ROOT Tutorial at La Plata

25/11 - 6/11  2013
Outline

• Day 1
  – Introduction to ROOT
  – Start using ROOT
  – Working with Histograms

• Day 2
  – ROOT I/O
  – Trees in ROOT

• Day 3
  – Parallel Tree data analysis using PROOF
  – Fitting in ROOT

• Day 4
  – Statistics analysis in ROOT with RooFit/RooStats
Introduction to ROOT

- What is ROOT?
- Overview of ROOT functionality
ROOT in a nutshell

• Framework for large scale data handling

• Provides, among others,
  – an efficient data storage, access and query system (PetaBytes)
  – advanced statistical analysis: histogramming, fitting, minimization and multi-variate analysis algorithms
  – scientific visualization: 2D and 3D graphics, Postscript, PDF, LateX
  – geometrical modeler
  – PROOF parallel query engine

• An Open Source Project
• Started in 1995 by R. Brun and F. Rademakers.
• Available (including the source) under GNU LGPL.
• 7 full time developers at CERN, plus 2 at Fermilab (Chicago).
• Many contributors from high energy physics experiments which uses ROOT as base for their software frameworks.
• Large number of part-time developers.
• Several thousands of users giving feedback and a very long list of small contributions.
Why ROOT?

• The analysis of data coming from LHC experiments (and also other experiments) requires a powerful and general toolkit
  – Visualisation
  – Statistical studies
  – Data reduction
  – Multivariate techniques

• A scalable and reliable persistency method is needed to write the data on disks and tapes.
ROOT Application Domains

Data Analysis & Visualization

Data Storage: Local, Network

General Framework
ROOT Libraries

- Overview of ROOT libraries and their dependencies
  - 1,700,000 lines of code.
  - More than 100 shared libraries
  - Fully cross-platform: Unix/Linux, MacOS and Windows.
  - More than 10000 downloads every month
The Interpreter: CINT

• ROOT is shipped with an interpreter, CINT*
  – C++ not trivial to interpret and not foreseen in the language standard!
• Provides interactive shell.
• Can interpret “macros” (not compiled programs)
  – Rapid prototyping possible.
• ROOT provides also Python bindings:
  – Can use Python interpreter directly after a simple import ROOT !!
New Interpreter: Cling

- Starting from ROOT 6 we will have a new interpreter
  - Cling based on LLVM/Clang
- An interpreter based on a real C++ compiler
  - use the JIT capabilities of Clang to interpret the code

- Can handle templated code at interpreter level
- Can parse also new C++ 11 syntax
  - this would have been impossible with CINT

- A real compiler
  - meaningful error reporting
  - some shortcut of CINT will not be allowed (e.g. using "." instead of "->")
• ROOT offers the possibility to write C++ objects to file
  – Extraordinary: impossible in native C++!
  – Used for petabytes/year rates of LHC detectors.
  – Achieved with serialization of the object using the reflection
    provided by the CINT dictionary.
• Single objects, collections, object trees
  – Basically one method for all ROOT objects: TObject::Write

Cornerstone of the storage of experimental data

LEP style flat n-tuples evolved in more efficient trees (fast access, read ahead)
Table of Ages (binned)

<table>
<thead>
<tr>
<th>Age</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-22</td>
<td>1</td>
</tr>
<tr>
<td>22-24</td>
<td>5</td>
</tr>
<tr>
<td>24-26</td>
<td>14</td>
</tr>
<tr>
<td>26-28</td>
<td>14</td>
</tr>
<tr>
<td>28-30</td>
<td>10</td>
</tr>
<tr>
<td>30-32</td>
<td>5</td>
</tr>
<tr>
<td>32-34</td>
<td>5</td>
</tr>
<tr>
<td>34-36</td>
<td>1</td>
</tr>
<tr>
<td>36-38</td>
<td>1</td>
</tr>
<tr>
<td>38-40</td>
<td>1</td>
</tr>
</tbody>
</table>

Shows distribution of ages, total number of entries (57 participants) and average: 27 years 10 months 6 days...
Histograms

Analysis result: often a histogram

Menu:
View / Editor
Analysis result: often a **fit** of a histogram
ROOT provides a reach set of mathematical libraries and tools needed for event reconstruction, simulation and statistical data analysis.
The ROOT framework provides many techniques to visualize multi-variable data sets from 2 until N variables.

- 2 variables visualization techniques are used to display Trees, Ntuples, 1D histograms, functions $y=f(x)$, graphs.

- 3 variables visualization techniques are used to display Trees, Ntuples, 2D histograms, 2D Graphs, 2D functions ...

- 4 variables visualization techniques are used to display Trees, Ntuples, 3D histograms, 3D functions ...

- N variables visualization techniques are used to display Trees and Ntuples ...

Can save graphics in many formats: \texttt{ps, pdf, svg, jpeg, png, c, root}
2 Variables Techniques: Histograms

Bar charts and lines are a common way to represent 1D histograms.

Pie charts can be also used to visualize 1D histograms.
Visualization of Variables with Errors (1D Histograms and Graphs)

Errors can be represented as bars, band, rectangles. They can be symmetric, asymmetric or bent.
Graphs can be drawn as simple lines, like functions. Can also visualize exclusion zones. Can be plotted in polar coordinates.
Several techniques to visualize 3 variables data sets (TH2, TGraph2D) in 2 dimensions.

Two variables are mapped on the X and Y axis and the 3rd one on some graphical attributes:

- the color or the size of a box;
- a density of points (scatter plot);
- writing the value of the bin content.
- Using contour plots. Some special projections (like Aitoff) are available to display such contours.
Lego and surface plots are a common way to display 3 variables data sets in 3D. They can be combined with color or light effects and displayed in non Cartesian coordinate systems like polar, cylindrical or spherical. 2D graphs can be drawn using the Delaunay triangulation technique.
4 variables data set representations (e.g. TH3, TF3)

Rectangles become boxes or spheres, contour plots become iso-surfaces. The scatter plots (density plots) are drawn in boxes instead of rectangles. The 4th variable can also be mapped on colors. The use of OpenGL allows to enhance the plots’ quality and the interactivity.
Functions like $t = f(x,y,z)$ and 3D histograms are 4 variables objects.

ROOT can render using OpenGL. It allows to enhance the plots’ quality and the interactivity. Cutting planes, projection and zoom allow to better understand the data set or function.
Specific visualization techniques are required for $N > 4$. ROOT provides:

- The parallel coordinates (top)
- the candle plots (right) which can be combined with the parallel coordinates.
- The spider plot (top right).

These three techniques, and in particular the parallel coordinates, require a high level of interactivity to be fully efficient.
Geometry

- Describes complex detector geometries
- Allows visualization of these detector geometries with e.g. OpenGL
- Optimized particle transport in complex geometries
- Working in correlation with simulation packages such as GEANT3, GEANT4 and FLUKA
• Visualize detector events (tracks, hits,... i.e. physics objects) together with detector geometry (OpenGL)
• Visually interact with the data, e.g. select a particular track and retrieve its physical properties
• PROOF: Parallel ROOT Facility
  – Multi-process approach to parallelism
  – A system to run ROOT queries in parallel on a large number of distributed computers
  – Proof-lite: does not need a farm, uses all the cores on a desktop machine
Summary

• ROOT: complete analysis framework offering a C++ interpreter, a powerful persistency mechanism, advanced mathematical tools and even a parallel facility
  – it is designed for data analysis of very large shared data sets
  – it is the result of 18 years of cooperation between the development team and thousands of heterogeneous users
  – it is used extensively and profitably by physicists
    • all Higgs discovery plots made with ROOT

• We will now explore in more detail some part of ROOT