The ROOT Data Analysis Framework

HASCO Summer School 2016

Olaf Nackenhorst
University of Geneva

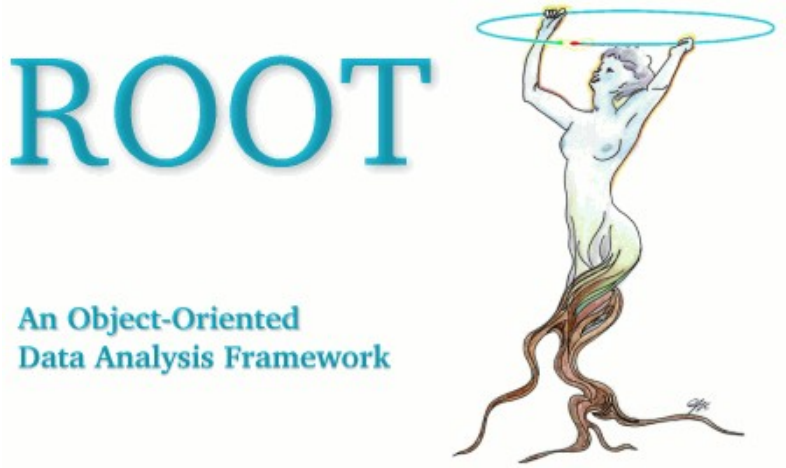
Introduction

Interactive ROOT

ROOT basics

How to make a nice plot

Outlook into advanced ROOT tools



Disclaimer & References

- Disclaimer
  - Not a real *lecture*, rather an introductory tutorial / workshop
  - Not complete, biased by personal point of view / experience
  - Not a ROOT expert, just an experienced user
  - Designed to match different experience levels
  - Mainly practical examples → learning by doing using Jupyter Notebook
- References
  - Parts based on last year's ROOT lecture by Andrea Knue
  - Inspired by ROOT primer (Guide for beginners)
  - ROOT web site: www.root.cern.ch
    - **THE** source of information and help for all users (beginner & experts)
    - User's Guide & Reference Guide (Documentation of classes)
    - Download & installation instructions
    - Topical manuals, tutorials, presentations, forum and more!

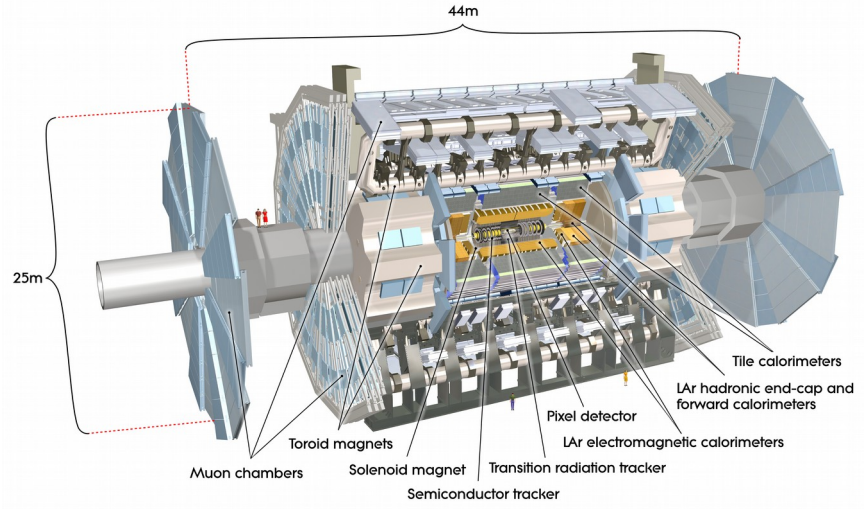
What is ROOT?

- ROOT is a modular scientific software framework for
  - Data processing
  - Data analysis
  - Data visualisation
  - Data storage
- ROOT is mainly written in C++
  - Bindings for Python (PyROOT) and other languages exist
- ROOT originally developed for particle physics
  - In 1995 by Rene Brun & Fons Rademakers
  - Used by many astrophysics & neuroscience projects
- ROOT is heavily used by LHC's experiments
  - Several petabytes a year, thousands of plots for publications

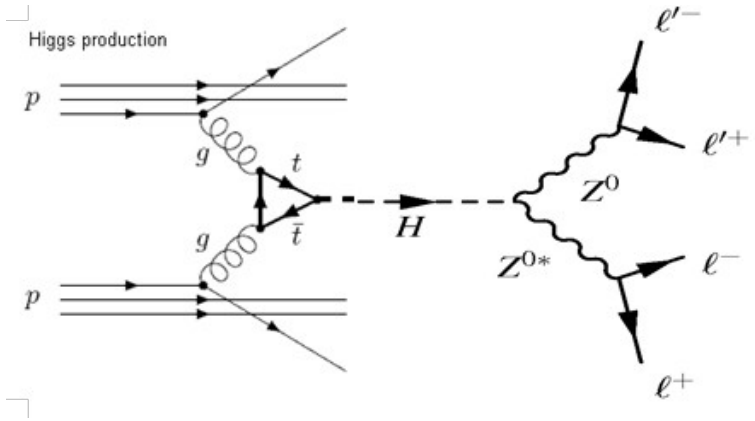
What we do in HEP (simplified)

UNIVERSITÉ DE GENÈVE

Record data, reconstruct and store!

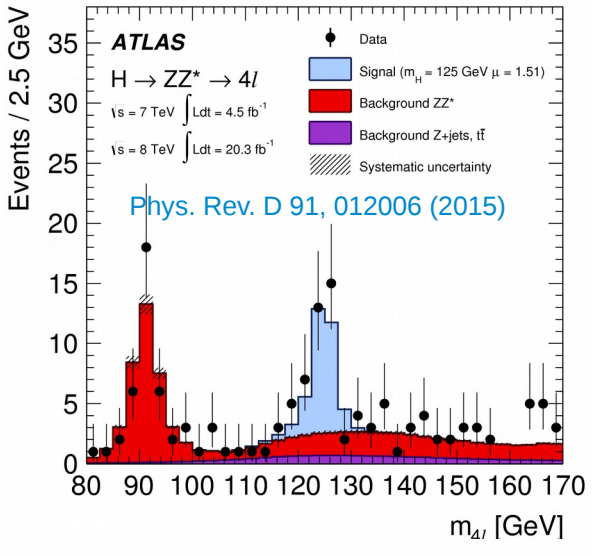


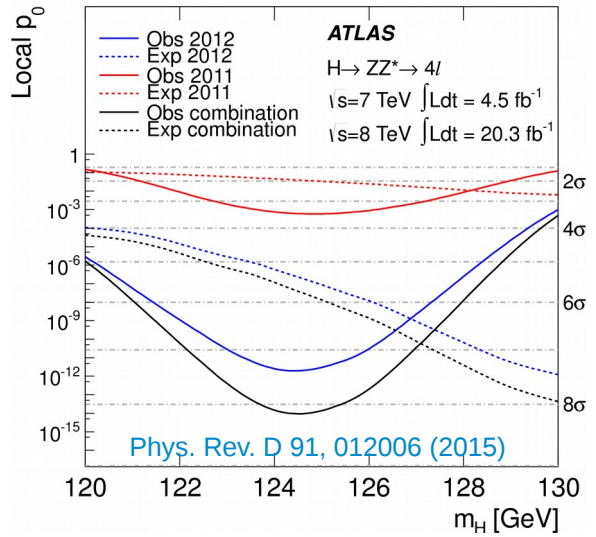
+



Process data, select and analyze!

Visualize data & publish!

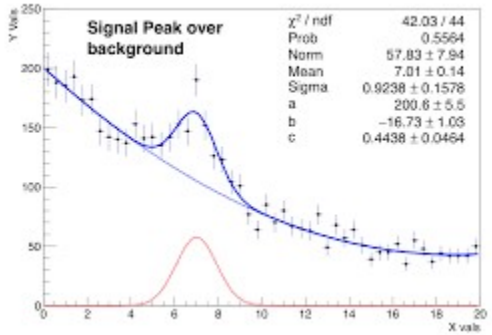


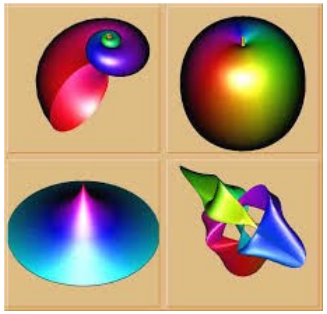


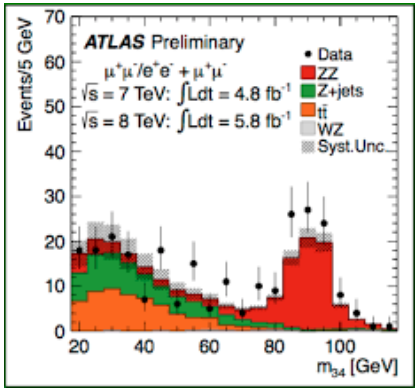
Standard tasks in HEP

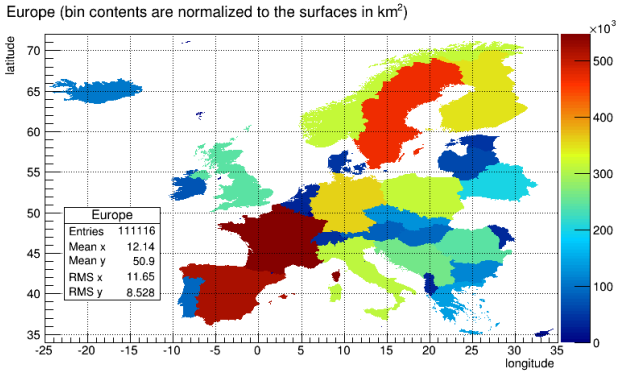
- Standard Tasks in High Energy Physics
  - Comparison of measurements to theoretical models
  - Visualization of data
  - Manipulation of data
  - Fitting of data
  - Statistical interpretation
  - Storage of large data
  - ...
- For all this and more: ROOT
  - The following is based on ROOT6
  - Jupyter Notebooks: > ROOT 6.05
    - Visualise & share live code
    - Web: http://jupyter.org/

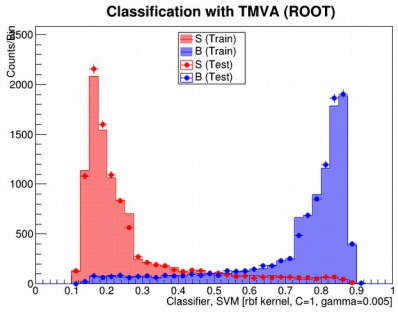
Visualization Examples

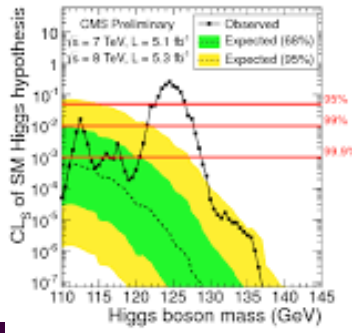


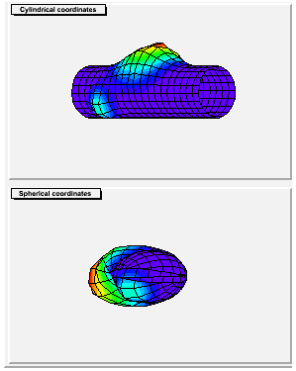


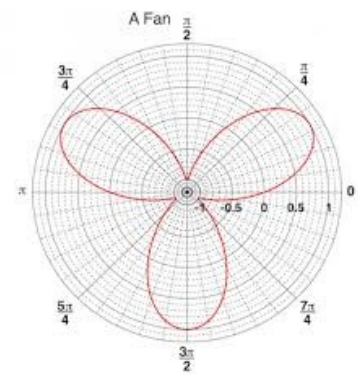


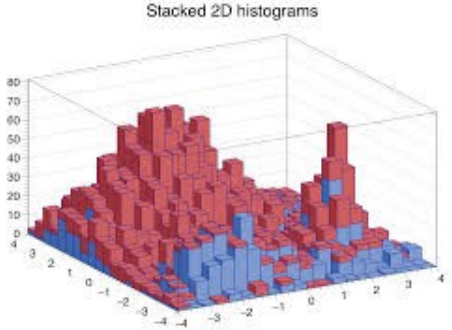


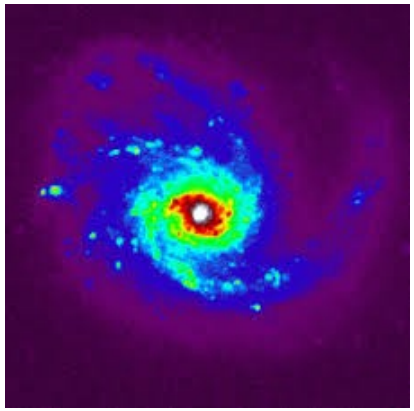


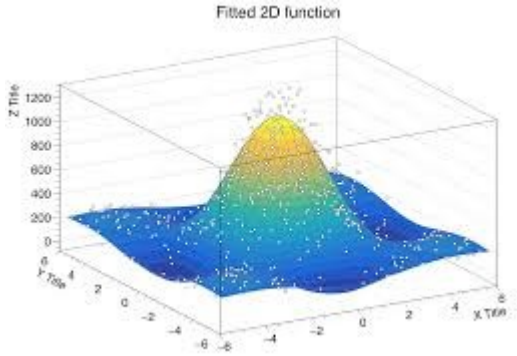


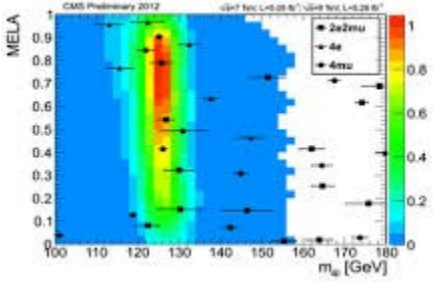


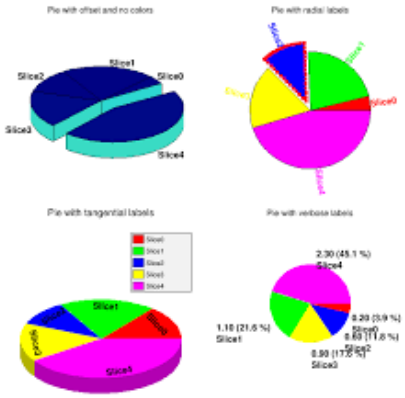












Compiled vs Interactive ROOT

UNIVERSITÉ DE GENÈVE

- Compiled ROOT
  - Programming framework, not an office suite
  - Write a program
  - Compile as standalone application

ExampleMacro.C

```
int main() {
  ExampleMacro();
  return 0;
}
```

```
> g++ -o ExampleMacro ExampleMacro.C `root-config --cflags --libs`
> ./ExampleMacro
```

- Interactive ROOT
  - Comes with a C++ interpreter (Cling/ACLiC)
  - Interactive C++ without the need of a compiler (like python)
  - Just in time compilation – non trivial in C++ !
  - Called **ROOT prompt** (interactive shell)
  - Can interpret macros (non compiled programs)

Interactive ROOT - First Steps

- Type **root** to start
  - **root -l**: no logo
  - **root -b**: no graphics
  - **root -h**: help (all options)
- Type **.help** or **.?**
  - **.L Plot.C**: load macro
    - Execute by **Plot();**
  - **.X Plot.C**: execute macro
    - **Plot()** needs to be defined!
- Compile a macro (ACLiC)
  - **root .L / .X Plot.C+**
- Access shell command
  - **.!<command>**
- Quit root: **.q** or **.qqqq** (force)

```
~ >root
 ------------------------------------------------------------
| Welcome to ROOT 6.06/04                        http://root.cern.ch |
|                                  (c) 1995-2016, The ROOT Team |
| Built for linuxx8664gcc                                     |
| From tag v6-06-04, 3 May 2016                               |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
 ------------------------------------------------------------

root [0]
```

```
root [0] .help

 Cling (C/C++ interpreter) meta commands usage
 All commands must be preceded by a '.', except
 for the evaluation statement { }
 ================================================================================
 Syntax: .Command [arg0 arg1 ... argN]

   .L <filename>                - Load the given file or library

   .(x|X) <filename>[args]      - Same as .L and runs a function with
                                  signature: ret_type filename(args)
```

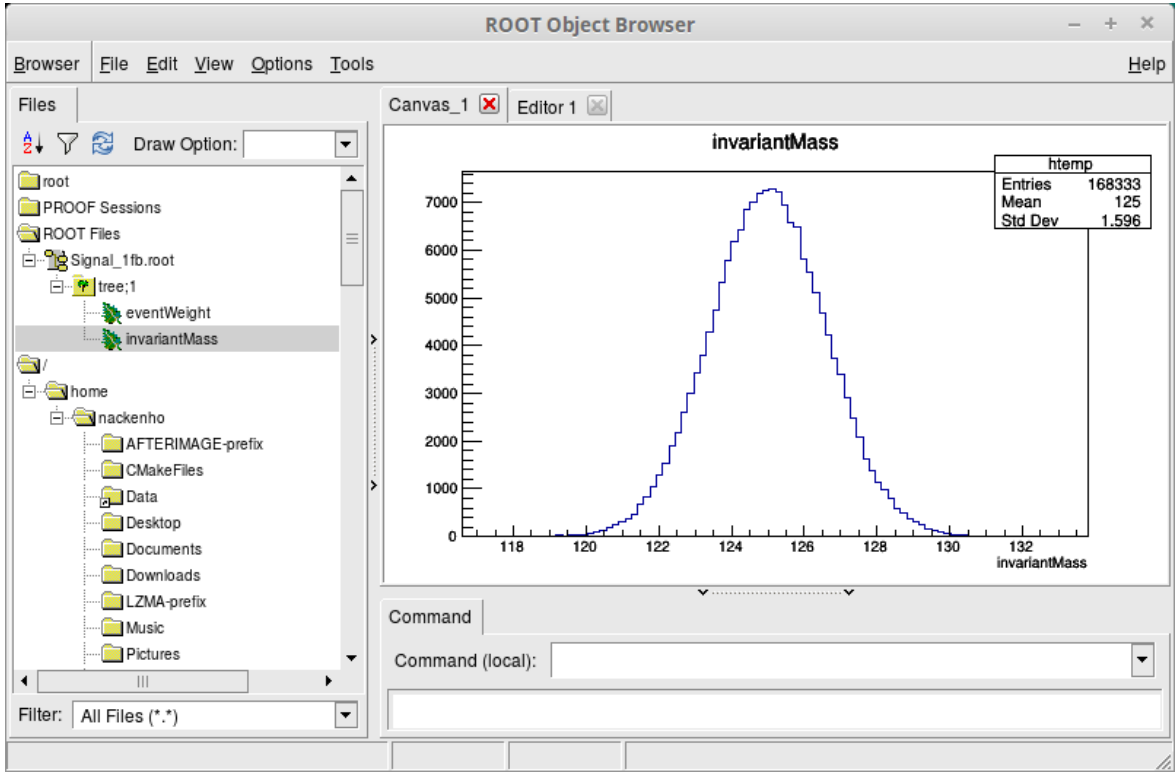
```
root [11] .!ls -lthr | grep ROOT
-rwxrwxrwx  1 nackenho nackenho  597 Jul  1 16:30 ROOTConfig-version.cmake
-rwxrwxrwx  1 nackenho nackenho  684 Jul  1 16:30 ROOTUseFile.cmake
```

```
root [0] .qqqq
Info in <TRint::ProcessLine>: Bye... (try '.qqqqqqq' if still running)
```

The TBrowser

- Load a root file during start
  - **root filename.root**
- Inspect file with GUI
  - **new TBrowser**
- Browse the file
  - **TTree::tree**
    - Datacontainer
  - **TBranch::invariantMass**
    - Content
- Draw content
  - Click on the branch
  - Use **TTree::Draw**
    - Selection & options

```
~ :root -l Signal_1fb.root
root [0]
Attaching file Signal_1fb.root as _file0...
(TFile *) 0x2963280
root [1] new TBrowser
(TBrowser *) 0x2bc5e30
```

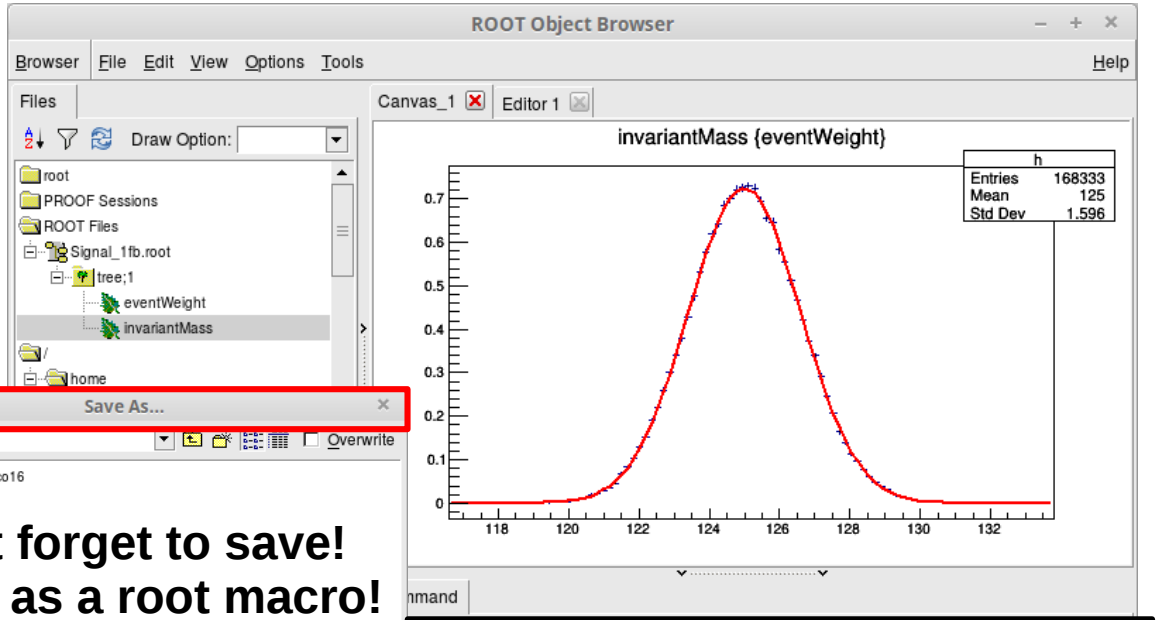


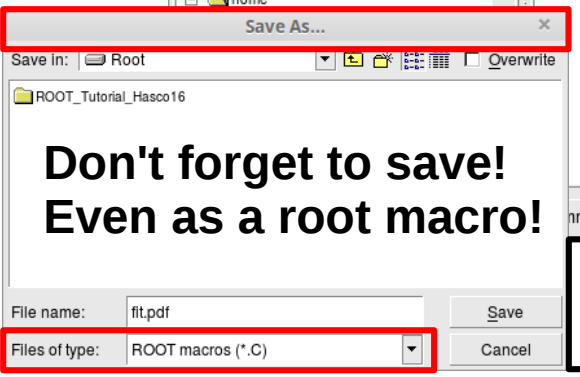

```
root [5] tree->Draw("invariantMass>>h","eventWeight*(invariantMass>115 && invariantMass<135)")
(Long64_t) 168333
```

The FitPanel

- Open the FitPanel
  - **Right click on graph → FitPanel**

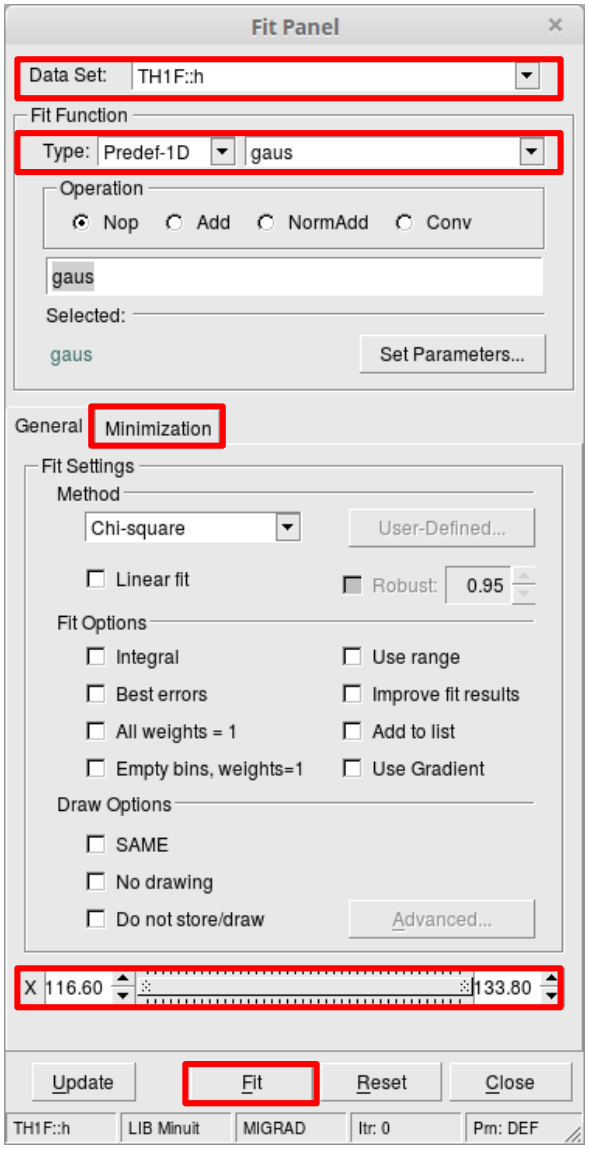
```
root [8]  FCN=94.4327 FROM MIGRAD    STATUS=CONVERGED      39 CALLS          40 TOTAL
                EDM=1.51534e-10    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     7.23859e-01   2.16354e-03   2.17158e-05  -1.39348e-03
   2  Mean         1.25000e+02   3.88943e-03   3.75001e-04   4.25919e-03
   3  Sigma        1.59484e+00   2.75875e-03   2.24184e-06  -1.15001e-02
```





Don't forget to save!
Even as a root macro!

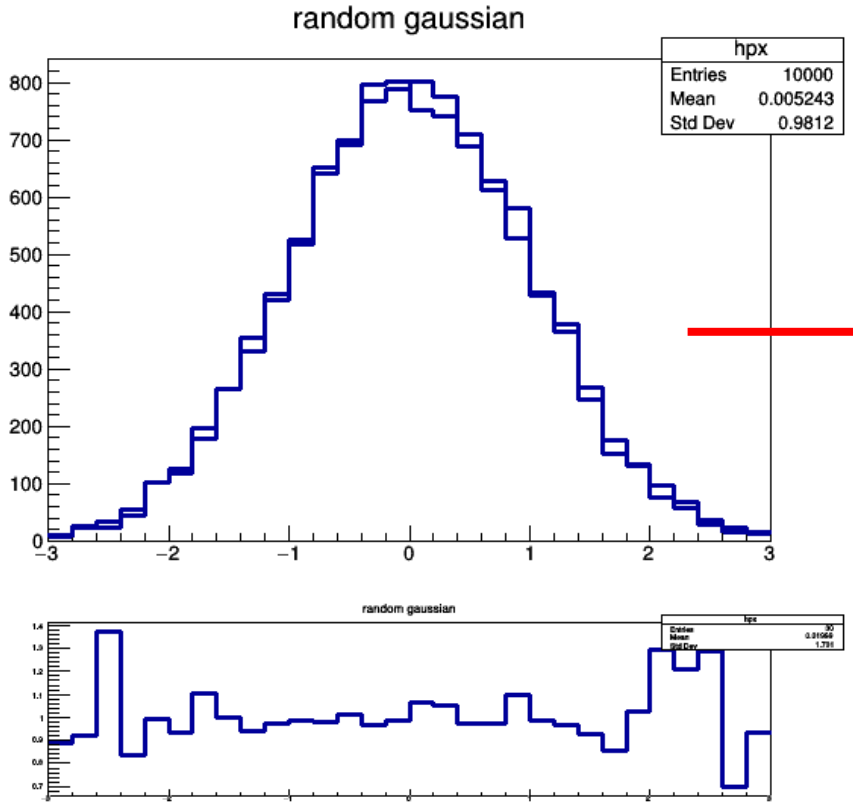
Nice, but write a script if you do something more than once!

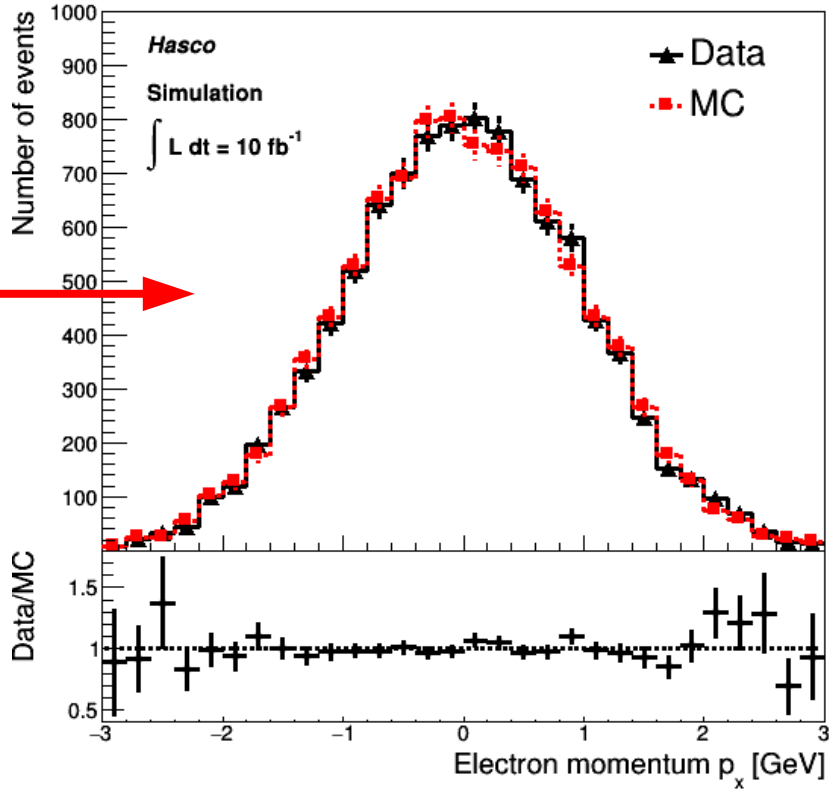


Intermezzo: Root Basics

Example: Good vs bad plot

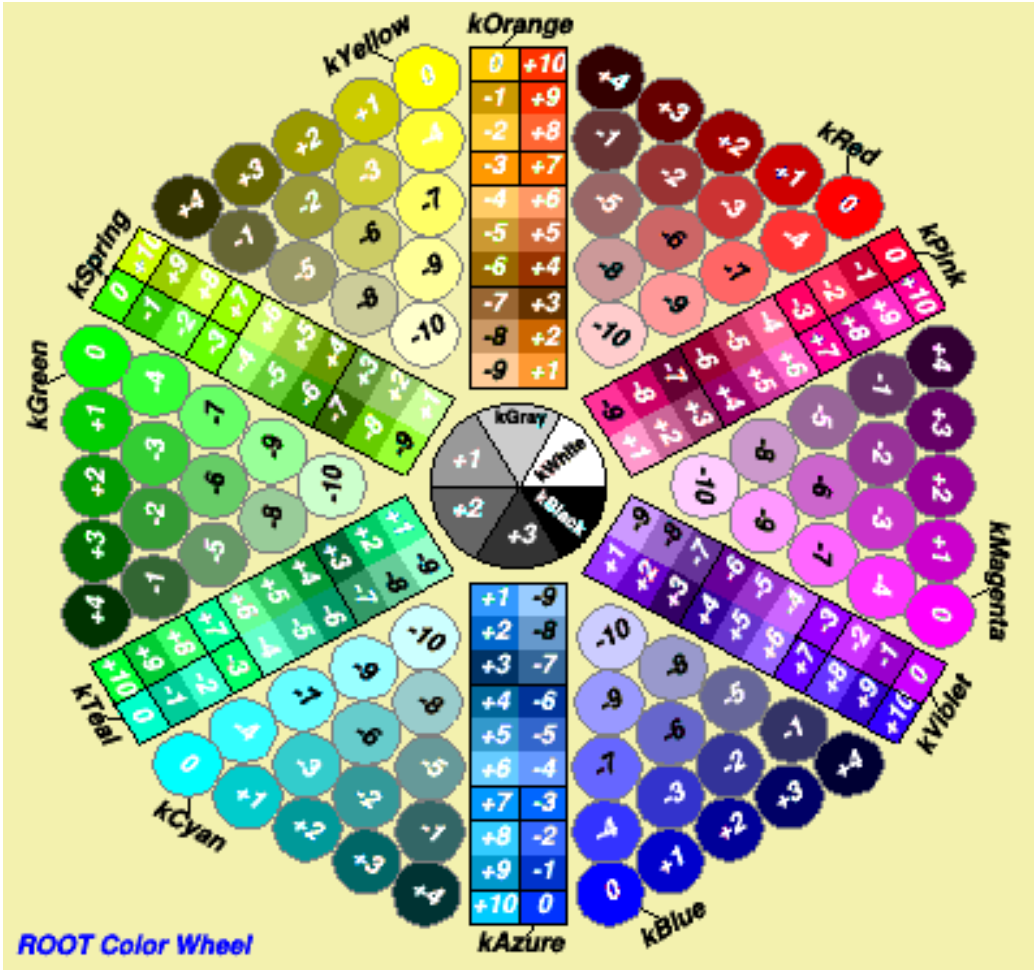
- Relatively easy to make plots in ROOT (left)
- Relatively difficult & time consuming to make nice plots (right)





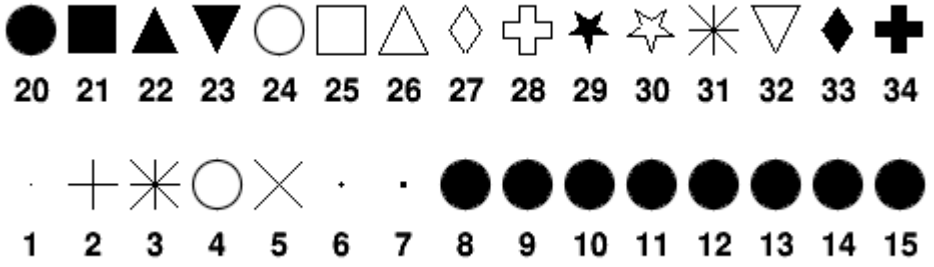
The ROOT Color Wheel

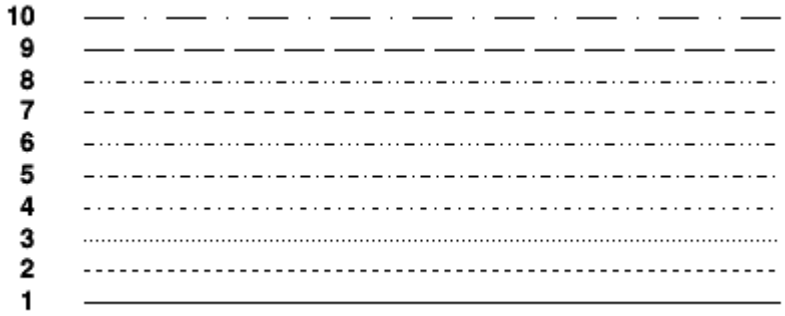
- TColorWheel: **216 colors** as used in web applications
- Special identifiers for colors
  - kWhite, kBlack, kGrey
  - kRed, kBlue, kGreen
  - KYellow, kMagenta, kCyan
  - And more…
- Colors in between
  - Obtained by ± [1,10]
- Colorize objects
  - SetLineColor()
  - SetFillColor()
- **Use colors smartly** (grayscale)

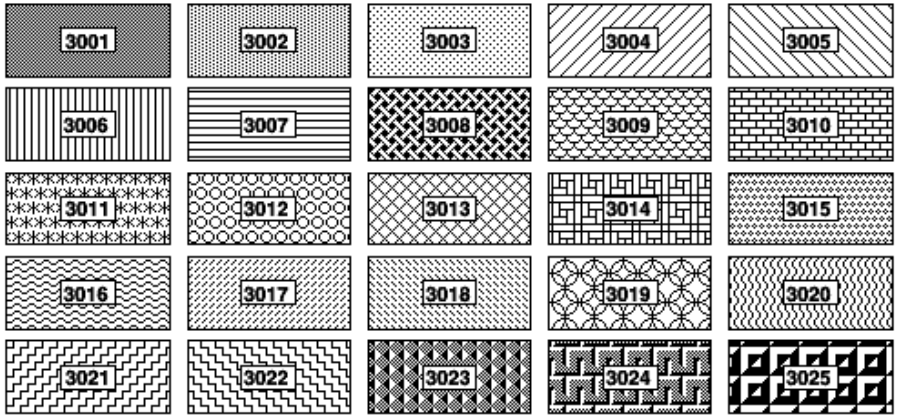


Other useful attributes

- Marker style
  - KDot, kPlus, kStar, kCircle, etc.
  - **SetMarkerStyle()**
- Line style
  - Fixed: 1-10
  - Customizable: SetLineStyleString()
  - **SetLineStyle()**
- Fill area style
  - Fixed: 3000 + 1-25
  - Customizable: FillStyle = 3ijk
    - i=space between each hatch
    - j=angle(<90), k=angle(>90)
  - **SetFillStyle()**







Intermezzo: Nice Plots

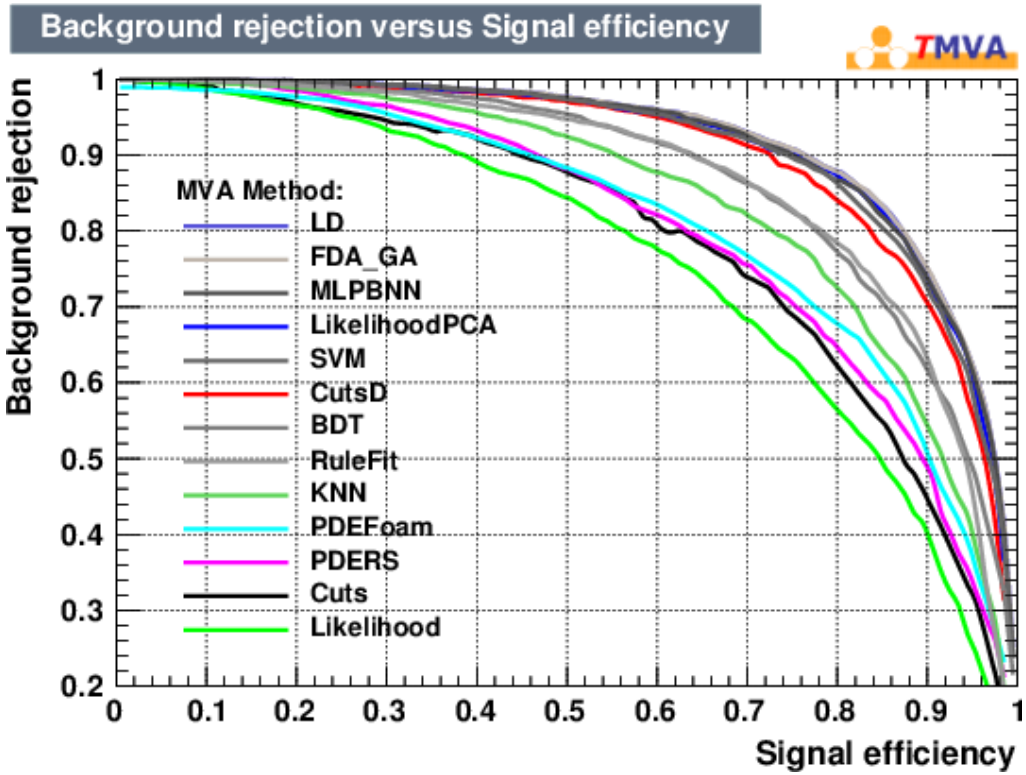
Advanced Root Tools

UNIVERSITÉ DE GENÈVE

- TMVA
  - Toolkit for mulitvariate analysis
  - Machine learning methods for classification and regression
  - Documentation: Users guide and web site
  - Exercise in tomorrow's tutorial!
- RooFit
  - Toolkit for data modelling
  - Maximum likelihood fitting, toy MC generator, visualisation
  - Documentation: Quick introduction, RooFit in 20 min & users manual
- PROOF
  - Parallel ROOT facility
  - Parallelize running on large sets of ROOT files
  - Documentation: Introduction

TMVA – Multivariate analysis

- Combine multiple inputs into a single (few) output
- Same processing for all methods, parallel run, easy to use, fair comparison
- Many machine learning techniques available (more coming)
  - Boosted decision tree (BDT)
  - Neural networks (NN)
  - Probability density estimation (PDE)
  - Function discriminant analysis (FDA)
  - Support Vector Machine (SVM)
  - Predictive learning (RuleFit)
  - etc.
- Many visualisations
- Many monitoring tools
- Modern ML not yet integrated

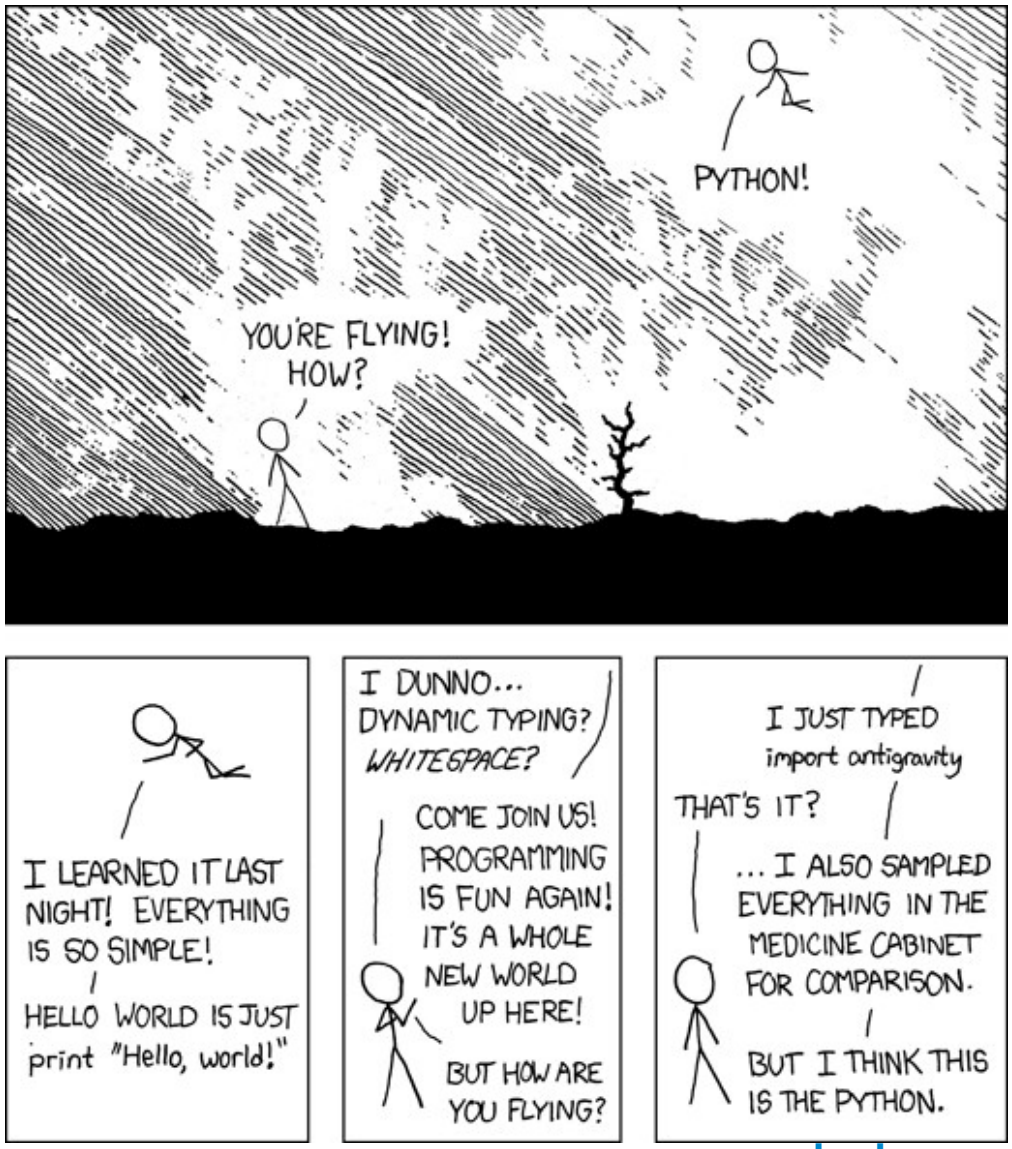


PyROOT

- ROOT interface to Python
- Real mix of the two languages
- Power of C++
  - compiled libraries
- Flexibility of Python
  - Easy to use

```
import ROOT
```

- Introduction
  - Tomorrow's tutorial!



xkcd.com

Conclusions

- ROOT is a powerful analysis framework for particle physics
  - Most of the daily tasks of HEP can be performed
  - C++ or PyROOT have both their advantages and disadvantages
  - Interactive ROOT only for very quick checks
  - Write macros or small programs for normal usage
  - Many basic ROOT objects introduced, much more exist
  - Nice plots time consuming and tricky, note B&W readibility!
- Only a very tiny fraction covered in this lecture
  - You are now experienced enough to use documentation!
  - Explore the examples in tutorials/ subdirectory of ROOT
- See you all again at tomorrow's tutorial!
  - $H \rightarrow \gamma\gamma$ search, PyROOT, TMVA introduction

Additional Slides