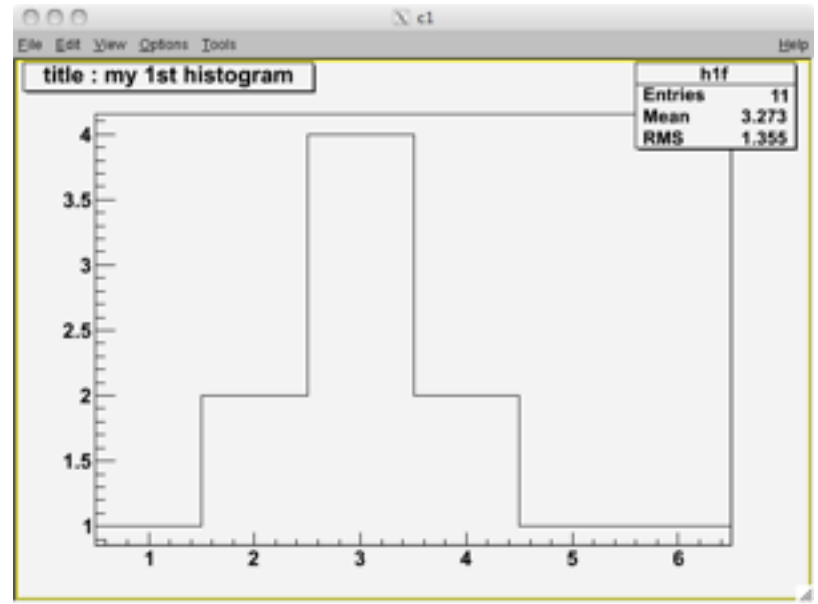
OK. Now I can see where the peak is, at least...

Otherwise I would have seen something (uninteresting) like



Histogram

So, how can we make it with ROOT?




```
//
// Fill a histogram
//
// E. Won (eunil@hep.korea.ac.kr)

void histo_fill()
{

  //
  // define a histogram
  //
  Float_t data[11] = {1,3,2,6,2,3,4,3,4,3,5};

  h1f = new TH1F("h1f","title : my 1st histogram",6,0.5,6.5);
  for (Int_t i=0; i<11; i++) h1f->Fill(data[i]);
  h1f->Draw();
}
```

I'm creating a new object called `h1f` *from the (pre-defined by ROOT) class* `TH1F`

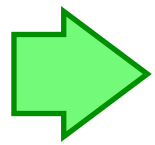
Looping over 11 data elements and "fill" into the histogram object `h1f`

And finally we draw the histogram that we filled

What is the keyword `Float_t` *? I know* `float`*, but...*

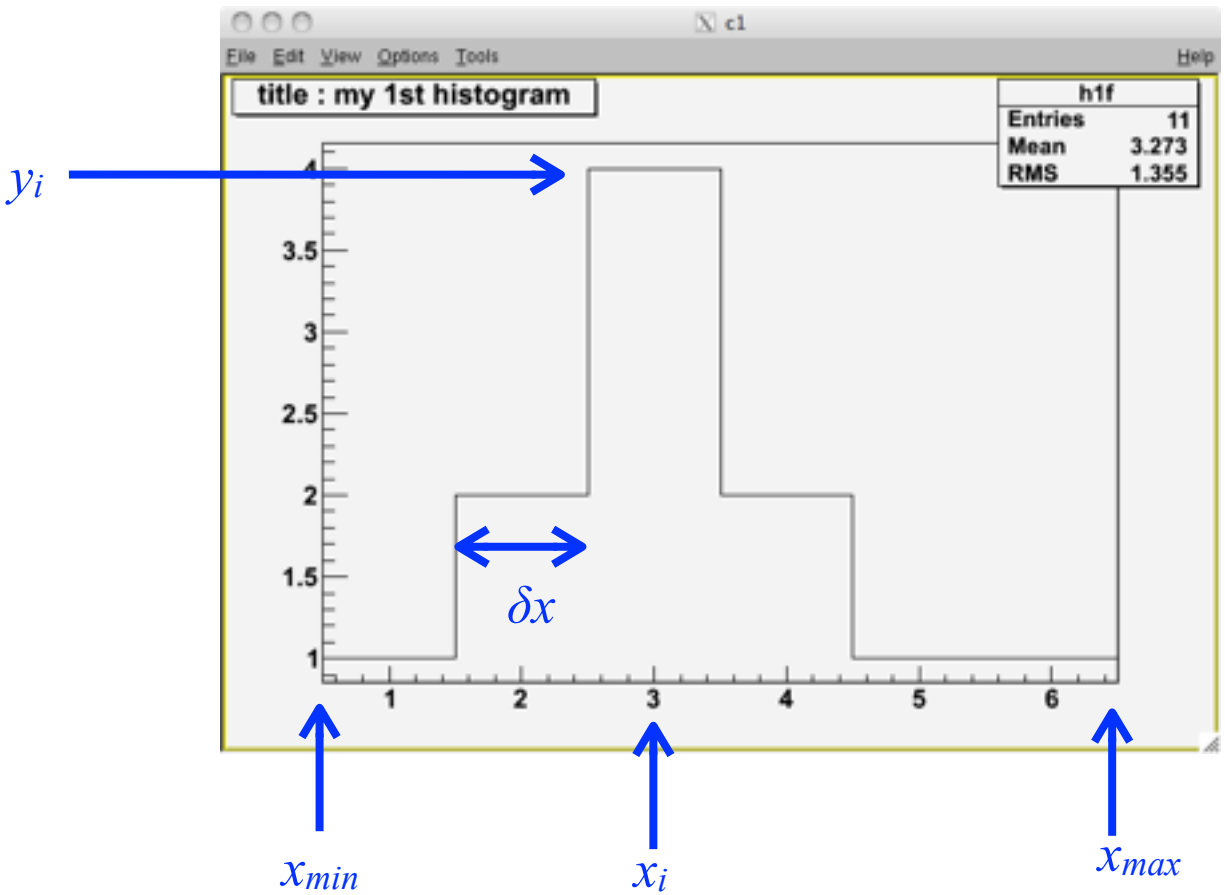
ROOT specific variable types

ROOT has its own data type (similar to C++)

ROOT		C++
`Int_t`		`int`
`Float_t`		`float`
`Double_t`		`double`
`...`		

You can still use C++ standard types. You need to know ROOT types to look at ROOT examples though.

Histogram - terminology



Statistics of the drawn histogram

$\delta x = (x_{max} - x_{min}) / nbin$

δx is called "bin width"

$nbin$ is called "number of bins"

Filling the histogram is also called "binning" the histogram

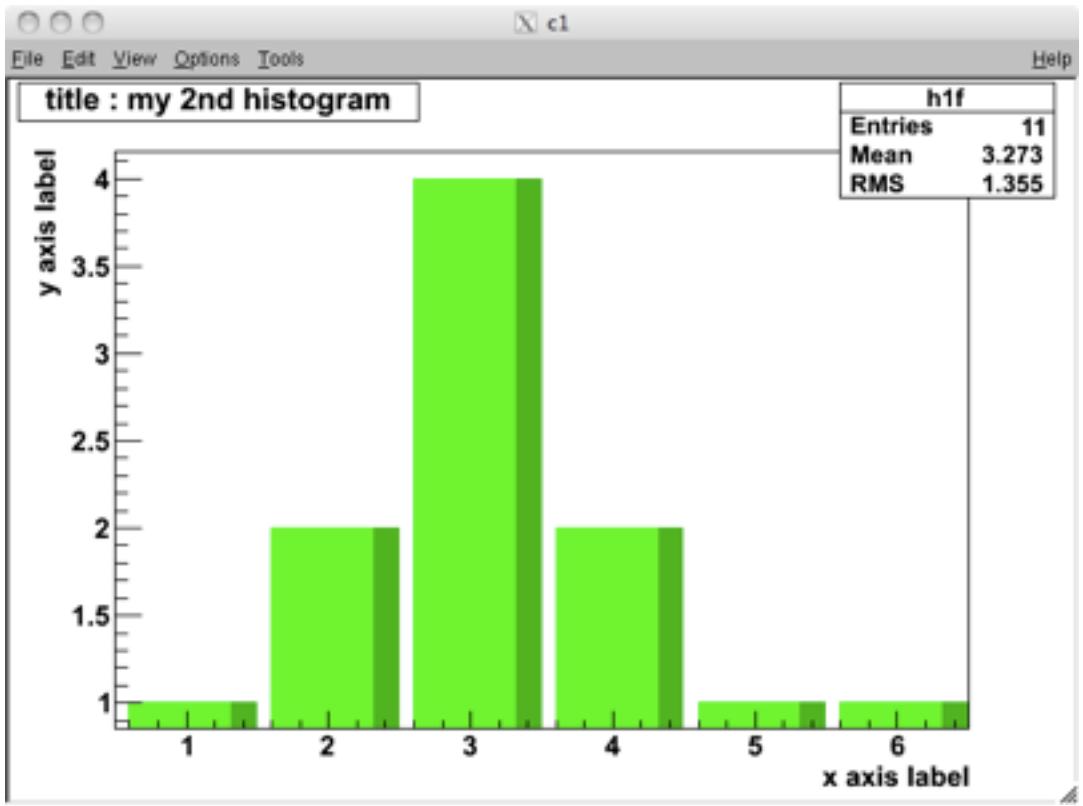
A TH1F constructor syntax:
TH1F("object", "title string", nbin, x_{min}, x_{max});

```
h1f = new TH1F("h1f","title : my 1st histogram",6,0.5,6.5);
```

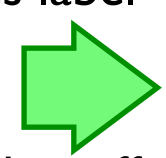
Question: so in our example so far, we have occurrence of data at a particular bin as the height (or number) at each bin. Can we talk the uncertainty associate with each height?

Histogram - cosmetics

- Previous histogram can be drawn as



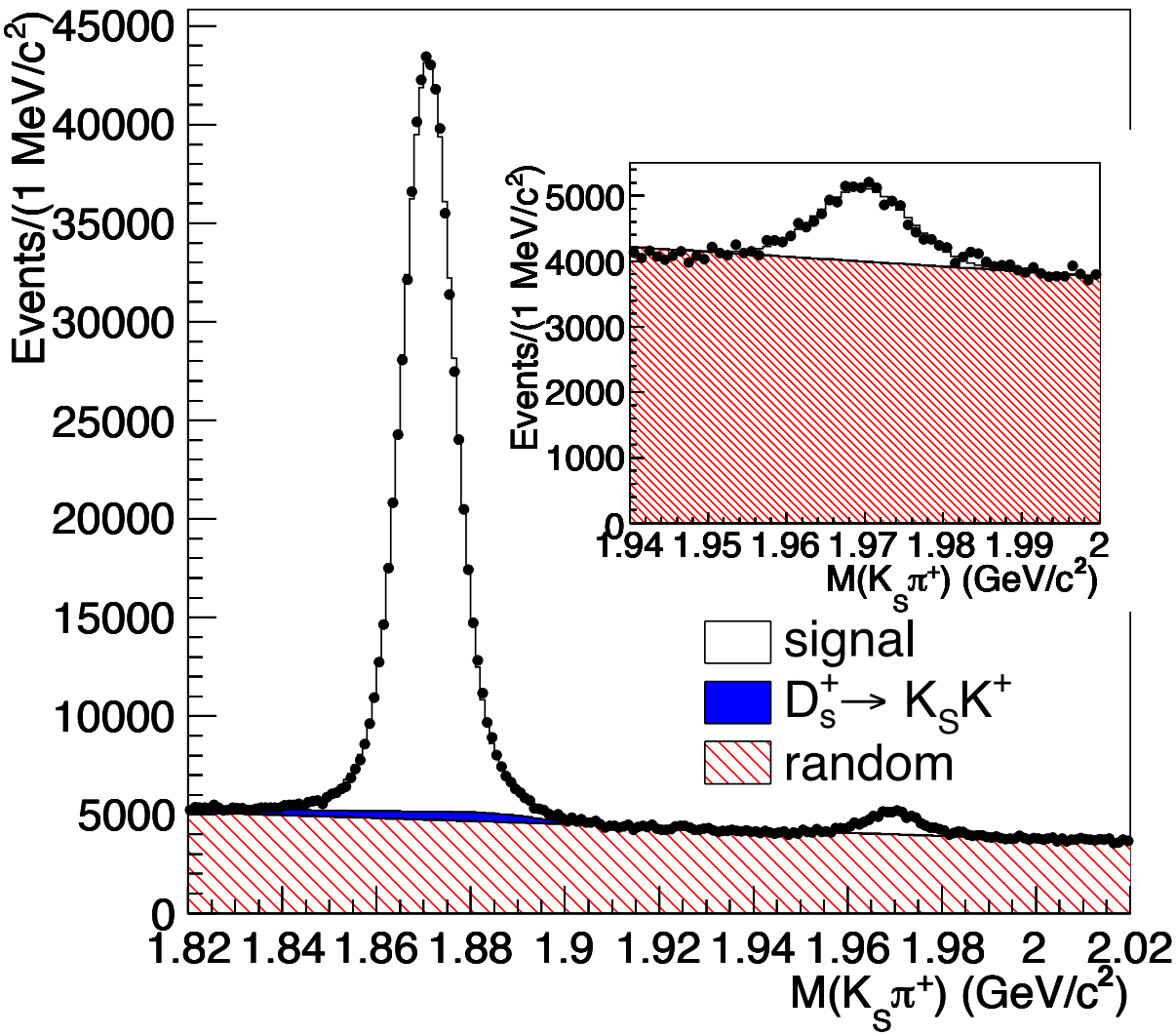
Different colors (background & histogram)
Axes label
....



It takes efforts to have a professional look for your plot (takes time!)

```
//
// Fill a histogram
//
// E. Won (eunil@hep.korea.ac.kr)

void histo_cos()
{

  //
  //
  gROOT->SetStyle("Plain");
  gROOT->ForceStyle();

  //
  // define a histogram
  //
  Float_t data[11] = {1,3,2,6,2,3,4,3,4,3,5};

  h1f = new TH1F("h1f","title : my 2nd histogram",6,0.5,6.5);
  for (Int_t i=0; i<11; i++) h1f->Fill(data[i]);
  h1f->SetBarWidth(0.9);
  h1f->SetFillColor(3);
  h1f->SetBarOffset(0.1);
  h1f->SetXTitle("x axis label");
  h1f->SetYTitle("y axis label");
  h1f->Draw("bar2");
}
```

Histogram - cosmetics
- how far can you go?



This is a frontier research quality plot made by me in 2009

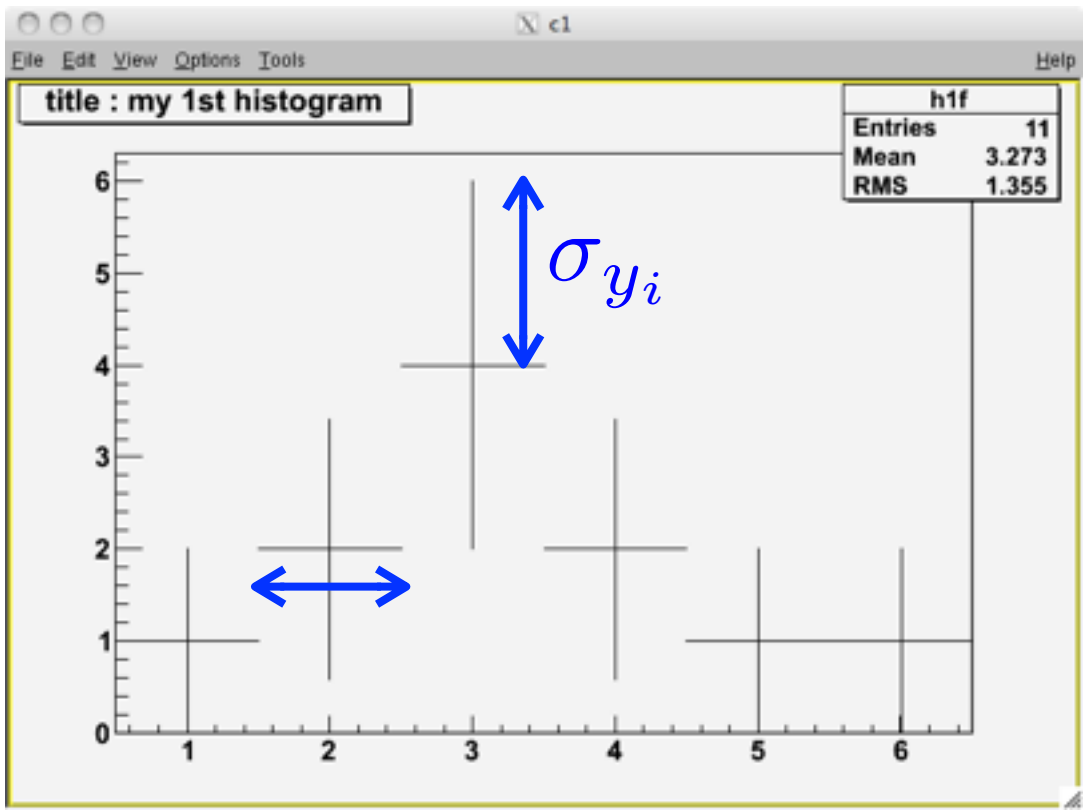
E. Won et al,
Physical Review D, 80 111101 (2009)

I'm not saying this is the best quality you can get, but it is pretty good one from a free software!

In your frontier research, it is important to make clear graphical presentation of results ! (as important as doing the research itself)

Histogram - "errors in each bin"

Note: following is true only when y_i represent the frequency of occurrence at *i-th* bin



This is a frontier research quality plot made by me in 2009

Full size of the horizontal bar represent the bin width...

Half size of the vertical bar shows the "error of y_i"

σ_{y_i} : error of y_i

$$\sigma_{y_i} = \sqrt{y_i}$$

Question: where is this from?

All I have to do to draw this is change he following line from the previous example

```
h1f->Draw("e");
```

Histogram - "errors in each bin"

Note: following is true only when y_i represent the frequency of occurrence at *i-th* bin

$$\sigma_{y_i} = \sqrt{y_i}$$ *Where is this from?*

Whether one number is contained in an entry or not - subjects to binomial probability

The probability of occurrence of *k*, for *n* trials is (*p*: probability of the occurrence)

$$P(k;p,n) = \frac{n!}{k!(n-k)!}p^k(1-p)^{n-k}$$

$$P(k;p,n) \sim \frac{e^\lambda \lambda^k}{k!} \qquad \text{when } n \gg 1 \qquad \lambda = np$$

This is called Poission probability and its standard deviation is $\sqrt{\lambda}$

and that's why we have $\sigma_{y_i} = \sqrt{y_i}$

(will discuss Poission and binomial probability distribution in more detail soon)

That's all for now for the histogram. Please practice to draw histograms to get familiar with them...