Tópicos Especiais de Física B IV: Introdução à análise de dados em FAE

O básico sobre o ROOT

ROOT-SYSTEM

ROOT PRIMER

https://root.cern.ch/root/htmldoc/guides/primer/ROOTPrimer.html

Tópicos

1) Alguns comandos do LINUX
2) O que é o ROOT
3) Uso interativo
   - Linha de comando vs. macro vs. código compilado
4) Abrindo arquivos e acessando informações
5) Histogramas, Tree e funções

Alguns comandos do LINUX

Linux command	What it does...
`ls`	List contents of a directory
`pwd`	Show present working directory
`mkdir test`	Make a new directory called test
`cd test`	Change to directory test
`cp file1.txt file2.txt`	Copy file1.txt to file2.txt
`mv file1.txt file3.txt`	Move file1.txt to file3.txt

`cat file2.txt`
`less file2.txt`
`more file2.txt`
	Print the contents of a file to the screen

`emacs -nw`
`vi`
`pico`
`...`
	Console text editors (no extra window pops up)

`emacs`
`xemacs`
`nedit`
`gedit`
`...`
	GUI text editors (extra window pops up)

What is ROOT?

Versatile software package developed for performing data analysis

- Read data from some source
- Write data to a file
- Select data with some criteria ("cuts")
- Perform calculations and fits
- Produce results as plots, graphs, numbers, fits, etc.
- Save results in some format (ROOT file, image of plot, ...)

ROOT can be used in many ways:

Command line

Good for quickly making plots, checking file contents, etc.

Unnamed macros

Execute commands as if you typed them on the command line
List of commands is enclosed by one set of { }.
Execute list of commands from command line by: "`.x file.C`" (without quotes)

Named macros / Compiled code

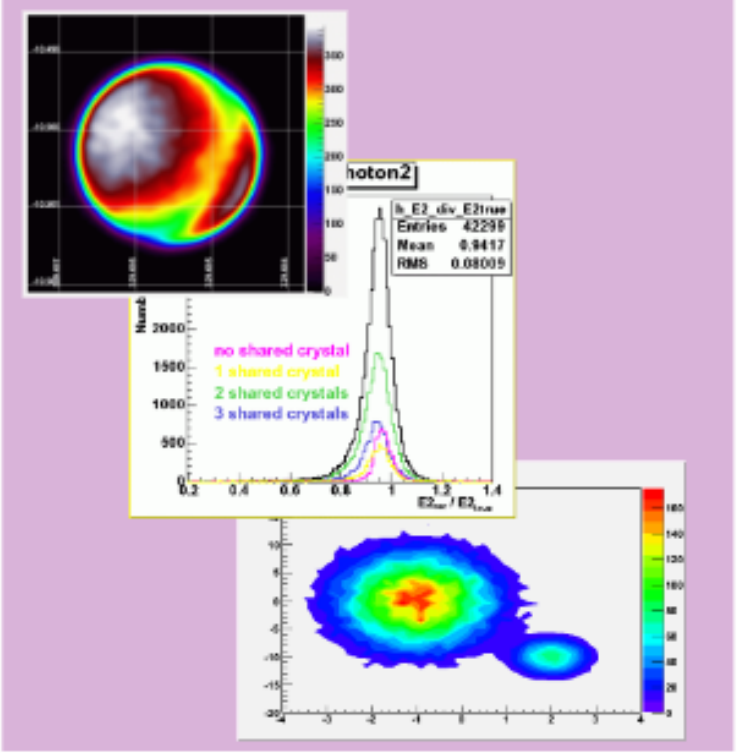
Best for analysis, can be compiled and run outside of ROOT, or loaded and executed during interactive session

Developed and supported by high energy physics community

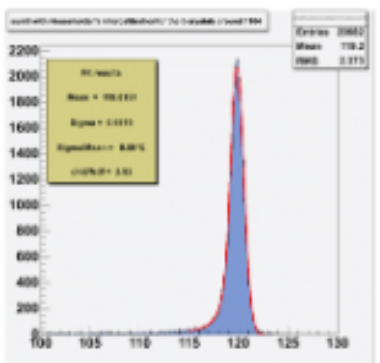
Homepage with documentation and tutorials (http://root.cern.ch)
Google will also find the documentation (for example, try "root TH1F")

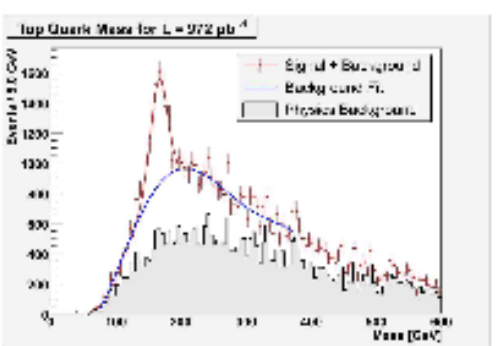
What ROOT can do:

Plotting

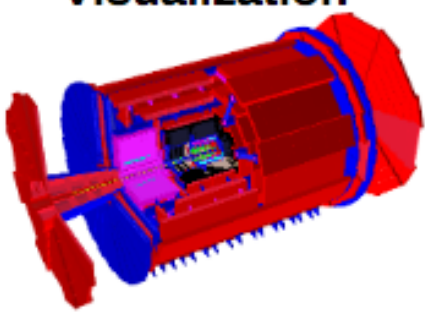


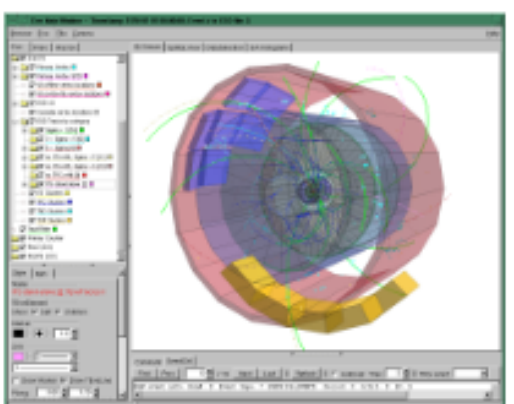
Fitting / Computation





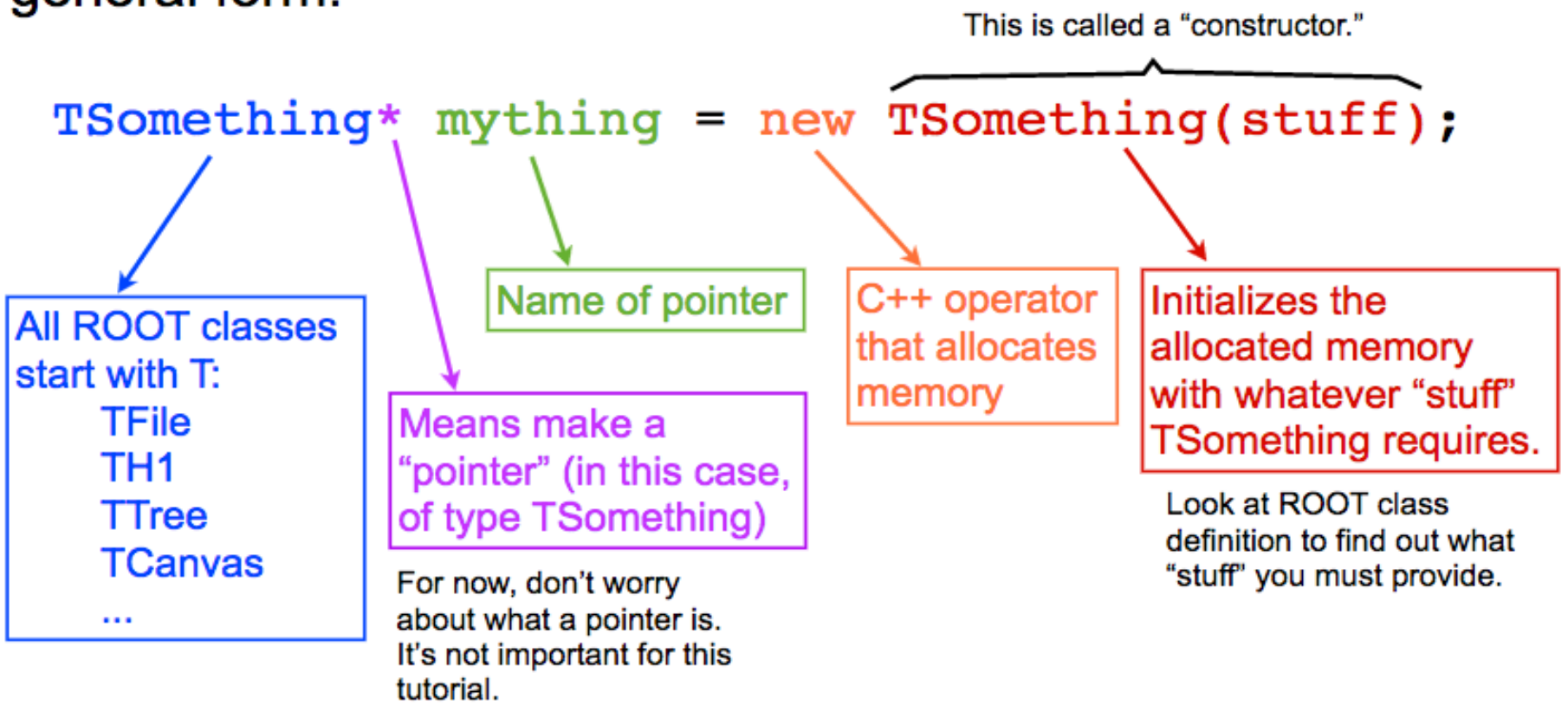
Detector Visualization





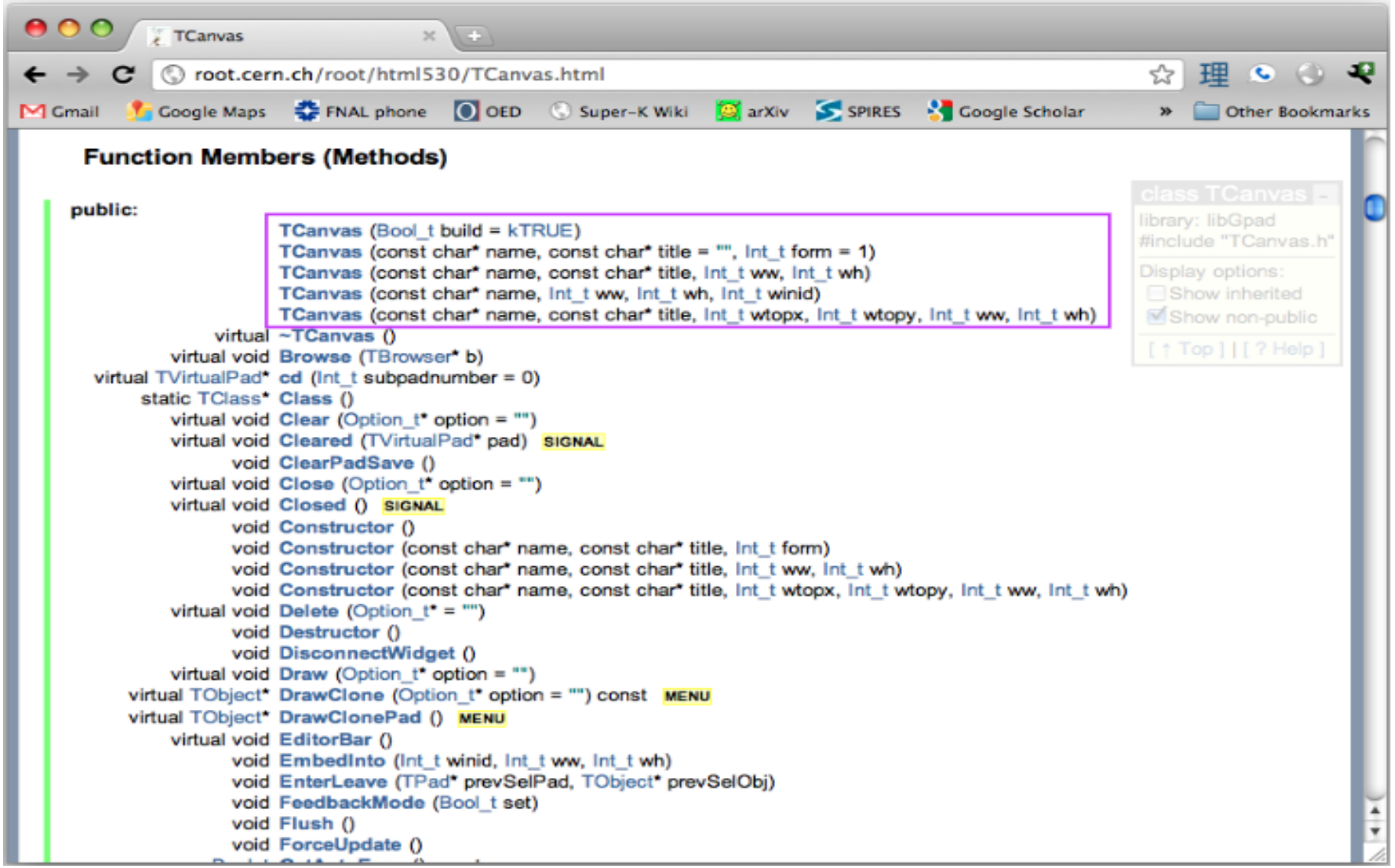
Syntax

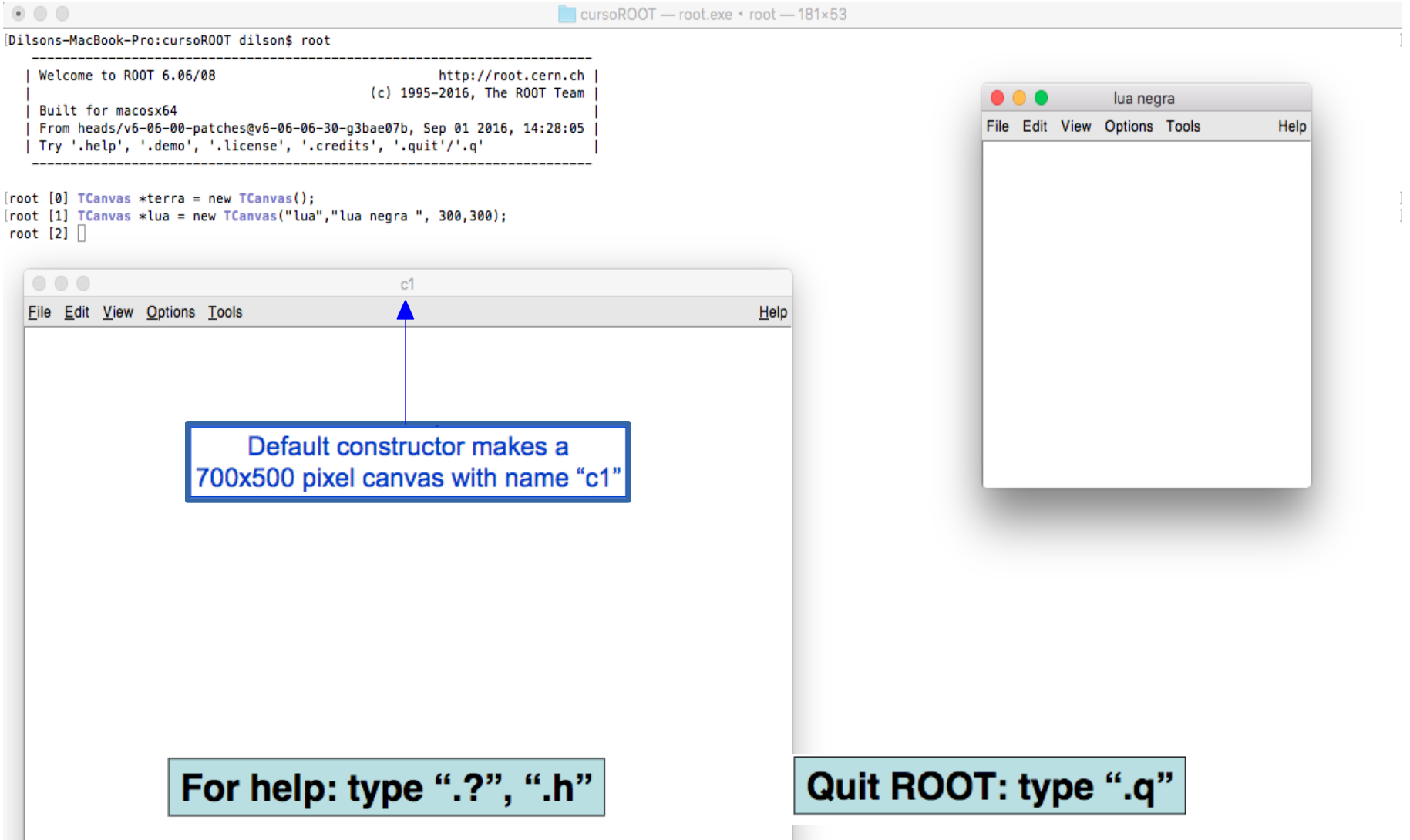
Many of the commands we will use will have this general form:



Note: In C++, if you allocate memory using the "new" operator, you must later use "delete mything" to release the memory... otherwise your code will have a memory leak. We will not worry about that today, but keep it in mind for your future code-writing.

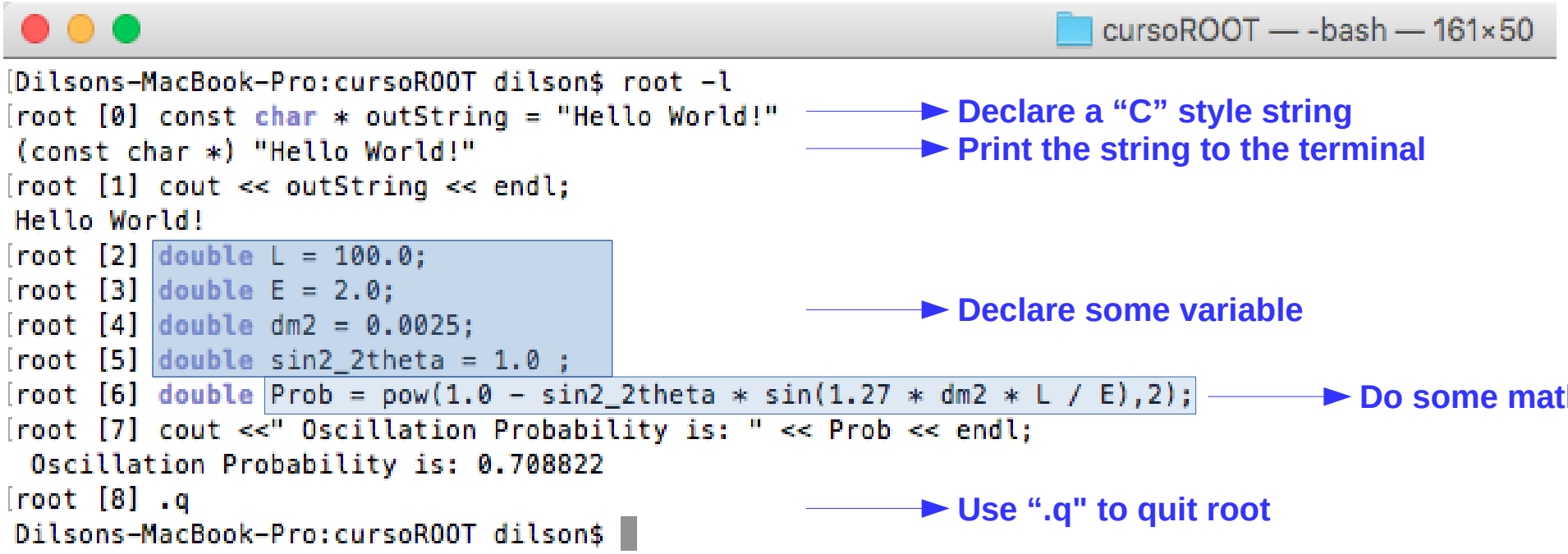
Example: TCanvas Constructors





Atenção: Comandos executados diretamente na linha de comando serão salvos um arquivo chamado: .root_hist (quem estará no seu diretório home).

ROOT command line: Hello world! And Calculator



ROOT command line: Hello world! And Calculator

```
root [6] double x=.5
(double) 5.000000e-01
root [7] int N=30
(int) 30
root [8] double geom_series=0
(double) 0.000000e+00
root [9] for (int i=0;i<N;++i)geom_series+=TMath::Power(x,i)
root [10] TMath::Abs(geom_series - (1-TMath::Power(x,N-1))/(1-x))
(Double_t) 1.862645e-09
```

ROOT: C++ at the prompt

```
root [0] using doubles = std::vector<double>;
root [1] auto pVec = [](const doubles& v){for (auto&& x:v) cout << x << endl;};
root [2] doubles v{0,3,5,4,1,2};
root [3] pVec(v);
0
3
5
4
1
2
root [4] std::sort(v.begin(),v.end());
root [5] pVec(v);
0
1
2
3
4
5
root [6] std::sort(v.begin(),v.end(),[](double a, double b){return a>b;});
root [7] pVec(v);
5
4
3
2
1
0
```

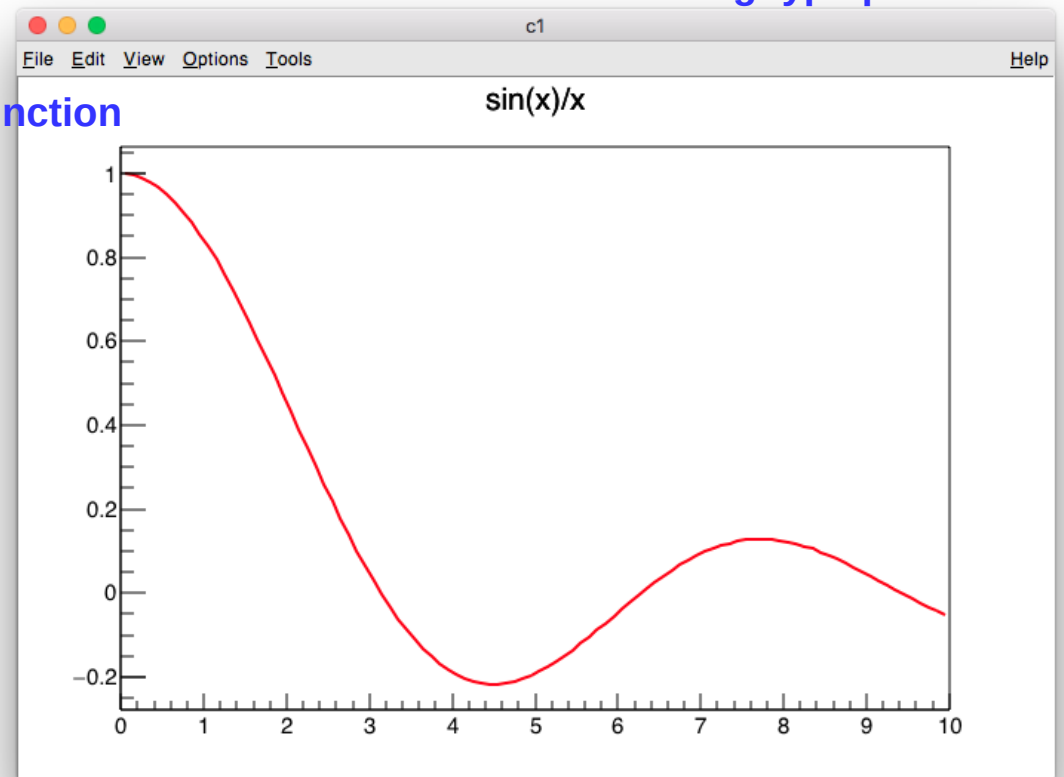
ROOT: as function plotter

```
Dilsons-MacBook-Pro:cursoROOT dilson$ root -l
root [0]  TF1 f1("f1","sin(x)/x",0.,10.);
root [1] f1.Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2]
```

f1 is an instance of a TF1 class

This string type parameter defines the function

The Draw method display the function

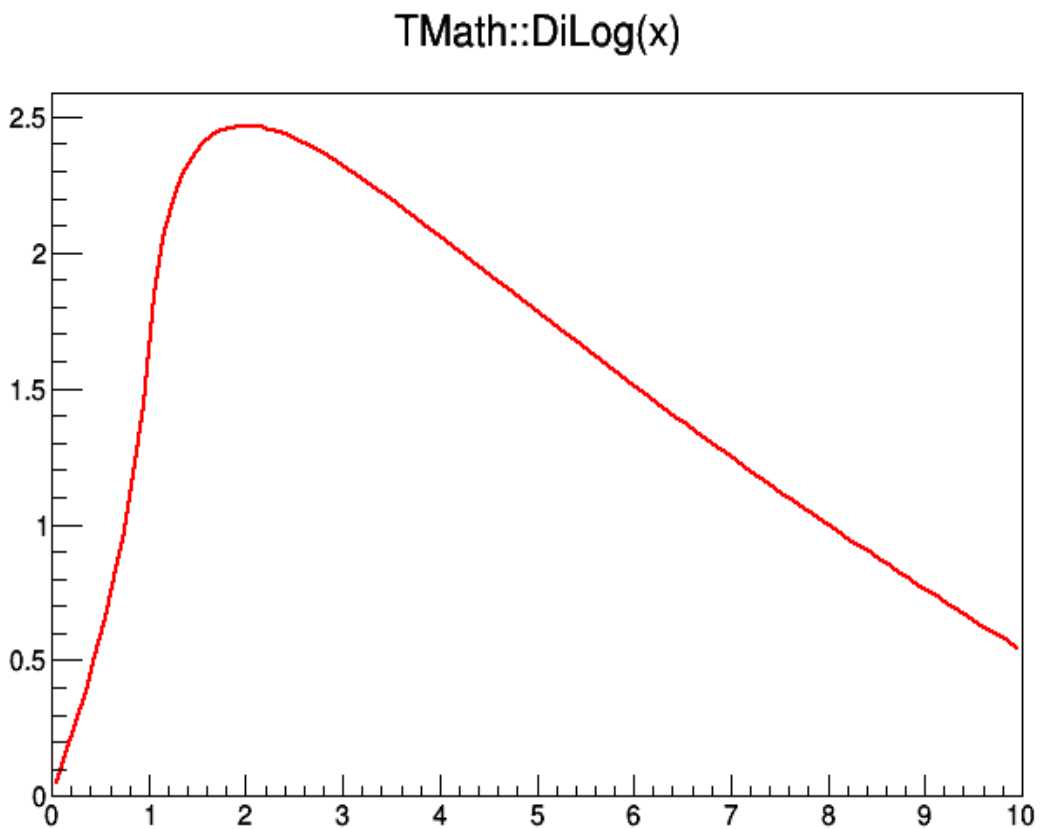


https://root.cern.ch/doc/master/classTF1.html

ROOT: as function plotter

```
{
   TF1 *fa2 = new TF1("fa2","TMath::DiLog(x)",0,10);
   fa2->Draw();
}
```

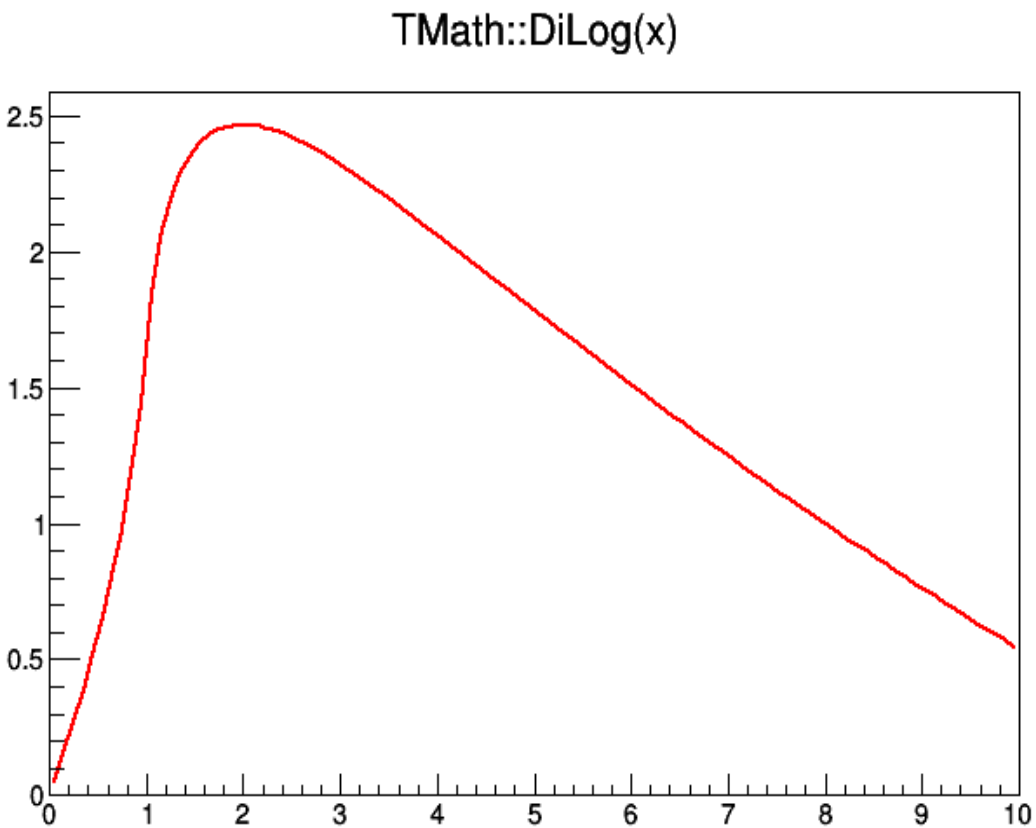
Using a ROOT function



ROOT: as function plotter

```
{
   TF1 *fa2 = new TF1("fa2","TMath::DiLog(x)",0,10);
   fa2->Draw();
}
```

Using a ROOT function



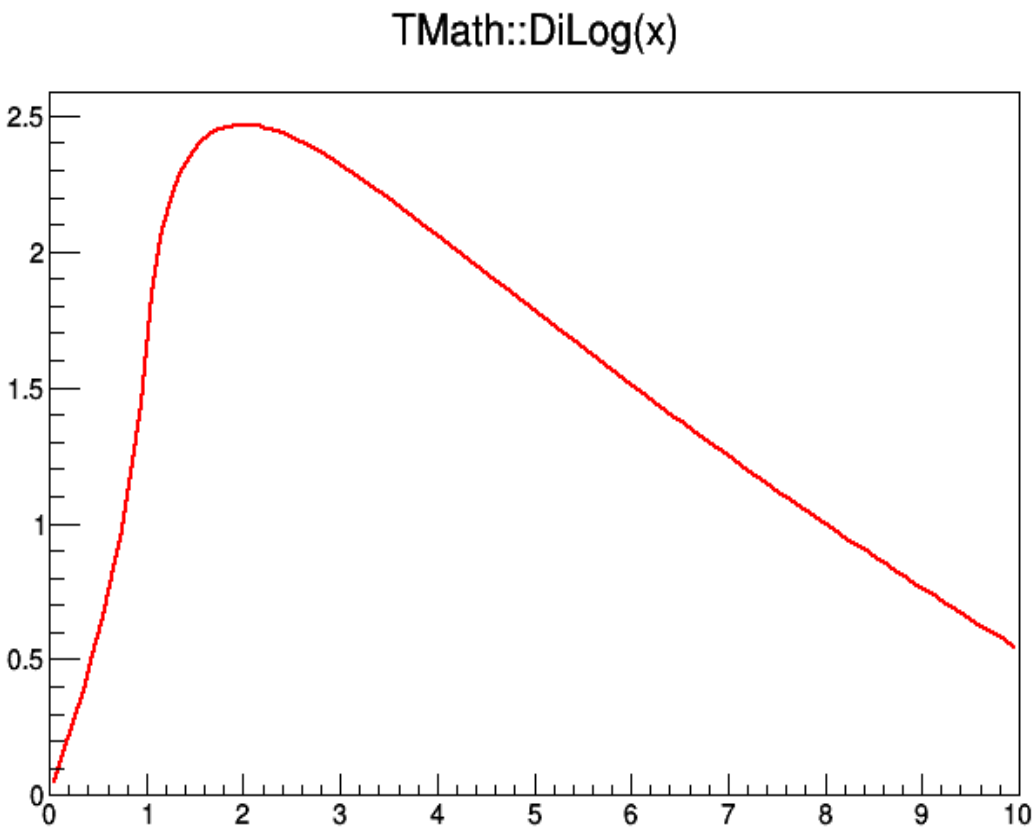
User defined

Double_t myFunc(double x) { return x+sin(x); }
....
TF1 *fa3 = new TF1("fa3","myFunc(x)",-3,5);
fa3->Draw();

ROOT: as function plotter

```
{
   TF1 *fa2 = new TF1("fa2","TMath::DiLog(x)",0,10);
   fa2->Draw();
}
```

Using a ROOT function



User defined

```
Double_t myFunc(double x) { return x+sin(x); }
....
TF1 *fa3 = new TF1("fa3","myFunc(x)",-3,5);
fa3->Draw();
```

- A partir daqui seguimos o ROOT Primer, capítulo 1 e 2
  - https://root.cern.ch/root/htmldoc/guides/primer/ROOTPrimer.html
  - Fazer a leitura e reproduzir os exemplos dos capítulos seguintes

Exercício

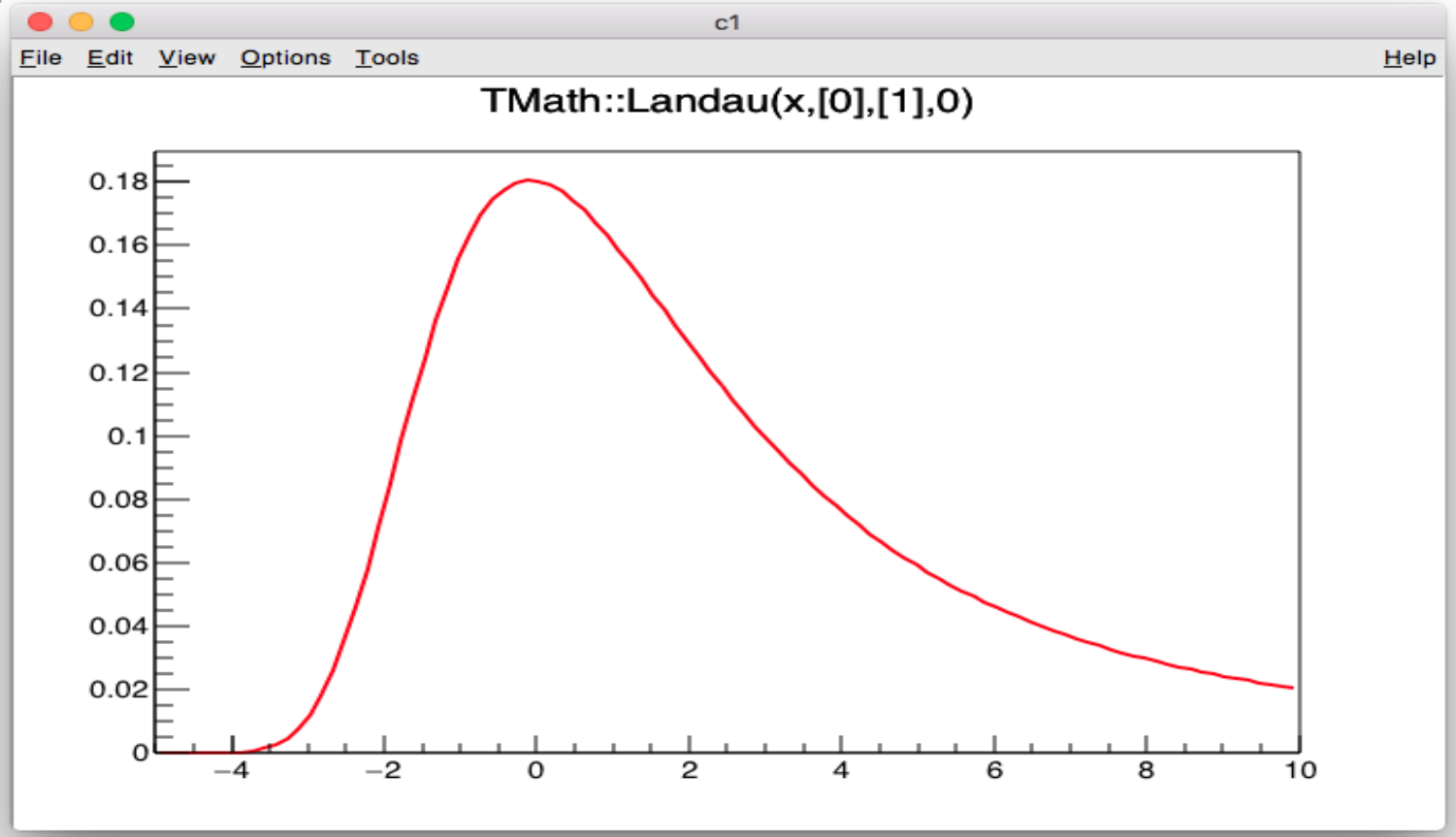
- Create and draw a Gaussian whose mean is 50 and standard deviation is 10 on the range [0, 100]

- Here is the formula: Note there are 3 parameters

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

ROOT: as function plotter

```
Dilsons-MacBook-Pro:cursoROOT dilson$ root -l
root [0] TF1 *fb2 = new TF1("fa3","TMath::Landau(x,[0],[1],0)",-5,10);
root [1] fb2->SetParameters(0.2,1.3);
root [2] fb2->Draw();
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3]
```



More sofisticated

Exercícios

- Opening a file and acessing its contents

```
root [0] TFile* f1 = new TFile("histograms.root");
root [1] f1->ls();
TFile**         histograms.root
 TFile*         histograms.root
  KEY: TH1F     histo1;1        Fancy 1-D Histogram
  KEY: TH2F     histo2;1        Schmancy 2-D Histogram
root [2] TCanvas *c1 = new TCanvas(<TAB>
TCanvas TCanvas(Bool_t build = kTRUE)
TCanvas TCanvas(const char* name, const char* title = "", Int_t form = 1)
TCanvas TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
TCanvas TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy, Int_t ww,
Int_t wh)
TCanvas TCanvas(const char* name, Int_t ww, Int_t wh, Int_t winid)
root [3] TCanvas *c1 = new TCanvas(
```

List the contents of the file.

This file contains 2 histograms

Tip: If you can't remember the correct syntax, press "Tab" for ROOT to help you complete a command.

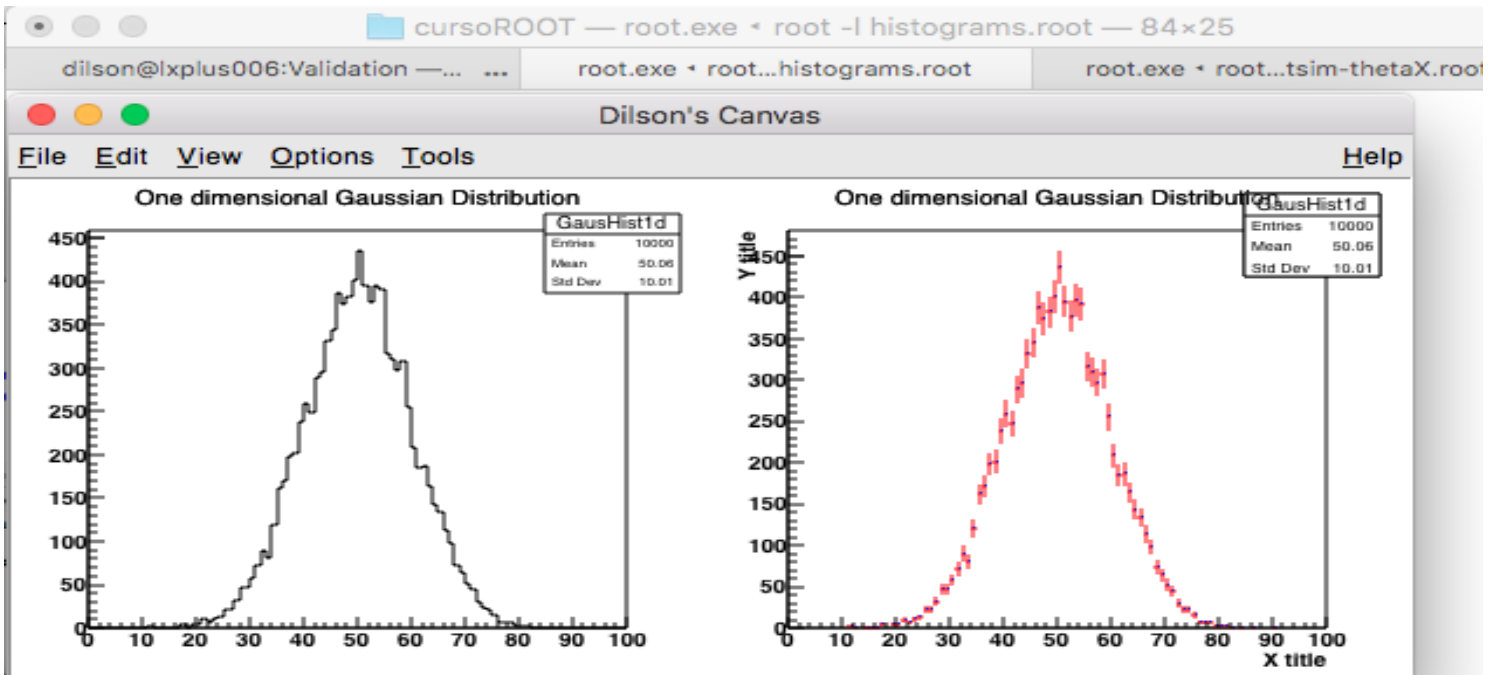
Exercise 1:

Complete the command to make a canvas that is 600 x 300 pixels.
Divide it into 2 regions. (Hint: See TPad class definition at root.cern.ch and look for method Divide)
Change to region 1 and draw histo1. (Hint: c1->cd(1))
Change to region 2 and draw histo2.

Note: If you don't make a canvas, ROOT will create one for you when you try to draw something. Personally, I like to make my own because I can control the dimensions.

Exercícios

- Change some attributes of the histograms
  - Try to use “lego2”, “colz”...



Check ThistPainer for even more drawing options

Exercícios

ROOT TTree : More about Arguments

- Arguments to many functions in ROOT objects are passed by character strings
- Strings are parsed for both logic and mathematics
- For trees
  - Any variable in the tree can be manipulated as part of an argument

```
root [11] mytree->Draw("py:px", "ebeam > 150.0", "lego" );
```

What to draw for each event
Semicolon ":" indicates adding a new dimension
Can be functions of variables: "sqrt(py)"
Can be combinations of variables : " ebeam/px :py**2"

Selection Cuts: I.e. which events or entries to draw
Multiple cuts are allowed, combined with C-style logic operators
Can be functions of variables.
Can be combinations of variables.

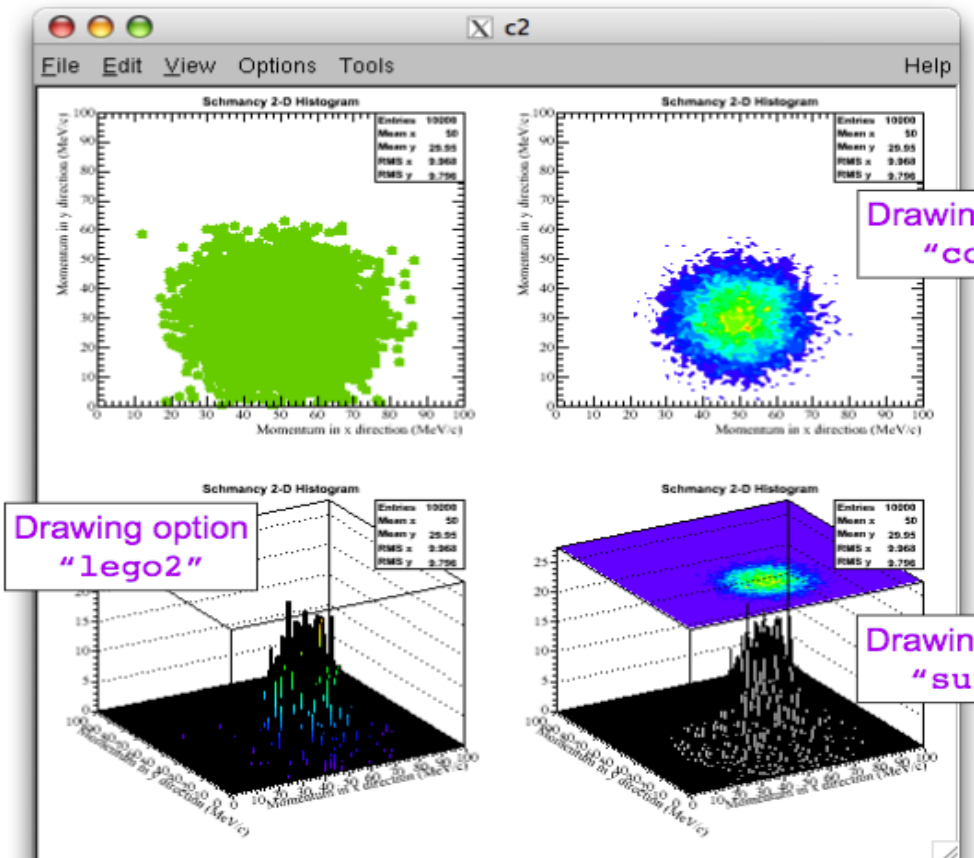
Drawing Options
Options for n-dimensional histograms go here as in previous examples

C-style logic operators

&&	- AND
||	- OR
==	- EQUALS
>	- GREATER
<	- LESS
>=	- > or EQUAL
<=	- < or EQUAL
!=	- NOT EQUAL

Exercícios

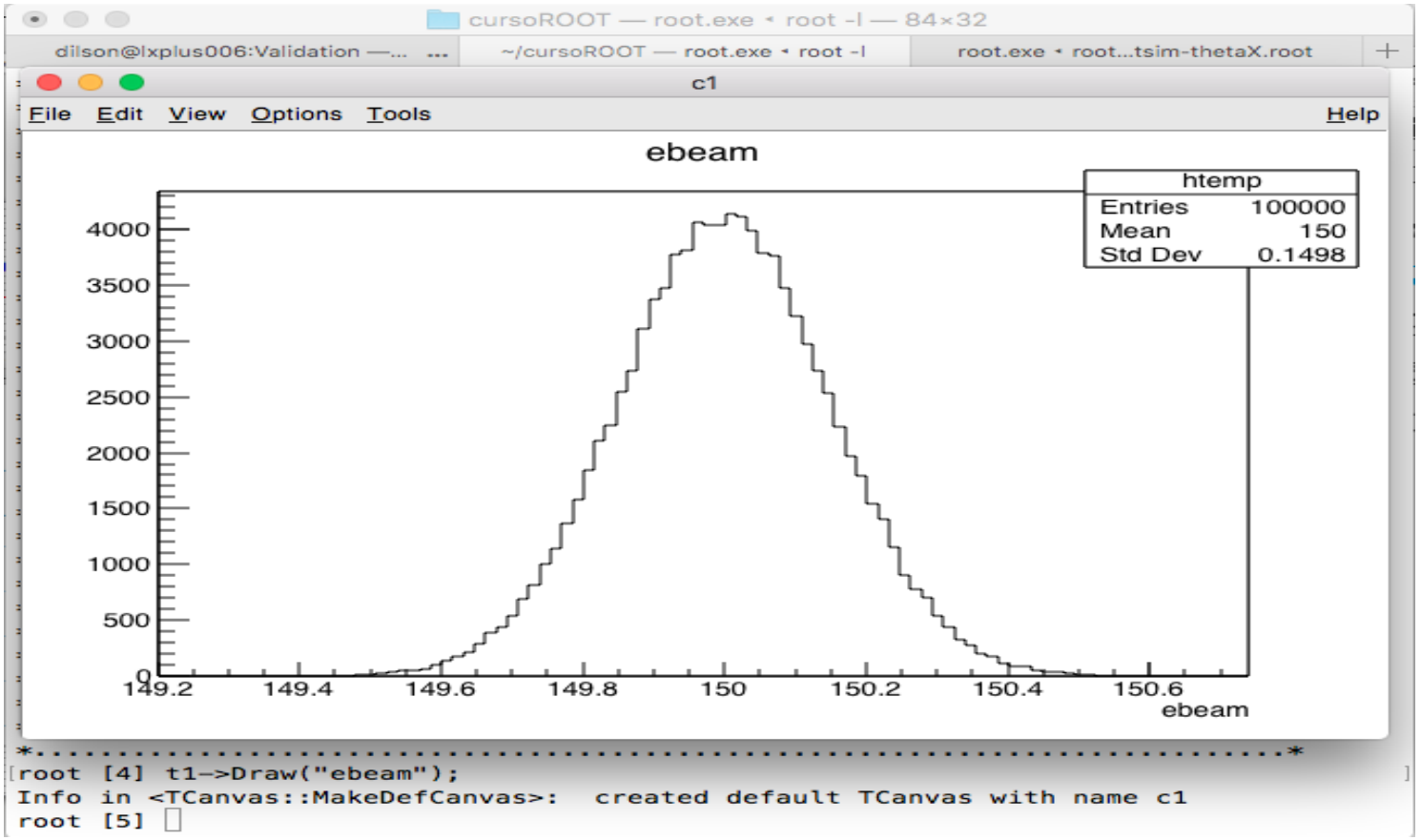
- Change some attributes of the histograms
  - Try to use “lego2”, “colz”...



You may have noticed that the default ROOT palette is hideous. Set this less offensive one by doing:

```
gStyle->SetPalette(1);
```

Exercício



Draw some other variables from the tree.

Draw a 2D plot of px vs. py

Hint: Look at the TTree Draw methods at http://root.cern.ch/root/html530/TTree.html

Draw px vs. py for events with ebeam > 150.0

Hint: The TTree Draw methods can take a "selection" character string

Projecting from a tree into a histogram

Sometimes you may want to put a variable from a tree into a histogram.

First define the histogram:

Name, Title, Number of bins, Low edge, High edge

```
root [7] TH1F *h_ebeam = new TH1F("h_ebeam","Beam energy",100,149.0,151.0);
```

Then use the `TTree` method "`Project`" to put the tree contents into the histogram:

```
root [8] t1->Project("h_ebeam", "ebeam", "(px > 10.0) || (py <= 5.0)");
```

Selection cuts: optional argument

To define complicated or often-used cuts:

```
TCut* cut1 = new TCut("px > 10.0");
TCut* cut2 = new TCut("py <= sqrt(2+px**2)");
TCut* cut3 = new TCut(*cut1 && *cut2);
```

Then use the TCut when you draw!

```
t1->Draw("ebeam", *cut3);
```

Exercise 5:
Try making some cuts on tree variables and drawing/projecting.

Cuts are specified using C logic

&&	AND
||	OR
==	equal
!=	NOT equal
>	greater than
<	less than
>=	greater or equal to
<=	less or equal to

ROOT : Other intersting tools

`TLorentzVector`

A general 4-vector class with implemented functionality to do almost everything you typically need to do with 4-vectors.

- dot products
- rotations
- boosting
- angle between vectors
- magnitude...

(next lecture)

`TFractionFitter`

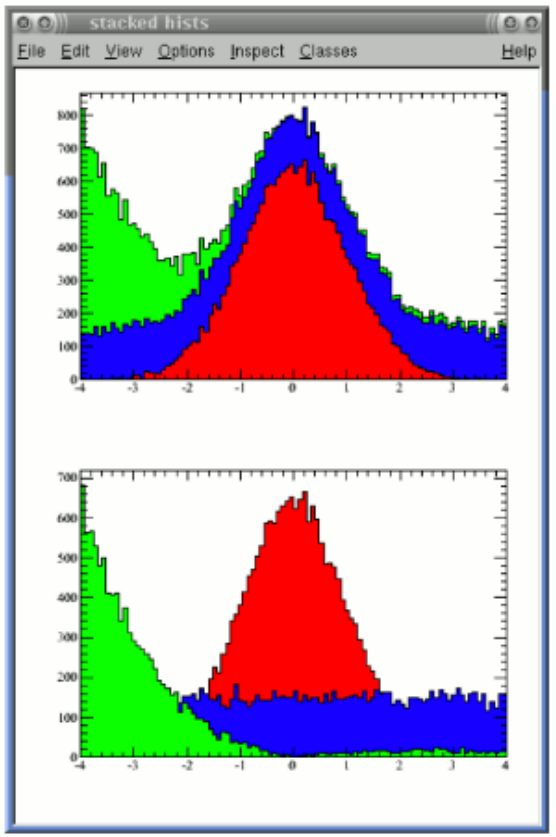
Fits data histogram using multiple MC histograms (instead of a defined function)

`TFitter, TMinuit`

Classes for fitting

`THStack`

Takes a collection of histograms and draws them "stacked" on each other.



Tutoriais utilizados para essas aulas

- Seguimos para o tutorial seguinte para vermos ajustes
  - http://hep.bu.edu/~jlraaf/2011REU/root_lecture02.pdf
- Para análise de dados, seguimos o seguinte tutorial
  - http://www.phy.duke.edu/~raw22/public/root.tutorial/basic_root_20100701.pdf

ROOT : Full Exercise

- Write a Macro to examine the contents of the file double.root
- Locate the TTree within and plot the “Mass” variable
- Plot the “Mass” variable on a separate canvas after selecting only positively charged events and those events that are within 30 degrees of the beam direction
- Project this distribution into an appropriate one-dimensional histogram and fit it using the sum of two gaussians
- What are the mean and standard deviation of each peak?
- What is the Chi Squared value of the fit?
- How many degrees of freedom?