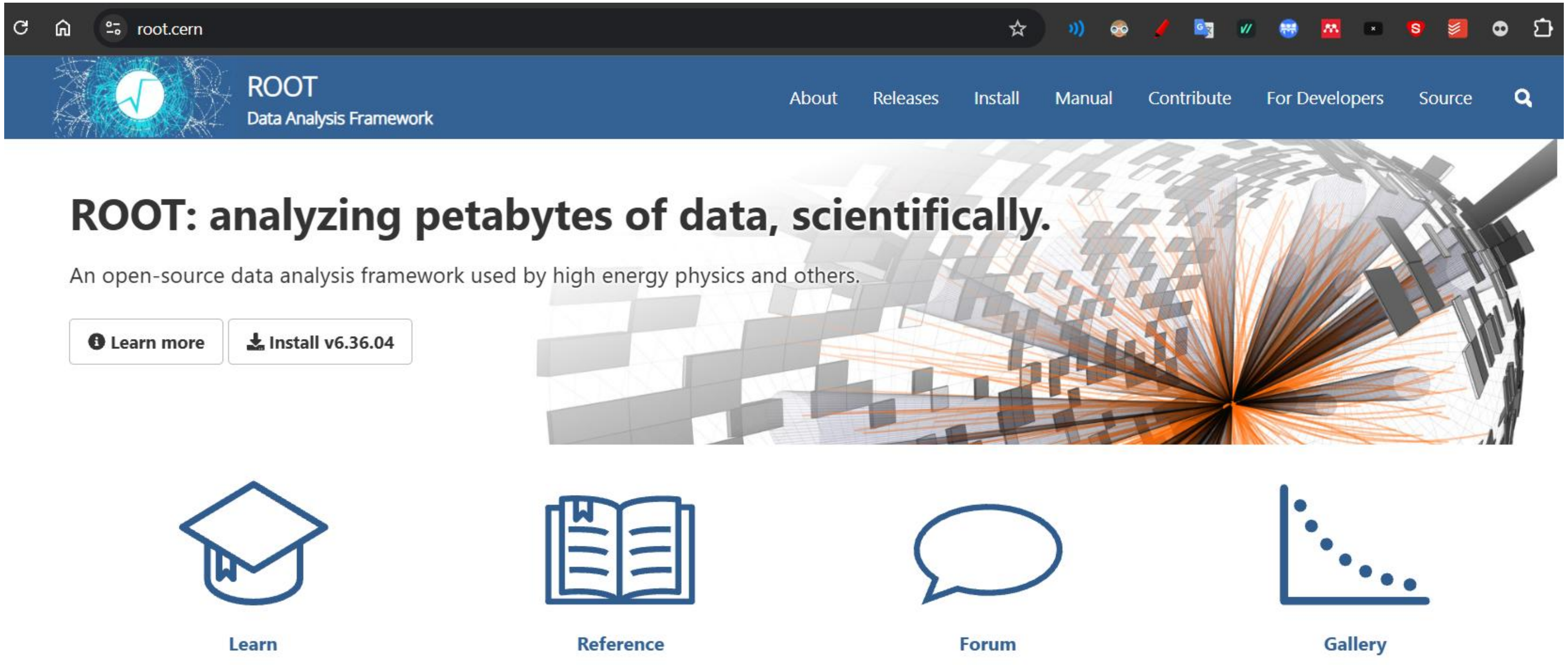
ROOT Primer

Installation + Basic Usage

Satyam


root.cern

ROOT
Data Analysis Framework

About Releases Install Manual Contribute For Developers Source

ROOT: analyzing petabytes of data, scientifically.

An open-source data analysis framework used by high energy physics and others.

ROOT is a scientific data-analysis framework heavily used in nuclear/particle physics (histograms, TTrees, fitting, etc.).

What is CERN ROOT?

Contents

PDF ^_^

Prerequisites: Basic C++ (specially, revise concept of pointers)

Installation

https://root.cern/install/

https://root.cern/manual/

Basic Usage

- **M1**: Using Interactive Shell
- **M2**: Histograms & Random Number Generation
- **M3**: TTree (*Saving Data*)
- **M4**: Reading & Analyzing

Installation

Depends on your operating system:

a. Linux => Use APT

b. macOS =>Use Homebrew

c. Windows => Install WSL first, then use APT

Alex Dobrenko
@Dobrenkz
therapist: whats your attachment style?
me: I dunno...pdf?
3:40 PM · 20 Apr 22 · Twitter Web App
850 Retweets 21 Quote Tweets 9,172

3 Ways:

a. Binary Installation (Easiest)

b. Source Installation (Slower, but configurable) For advanced users (PDF ^_^)

c. Using Package Managers (**Homebrew**, Conda, **APT**)

Windows: <Powershell> `wsl --install`

Linux (APT)

- `sudo apt update`
- `sudo apt install root-system`

macOS (Homebrew)

- `brew update`
- `brew install root`

To confirm installation: `root --version`

To see the installation directory: `snap info root-framework`

Module 1: The Interactive Shell

ROOT has a built-in C++ interpreter called **CLING**. You can type C++ line-by-line and see results immediately.

a. Calculator
b. Function Plotter

ROOT as a Calculator

```
root [0] 1+1
(int) 2
root [1] 2*(4+2)/12.
(double) 1.000000
root [2] sqrt(3.)
(double) 1.732051
root [3] 1 > 2
(bool) false
root [4] TMath::Pi()
(double) 3.141593
root [5] TMath::Erf(.2)
(double) 0.222703
```

Using Function Plotter of ROOT (TF1 => The Formula 1-Dimensional)

Syntax: `new TF1("InternalName", "Formula", Min, Max);`

```
// Draw a simple parabola x² from -10 to 10
TF1 *f1 = new TF1("func1", "x*x", -10, 10);
f1->Draw();

// Draw a sine wave
TF1 *f2 = new TF1("func2", "sin(x)", 0, 10);
f2->Draw();
```

Parametric Functions

Math: $f(x) = p_0 * x + p_1$

ROOT: "[0]*x + [1]”

```
// Define a line with 2 parameters ([0] and [1])

TF1 *line = new TF1("line", "[0]*x + [1]", 0, 10);

// Set the values of the parameters

line->SetParameter(0, 2.5); // Slope (m)

line->SetParameter(1, 5.0); // Intercept (c)

line->Draw();
```

```
// Arguments: ("Internal_Name", "Title; X-Axis; Y-Axis", Bins, Min, Max)
TH1F *hist = new TH1F("hist", "Invariant Mass; Mass [GeV]; Counts", 100, 0, 10);
```

Module 2: Histograms & Random Numbers

In High Energy Physics, we don't look at single events; we look at statistical distributions. We use Histograms (TH1) to count events. Since we are doing Monte Carlo (MC), we need to generate random data.

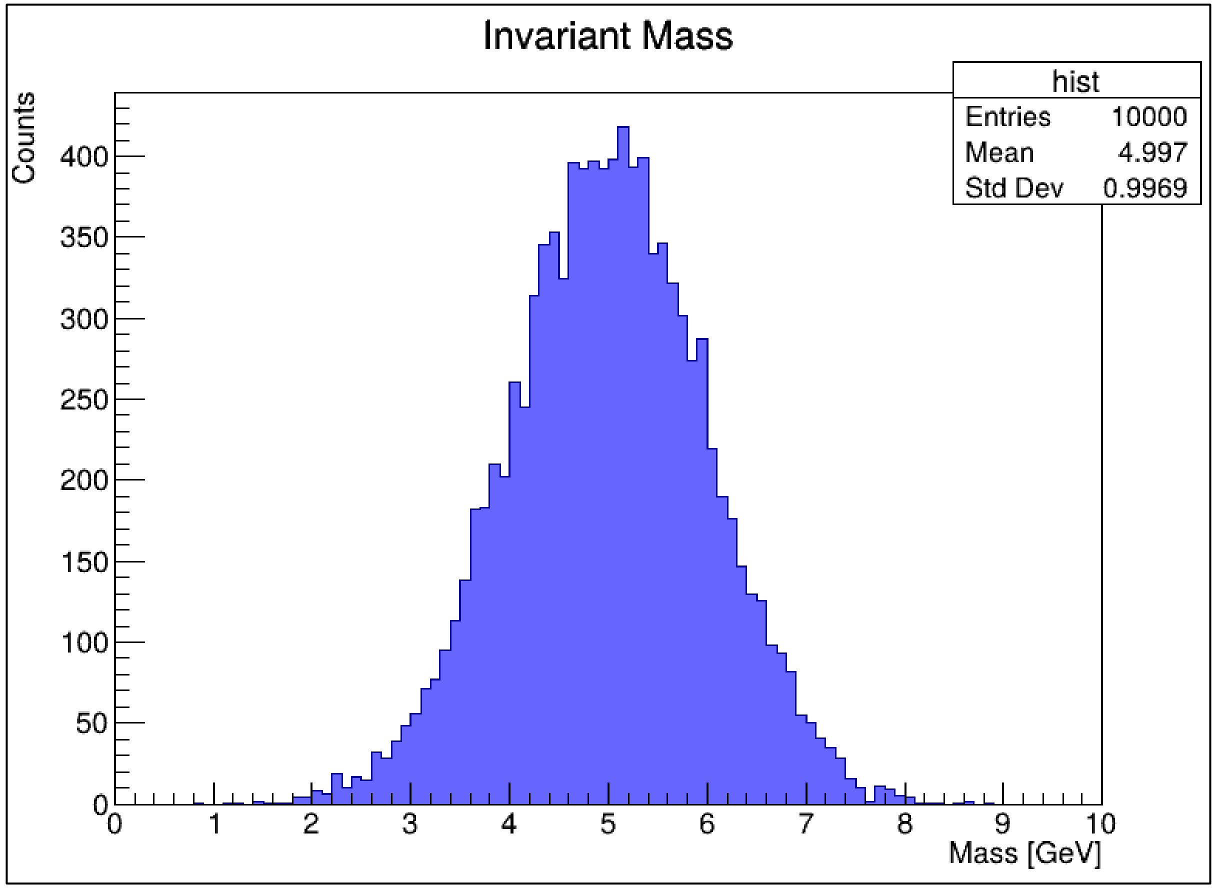
a. TRandom3 is the best random engine in ROOT

b. Filling the histogram

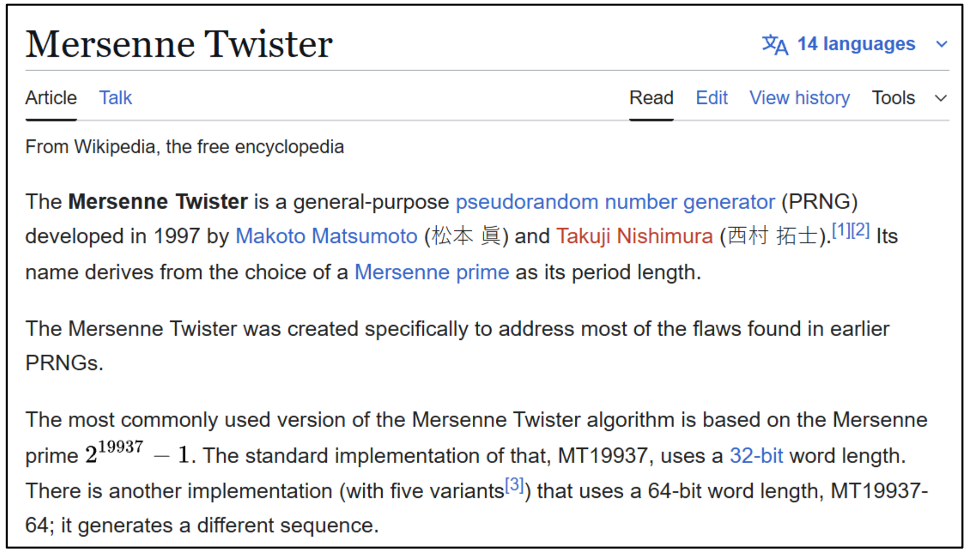
Module-2/basics.C

Module-2/basics.C

This ROOT macro simulates 10,000 Gaussian-distributed mass values (mean 5 GeV, sigma 1 GeV) and fills them into a histogram ranging from 0 to 10 GeV, then displays it on a canvas.



(Extras) TRandom3


Mersenne Twister

14 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

The **Mersenne Twister** is a general-purpose pseudorandom number generator (PRNG) developed in 1997 by Makoto Matsumoto (松本 眞) and Takuji Nishimura (西村 拓士).[1][2] Its name derives from the choice of a Mersenne prime as its period length.

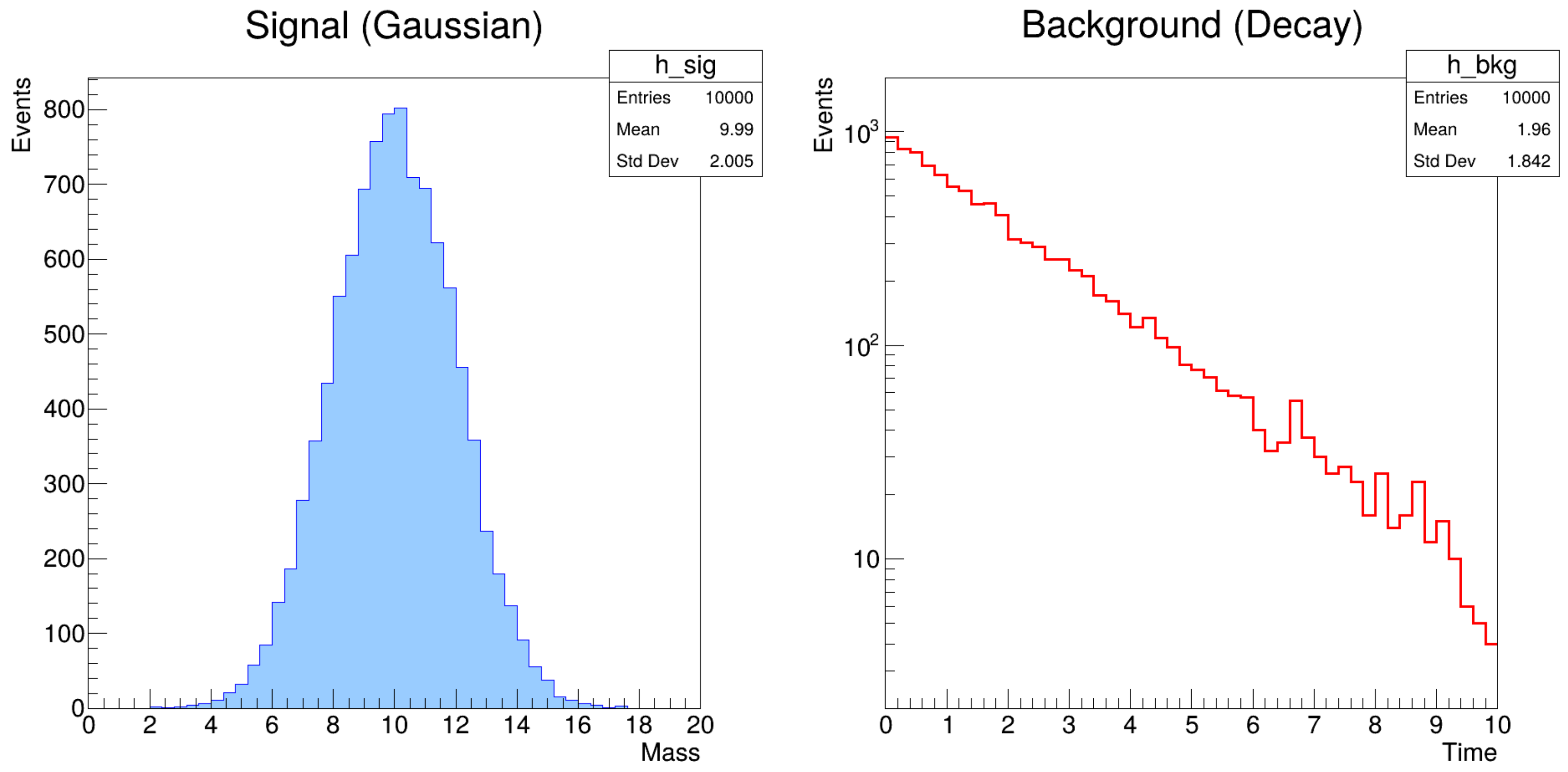
The Mersenne Twister was created specifically to address most of the flaws found in earlier PRNGs.

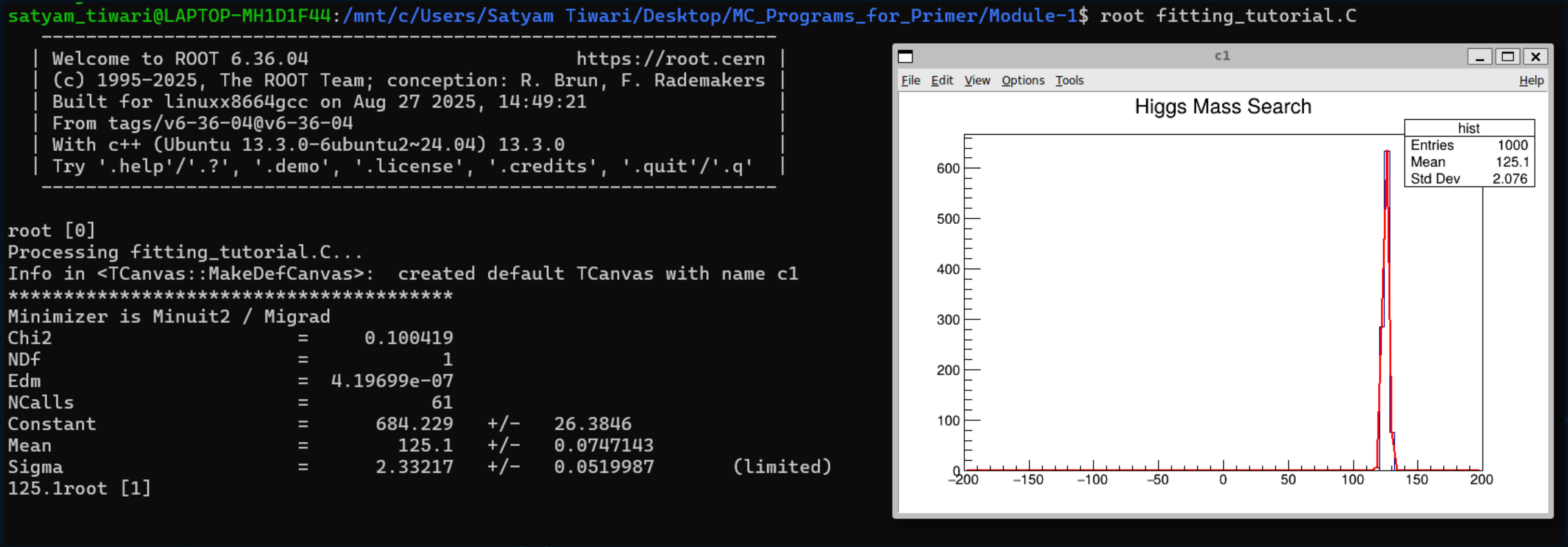
The most commonly used version of the Mersenne Twister algorithm is based on the Mersenne prime $2^{19937}-1$. The standard implementation of that, MT19937, uses a 32-bit word length. There is another implementation (with five variants[3]) that uses a 64-bit word length, MT19937-64; it generates a different sequence.

```
TRandom3 *r = new TRandom3(0);

// 1. Flat Distribution (Noise) from 0 to 10
double x = r->Uniform(0, 10);

// 2. Exponential Decay (Lifetime of particles)
// Probability drops as e^(-t/tau)
double t = r->Exp(2.5); // Mean lifetime = 2.5

// 3. Poisson (Counting Statistics)
// "I expect 5 events. How many did I actually see?"
int n = r->Poisson(5);
```

Advanced Stuff with TCanvas (Making Subplots and log scaling)

`Module-2/advanced_tutorial.C`





```
satyam_tiwari@LAPTOP-MH1D1F44:/mnt/c/Users/Satyam Tiwari/Desktop/MC_Programs_for_Primer/Module-1$ root fitting_tutorial.C
   ------------------------------------------------------------------
  | Welcome to ROOT 6.36.04                        https://root.cern |
  | (c) 1995-2025, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Aug 27 2025, 14:49:21                 |
  | From tags/v6-36-04@v6-36-04                                      |
  | With c++ (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0]
Processing fitting_tutorial.C...
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
****************************************
Minimizer is Minuit2 / Migrad
Chi2                      =     0.100419
NDf                       =            1
Edm                       =  4.19699e-07
NCalls                    =           61
Constant                  =      684.229   +/-   26.3846
Mean                      =        125.1   +/-   0.0747143
Sigma                     =      2.33217   +/-   0.0519987       (limited)
125.1root [1]
```

Advanced Stuff with TF1 (& TF1H)

- Module-1/fitting_tutorial.C => Fitting using TF1 (`hist->Fit("gaus")`)
- Module-1/weird_mc.C => Generating from Custom Distribution using TF1 (`f1->GetRandom()`)

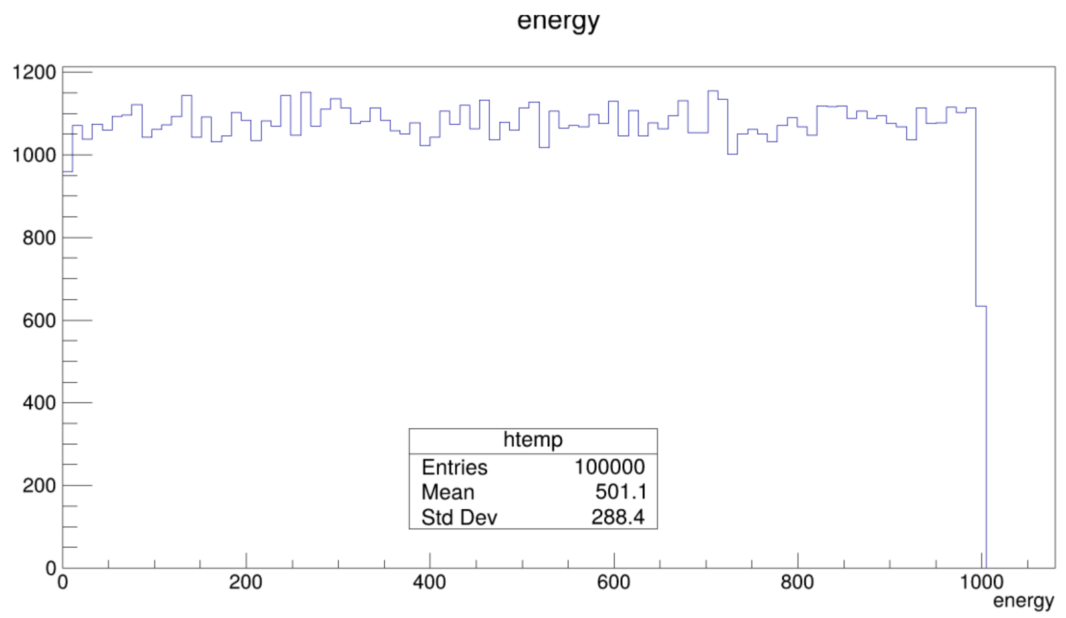
Module 3: TTree

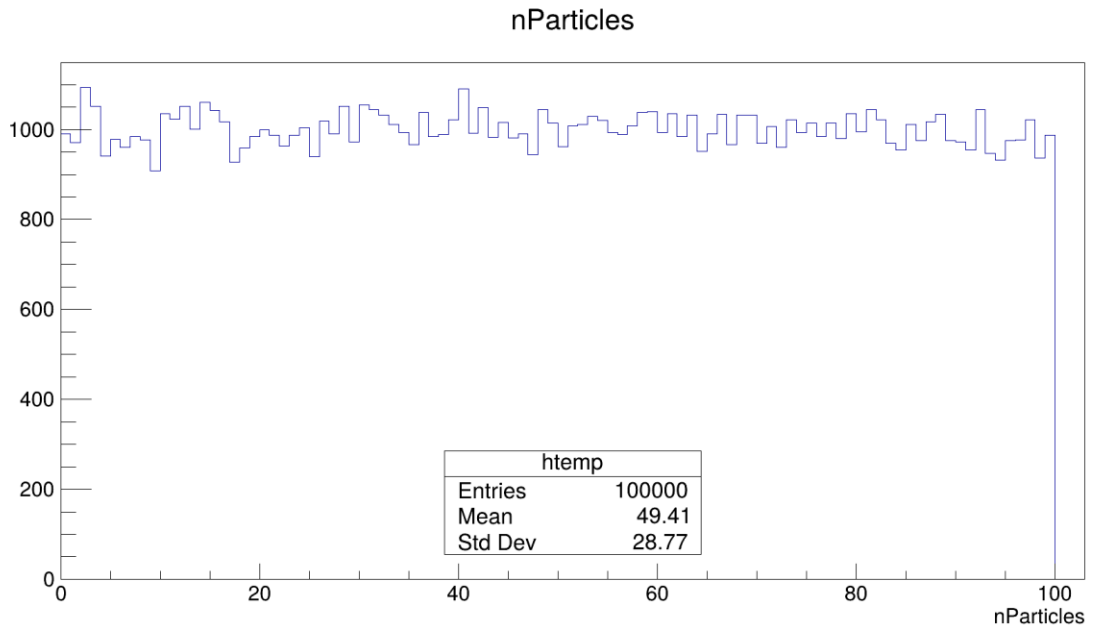
A Histogram is just a picture. If you want to save the raw data to analyze later, you use a TTree.

- **Tree:** The Table.
- **Branch:** The Column.
- **Leaf/Entry:** The Row.

`Module-3/create_data.C`

This ROOT macro creates a file named `experiment_data.root`, generates a tree with 100,000 entries of random energy (0–1000) and particle counts (0–100), and saves it.

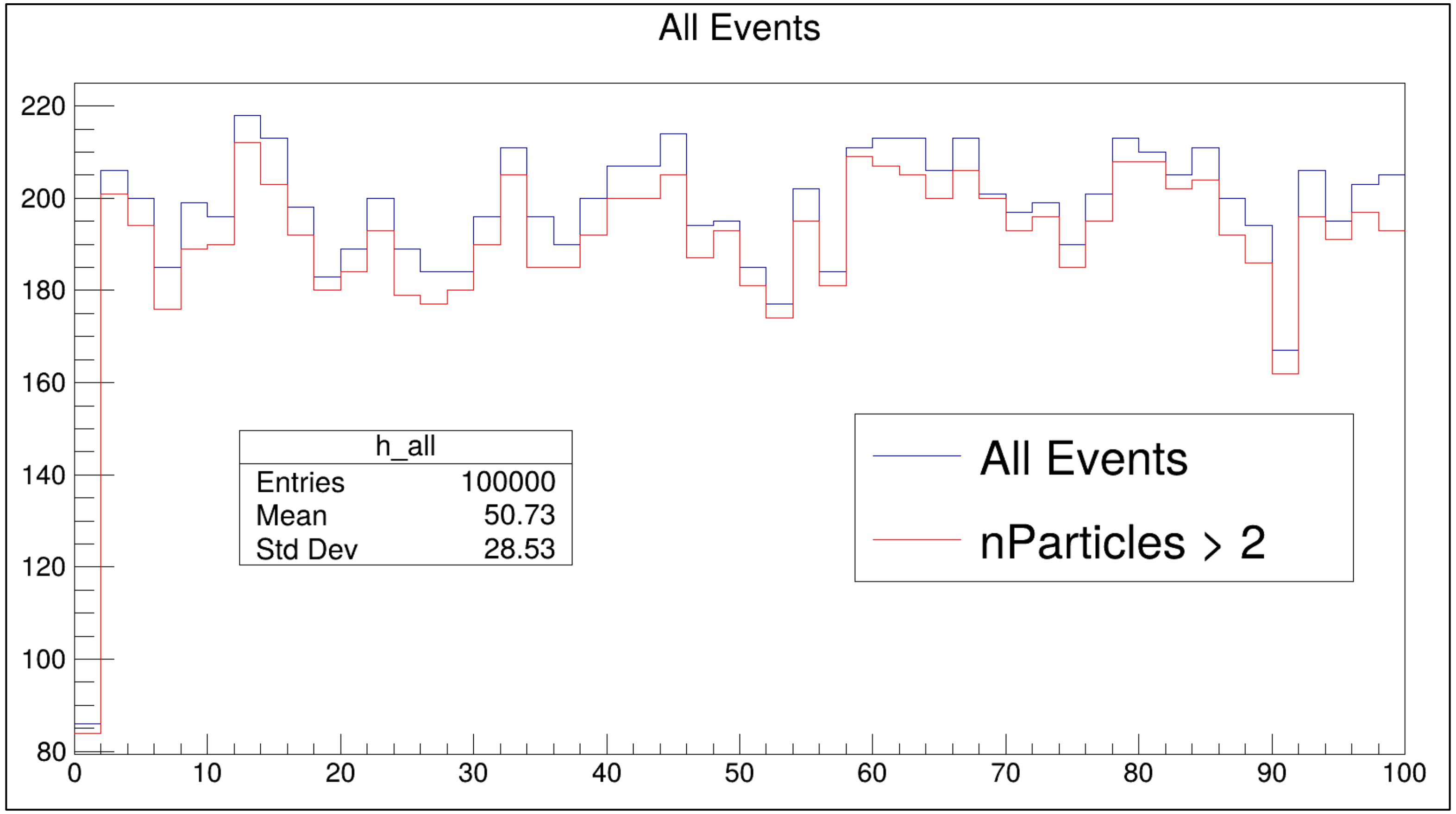




Module 4: Reading & Analysing

This ROOT macro reads experiment_data.root, loops over all tree entries, fills one histogram with all energies and another with energies from events where nParticles > 2, and then draws both histograms on the same canvas with a legend.

Module-4/analyze.C



Thank You