Introduction to ROOT

Prof. Silvia Masciocchi
dr. Federica Sozzi

Heidelberg University

Day 2 – First steps with ROOT

Based on the slides created by dr. Jens Wiechula, Frankfurt University

Outline

- Where to get information
- First steps with ROOT / the C INTerpreter
- Data types in ROOT
- Using macros
- Histograms
- Graphs
- Functions
- Fitting histograms and graphs
- Canvases
- Legends

Where to get information

- The ROOT homepage

  http://root.cern.ch
- User's guide

  http://root.cern.ch/drupal/content/users-guide
- Class documentation

  http://root.cern.ch/root/html/ClassIndex.html
- Tutorials

  http://root.cern.ch/root/html/tutorials/
- A ROOT Guide For Beginners

  https://root.cern.ch/guides/primer

Starting root

```
~$ root -h
Usage: root [-l] [-b] [-n] [-q] [dir] [[file:]data.root] [file1.C ... fileN.C]
Options:
  -b : run in batch mode without graphics
  -n : do not execute logon and logoff macros as specified in .rootrc
  -q : exit after processing command line macro files
  -l : do not show splash screen
  -x : exit on exception
 dir : if dir is a valid directory cd to it before executing

  -?       : print usage
  -h       : print usage
  --help   : print usage
  -config  : print ./configure options
  -memstat : run with memory usage monitoring
```

Starting root

When starting root you will see the root prompt

ROOT 5

```
GSI: wiechula@milamber: ~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day01
File  Edit  View  Bookmarks  Settings  Help

wiechula@milamber:~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day01$ root
  ****************************************
  *                                      *
  *        W E L C O M E  to  R O O T    *
  *                                      *
  *   Version   5.34/02 21 September 2012*
  *                                      *
  *  You are welcome to visit our Web site*
  *          http://root.cern.ch         *
  *                                      *
  ****************************************

ROOT 5.34/02 (tags/v5-34-02@46115, Jan 11 2013, 14:04:11 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.

starting jens' environment

root [0]
```

ROOT 6

```
Terminal - fsozzi@kp1nbg055: ~
File  Edit  View  Terminal  Tabs  Help

fsozzi@kp1nbg055:~$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.07/07                        http://root.cern.ch |
  |                                  (c) 1995-2016, The ROOT Team     |
  | Built for linuxx8664gcc                                           |
  | From heads/master@v6-07-06-2029-g54e19b3, Sep 27 2016, 07:42:17   |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'        |
   ------------------------------------------------------------------

root [0]
```

ROOT5 - CINT

The C INTerpreter of root

- Commands of the ROOT prompt are forwarded to “CINT”
- CINT is a C(++) interpreter
- Covers most of ANSI C and ISO C++ 2003
  - BUT: is not as strict as a compiler

    → Be careful, code that executes in CINT does not necessarily compile!
- Allows to type code directly into a command line
  - Make use and test root features quickly
- Allows for fast code development and is ideal for small tasks
- Is much slower than compiled code
- Can be used as a calculator

ROOT 5 vs ROOT 6

In ROOT6 new compiler CLING replaces interpreter CINT

	CINT	CLING
Type	Interpreter	Just-in-time compiler
Maintainance	ROOT	Mostly by LLVM team
Language standard	C++98	C++11, C++14, …
C++ Language support	Partial	Full
Language correctness	Allows non-standard code	Strict
STL support	Partial	Full

ROOT prompt

ROOT as a calculator

GSI: wiechula@milamber: ~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day01

File Edit View Bookmarks Settings Help

```
  *                                          *
  ********************************************

ROOT 5.34/02 (tags/v5-34-02@46115, Jan 11 2013, 14:04:11 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.

starting jens' environment

root [0] 5.32*2.7
(const double)1.43640000000000025e+01
root [1] sqrt(5)
(const double)2.23606797749978981e+00
root [2] exp(2.9)
(const double)1.81741453694430604e+01
root [3] sin(90./180.*3.1415)
(const double)9.99999998926914047e-01
root [4] for (int i=0; i<10; ++i) { cout << i <<" ";}; cout<<endl;
0 1 2 3 4 5 6 7 8 9
root [5]
```

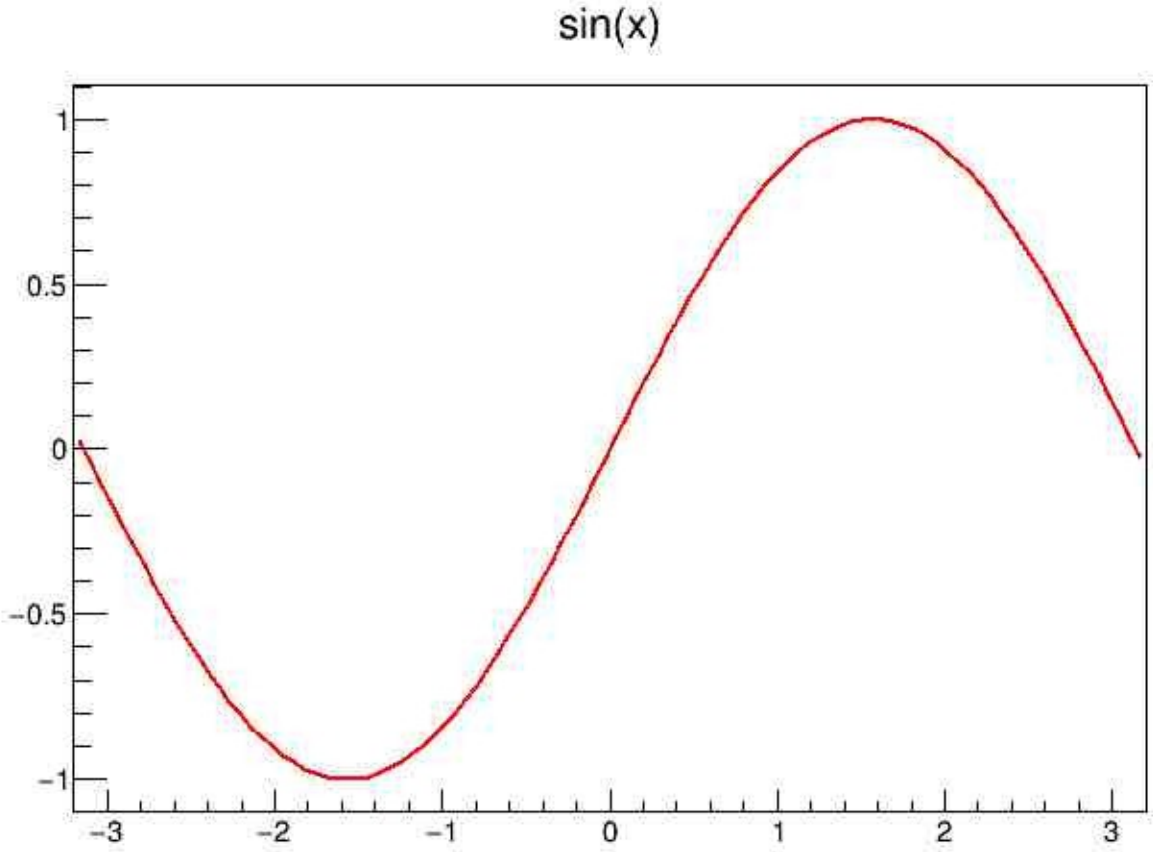
ROOT prompt

drawing example

Drawing a function using ROOT classes

```
root[] TF1 f("f","sin(x)",-3.2,3.2)
root[] f.Draw()
```

Major part of ROOT object names
start with "T"



ROOT prompt

Most important commands / features

root[] .q // end the root session
root[] .? //this command will list all the CINT commands
root[] .L <filename> //load [filename]
root[] .x <filename> //load and execute [filename]

Shell commands can be executed after a .!
root[] .! ls
root[] .! cat /tmp/fileX

Blocks of code can be included in {}
root [] {
end with '}', '@':abort > for (int i=0; i<10; ++i) {
end with '}', '@':abort > cout << i <<" ";
end with '}', '@':abort > cout << endl;
end with '}', '@':abort > }
end with '}', '@':abort > }

Unix shell like tab completion, shows hint, e.g. the constructors of a class
root [9] TH1F h(<TAB>
TH1F TH1F()
TH1F TH1F(const char* name, const char* title, Int_t nbinsx, Double_t xlow, Double_t xup)
TH1F TH1F(const char* name, const char* title, Int_t nbinsx, const Float_t* xbins)
TH1F TH1F(const char* name, const char* title, Int_t nbinsx, const Double_t* xbins)
TH1F TH1F(const TVectorF& v)
TH1F TH1F(const TH1F& h1f)

Macros

Basics

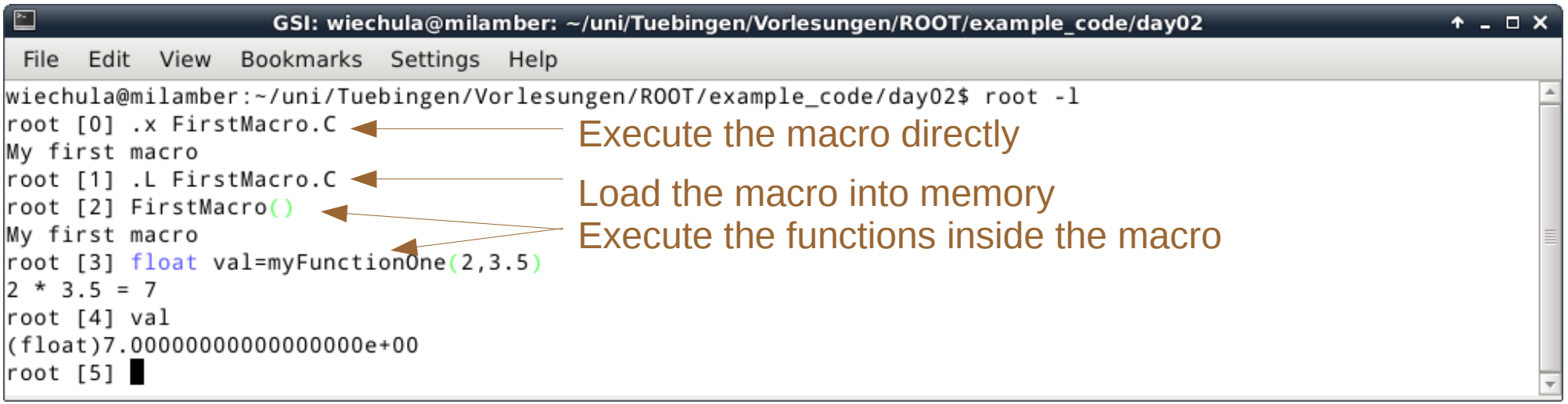
- The code to be interpreted can be written inside a text file, called a macro
- By convention the macro has the extension '.C'
- Macros can hold any C++ code, e.g. several functions, classes ...
- The interpretation by CINT has some limitations
  - For most things not recognisable
  - Again: Code that CINT manages to interpret does not need to compile

Macros

Simple example

```
1  void FirstMacro()
2  {
3    cout << "My first macro" << endl;
4  }
5
6  float myFunctionOne(float value, float multiplicator)
7  {
8    float returnValue=value*multiplicator;
9    cout << value << " * " << multiplicator << " = " << returnValue << endl;
10   return returnValue;
11 }
```

Note: do you notice main differences with the C++ files used in the lecture of yesterday?



Macros can also be run directly from the shell:

shell$ root FirstMacro.C

FirstMacro.C

Macros

ACLiC - The Automatic Compiler of Libraries for CINT

- Macros can be compiled on the fly by adding a '+' or '++' behind the file name (handled by ACLiC)
- +: only compile again if changes were made since the last compilation
- ++: force recompilation of the code
- + and ++ can be combined with 'g' or 'O'
  - g: add debugging symbols to the executable (like -g of g++)
  - O: better optimisation of the code (like -O of g++)
- In order to be able to compile the macro, *#include* statements for all classes used need to be added to the macro

Macros

ACLiC – example

```
1  #include <iostream>
2
3  using namespace std;
4
5  void FirstMacro2()
6  {
7    cout << "My first macro" << endl;
8  }
9
10 float myFunctionOne(float value, float multiplicator)
11 {
12   float returnValue=value*multiplicator;
13   cout << value << " * " << multiplicator << " = " << returnValue << endl;
14   return returnValue;
15 }
```

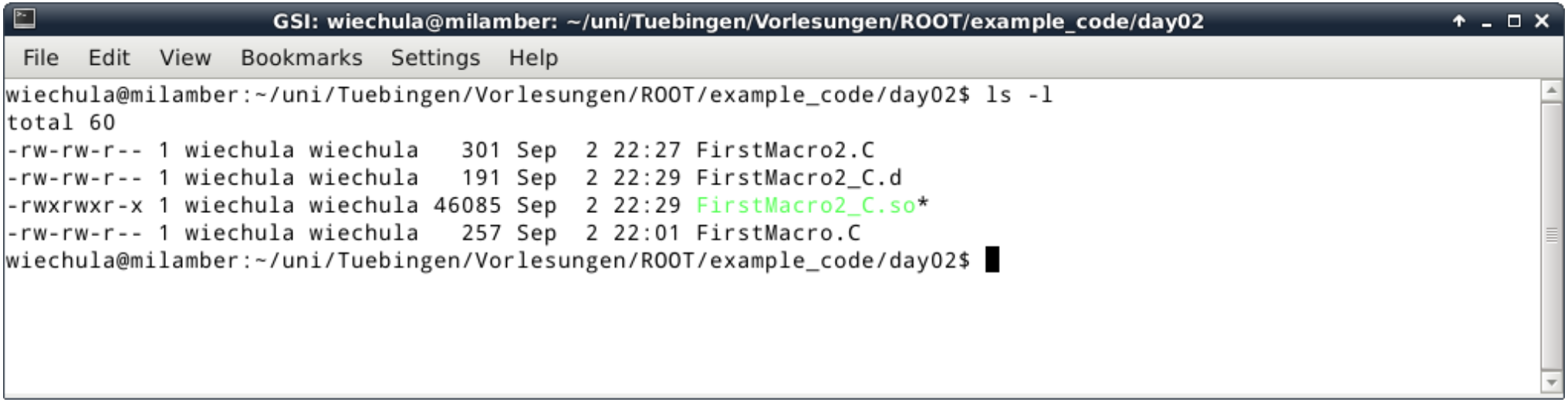
```
GSI: wiechula@milamber: ~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day02
File  Edit  View  Bookmarks  Settings  Help
wiechula@milamber:~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day02$ root -l
root [0] .L FirstMacro2.C+
Info in <TUnixSystem::ACLiC>: creating shared library /home/wiechula/uni/Tuebingen/Vorlesungen/ROOT/example_cod
e/day02/./FirstMacro2_C.so
root [1] FirstMacro2()
My first macro
root [2] float val=myFunctionOne(2,3.5)
2 * 3.5 = 7
root [3] val
(float)7.00000000000000000e+00
root [4]
```

FirstMacro2.C

Macros

ACLiC – example

- ACLiC creates two new files for each compiled macro
  - A dependency file '_C.d' – keeps track of header files and code used for the compilation
  - A shared object file '_C.so' – keeps the compiled code



- ROOT6: a file with extension “pcm” is also created. This latter file is fundamental for the correct functioning of the dictionary at runtime.

Data types

- ROOT provides aliases for the standard data types for better cross-platform compatibility
- E.g. int → Int_t, float → Float_t, etc.
- Its use is not necessary, but recommended, especially for larger projects

Histograms

Introduction

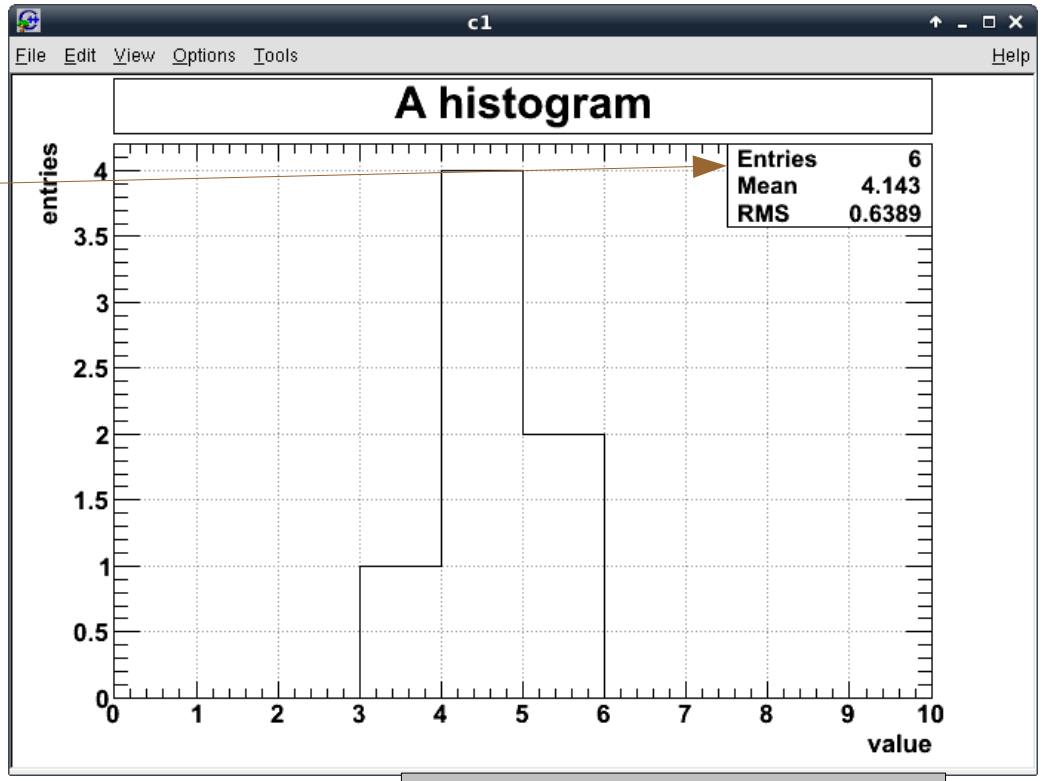
- Histograms are one of the most important tools to represent and analyse data
- They 'bin' continuous distributions to visualise frequency distributions
- ROOT offers classes to handle, draw, fit histograms in 1d, 2d, 3d, but also arbitrary dimensions
- Histograms are used for any graphical output requiring axes

Histograms

A first example

```
root [0] TH1F myFirstHisto("myFirstHisto","A histogram;value;entries",10,0,10);
root [1] myFirstHisto.Fill(3);
root [2] myFirstHisto.Fill(4);
root [3] myFirstHisto.Fill(4);
root [4] myFirstHisto.Fill(4);
root [5] myFirstHisto.Fill(4);
root [6] myFirstHisto.Fill(5,2);
root [7] myFirstHisto.Draw("HIST");
```

NB: in ROOT6 one needs to specify the option "HIST" to have the representation without errors.
If you do not use it, errors will be displayed.
Note that the errors are not poissonian when you fill with weights as here. If using ROOT6, use this options in the examples in the next slides



histogram_examples.txt

Stat box:
- Shown per default after drawing
- Default settings (can be changed via gStyle->SetOptStat(...)):
  Show entries, mean and RMS
- CAREFUL:
  RMS it NOT the root mean square:

$$x_{\mathrm{rms}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}.$$

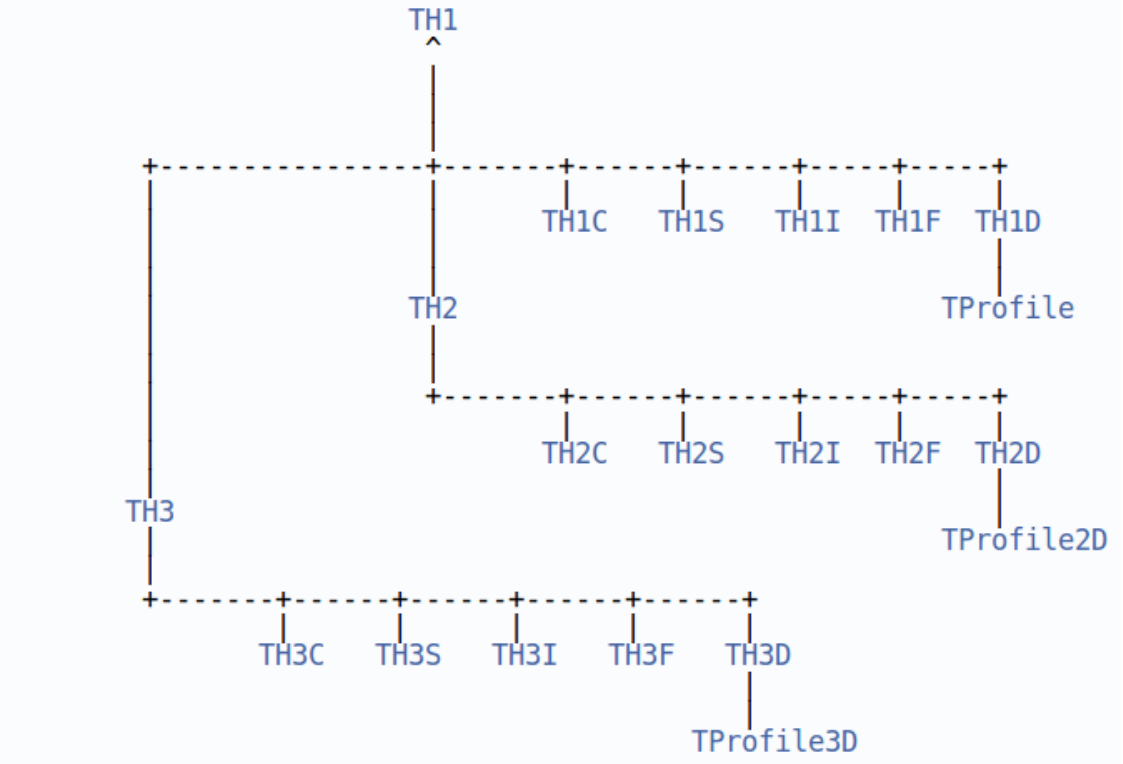
but the standard deviation (sigma):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Histograms

The TH1 family

http://root.cern.ch/root/html/TH1.html



- TH1 is the base class of all 1d,2d,3d histograms
  - They can all use the same interface

Histograms

Constructors – example from TH1F

TH1<T>(const char* **name**, const char* **title**, Int_t **nbinsx**, Double_t **xlow**, Double_t **xup**)

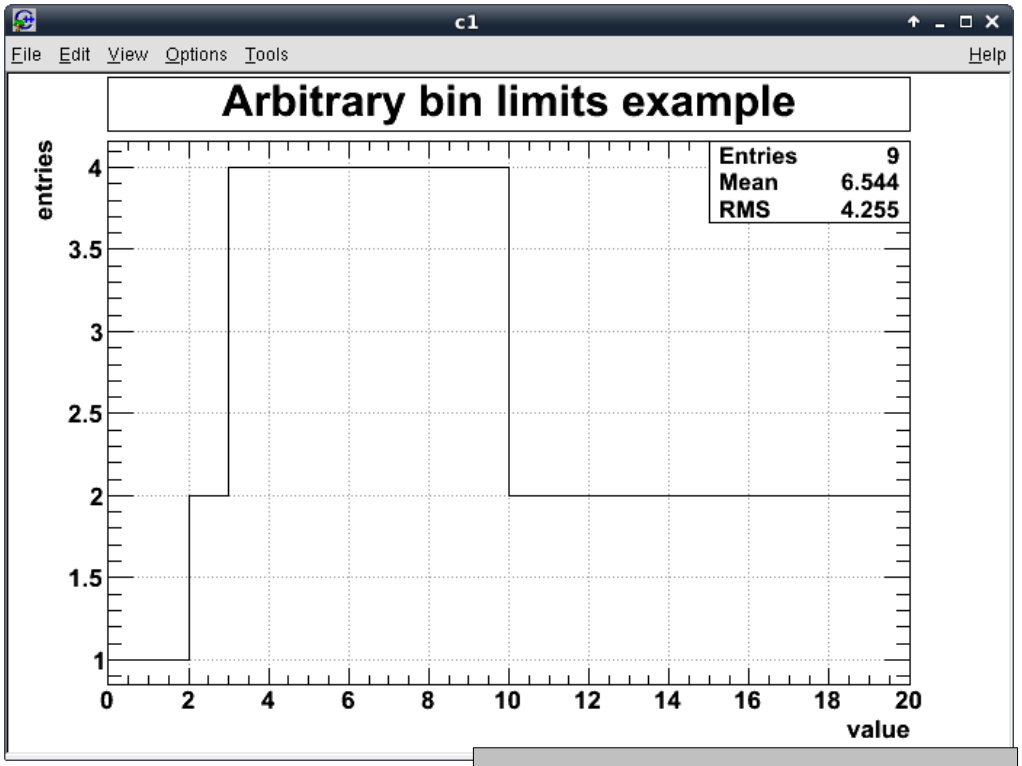
TH1<T>(const char* **name**, const char* **title**, Int_t **nbinsx**, const Float_t* **xbins**)

<T>	data type filled in the hist: C,D,F,I,S (char,double,float,int,short)
name	Many objects in ROOT are identified by a name, by convention the name and the variable are identical
title	The title shown when the histogram is drawn. Axis-titles can be separately given after a ';': “histogram title;x-axis title; y-axis title; z-axis title”
nbinsx	Number of bins on the x-axis
xlow	Lower bound on the x-axis
xup	Upper bound on the x-axis → In the case of this constructor equally spaced bins between xlow and xup will be created on the axis
xbins	Bin limits for arbitrary binning, the size of the array needs to be nbinsx+1, since the lower and upper bound need to be given

Histograms

Example for arbitrary binning

```
12 //example for arbitrary binning
13 // NOTE: the number of entries in the array is 5
14 Float_t binLimits[5]={0.,2.,3.,10.,20.};
15 //       which corresponds to 4 bins
16 //
17 //     0.     2.      3.     10.    20.
18 //     | bin1 |  bin2 | bin3 |  bin4 |
19 //
20 TH1F histo("histo","Arbitrary bin limits example;value;entries",4,binLimits);
21 histo.Fill(1);    //will go into bin1
22 histo.Fill(2);    //will go into bin2
23 histo.Fill(2.9);  //will go into bin2
24 histo.Fill(4);    //will go into bin3
25 histo.Fill(7);    //will go into bin3
26 histo.Fill(8);    //will go into bin3
27 histo.Fill(9);    //will go into bin3
28 histo.Fill(10);   //will go into bin4
29 histo.Fill(15);   //will go into bin4
30 histo.Draw();
```

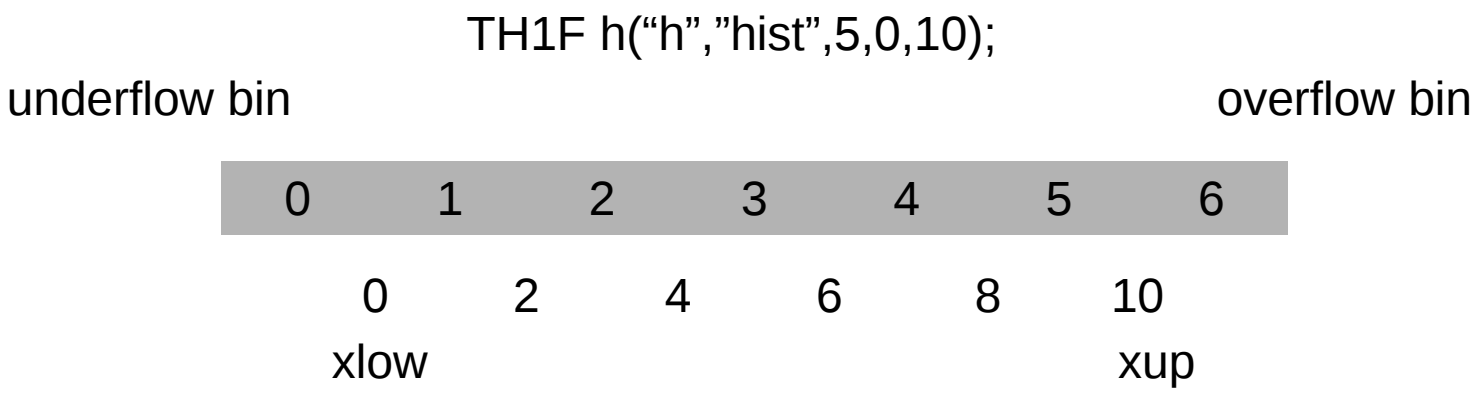


histogram_examples.txt

Histograms

More on binning

- A histogram contains two more bins than requested in the contructor (**nbinsx**+2)
  - The underflow bin: all entries smaller than **xlow**, or the smallest value in **xbins** (arbitrary binning) are filled here. Bin number = 0
  - The overflow bin: all entries larger or equal **xup**, or the largest value in **xbins** (arbitrary binning) are filled here. Bin number = nbinsx+1



Histograms

Most important functions

Filling Entries into a histogram

Int_t Fill(Double_t **x**)

Int_t Fill(Double_t **x**, Double_t **w**)

x	The value to be filled
w	A weight with which to fill the value

Acting directly on the bin content

SetBinContent(Int_t **bin**, Double_t **content**)

GetBinContent(Int_t **bin**)

SetBinError(Int_t **bin**, Double_t **error**)

GetBinError(Int_t **bin**)

bin	bin number to act on
content	content to fill to the bin
error	error to assign to a bin

Titles

SetTitle(const char* **title**)

title	see slide 21

Drawing

Draw(Option_t* **option** = "")

option	draw option, for details see http://root.cern.ch/root/html/THistPainter.html

Histograms

Most important functions

Math operations on histograms

Add(TF1* **f1**, Double_t **c1** = 1)

Add(const TH1* **h1**, Double_t **c1** = 1)

Multiply(TF1* **h1**, Double_t **c1** = 1)

Multiply(const TH1* **h1**)

Divide(TF1* **f1**, Double_t **c1** = 1)

Divide(const TH1* **h1**)

// To scale the histogram

// (both contents and errors are scaled)

// - e.g. for normalisation

Scale(Double_t **c1** = 1, Option_t* **option** = "")

f1 function, evaluated at the bin centre

h1 histogram to be added/multiplied/divided

c1 normalisation value multiplied to f1 or h1 or the histogram itself (for Scale)

option If it contains "width", bin content and error are divided by the bin width

Important:

BEFORE filling the histogram, call *Sumw2()* to create structure for squares of weights. It is required for proper error propagation in such math operations!

Histograms

A few examples

GSI: wiechula@milamber: ~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day02

File Edit View Bookmarks Settings Help

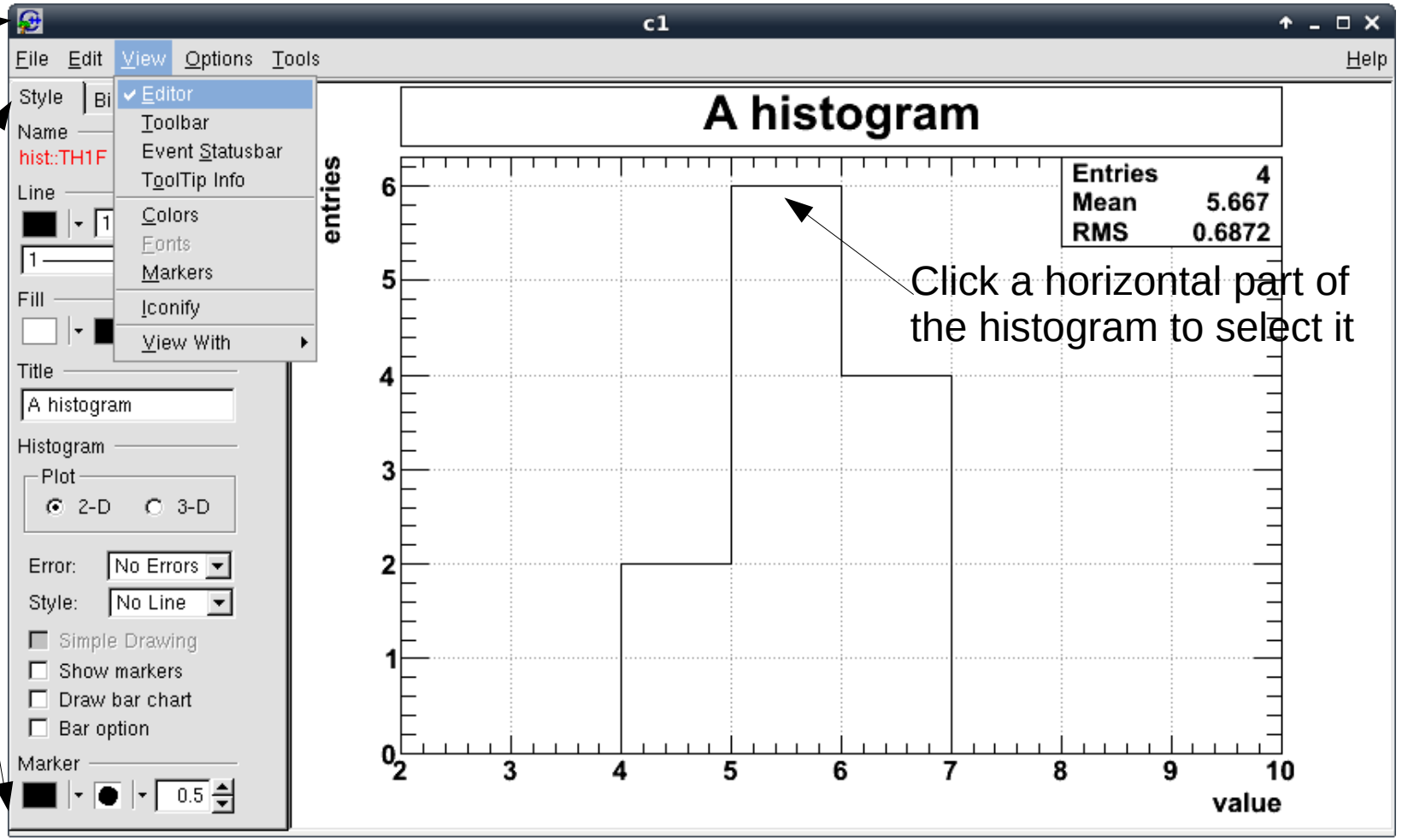
```
wiechula@milamber:~/uni/Tuebingen/Vorlesungen/ROOT/example_code/day02$ r
root [0] TH1F histo("hist","A histogram;value;entries",8,2,10);
root [1] histo.SetBinContent(3,2);
root [2] histo.SetBinContent(4,6);
root [3] histo.SetBinContent(5,4);
root [4]
root [4] histo.Fill(0);
root [5]
root [5] histo.GetBinContent(0)
(const Double_t)1.00000000000000000e+00
root [6] histo.GetBinContent(4)
(const Double_t)6.00000000000000000e+00
root [7]
```

histogram_examples.txt

Histograms

Changing the style – using the gui 1

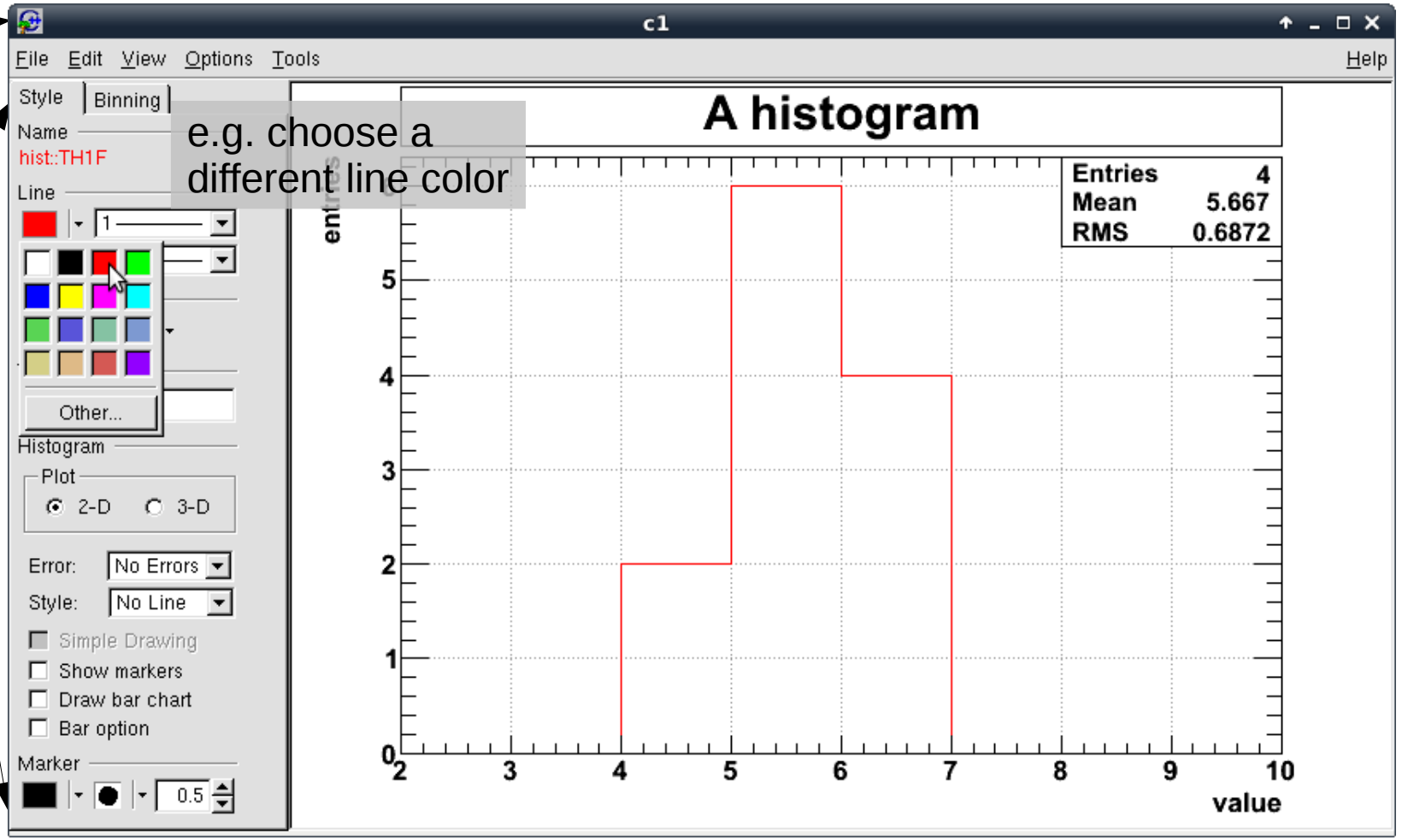
- The easiest way to change the style is to use the editor (Check: View → Editor – in the menu of the canvas)



Histograms

Changing the style – using the gui 2

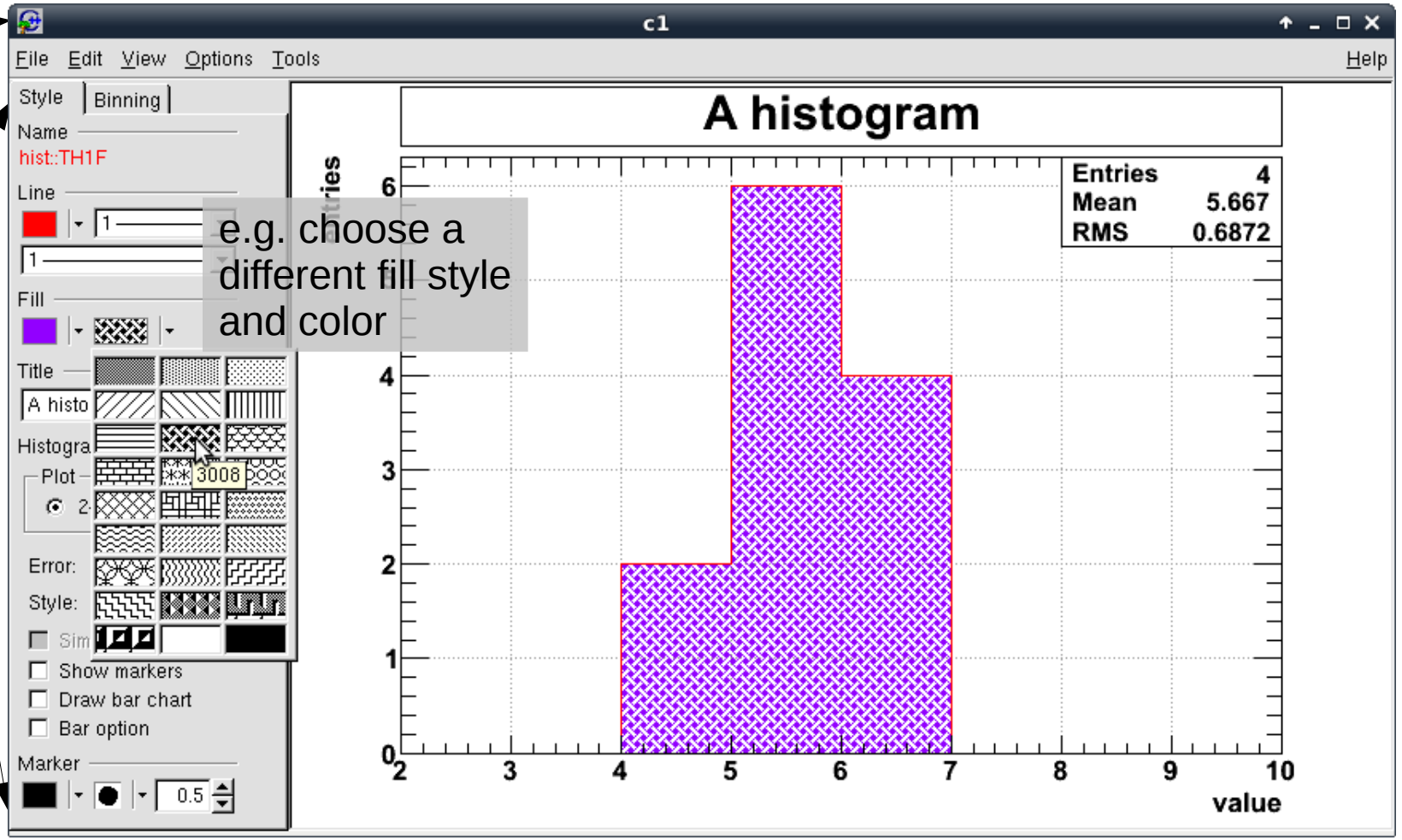
- The easiest way to change the style is to use the editor (Check: View → Editor – in the menu of the canvas)



Histograms

Changing the style – using the gui 3

- The easiest way to change the style is to use the editor (Check: View → Editor – in the menu of the canvas)

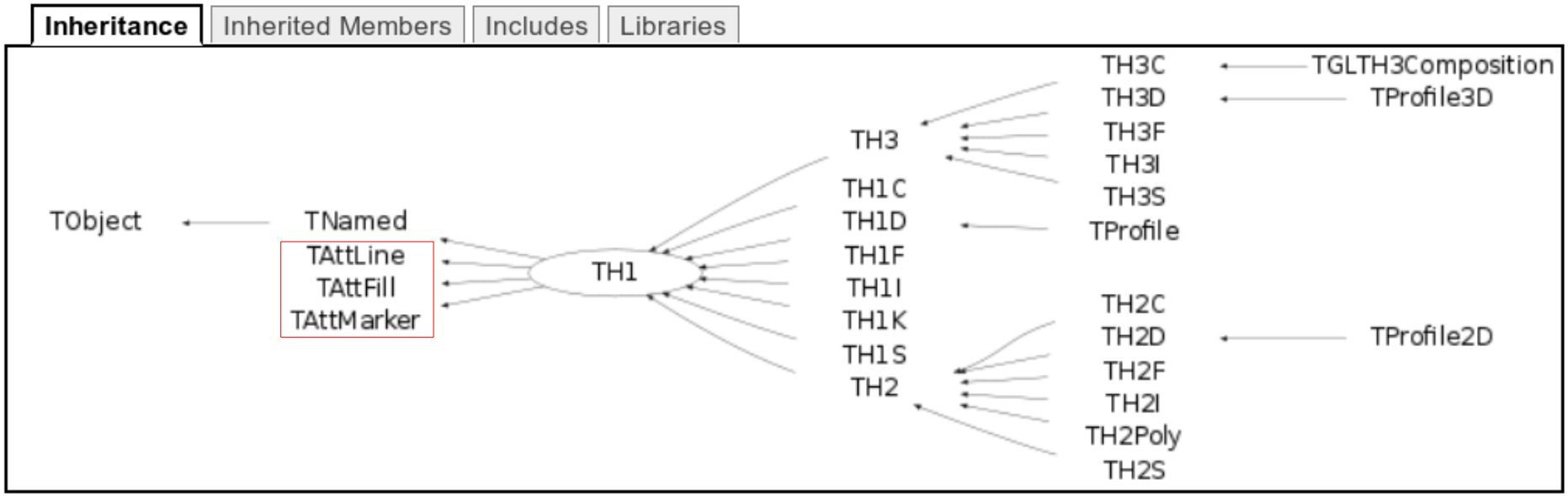


Histograms

Changing the style – using class functions

- Often histograms are created inside a macro, then it is impractical to always use the gui
- The histogram classes inherit from attribute classes (TAttLine, TAttFill, TAttMarker) → Those have functions to change the style

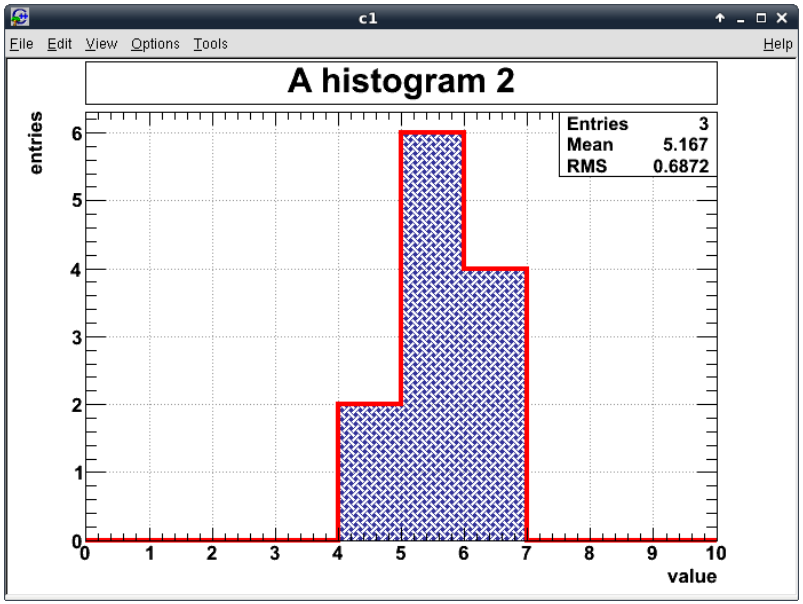
Class Charts http://root.cern.ch/root/html/TH1.html



Histograms

Changing the style – using class functions - examples

```
1  void histo_style()
2  {
3    // Attention the histogram below will not show up in
4    // the canvas, because at the end of the function
5    // it goes out of scope and is deleted
6    TH1F histo("histo","A histogram",10,0,10);
7    histo.Fill(4,2);
8    histo.Fill(5,6);
9    histo.Fill(6,4);
10
11   histo.SetLineColor(kRed);
12   histo.SetFillStyle(3008);
13   histo.SetFillColor(kBlue-2);
14   histo.Draw();
15
16
17   // In order to get a histogram which will stay on the canvas
18   // you need to create it with 'new'
19   // NOTE: the acess operator changed
20   TH1F *histo2 = new TH1F("histo2","A histogram 2",10,0,10);
21   histo2->Fill(4,2);
22   histo2->Fill(5,6);
23   histo2->Fill(6,4);
24 ..
25   histo2->SetLineColor(kRed);
26   histo2->SetLineWidth(4);
27   histo2->SetFillStyle(3008);
28   histo2->SetFillColor(kBlue-2);
29   histo2->Draw();
30 }
```

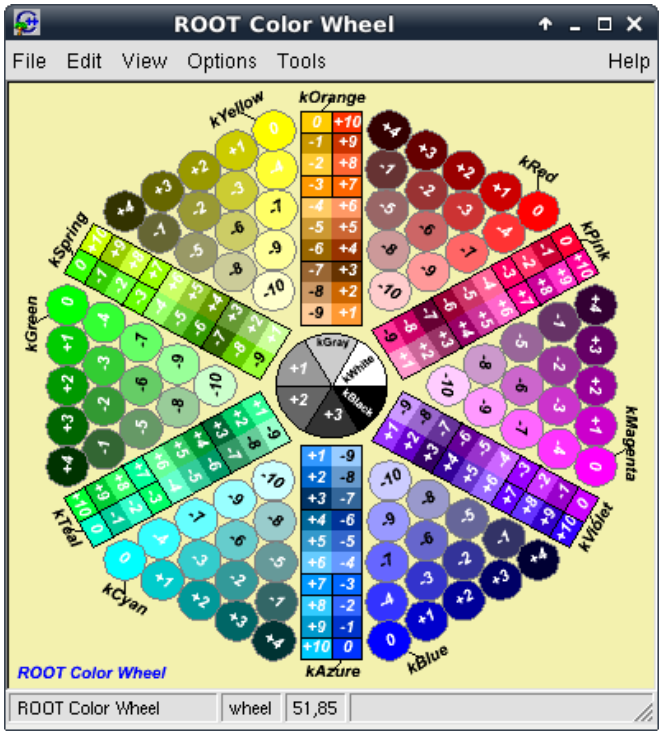


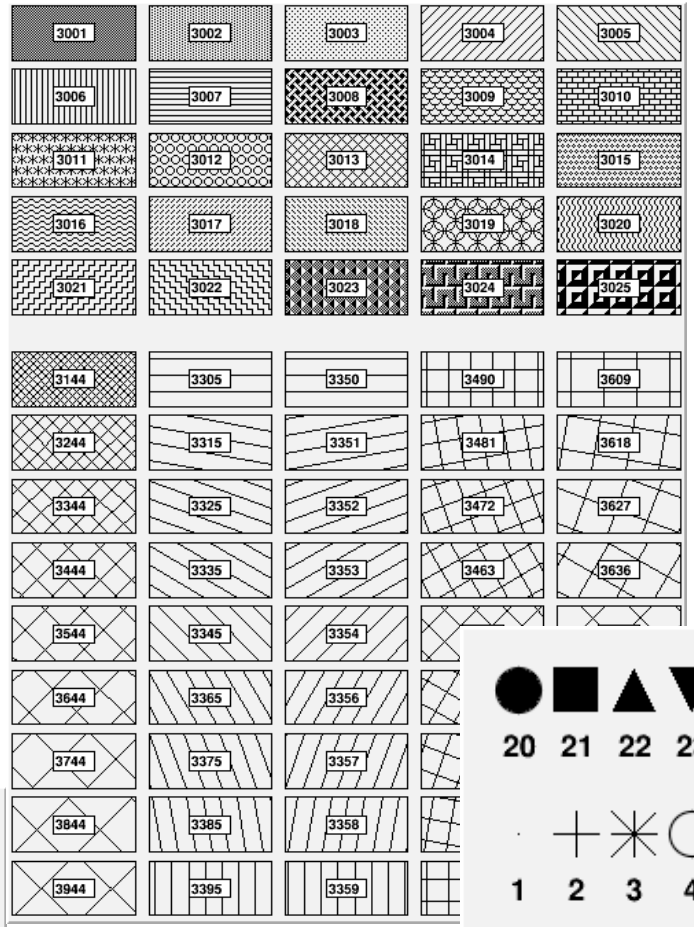
histo_style.C

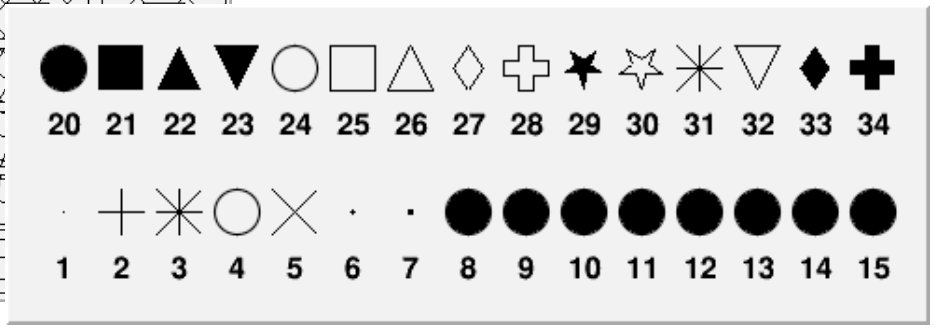
Histograms

Introduction continued

- Many classes in ROOT inherit from the basic attribute classes. For those classes you can use the same functions to change the style







Histograms

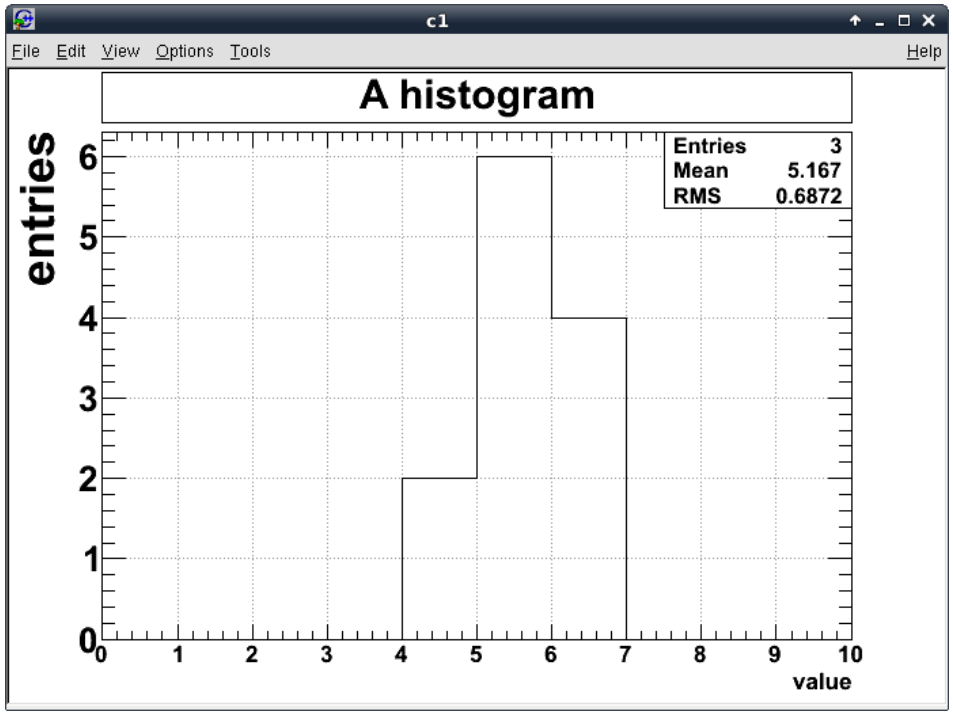
Axes

- The axes of histograms are own classes (TAxis) you get them e.g. with GetXaxis()
- Most important properties are:
  - Title (SetTitle(...))
  - Distance of the title to the axis (SetTitleOffset(...))
  - Distance of axis labels to the axis (SetLabelOffset(...))
  - Text size of the title (SetTitleSize(...))
  - Text size of the labels (SetLabelSize(...))
  - Number of divisions (SetNdivisions(...))

Histograms

Axes – example

```
3   // don't set the axis title here, but below, directly on the axis
4   TH1F *histo = new TH1F("histo","A histogram 2",10,0,10);
5   histo->Fill(4,2);
6   histo->Fill(5,6);
7   histo->Fill(6,4);
8
9   // The histogram axis are own classes (TAxis), the can be accessed
10  // using GetXaxis(), GetYaxix(), GetZaxis()
11  TAxis *xaxis=histo.GetXaxis();
12  TAxis *yaxis=histo.GetYaxis();
13
14  // for example change the title
15  xaxis->SetTitle("value");
16  yaxis->SetTitle("entries");
17  // for example change the size and offset of the
18  yaxis->SetTitleSize(0.08);
19  yaxis->SetTitleOffset(0.55);
20  // for example change the label size
21  yaxis->SetLabelSize(0.06);
22 ..
23  histo->Draw();
```

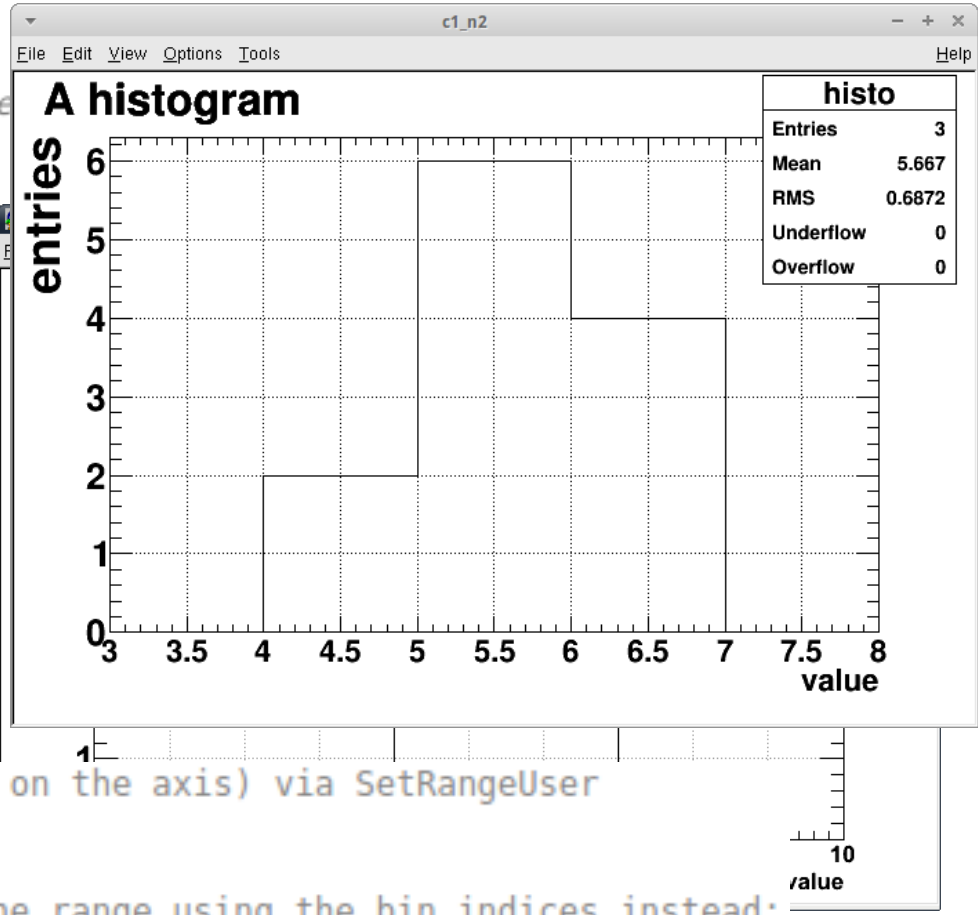


histo_axis.C

Histograms

Axes – setting the range

```
3   // don't set the axis title here, but below, directly on the axis
4   TH1F *histo = new TH1F("histo","A histogram 2",10,0,10);
5   histo->Fill(4,2);
6   histo->Fill(5,6);
7   histo->Fill(6,4);
8
9   // The histogram axis are own classes (TAxis), the
10  // using GetXaxis(), GetYaxix(), GetZaxis()
11  TAxis *xaxis=histo.GetXaxis();
12  TAxis *yaxis=histo.GetYaxis();
13
14  // for example change the title
15  xaxis->SetTitle("value");
16  yaxis->SetTitle("entries");
17  // for example change the size and offset of the
18  yaxis->SetTitleSize(0.08);
19  yaxis->SetTitleOffset(0.55);
20  // for example change the label size
21  yaxis->SetLabelSize(0.06);
22
23  histo->Draw();
```

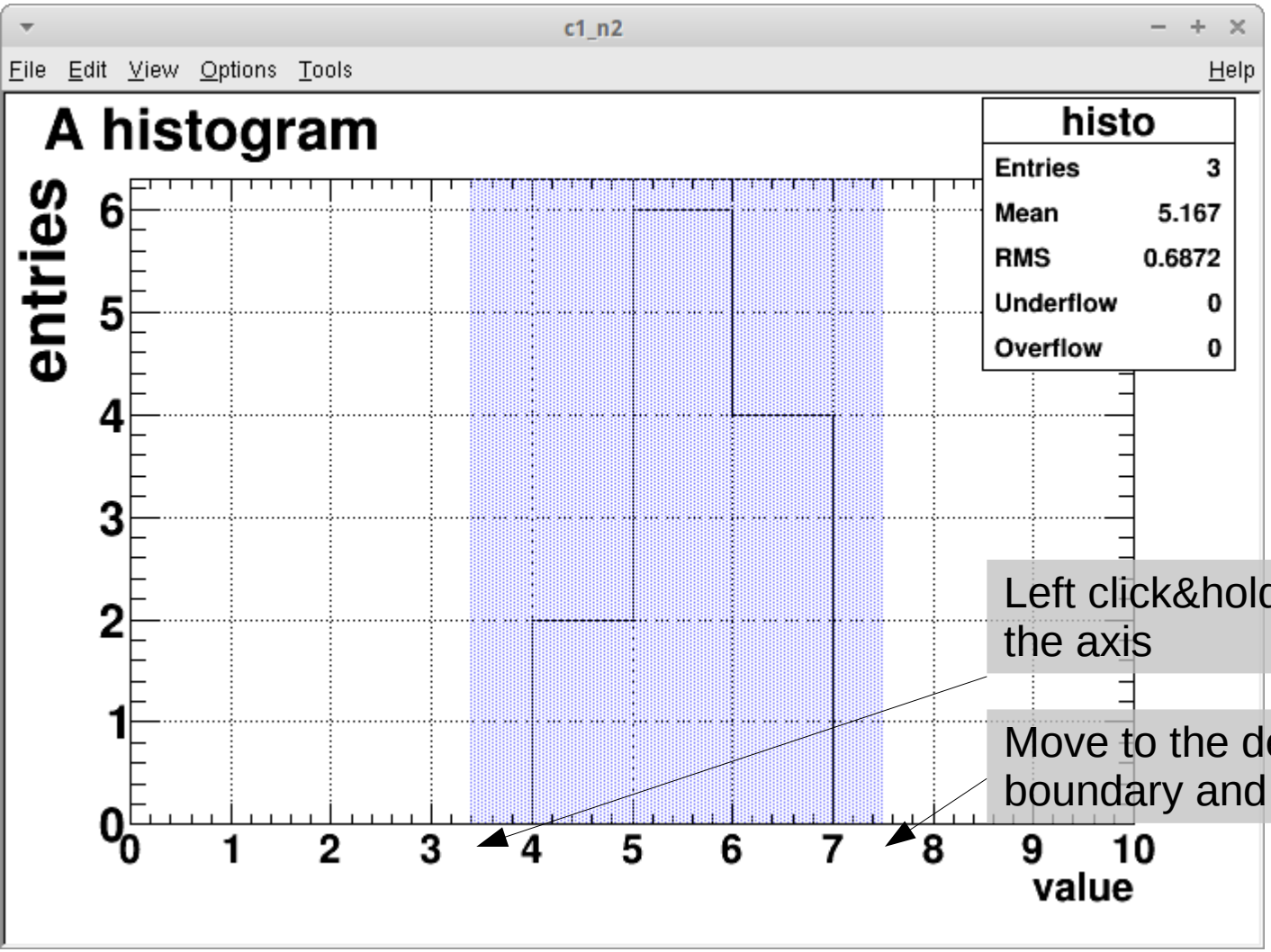



```
30  // Set the range of the axis from 3 to 8 (value on the axis) via SetRangeUser
31  xaxis->SetRangeUser(3, 8);
32
33  // Alternatively, one can use SetRange to set the range using the bin indices instead:
34  //xaxis->SetRange(4, 8);
35  histo->Draw();
```

histo_axis.C

Histograms

Axes – setting the range using the GUI



- Works for any axis, not only for x-axis

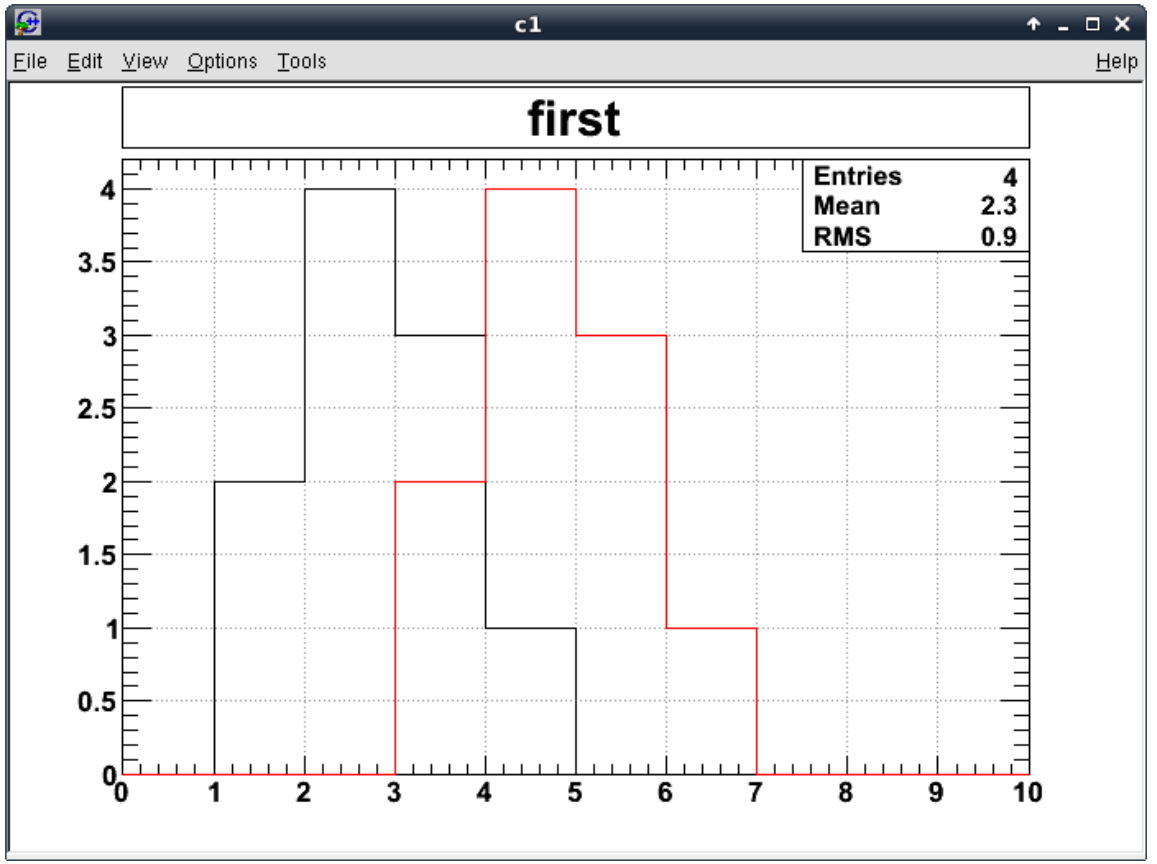
- NOTE: The boundaries of the range need to be bin edges, i.e., if a bin ranges from 3 to 4 and you select a range starting or ending at 3.x, ROOT eventually sets the range start/end to 3 or 4 (same holds true for calling SetRangeUser(...))

Histograms

Overlaying histograms

- Several histograms can be drawn on top of each other. This can be done using the draw option 'same'

```
45 // draw 2 histograms
46 TH1F *h1 = new TH1F("h1","first",10,0,10);
47 h1->Fill(1,2);
48 h1->Fill(2,4);
49 h1->Fill(3,3);
50 h1->Fill(4,1);
51
52 TH1F *h2 = new TH1F("h2","second",10,0,10);
53 h2->Fill(3,2);
54 h2->Fill(4,4);
55 h2->Fill(5,3);
56 h2->Fill(6,1);
57 h2->SetLineColor(kRed);
58
59 h1->Draw();
60 h2->Draw("same");
```



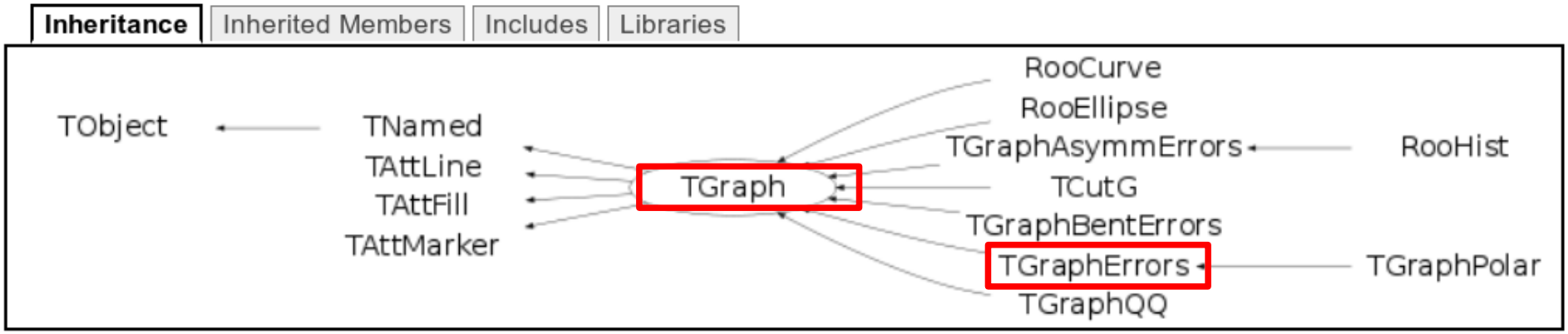
histogram_examples.C

Graphs

Basics – the TGraph family

- Graphs are simple xy plots
- The style can be changed as for histograms

Class Charts http://root.cern.ch/root/html/TGraph.html

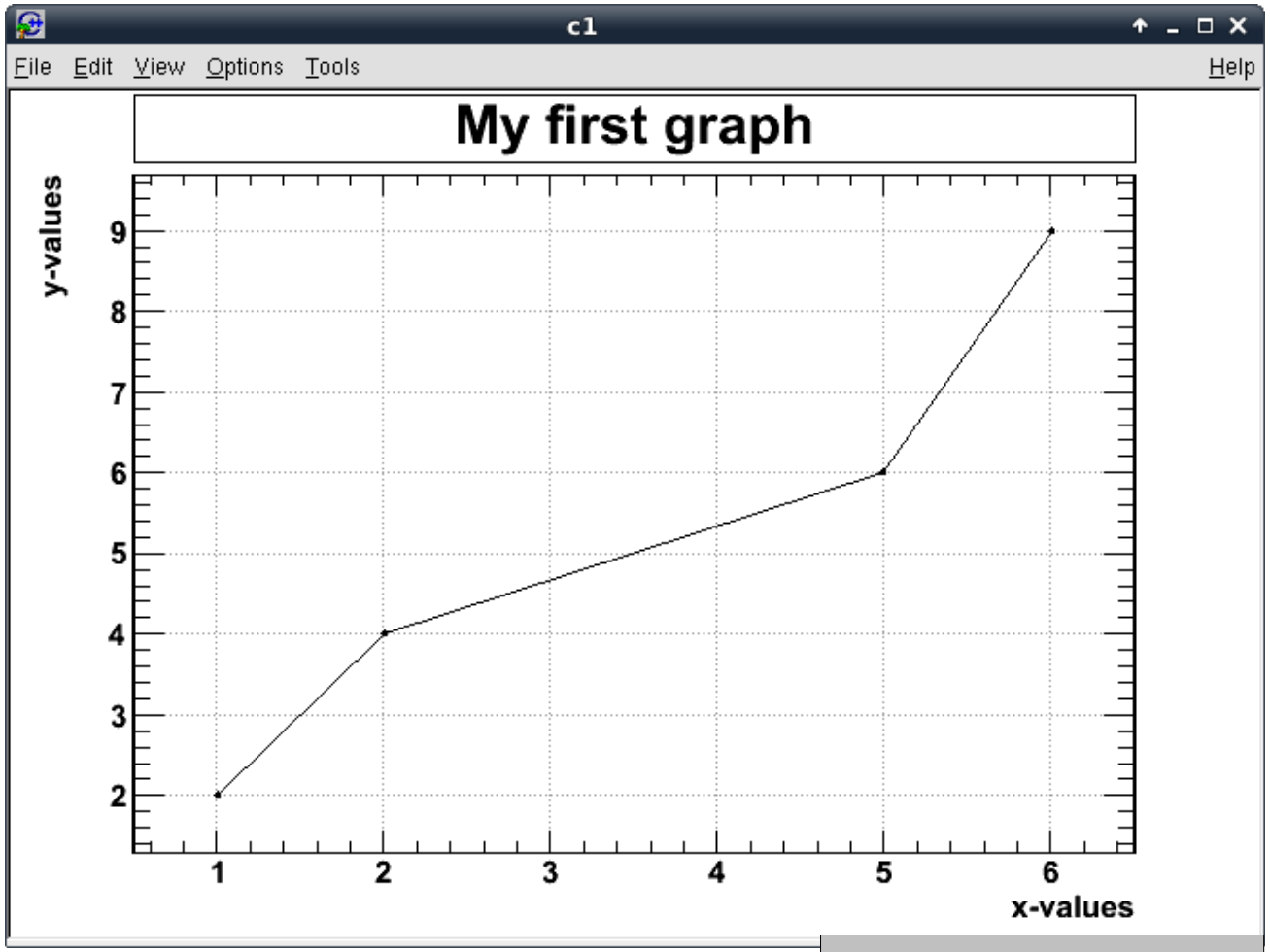


- The most important derived class is TGraphErrors a graph with error bars on the x and y values

Graphs

A first example

```
1 // first example for a graph
2 TGraph gr;
3 gr.SetTitle("My first graph;x-values;y-values");
4 gr.SetPoint(0,1.,2.);
5 gr.SetPoint(1,2.,4.);
6 gr.SetPoint(2,5.,6.);
7 gr.SetPoint(3,6.,9.);
8 gr.Draw("alp");
```

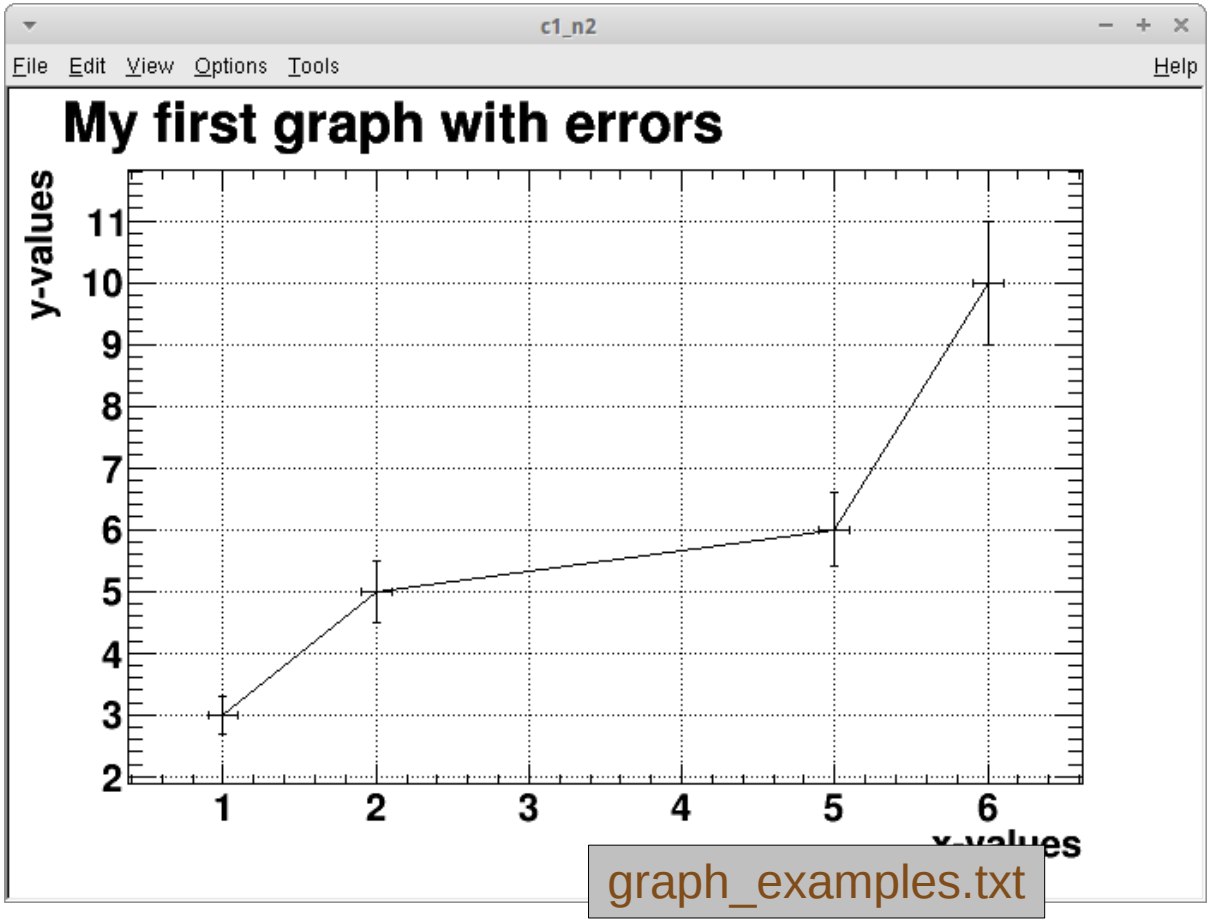


graph_examples.txt

Graphs

A simple example for graphs with errors

```
10  // first example for a graph with errors
11  new TCanvas;
12  TGraphErrors grErr;
13  grErr.SetTitle("My first graph with errors;x-values;y-values");
14  grErr.SetPoint(0,1.,3.);
15  grErr.SetPointError(0,0.1,0.3);
16  grErr.SetPoint(1,2.,5.);
17  grErr.SetPointError(1,0.1,0.5);
18  grErr.SetPoint(2,5.,6.);
19  grErr.SetPointError(2,0.1,0.6);
20  grErr.SetPoint(3,6.,10.);
21  grErr.SetPointError(3,0.1,1.0);
22  grErr.Draw("alp");
```



graph_examples.txt

Note: The TGraphErrors has the data points AND the error.

Common misunderstanding: TGraph with data points and then an additional TGraphErrors only containing the errors

Graphs

Constructors

TGraph()

TGraph(Int_t **n**)

TGraph(Int_t **n**, const Double_t* **x**, const Double_t* **y**)

And a few more constructors

TGraphErrors()

TGraphErrors(Int_t **n**)

TGraphErrors(Int_t **n**, const Double_t* **x**, const Double_t* **y**, const Double_t* **ex** = 0, const Double_t* **ey** = 0)

n	Number of points in the graph. If not given the internal arrays will be automatically extended if a new point is added with 'AddPoint'
x	array of x values of size **n**
y	array of y values of size **n**
ex	array of errors for the x values of size **n**
ey	array of errors for the y values of size **n**

Graphs

Most important functions

Setting points and errors

SetPoint(Int_t **i**, Double_t **x**, Double_t **y**)

SetPointError(Int_t **i**, Double_t **ex**, Double_t **ey**)

Getting point values

GetPoint(Int_t **i**, Double_t& **x**, Double_t& **y**)

GetErrorX(Int_t **i**)

GetErrorY(Int_t **i**)

Titles

SetTitle(const char* **title**)

Drawing

Draw(Option_t* **option** = "")

i	The point number
x	x-value of the point
y	y-value of the point
ex	x-error of the point
ey	y-error of the point

Note: Use TGraphErrors if you want errors and corresponding functions!

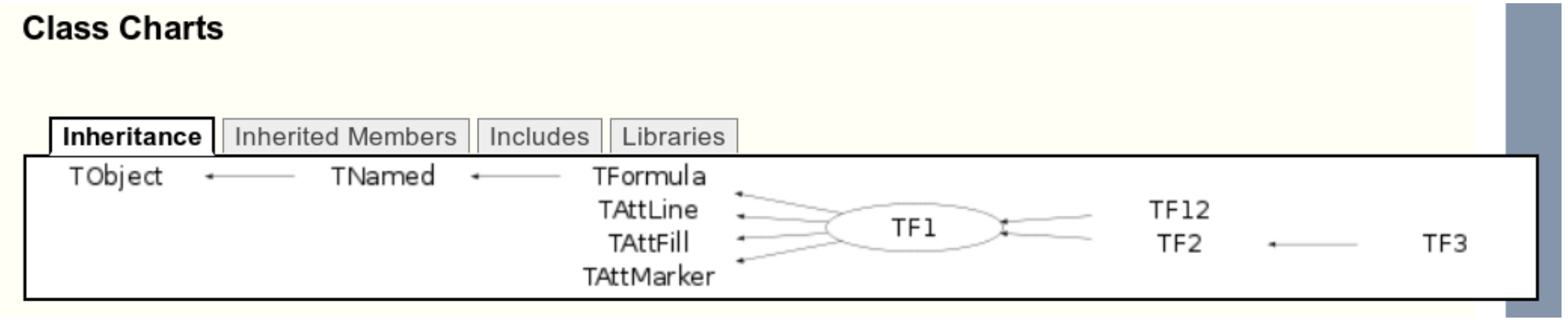
title	As for TH1 - see slide 22

option	draw option a:axis, l:line, p:marker for details see class docu

Functions

Introduction

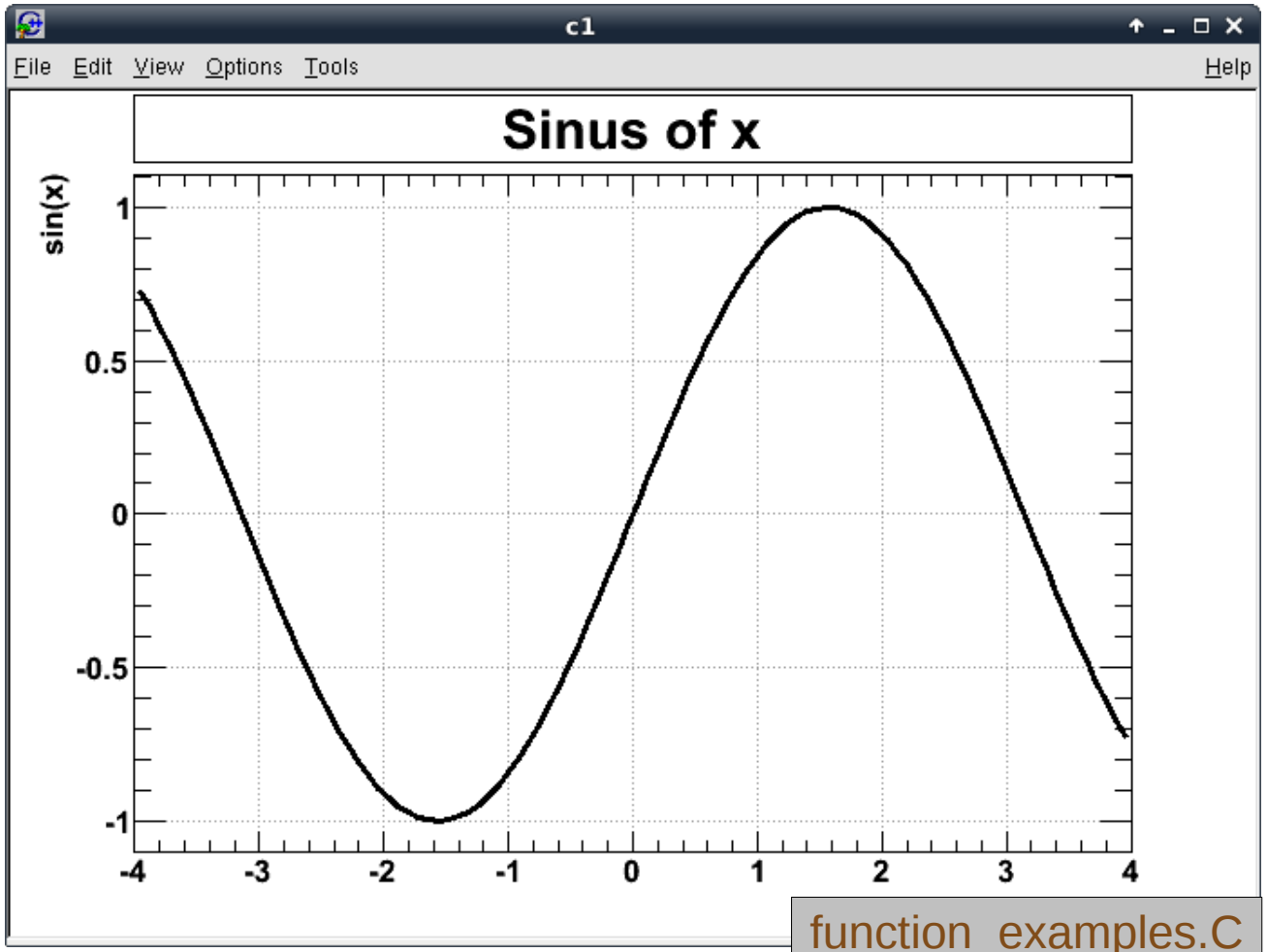
- Arbitrary definitions of functions can be drawn and used for fitting histograms and graphs
- The classes are TF1, TF2, TF3 (1-3 dimensions)
- The style can be changed as for histograms



Functions

A first example

```
1 // first example for a function
2 TF1 f("muFunction","sin(x)",-4,4);
3 f.SetTitle("Sinus of x;x;sin(x)");
4 f.Draw();
```



function_examples.C

Functions

Constructors

TF1(const char* **name**, const char* **formula**, Double_t **xmin** = 0, Double_t **xmax** = 1)

TF1(const char* **name**, void* **fcn**, Double_t **xmin**, Double_t **xmax**, Int_t **npar**)

name name of the function, used as identifier

formula function definition

xmin minimal x value to start plotting from

xmax maximum x value until which to plot

fcn pointer to a c-function

npar number of parameters in the c-function

Functions

Parameters

- Functions can carry any number of parameters, they are added in the function definition with [<parnum>], e.g.

```
TF1 f("myFunction","[0]+[1]*sin(x+[2])",-4,4);
```

has three parameters

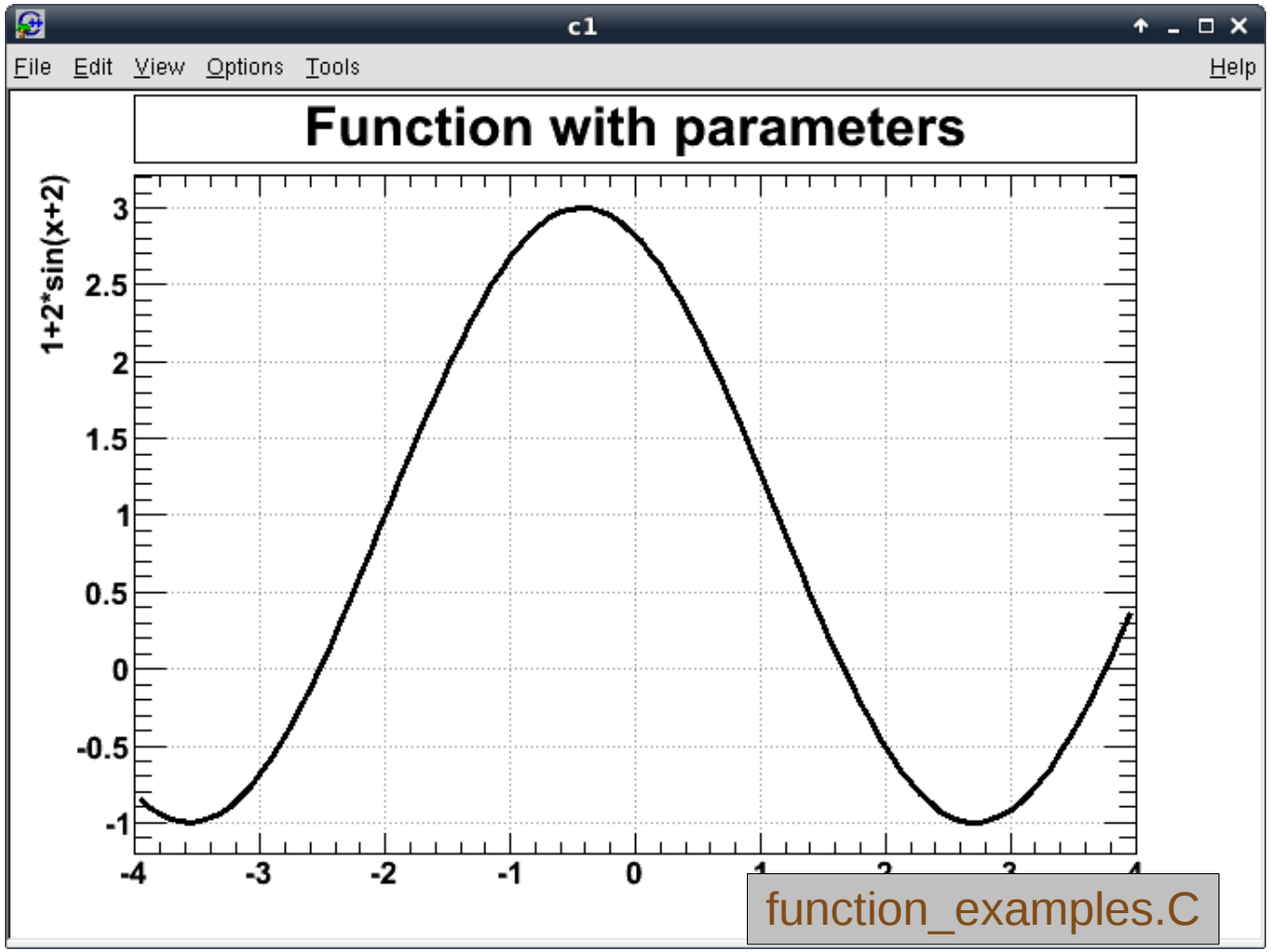
- Parameters can be set using the functions

```
SetParameter(Int_t ipar, Double_t parvalue)
SetParameters(Double_t p0, Double_t p1, Double_t p2 = 0, Double_t p3 = 0, … )
```

Functions

Parameters – example

```
7  // example with parameters
8  TF1 f("myFunction","[0]+[1]*sin(x+[2])",-4,4);
9  f.SetParameters(1,2,2);
10 f.SetTitle("Function with parameters;x;1+2*sin(x+2)");
11 f.Draw();
```



Functions

use of the namespace TMath

https://root.cern.ch/root/html524/TMath.html

- The TMath name space offers wrappers for cmath functions and other mathematical and physics functions plus constants
- These functions and constants can be used in TF1 definition

Functions

use of the namespace TMath

```
root [0] TMath::Pi()  // pi
(Double_t)3.14159265358979312e+00
root [1] TMath::E()   // eulers number
(Double_t)2.71828182845904509e+00
root [2] TMath::R()   // gas constant
(Double_t)8.31447214513609723e+00
root [3] TMath::Na()  // avogadro number
(Double_t)6.02214198999999998e+23
root [4] TMath::Abs(-0.88)
(Double_t)8.80000000000000004e-01
root [5] TMath::Gaus(0)
(Double_t)1.00000000000000000e+00
root [6] TMath::Sin(TMath::PiOver2())
(Double_t)1.00000000000000000e+00
root [7] TMath::RadToDeg()*TMath::Pi()
(double)1.80000000000000000e+02
root [8] Float_t
numbers[5]={1.,2.,3.,4.,5.}
root [9] TMath::Mean(5,numbers)
(Double_t)3.00000000000000000e+00
root [10] TMath::RMS(5,numbers)
(Double_t)1.41421356237309515e+00
```

Note once more that TMath::RMS actually gives the standard deviation, not the root mean square

And many many more …

Example of use in a TF1:
Root[] TF1 f("f","TMath::Cos(x)",-TMath::Pi(),TMath::Pi())

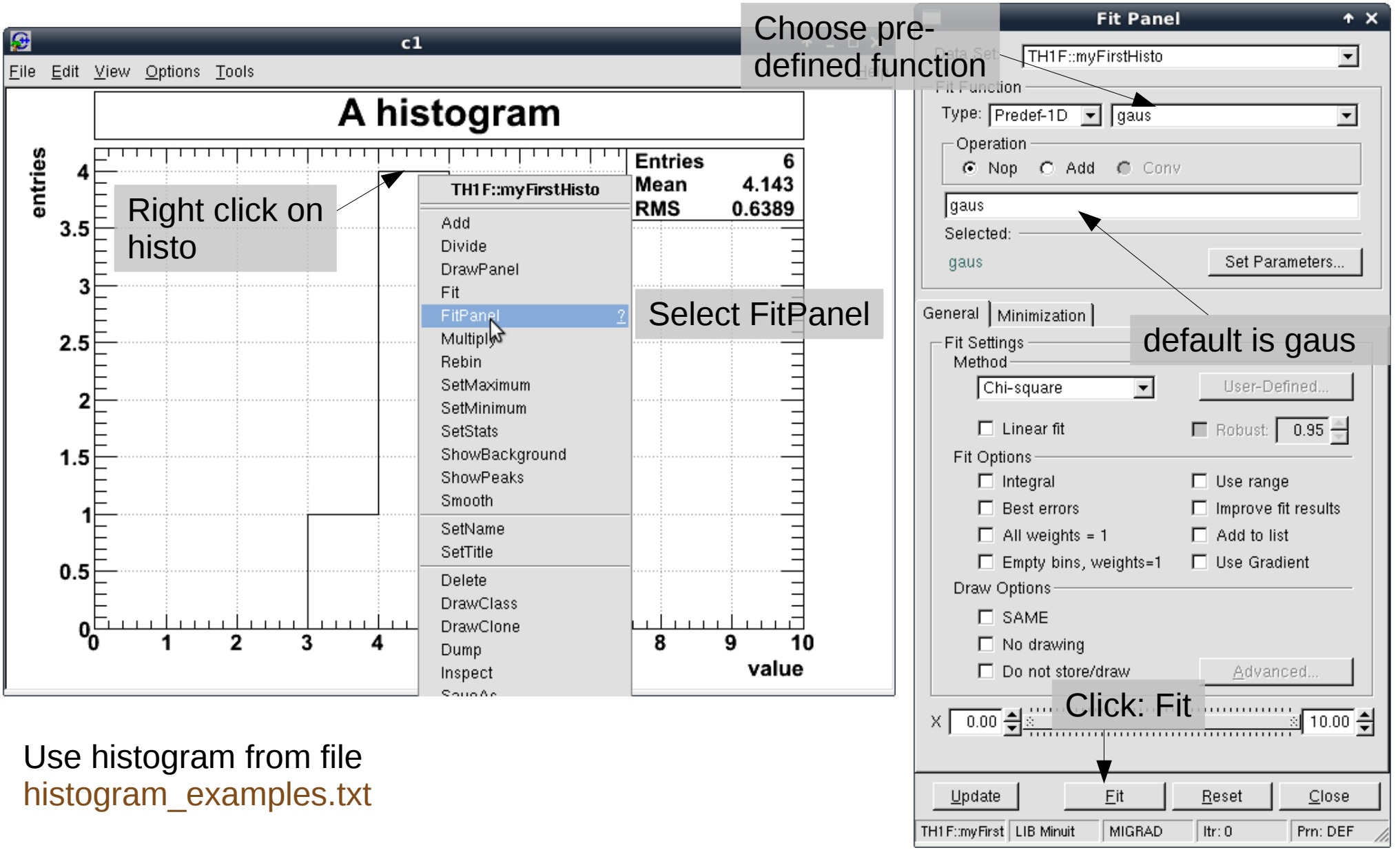
Fitting histograms and function

Introduction

- A very important method in data analysis is to be able to fit functions to a measured histogram / graph
- ROOT allows to fit any function to a histogram / graph
- As for most other things, this can be done graphically or inside the code

Fitting histograms and function

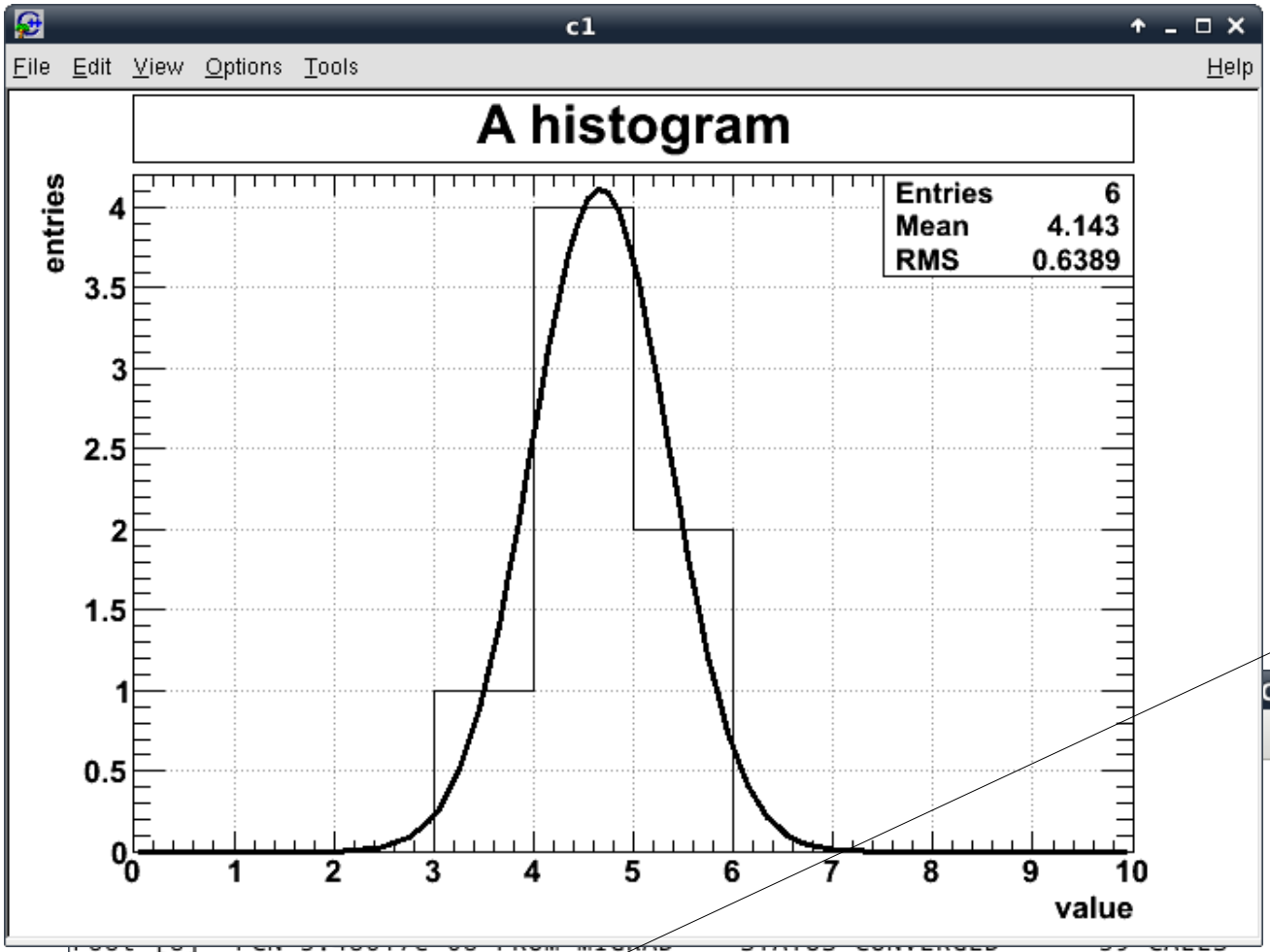
The graphical way 1



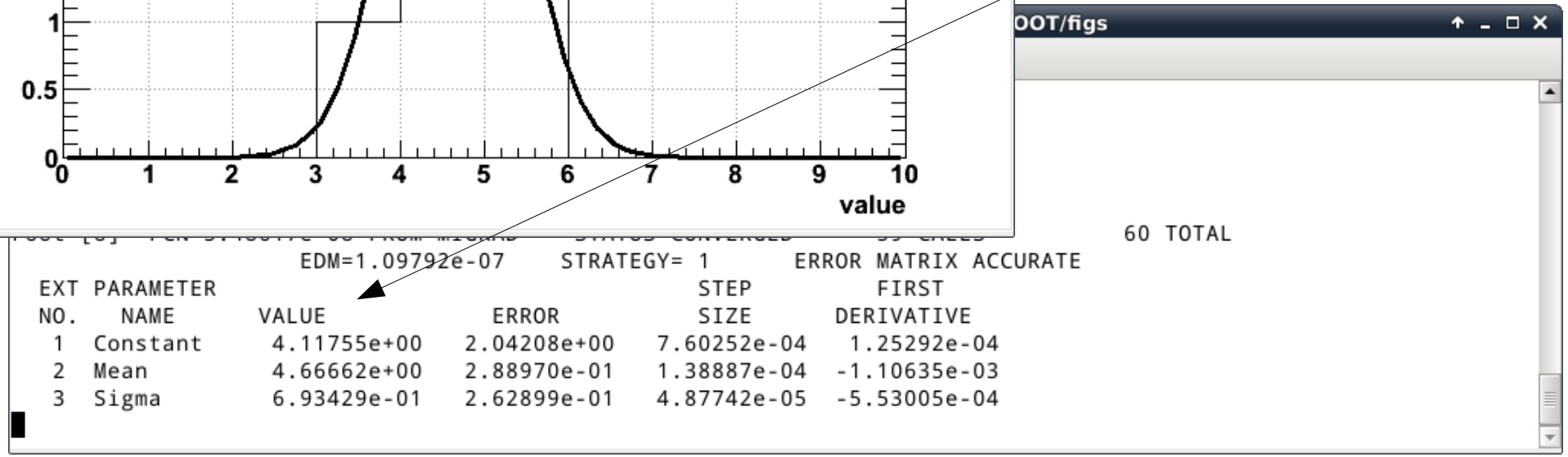
Use histogram from file
histogram_examples.txt

Fitting histograms and function

The graphical way 2



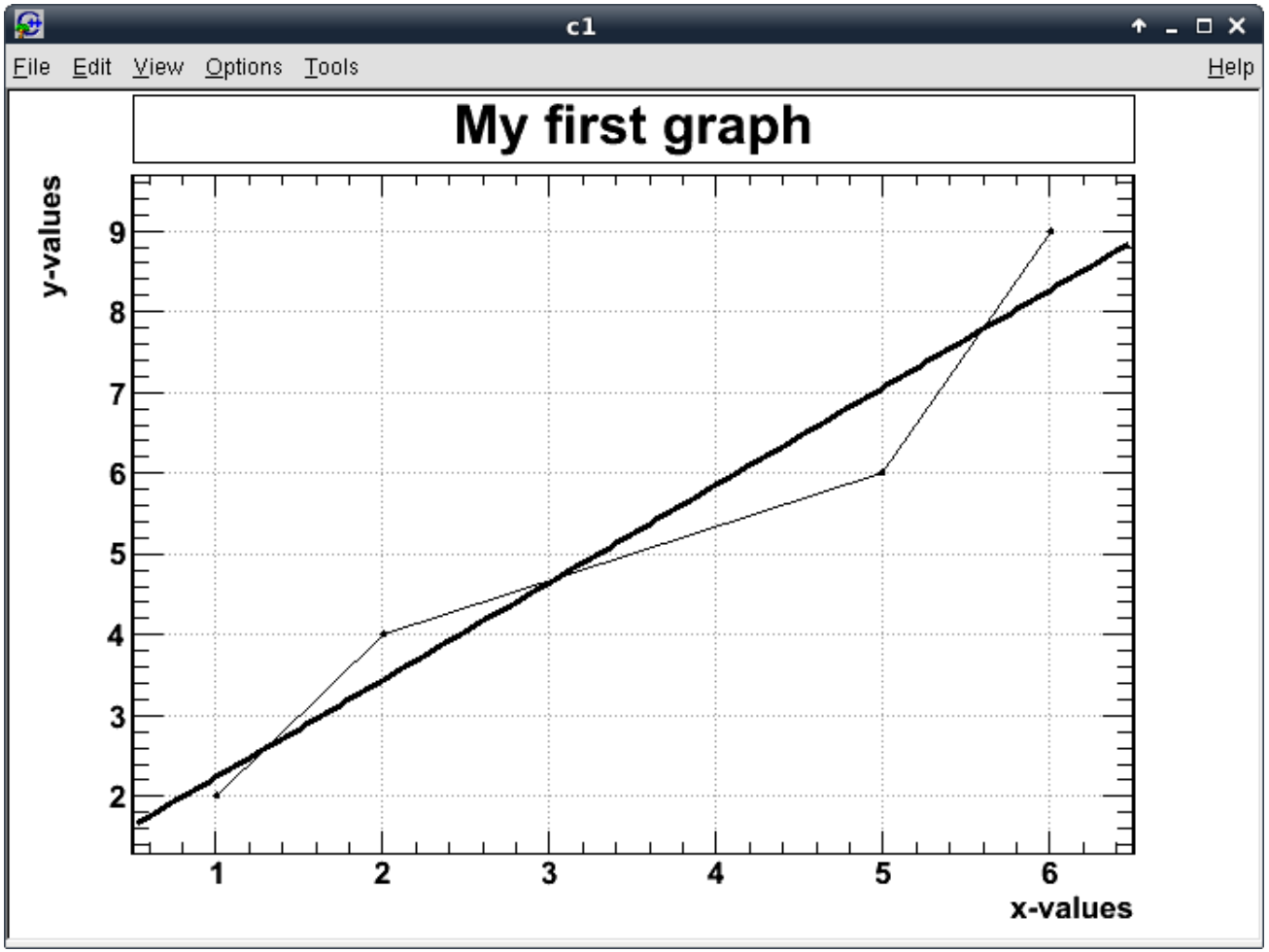
Fit results are written in the ROOT shell



Fitting histograms and function

The graphical way 3

Use the fit panel as for histograms and choose 'pol1' as a function



Use graph from file
graph_examples.txt

Fitting histograms and graphs

Inside the code

```
Fit(TF1* f1, Option_t* option = "", Option_t* goption = "", Double_t xmin = 0, Double_t xmax = 0)
```

f1 pointer to the function

option fitting option (e.g. R: use the range specified in the function)

The options slightly differ for histo and graph, for details see documentation of Fit method

goption graphics options (see web)

xmin minimal x value to start fitting from

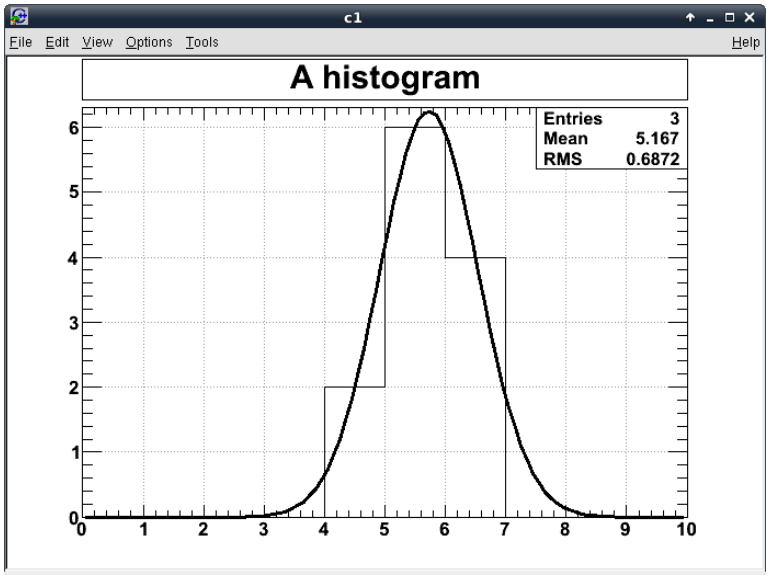
xmax maximum x value until which to fit

Fitting histograms and function

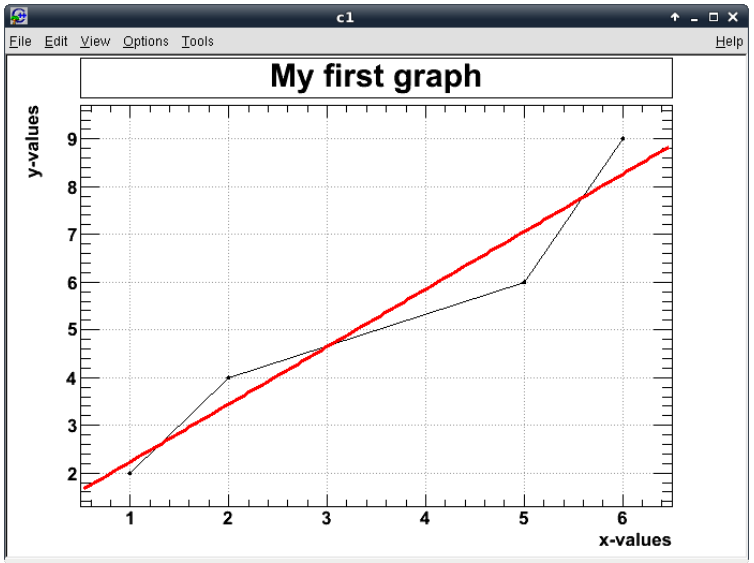
Inside the code – examples

```
53  TH1F *histo = new TH1F("histo","A histogram",10,0,10);
54  histo->Fill(4,2);
55  histo->Fill(5,6);
56  histo->Fill(6,4);
57
58  // define the function yourself
59  TF1 *myFun = new TF1("myGaus","[0]*exp( -(x-[1])**2/([2]**2) )",-10,10);
60  // you will need to give some initial parameters, this can be tedious ...
61  myFun->SetParameters(1,1,1);
62
63  histo->Fit(myFun);
64  PrintFunctionParameters(myFun);
```

```
43  // do the fit with a predefined function.
44  // This list can be found in the FitPanal of the GUI
45  histo->Fit("gaus");
```




```
70  TGraph *gr = new TGraph;
71  gr->SetTitle("My first graph;x-values;y-values");
72  gr->SetPoint(0,1.,2.);
73  gr->SetPoint(1,2.,4.);
74  gr->SetPoint(2,5.,6.);
75  gr->SetPoint(3,6.,9.);
76  gr->Draw("alp");
77
78  // define a linear function
79  TF1 *myLine = new TF1("myLine","[0]+[1]*x",0.,10.);
80  myLine->SetLineColor(kRed);
81
82  // fit the graph
83  gr->Fit(myLine);
84  PrintFunctionParameters(myLine);
```



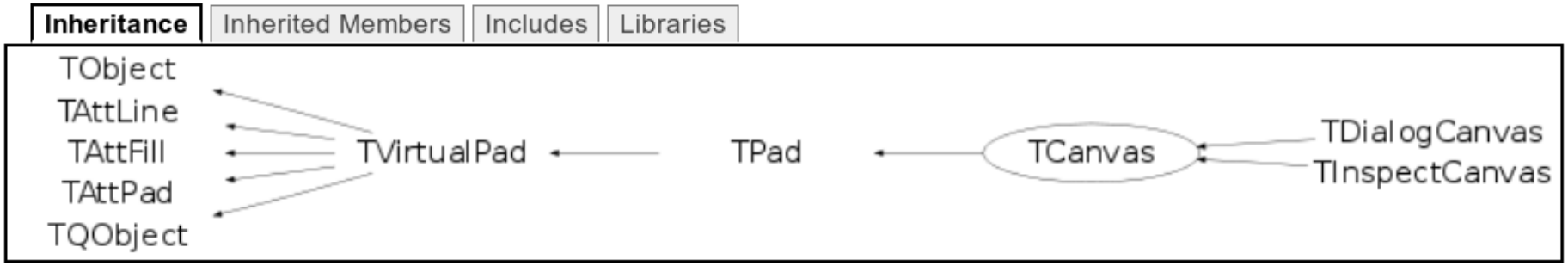
fitting.C

Canvases

Introduction

- The graphical output in ROOT is done in a so-called 'canvas' (TCanvas)
- By default, if the Draw function of an object is called, a canvas is automatically created
- For many purposes it is necessary to create canvases manually: a subdivided canvas, several canvases, …
- The graphical content is drawn on a 'pad' inside the canvas (TPad) – a canvas is a pad itself

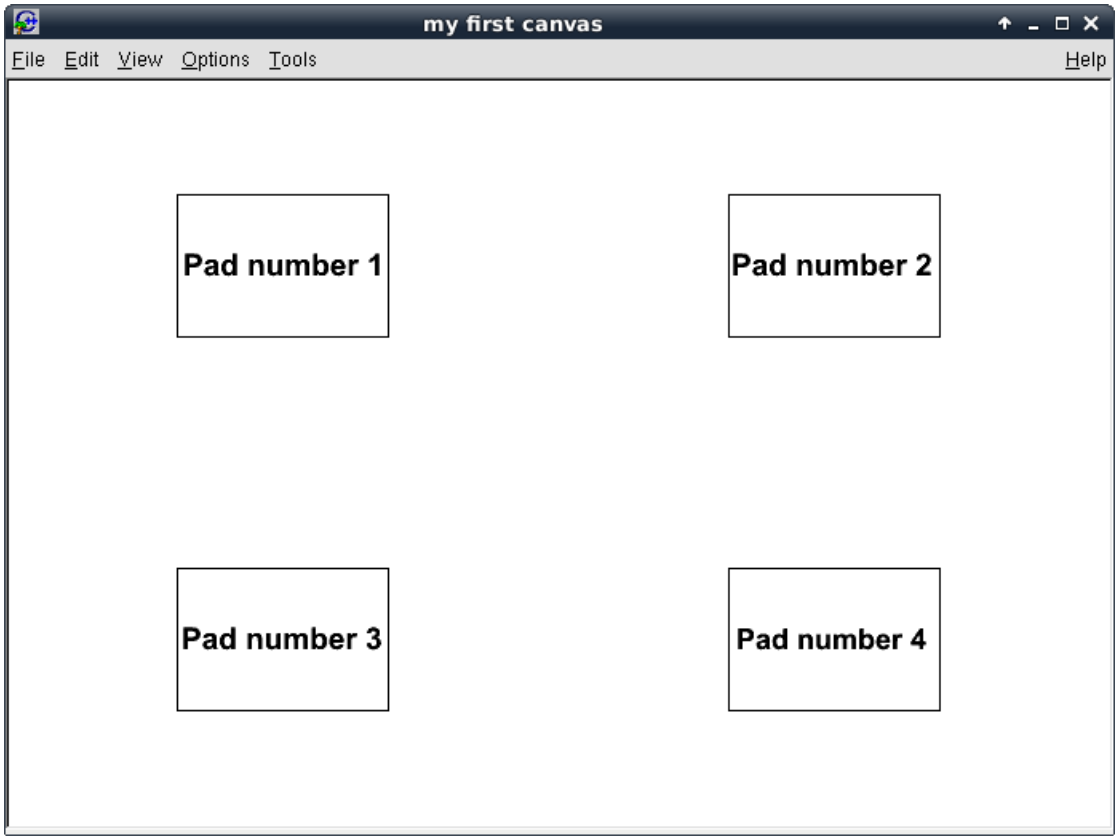
Class Charts



Canvases

A first example

```
2  TCanvas *c=new TCanvas("c","my first canvas");
3  c->Divide(2,2);
4  TPaveText t; //output text
5  for (Int_t iPad=0; iPad<4; ++iPad) {
6    // note that the sub-pads start with '1'
7    // therefore the '+1' is needed
8    // Pad '0' is the background pad
9    c->cd(iPad+1);
10   TPaveText *p=new TPaveText(.3,.3,.7,.7);
11   p->SetFillColor(10); p->SetBorderSize(1);
12   p->AddText(Form("Pad number %d",iPad+1));
13   p->Draw();
14 }
```



canvas_examples.txt

Canvases

Constructors

TCanvas(const char* **name**, const char* **title** = "", Int_t **form** = 1)
TCanvas(const char* **name**, const char* **title**, Int_t **ww**, Int_t **wh**)
TCanvas(const char* **name**, const char* **title**, Int_t **wtopx**, Int_t **wtopy**, Int_t **ww**, Int_t **wh**)

name	name of the canvas, used as identifier
title	title shown in the canvas
form	1: 700x500 at 10,10; 2: 500x500 at 20,20
ww	width in pixels
wh	height in pixels
wtopx	left corner in pixels
wtopy	top corner in pixels

Canvases

Most important functions

Divide into several sub-pads

Divide(Int_t **nx** = 1, Int_t **ny** = 1, Float_t **xm** = 0.01, Float_t **ym** = 0.01, Int_t **color** = 0)

Change to a sub-pad

cd(Int_t padnum = 0)

Empty the canvas

Clear(Option_t* **option** = "")

Change margins

SetTopMargin(Float_t **margin**)

SetBottomMargin(Float_t **margin**)

SetLeftMargin(Float_t **margin**)

SetRightMargin(Float_t **margin**)

nx number of pads in x-direction

ny number of pads in y-direction

xm space between pads in x (% of the canvas)

ym space between pads in y (% of the canvas)

color background color of the new pad (0=the same as the mother)

padnum pad number to change to, the sub-pads start with '1'

margin margin size in % of the pad size

Canvases

Pointer to the current pad (gPad)

- To access the current graphics pad in a simple way, the global pointer gPad is provided
- This e.g. allows to easily change properties of the current pad (like margins) or clear it ...

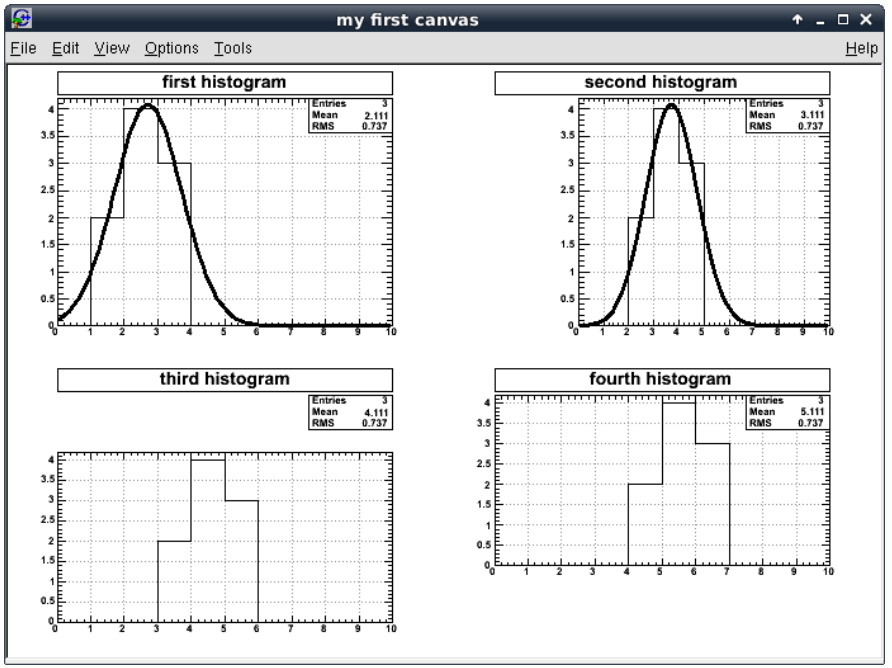
```
TCanvas *c1 = new TCanvas("c1","my first canvas");
c1->Divide(2,2);
c1->cd(1);
gPad->SetLeftMargin(0.3);
```

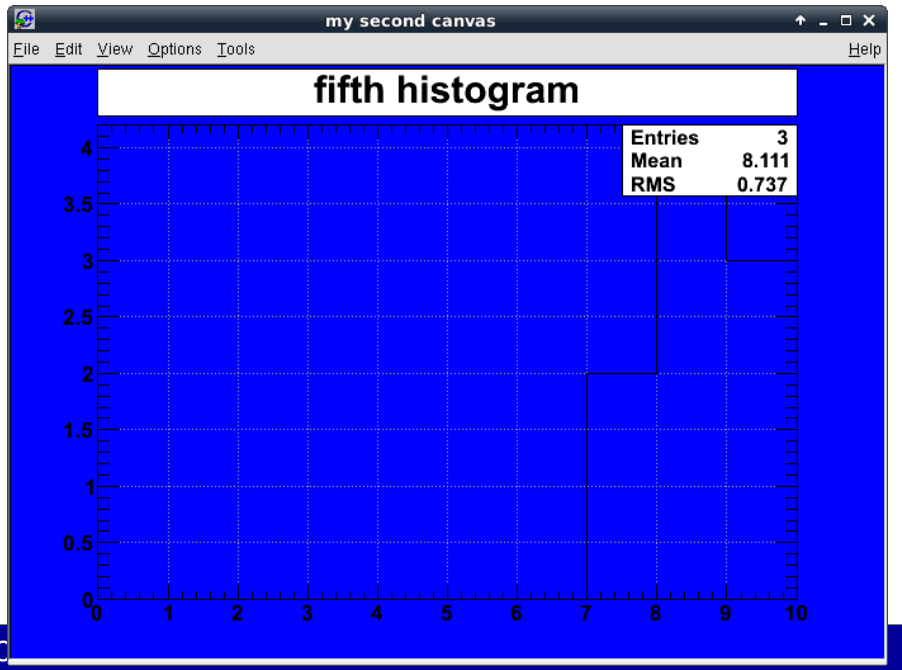
Canvases

Another example

```
29   // create first canvas
30   TCanvas *c1 = new TCanvas("c1","my first canvas");
31   c1->Divide(2,2);
32
33   c1->cd(1);
34   h1->Draw();
35   h1->Fit("gaus");
36
37   c1->cd(2);
38   gPad->SetLeftMargin(0.3);
39   h2->Draw();
40   h2->Fit("gaus");
41
42   c1->cd(3);
43   gPad->SetTopMargin(0.3);
44   h3->Draw();
45
46   c1->cd(4);
47   gPad->SetBottomMargin(0.3);
48   h4->Draw();
49 ..
50   // create second canvases
51   TCanvas *c2 = new TCanvas("c2","my second canvas");
52   c2->SetFillColor(kBlue);
53   h5->Draw();
```

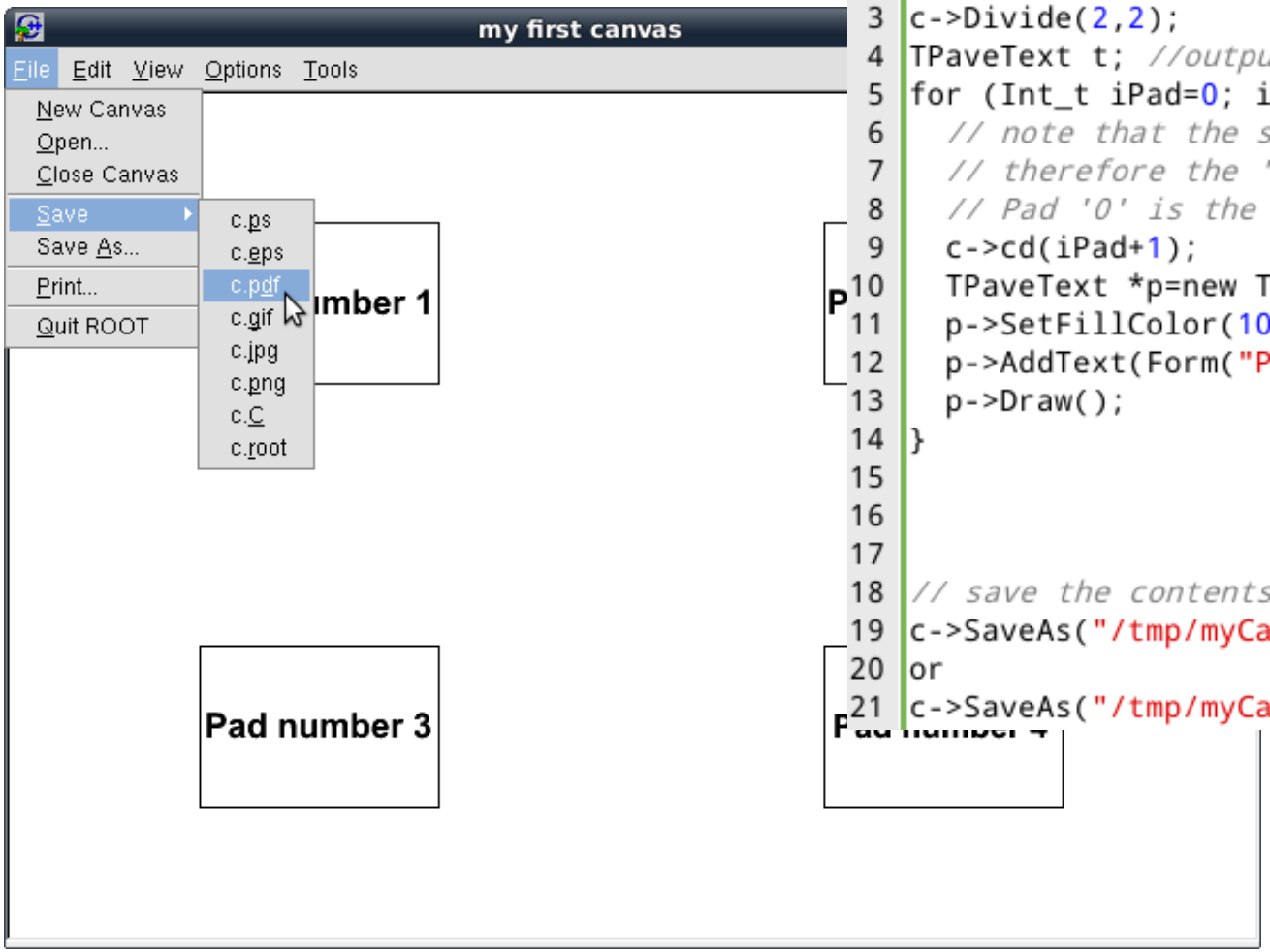
canvas.C





Canvases

Saving to file




```
1  // First example for a canvas to show how to devide in pas
2  TCanvas *c=new TCanvas("c","my first canvas");
3  c->Divide(2,2);
4  TPaveText t; //output text
5  for (Int_t iPad=0; iPad<4; ++iPad) {
6    // note that the sub-pads start with '1'
7    // therefore the '+1' is needed
8    // Pad '0' is the background pad
9    c->cd(iPad+1);
10   TPaveText *p=new TPaveText(.3,.3,.7,.7);
11   p->SetFillColor(10); p->SetBorderSize(1);
12   p->AddText(Form("Pad number %d",iPad+1));
13   p->Draw();
14 }
15
16
17
18 // save the contents of a canvas:
19 c->SaveAs("/tmp/myCanvas.png");
20 or
21 c->SaveAs("/tmp/myCanvas.jpg");
```

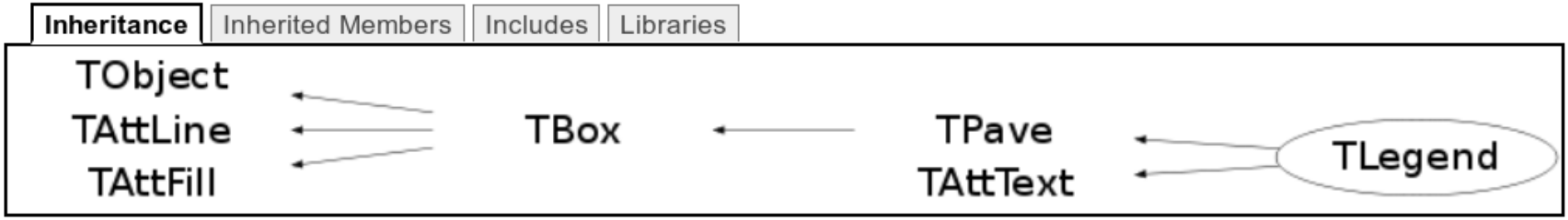
canvas_examples.txt

Legends

Introduction

- Often when drawing several objects into one pad a legend is helpful to label them
- This can be done using the TLegend class
- The style of the legend can be set using the attribute classes

Class Charts http://root.cern.ch/root/html/TLegend.html



Legends

Constructors

TLegend(Double_t **x1**, Double_t **y1**, Double_t **x2**, Double_t **y2**,
const char* **header** = "", Option_t* **option** = "brNDC")

x1	left x-coordinate where the legend will be drawn
y1	bottom y-coordinate where the legend will be drawn
x2	right x-coordinate to which the legend will be drawn
y2	top y-coordinate to which the legend will be drawn
header	Title of the legend
option	drawing options, for details see http://root.cern.ch/root/html/TPave.html#TPave:TPave@1

Legends

Most important functions

Add an object to the legend

AddEntry(const TObject* **obj**, const char* **label** = "", Option_t* **option** = "lpf")

Display the legend

Draw()

obj pointer to an object (histogram, graph, ...)

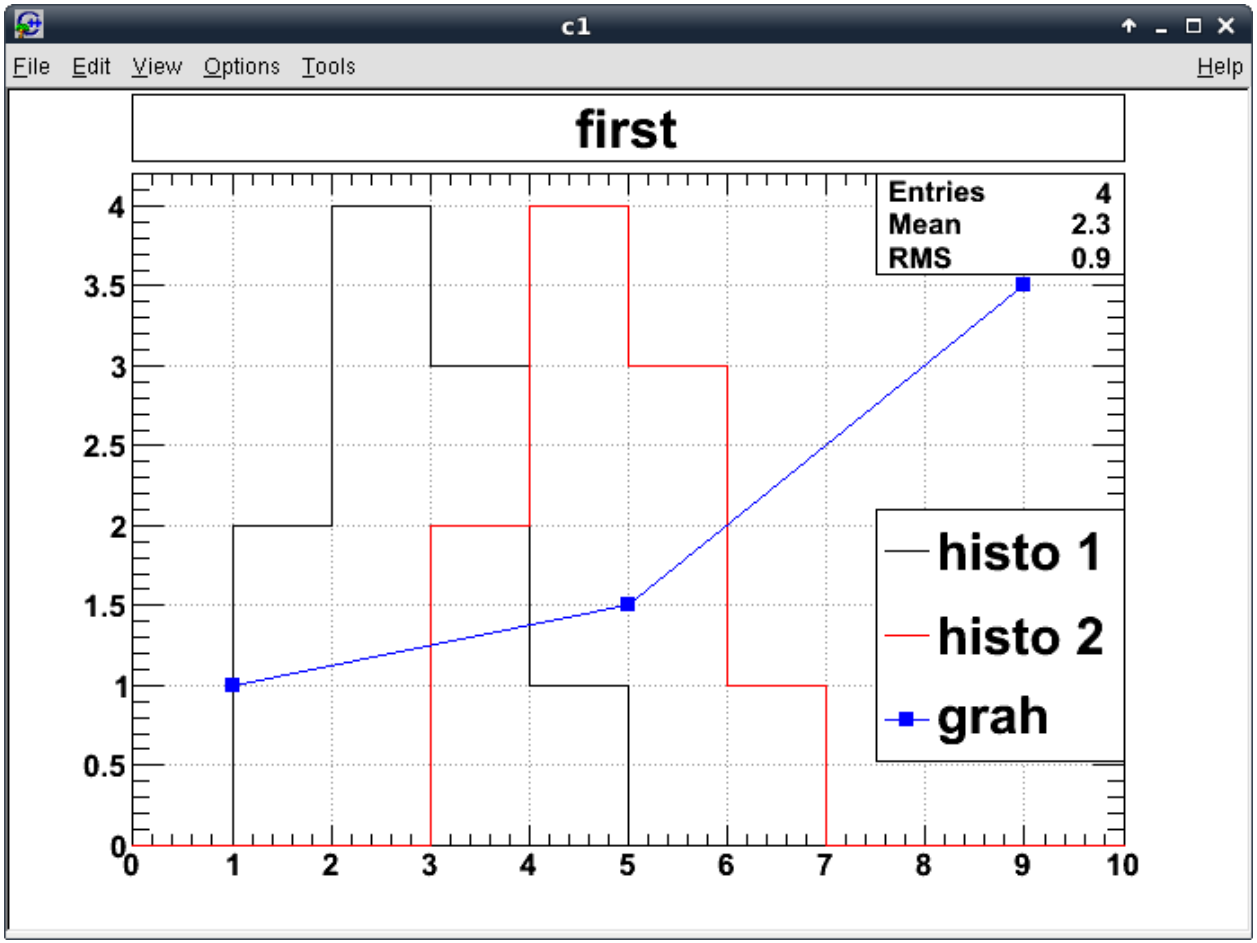
label The text to be displayed to label the object

option what kind of information to display to identify the object; l=line, p=marker, f=filling

Legends

Example

```
33  TLegend *leg = new TLegend(.7,.2,.9,.5);
34  leg->SetFillColor(10);
35  leg->SetBorderSize(1);
36  leg->AddEntry(h1,"histo 1","l");
37  leg->AddEntry(h2,"histo 2","l");
38  leg->AddEntry(gr,"grah","lp");
39  leg->Draw();
```

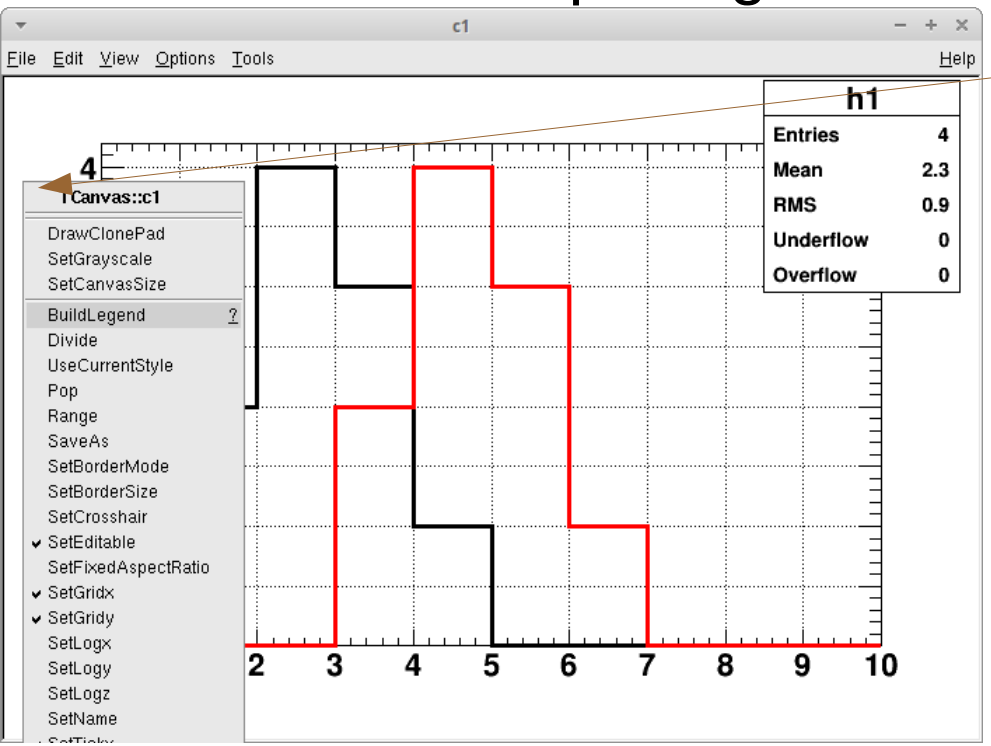


legend.C

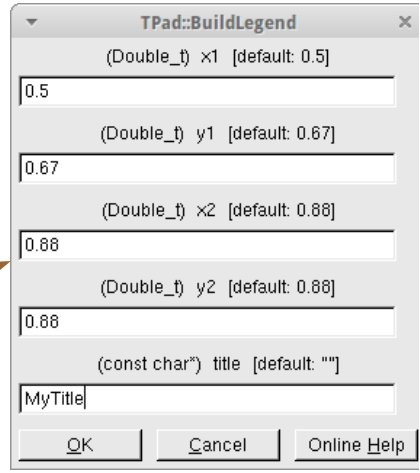
Legends

Quick (and sometimes dirty)

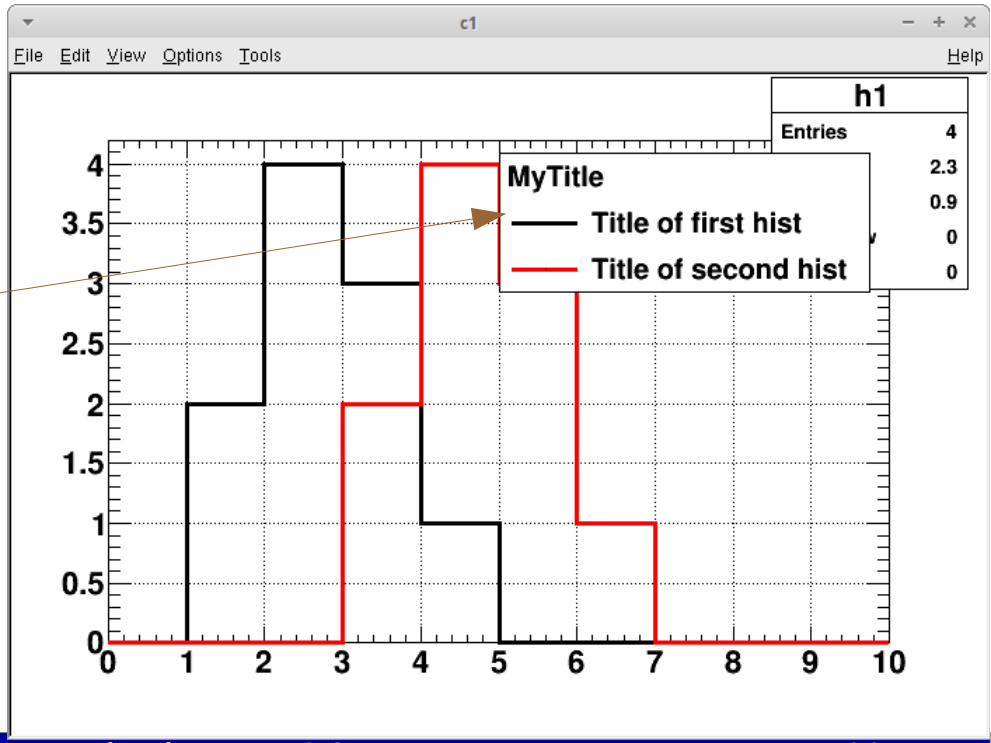
- Create a simple legend using the GUI



Right-click on the pad outside the axis/histo area and select "BuildLegend"



Specify legend coordinates ((x1,y1) → bottom left point, (x2,y2) → top right point; both in mother pad reference frame) and legend title



Legend is drawn with desired title and containing all drawn objects
- style (e.g. line colour and width) is taken from the actual objects
- text is the title of each object

=> Use mouse to move or resize the legend

=> Quick and easy, but more advanced legends should be created inside the code (also: much easier to reproduce!)

Do and Don't

- Use the ROOT data types
- It is better to compile macros
- Give a proper title to histograms, especially the axes
- Use legends if you overlay histograms / graphs

Exercises: Functions

- Functions:

  define the function $a_0\, x^{a1} \sin(x)$ and draw it for different parameters values.

  Fix the values as $a_0=3$, $a_1=2$ and compute:

  - Function value at x = 2
  - Derivative value at x =2
  - Integral value in the range [0,1]

  Solution: exercises/day02/Functions.C

Exercises: Histograms

- Use the root pages to find out about the 'FillRandom' function (in TH1) and use it to fill two histograms with random numbers :
  - Uniform distribution between -5 and 5, 50 bins, 10000 entries
  - Gaussian distribution between -5 and 5, 50 bins, 1000 entries

  Add the two histograms to a new one

  Normalize the sum histogram to unity

  Draw the histogram with option “E” (show the errors). Do the error make sense ?

Solution: exercises/day02/Histograms.C

Exercises: Fits

- Fill the data you brought from your lab in a histogram or graph and draw it
  - Fit the histogram/graph with the proper function

- Alternatives:

  1) you can fill random numbers from two Gaussian distributions with different mean value and entries into one histogram.

  Afterwards, you fit the histogram with the sum of two Gaussians

  (Solution: exercises/day02/FillRandomAndFit.C)

  2) Run the macro example_code/drawK0mass.C

  Add in the macro the fit of the distribution, as sum of two functions (which ones ?)

  (Solution: exercises/day02/drawK0mass.C)

Hint: look at the TF1 documentation for a simple way to define a function as the sum of two predefined functions

Don't forget to set reasonable initial parameters for your fit function!

Exercises: 2D histograms, canvases

- Try to define a 2 dimensional histogram (TH2F) and fill it with random numbers (draw it with the 'colz' or 'surf2' option)

  (Solution: exercises/day02/TH2_examples.C)

- Create a canvas with 6 pads inside a macro and fill it with histograms and graphs with different fill/line/marker properties
  - Save the canvas

  (Solution: exercises/day02/canvas_playing.C)