Lecture 3

Finding your way in ROOT memory:
Names, Lists, Directories, Browsers and Files
Exercises

① Use the browser to find out *which* standard presentation styles are available in ROOT. Hint: there are 5
Exercises

① Use the browser to find out *which* standard presentation styles are available in ROOT. Hint: there are 5

**Solution:**

KVIQ75:tmp:906>root -l
root [0] new TBrowser
(class TBrowser*)0x8c758a8
Exercises

1. Use the browser to find out *which* standard presentation styles are available in ROOT. Hint: there are 5

Solution:
KVIQ75:tmp:906>root -l
root [0] new TBrowser
(class TBrowser*)0x8c758a8
Exercises

① Use the browser to find out which standard presentation styles are available in ROOT. Hint: there are 5

Solution:
KVIQ75:tmp:906>root -l
root [0] new TBrowser
(class TBrowser*)0x8c758a8
Exercises

① Use the browser to find out *which* standard presentation styles are available in ROOT. Hint: there are 5

Solution:
KVIQ75:tmp:906>root -l
root [0] new TBrowser
(class TBrowser*)0x8c758a8
Exercises

② Read http://root.cern.ch/root/html/TFile.html and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there....
Exercises

② Read http://root.cern.ch/root/htmldoc/TFile.html and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there.

Solution:
KVIQ75:tmp:910>root -l
root [0] TFile* file = new TFile("test.root","create")
root [1] TH1D* myHistoPtr = new TH1D("histName","histTitle",100,0,1)
root [2] file->Write()
(Int_t)249
root [4] .q
KVIQ75:tmp:912>root -l
root [0] new TBrowser
(class TBrowser*)0x8c758a8
Exercises

2. Read http://root.cern.ch/root/html/TFile.html and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there....
Exercises

② Read http://root.cern.ch/root/htmldoc/TFile.html and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there....
Exercises

2. Read http://root.cern.ch/root/htmldoc/TFile.html and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there ....
Exercises

Read http://root.cern.ch/root/htmldoc/TFile.html and the ROOT tutorial #6 on the web. Create a file with a histogram in it. Make sure you close the file. Start ROOT again and open the file you just created with the browser. See if the histogram is indeed there....
Exercises

③ Open the file you created in (2) in update mode and change the title of the histogram. Describe what you did.
Exercises

3. Open the file you created in (2) in *update* mode and change the title of the histogram. Describe what you did.

Solution:

```
KVIQ75:tmp:912>root -l
root [0] TFile* f = new TFile("test.root","update")
root [1] TH1D* hist = (TH1D*)f->Get("histName")
root [2] hist->SetName("newTitle")
root [3] f->Write()
(Int_t)250
root [0] new TBrowser
(class TBrowser*)0x8c758a8
```
Exercises

4. From gEnv (an instance of the TEnv class), get the name of the default fitter in ROOT: "Root.Fitter". Hint: for $dflt$ use ""
Exercises

4 From gEnv (an instance of the TEnv class), get the name of the default fitter in ROOT: "Root.Fitter". Hint: for dfilt use ""
Exercises

④ From gEnv (an instance of the TEnv class), get the name of the default fitter in ROOT: "Root.Fitter". Hint: for dflt use ""

Solution:

KVIQ75:tmp:1186>root
root [0] gEnv->GetValue("Root.Fitter","")
(const char* 0x87ab17c)"Minuit"
Graphs

A graph is a graphics object made of two arrays \( X \) and \( Y \), holding the \( x \), \( y \) coordinates of \( n \) points. There are several graph classes, e.g.: \texttt{TGraph}, \texttt{TGraphErrors} and \texttt{TGraphAsymmErrors}. There are also 2D versions: \texttt{TGraph2D} and \texttt{TGraph2DErrors}, with \( x \), \( y \) and \( z \) coordinates.

Constructors:

\begin{verbatim}
TGraph()
TGraph(Int_t n)
TGraph(Int_t n, const Int_t* x, const Int_t* y)
TGraph(Int_t n, const Float_t* x, const Float_t* y)
TGraph(Int_t n, const Double_t* x, const Double_t* y)
TGraph(const TGraph& gr)
TGraph(const TVector& vx, const TVector& vy)
TGraph(const TVectorD& vx, const TVectorD& vy)
TGraph(const TH1* h)
TGraph(const TF1* f, Option_t* option)
TGraph(const char* filename, const char* format = "%lg %lg", Option_t* option)
\end{verbatim}

Example:

\begin{verbatim}
root [0] Float_t x[5] = {1,2,4,8,12};
root [1] Float_t y[5] = {6,7,8,9,10};
root [2] TGraph *gr1 = new TGraph (5, x, y);
\end{verbatim}
Examples of Graphs
Drawing Options for Graphs

Graph Draw Options

- "L" A line between every points is drawn
- "F" A fill area is drawn
- "A" Axis are drawn around the graph (needed for stand-alone graph!!!)
- "C" A smooth curve is drawn
- "*" A star is plotted at each point
- "P" The current marker of the graph is plotted at each point
- "B" A bar chart is drawn at each point
- "[]" Only the end vertical/horizontal lines of the error bars are drawn. This option only applies to the TGraphAsymmErrors.
Examples of Graphs Drawing Options

Option = AP

Option = AL

Option = AC

Option = AF
## Things You Can Do With Graphs

<table>
<thead>
<tr>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print</td>
<td><code>graph-&gt;Print()</code></td>
</tr>
<tr>
<td>Set marker properties</td>
<td><code>graph-&gt;SetMarkerStyle(20)</code></td>
</tr>
<tr>
<td></td>
<td><code>graph-&gt;SetMarkerColor(kRed)</code></td>
</tr>
<tr>
<td></td>
<td><code>graph-&gt;SetMarkerSize(2)</code></td>
</tr>
<tr>
<td>Set line properties</td>
<td><code>graph-&gt;SetLineWidth(2)</code></td>
</tr>
<tr>
<td></td>
<td><code>graph-&gt;SetLineStyle(2)</code></td>
</tr>
<tr>
<td>Set main title</td>
<td><code>graph-&gt;SetTitle(&quot;main-title&quot;)</code></td>
</tr>
<tr>
<td>Set axis titles</td>
<td><code>graph-&gt;GetXaxis()-&gt;SetTitle(&quot;axis-title&quot;)</code></td>
</tr>
<tr>
<td></td>
<td><code>graph-&gt;GetYaxis()-&gt;SetTitle(&quot;axis-title&quot;)</code></td>
</tr>
<tr>
<td>Interpolate</td>
<td><code>graph-&gt;Eval(x)</code></td>
</tr>
<tr>
<td>Fit</td>
<td><code>graph-&gt;FitPanel()</code></td>
</tr>
<tr>
<td></td>
<td><code>graph-&gt;Fit(&quot;function-name&quot;)</code></td>
</tr>
<tr>
<td>Calculate correlation</td>
<td><code>graph-&gt;GetCorrelationFactor()</code></td>
</tr>
<tr>
<td>Calculate covariance</td>
<td><code>graph-&gt;GetCovariance()</code></td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

A lot more examples are in the user guide:
**Histograms**

There are several histograms classes available in ROOT, which contain data in the form of a number of (weighted) counts $N$ for a collection of consecutive bins in $x$ (1-D), $(x, y)$ (2-D) or $(x, y, z)$ (3-D). The corresponding classes are

\[ \text{THNS} \]

where $N = 1, 2, 3$ for 1-D, 2-D and 3-D and $S = "C, S, I, F, D"$ for 1 (Char_t), 2 (Short_t), 4 (Int_t), 4 (Float_t) or 8 (Double_t) bytes of storage volume per bin.

All histogram classes inherit from \text{TH1}.
Histogram Constructor

Contractors of **TH1D:**

TH1D()

TH1D(const char* name, const char* title, Int_t nbinsx, Axis_t xlow, Axis_t xup)

TH1D(const char* name, const char* title, Int_t nbinsx, const Float_t* xbins)

TH1D(const char* name, const char* title, Int_t nbinsx, const Double_t* xbins)

TH1D(const TVectorD& v)

TH1D(const TH1D& h1d)

**Example for fixed-bin width:**

```
root [0] TH1D* hPtr = new TH1D("histoName","Fixed Bin Width",10,-1,1)
```

**Example for variable-bin width:**

```
root [0] Double_t bins[5] = {1,2,4,8,16}
root [1] TH1D* hPtr = new TH1D("histoName","Variable Bin Width",4,bins)
```
Conventions for Axes

You can look at the boundaries of the axes. Histogram axes are implemented via the TAxis class:

```
root [10] h2->GetXaxis()->GetBinLowEdge(1) (const Axis_t)0.00000000000000000e+00
root [11] h2->GetXaxis()->GetBinUpEdge(1) (const Axis_t)1.00000000000000000e+00
root [12] h2->GetXaxis()->GetBinCenter(1) (const Axis_t)5.0000000000000001e-01
root [13] h2->GetXaxis()->GetBinWidth(1) (const Axis_t)1.00000000000000000e+00
```

Bin Numbering Convention

```
x_i < x < x_{i+1}
```

underflow (x<xlow) | overflow (x>xup)
# Filling a Histogram

There are many ways to fill a histogram. A selection:

<table>
<thead>
<tr>
<th>Method</th>
<th>What happens?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddBinContent(Int_t bin)</code></td>
<td>increment content of ‘bin’ by 1</td>
</tr>
<tr>
<td><code>AddBinContent(Int_t bin, Stat_t w)</code></td>
<td>increment content of ‘bin’ by ‘w’</td>
</tr>
<tr>
<td><code>Eval(TF1* f1, Option_t* option)</code></td>
<td>by evaluation function ‘f1’ at bin centers</td>
</tr>
<tr>
<td><code>Fill(Axis_t x)</code></td>
<td>increment bin in which ‘x’ falls by 1</td>
</tr>
<tr>
<td><code>Fill(Axis_t x, Stat_t w)</code></td>
<td>increment bin in which ‘x’ falls by ‘w’</td>
</tr>
<tr>
<td><code>FillRandom(const char* fname, Int_t ntimes = 5000)</code></td>
<td>‘ntimes’ random events distributed as <code>fname</code></td>
</tr>
<tr>
<td><code>Reset(Option_t* option)</code></td>
<td>set all contents to zero</td>
</tr>
<tr>
<td><code>SetBinContent(Int_t bin, Stat_t content)</code></td>
<td>set content of ‘bin’ to ‘content’</td>
</tr>
</tbody>
</table>
**Inspecting Histogram Content**

The simplest way to look at the histogram content is by **drawing it**

`histo->Draw()`

However, there are many things you can do from the command line:

<table>
<thead>
<tr>
<th>Method</th>
<th>What happens?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>histo-&gt;GetBinContent(bin)</code></td>
<td>get content of bin</td>
</tr>
<tr>
<td><code>histo-&gt;GetBinError(bin)</code></td>
<td>get error of bin</td>
</tr>
<tr>
<td><code>histo-&gt;GetMinimum()</code></td>
<td>get minimum bin content</td>
</tr>
<tr>
<td><code>histo-&gt;GetMaximum()</code></td>
<td>get maximum bin content</td>
</tr>
<tr>
<td><code>histo-&gt;GetMinimumBin()</code></td>
<td>get bin in which minimum occurs</td>
</tr>
<tr>
<td><code>histo-&gt;GetMaximumBin()</code></td>
<td>get bin in which maximum occurs</td>
</tr>
<tr>
<td><code>histo-&gt;GetEntries()</code></td>
<td>get the number of entries</td>
</tr>
<tr>
<td><code>histo-&gt;GetSum()</code></td>
<td>get the sum of the bin contents</td>
</tr>
<tr>
<td><code>histo-&gt;Integral(bin1,bin2)</code></td>
<td>get the bin contents from bin1 to bin2 (incl.)</td>
</tr>
<tr>
<td><code>histo-&gt;GetMean()</code></td>
<td>get mean of $x$</td>
</tr>
<tr>
<td><code>histo-&gt;GetRMS()</code></td>
<td>Get RMS of $x$</td>
</tr>
</tbody>
</table>

....
Histogram Errors

By default, the error for a histogram bin is calculated as

\[ \delta N = \sqrt{N} \]

with \( N \) the content of the bin. This is only correct if Poissonian statistics are applicable, i.e. if \( N \) represents counts. For weighted events, the error is given by

\[ \delta N = \sqrt{\sum w_i} \]

This is has to be set explicitly, before filling the histogram:

\package{histo}\rightarrow \Call{Sumw2}{}

You can also set the errors by hand:

\package{histo}\rightarrow \Call{SetBinError}(\text{bin},\text{error})
## Manipulating Histogram Content

<table>
<thead>
<tr>
<th>Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group bins</td>
<td>histo-&gt;Rebin(2)</td>
</tr>
<tr>
<td>Add another histogram</td>
<td>histo-&gt;Add(otherhisto,1)</td>
</tr>
<tr>
<td>Add a function</td>
<td>histo-&gt;Add(afunction,-3.1415)</td>
</tr>
<tr>
<td>Add two histos</td>
<td>histo-&gt;Add(h1,h2,12,-11)</td>
</tr>
<tr>
<td>Divide with another histogram</td>
<td>histo-&gt;Divide(anotherhisto)</td>
</tr>
<tr>
<td>Multiply with another histogram</td>
<td>histo-&gt;Multiply(anotherhisto)</td>
</tr>
<tr>
<td>Scale with a factor</td>
<td>histo-&gt;Scale(100)</td>
</tr>
<tr>
<td>Smooth histogram</td>
<td>histo-&gt;Smooth(3)</td>
</tr>
</tbody>
</table>
More documentation

A lot more information and examples on histograms can be found in the USER MANUAL on


Also, on http://root.cern.ch/root/Tutorials.html check tutorials 6,7,8,12,13,24,32–34 for more histogramming examples.
Exercises

① Modify example 25. of the tutorial to display three full periods of a sin-wave (and get rid of the ugly brown background color). Make the marker a full square and change the line color to yellow.

② Experiment with the histogram drawing options, starting from example 24. of the tutorials.