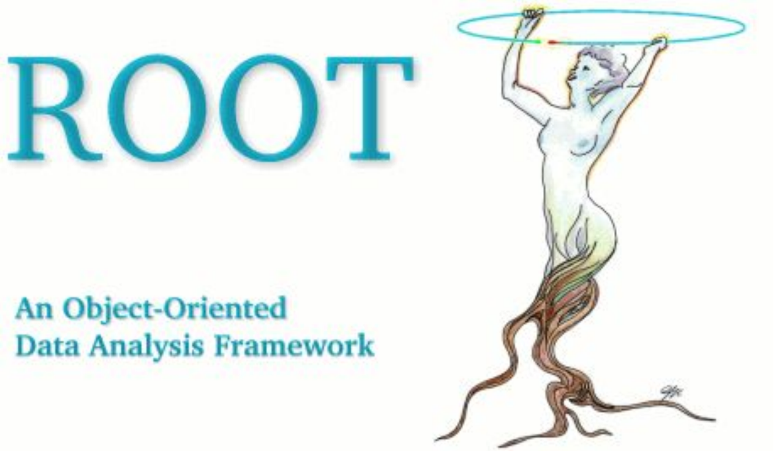
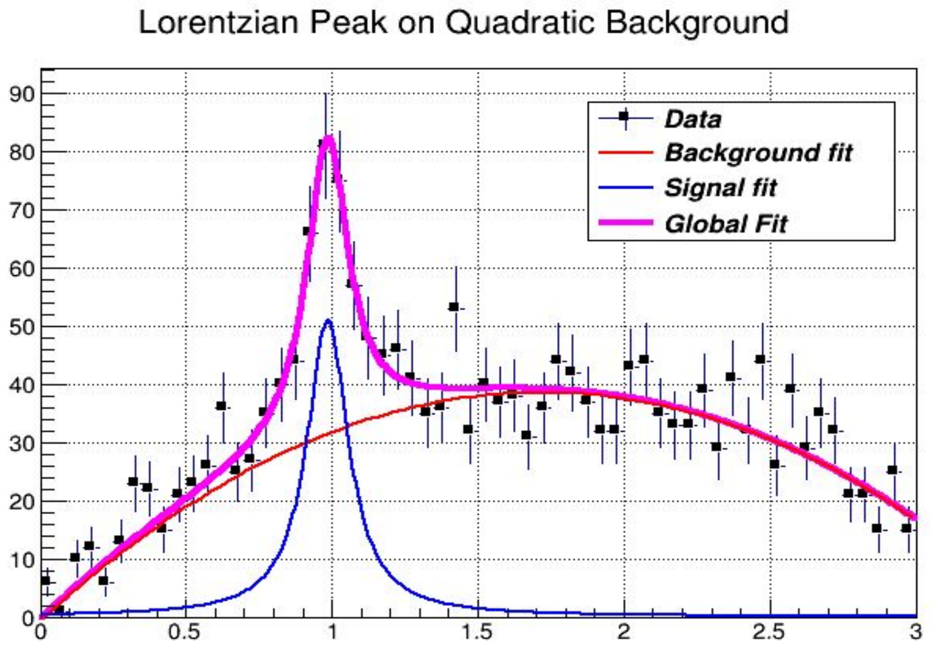
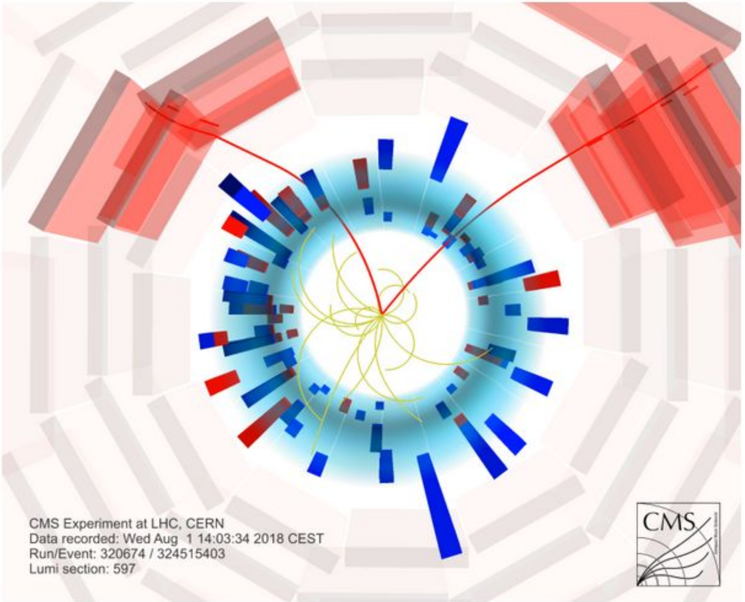






Big data en el CERN y otros contextos

Jhovanny Andres Mejia Guisao
UNIVERSIDAD DE ANTIOQUIA, COLOMBIA

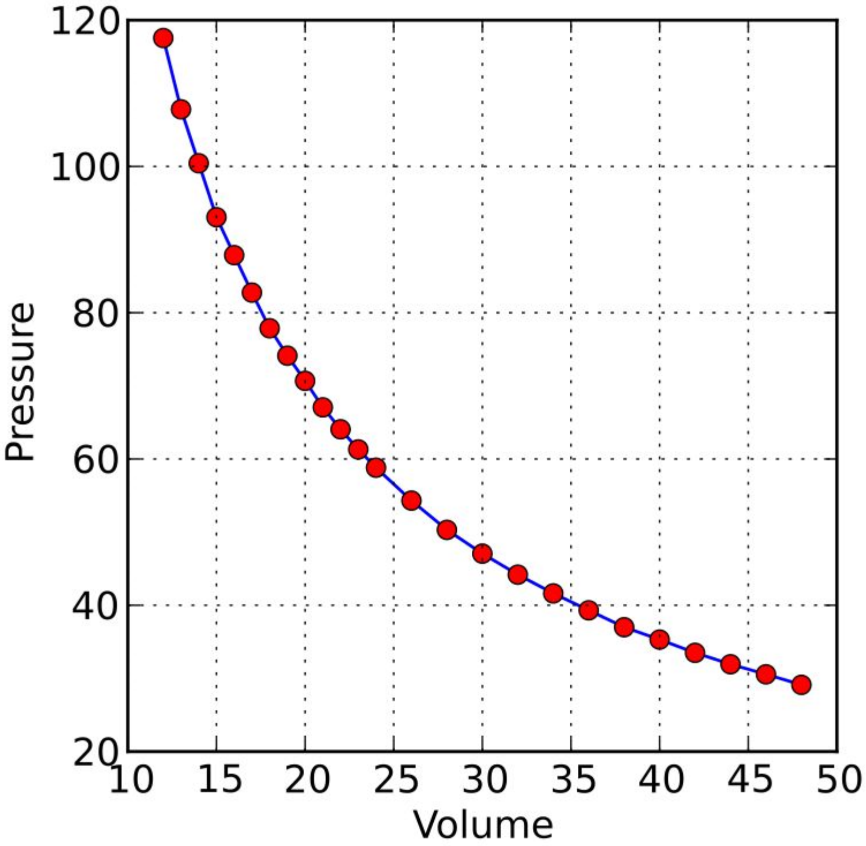
Motivation

What we hope to discuss about scientific data analysis?

- **Advanced graphical user interface**
- **Interpreter for the C++ programming language**
- **Persistency mechanism for C++ objects**
- **Used to write every year petabytes of data recorded by the Large Hadron Collider experiments**

Input and plotting of data from measurements and fitting of analytical functions.

Motivation

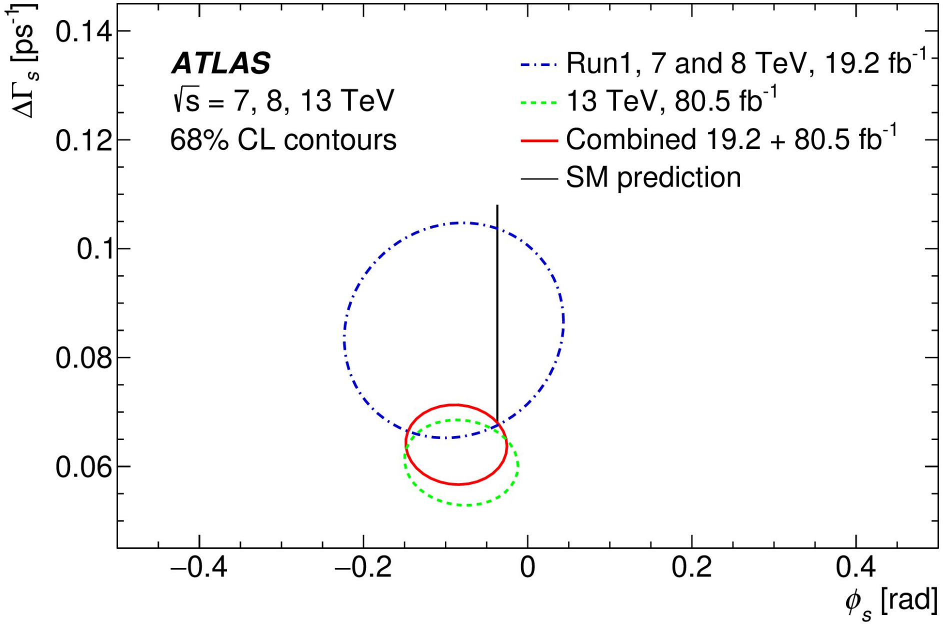


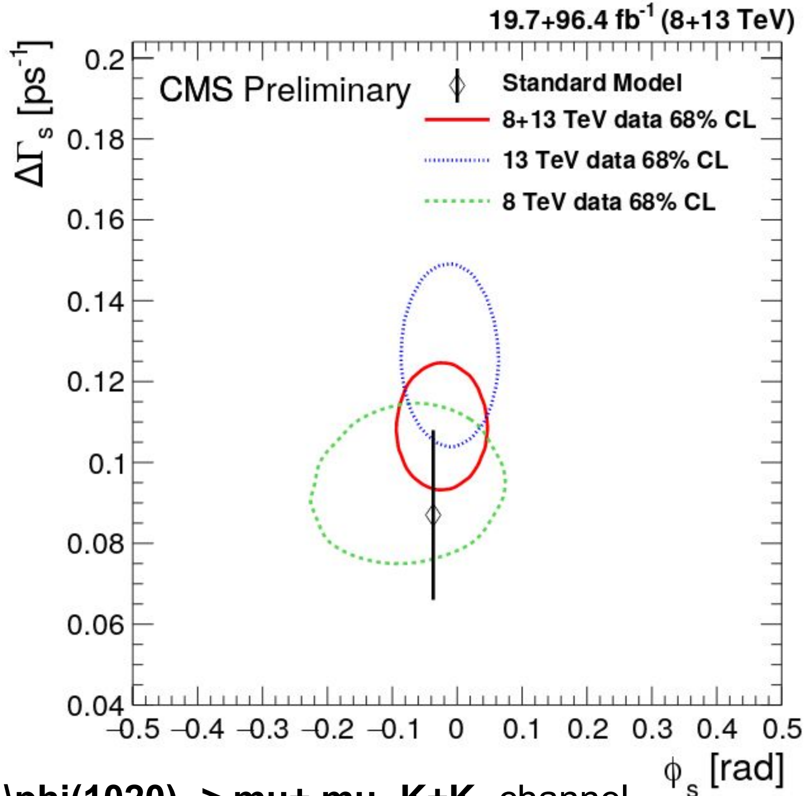
Para un gas a temperatura constante, el volumen es inversamente proporcional a la presión sobre éste[link]

Se puede explicar matemáticamente con:
pV = k

y la tarea del experimentador consiste en determinar la constante, k, a partir de un conjunto de medidas.

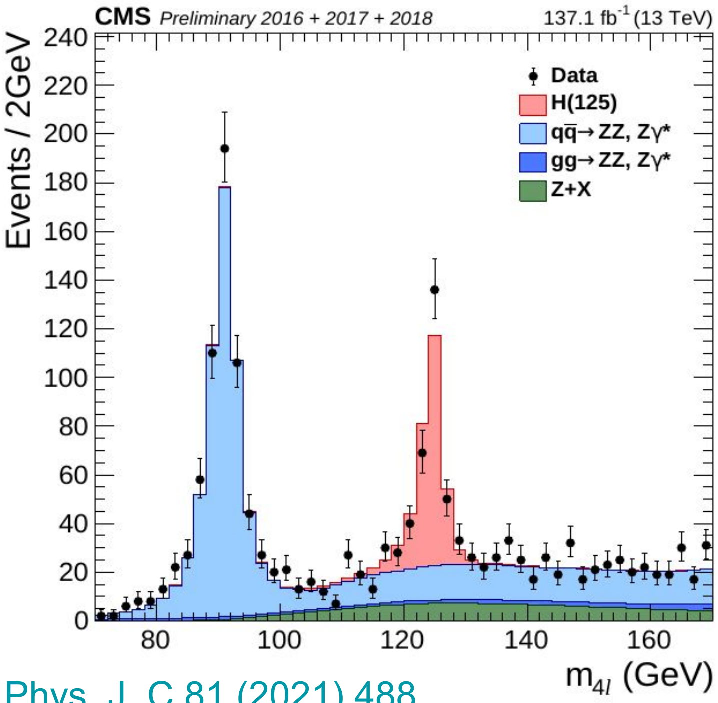
Motivation

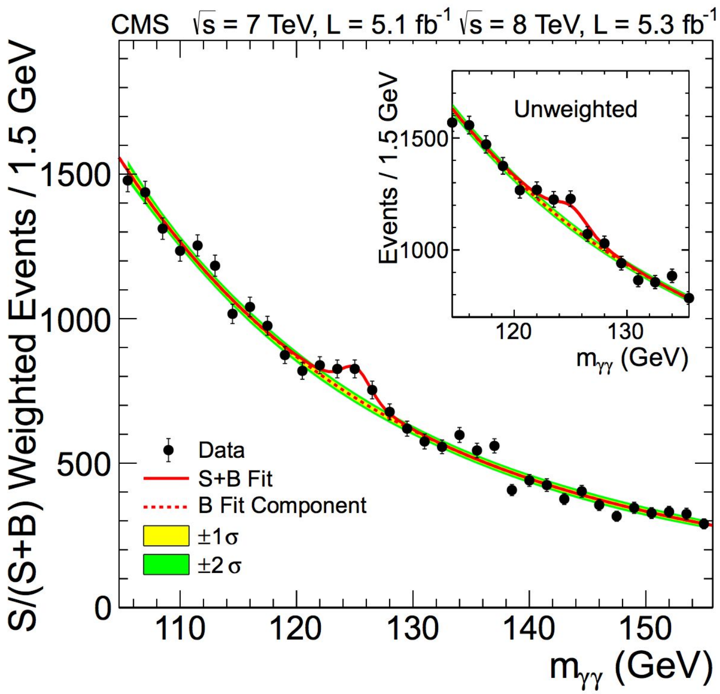




Measurement of the **CP violating phase** in the **Bs-> J/\psi \phi(1020) -> mu+ mu- K+K-** channel in proton-proton collisions at √s = 13 TeV ([PLB 816 (2021) 136188](PLB 816 (2021) 136188))

Motivation



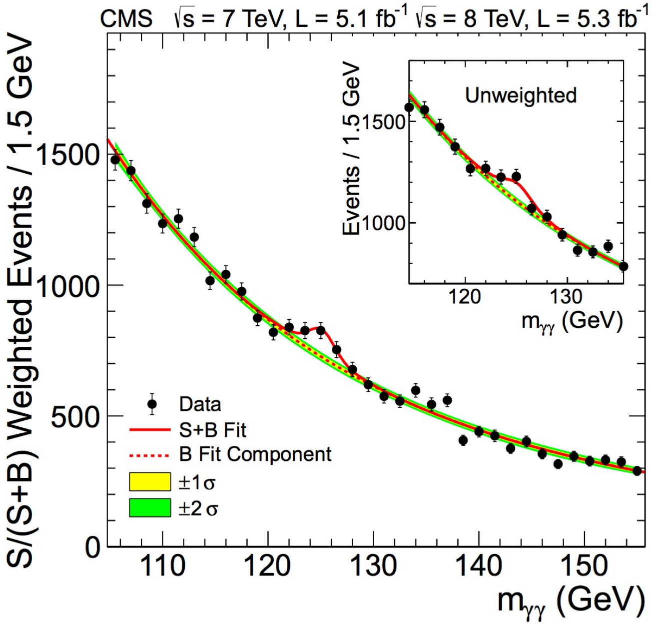


Eur. Phys. J. C 81 (2021) 488

Direct (left) and indirect (right) evidence for the Higgs boson. Left: the invariant mass of four leptons, showing evidence for the Higgs boson at Mh~ 126 GeV. Right: the world average of the W boson mass vs the top-quark mass is shown. Overlayed are the contour plots when the direct Higgs boson mass measurement is used. It is evident that the indirect determination of the Higgs boson mass is quite consistent with the direct measurement.

Motivation

In Quantum mechanics, models typically only predict the **probability density function ("pdf")** of measurements depending on a number of parameters, and the aim of the experimental analysis is to extract the parameters from the **observed distribution** of frequencies at which certain values of the measurement are observed. Measurements of this kind require means to generate and visualize frequency distributions, **so-called histograms,** and **stringent statistical treatment to extract the model parameters from purely statistical distributions.**



Motivation

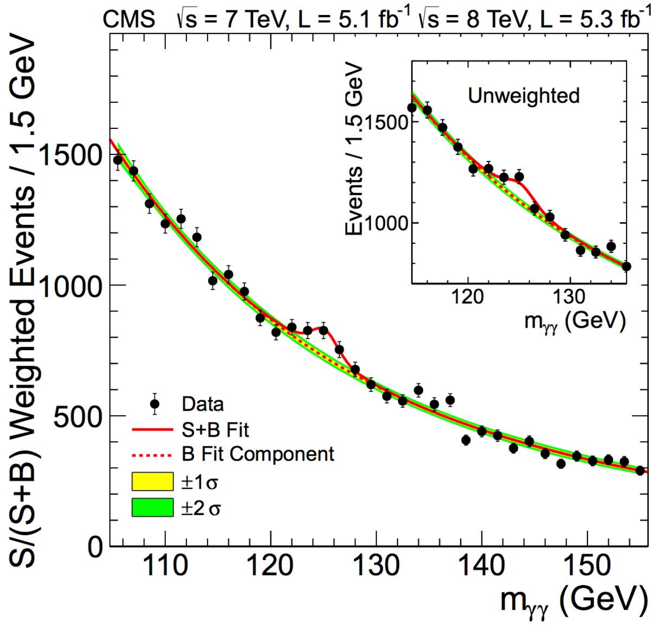
Visualization of the data

Corrections or parameter transformations?

One specialty of experimental physics are the inevitable uncertainties affecting each measurement.

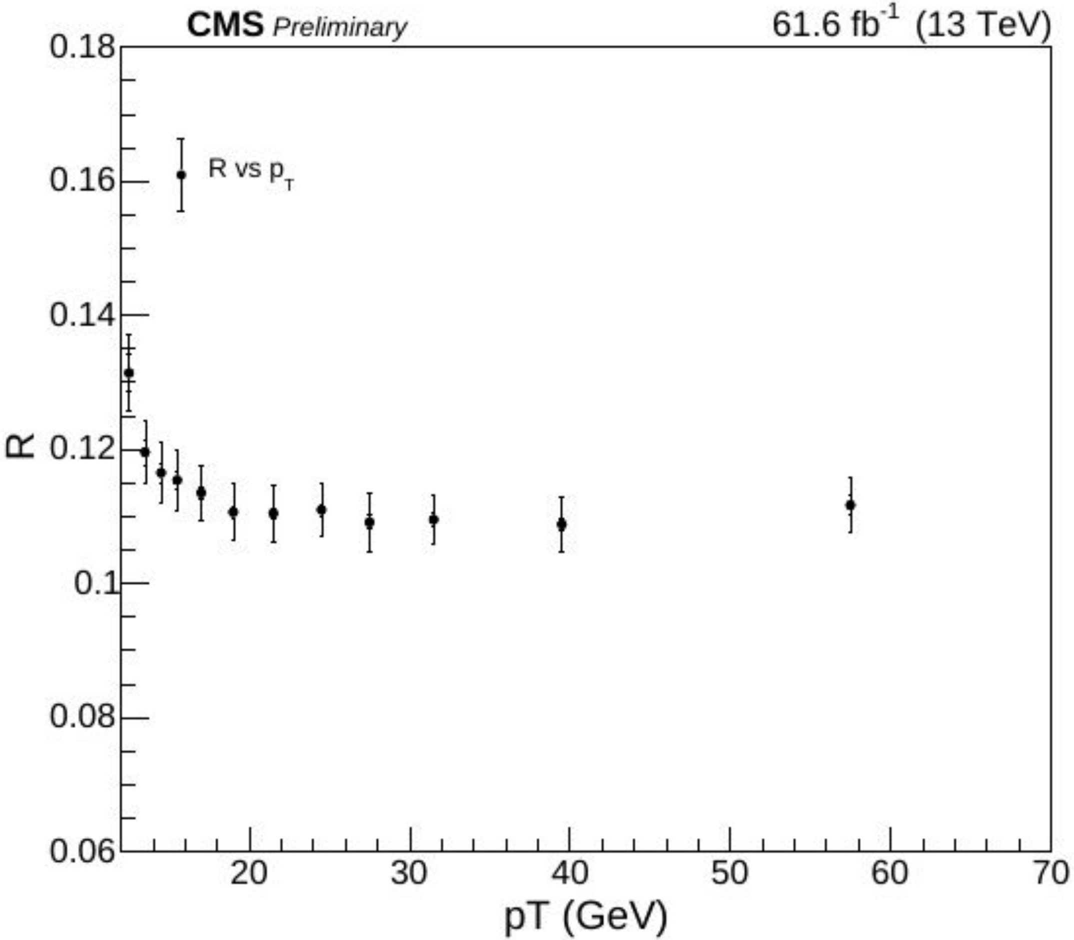
The statistical nature of the errors must be handled properly.

Quite often, the data volume to be analyzed is large - think of fine-granular measurements accumulated with the aid of computers. A usable tool therefore must contain easy-to-use and efficient methods for storing and handling data.



Simulation of expected data is another important aspect in data analysis. By repeated generation of "**pseudo-data**", which are analysed in the same manner as intended for the real data, analysis procedures can be validated or compared.

Motivation



what does this distribution tell us?

why there are two errors? or more?

how can we understand/handle these errors?

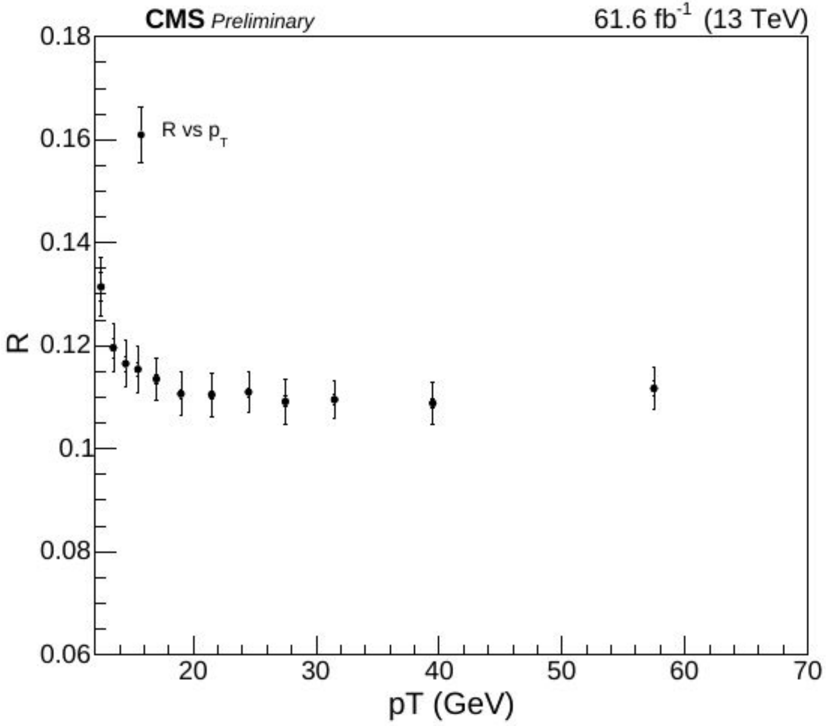
[BPH-21-001](BPH-21-001)

Motivation

Table 11: $\mathcal{R}$ and uncertainty (σ_{tot}), including the statistical (σ_{stat}) uncertainty and the fully correlated and uncorrelated systematic uncertainties among the samples ($\sigma_{\text{sys}}^{\text{uncor}}$, $\sigma_{\text{sys}}^{\text{cor}}$). Correlations stem from the common tracking and fit model uncertainties.

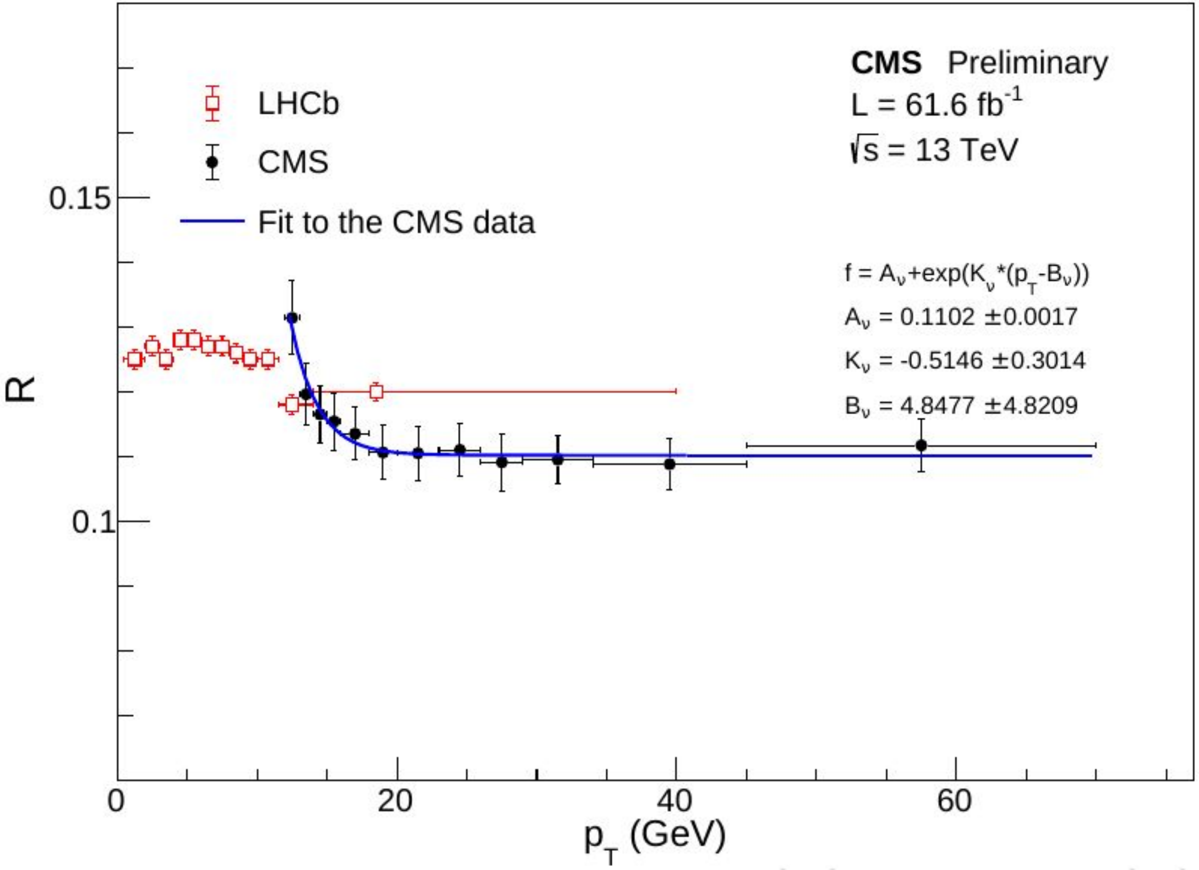
p_T (GeV)	$\mathcal{R}$	σ_{tot}	σ_{stat}	$\sigma_{\text{sys}}^{\text{uncor}}$	$\sigma_{\text{sys}}^{\text{cor}}$
12 – 13	0.1314	0.0058	0.0028	0.0033	0.0038
13 – 14	0.1196	0.0046	0.0019	0.0015	0.0039
14 – 15	0.1165	0.0041	0.0015	0.0016	0.0035
15 – 16	0.1154	0.0042	0.0014	0.0018	0.0036
16 – 18	0.1135	0.0040	0.0009	0.0022	0.0032
18 – 20	0.1106	0.0041	0.0009	0.0022	0.0033
20 – 23	0.1105	0.0042	0.0008	0.0024	0.0034
23 – 26	0.1110	0.0040	0.0009	0.0023	0.0031
26 – 29	0.1091	0.0044	0.0010	0.0020	0.0038
29 – 34	0.1095	0.0037	0.0010	0.0022	0.0028
34 – 45	0.1088	0.0040	0.0009	0.0023	0.0032
45 – 70	0.1117	0.0041	0.0014	0.0021	0.0033

$$\begin{pmatrix}
3.214 & 0.663 & 0.545 & 0.571 & 0.450 & 0.513 & 0.520 & 0.421 & 0.677 & 0.313 & 0.464 & 0.473 \\
0.663 & 2.242 & 0.645 & 0.676 & 0.533 & 0.608 & 0.615 & 0.498 & 0.801 & 0.370 & 0.549 & 0.560 \\
0.545 & 0.645 & 2.005 & 0.556 & 0.438 & 0.499 & 0.506 & 0.410 & 0.658 & 0.304 & 0.451 & 0.460 \\
0.571 & 0.676 & 0.556 & 1.973 & 0.459 & 0.523 & 0.530 & 0.429 & 0.690 & 0.319 & 0.473 & 0.482 \\
0.450 & 0.533 & 0.438 & 0.459 & 1.721 & 0.413 & 0.418 & 0.338 & 0.544 & 0.251 & 0.373 & 0.380 \\
0.513 & 0.608 & 0.499 & 0.523 & 0.413 & 1.766 & 0.476 & 0.386 & 0.620 & 0.287 & 0.425 & 0.433 \\
0.520 & 0.615 & 0.506 & 0.530 & 0.418 & 0.476 & 1.781 & 0.391 & 0.627 & 0.290 & 0.430 & 0.439 \\
0.421 & 0.498 & 0.410 & 0.429 & 0.338 & 0.386 & 0.391 & 1.585 & 0.508 & 0.235 & 0.349 & 0.355 \\
0.677 & 0.801 & 0.658 & 0.690 & 0.544 & 0.620 & 0.627 & 0.508 & 1.966 & 0.378 & 0.560 & 0.571 \\
0.313 & 0.370 & 0.304 & 0.319 & 0.251 & 0.287 & 0.290 & 0.235 & 0.378 & 1.380 & 0.259 & 0.264 \\
0.464 & 0.549 & 0.451 & 0.473 & 0.373 & 0.425 & 0.430 & 0.349 & 0.560 & 0.259 & 1.611 & 0.392 \\
0.473 & 0.560 & 0.460 & 0.482 & 0.380 & 0.433 & 0.439 & 0.355 & 0.571 & 0.264 & 0.392 & 1.691
\end{pmatrix} \cdot 10^{-05}$$



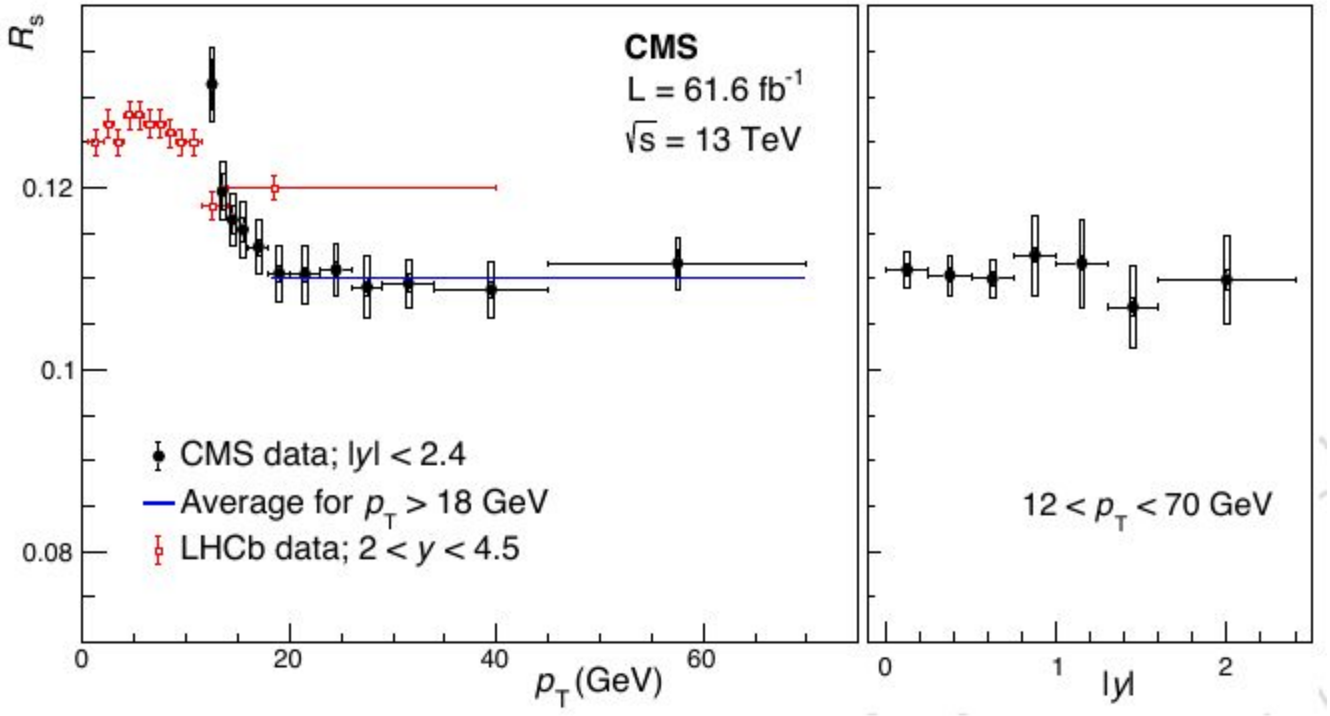
BPH-21-001

Motivation



BPH-21-001

Motivation



BPH-21-001

ROOT es muy flexible y proporciona una interfaz de programación para usar en aplicaciones propias y una interfaz gráfica de usuario para el análisis interactivo de datos.

Descripción general y justificación del curso:

Este curso pretende aportar a la formación de los estudiantes de los programas de pregrado de física y astronomía, aportando a sus conocimientos las técnicas y métodos computacionales necesarios para un mejor entendimiento de los procesos involucrados en el tratamiento de los datos procesados en el CERN en colisiones de partículas. Dada la magnitud de los datos tomados, se han desarrollado técnicas para el manejo de cantidades masivas de información. Se pretende que los estudiantes adquieran habilidades computacionales que le permitan desenvolverse en este medio, ya que es un requisito esencial para poder participar de proyectos relacionados con el área de física de partículas experimental. Adicional a esa idea, sin embargo, es importante decir que también de esta forma se garantiza que el estudiante tenga una formación integral tanto en las bases conceptuales aprendidas en la carrera como en las habilidades computacionales que le serán de gran utilidad enfrentando problemas tanto de tipo académico asi como en el ámbito de la industria en el manejo de grandes cantidades de datos.

Objetivo general:
Adquirir conocimientos básicos y experiencia en el uso del marco de análisis de datos usando el software ROOT del CERN y adquirir habilidades para manejo de grandes volúmenes de datos con el mismo.

la clase de hoy, tomada directamente de:
https://root.cern/get_started/courses/

ROOT in a Nutshell

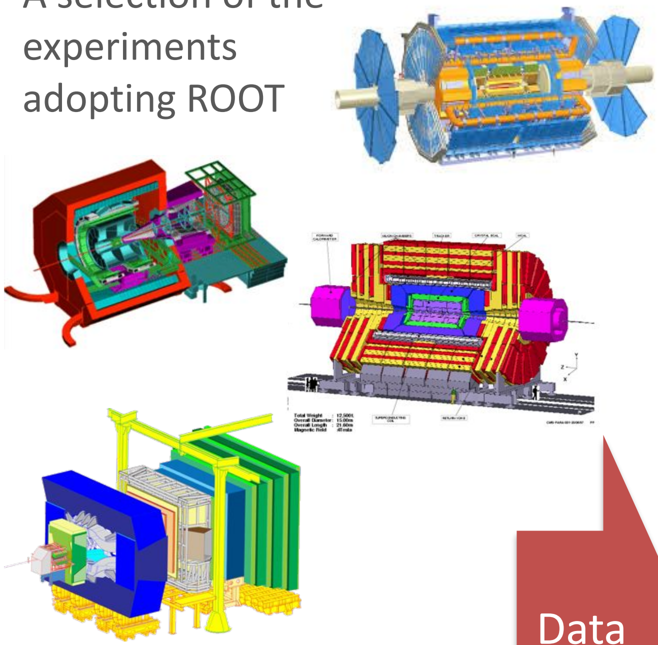
- ROOT is a software framework with building blocks for:
  - Data processing
  - Data analysis
  - Data visualisation
  - Data storage
- ROOT is written mainly in C++ (C++11/17 standard)
  - Bindings for Python available as well
- Adopted in High Energy Physics and other sciences (but also industry)
  - 1 EB of data in ROOT format
  - Fits and parameters’ estimations for discoveries (e.g. the Higgs)
  - Thousands of ROOT plots in scientific publications

ROOT in a Nutshell

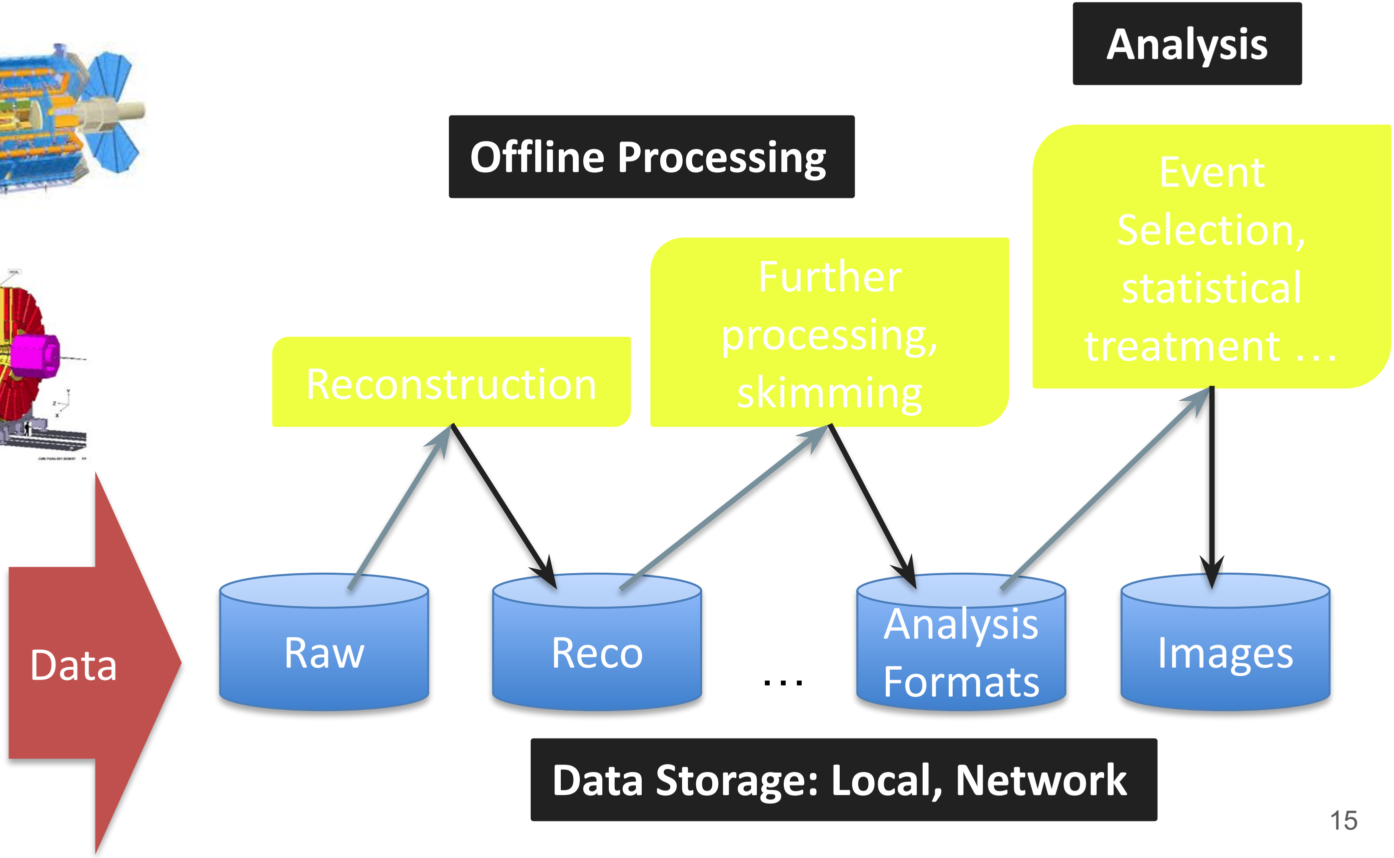
- ROOT can be seen as a collection of building blocks for various activities, like:
  - **Data analysis: histograms, graphs, functions**
  - **I/O: row-wise, column-wise** storage of any C++ object
  - **Statistical tools** (RooFit/RooStats): rich modeling and statistical inference
  - Math: **non trivial functions** (e.g. Erf, Bessel), optimised math functions
  - **C++ interpretation**: full language compliance
  - **Multivariate Analysis** (TMVA): e.g. Boosted decision trees, NN
  - **Advanced graphics** (2D, 3D, event display)
  - **Declarative Analysis**: RDataFrame

ROOT Application Domains

A selection of the experiments adopting ROOT



Event Filtering



Interpreter

- ROOT has a built-in interpreter : CLING
  - C++ interpretation: highly non trivial and not foreseen by the language!
  - One of its kind: Just In Time (JIT) compilation
  - A C++ interactive shell

```
$ root
root[0] 3 * 3
(const int) 9
```

- Can interpret “macros” (non compiled programs)
  - Rapid prototyping possible
- ROOT provides also Python bindings
  - Can use Python interpreter directly after a simple *import ROOT*
  - Possible to “mix” the two languages (????)

Persistency or Input/Output (I/O)

- ROOT offers the possibility to write C++ objects into files
  - This is impossible with C++ alone
  - Used the LHC detectors to write several petabytes per year
- Achieved with serialization of the objects using the reflection capabilities, ultimately provided by the interpreter
  - Raw and column-wise streaming
- As simple as this for ROOT objects: one method - *TObject::Write*

Cornerstone for storage of experimental data

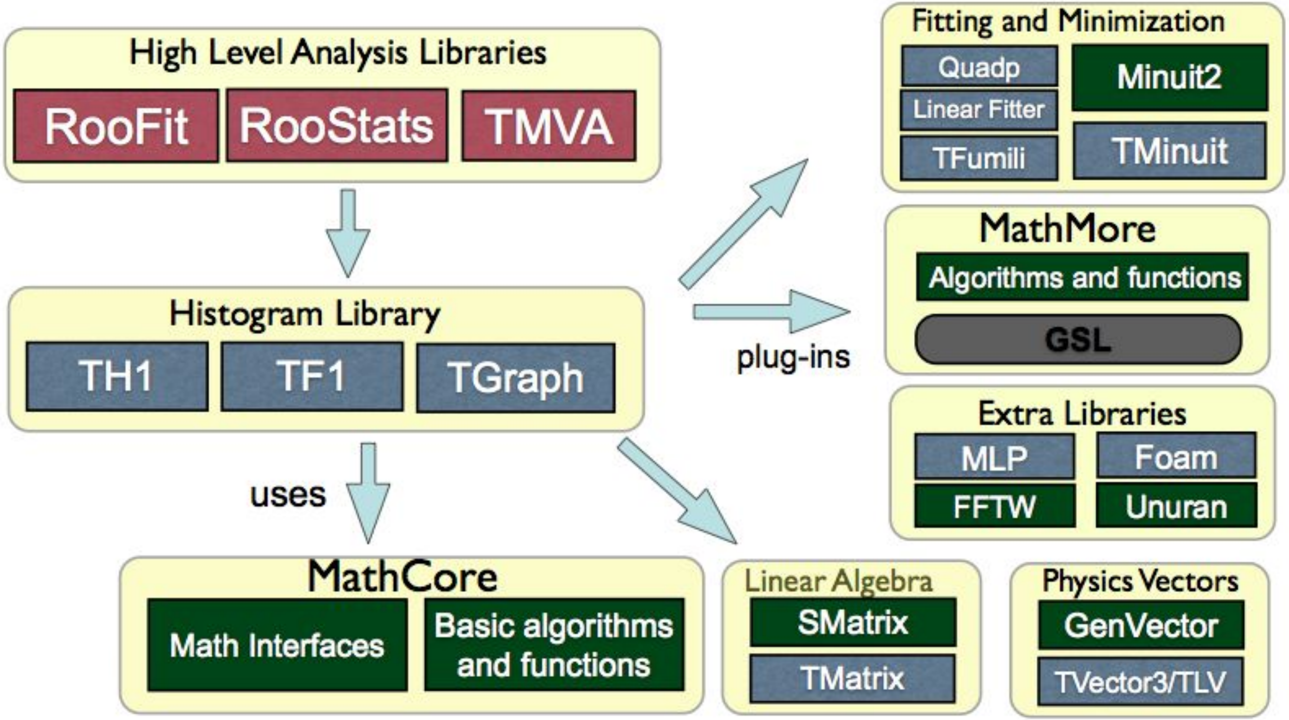
LHC Data in ROOT Format

1 EB

as of 2017

Mathematics and Statistics

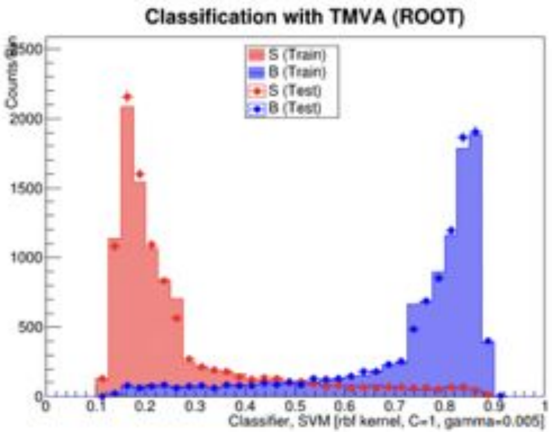
- ROOT provides a rich set of mathematical libraries and tools for sophisticated statistical data analysis



Machine Learning: TMVA

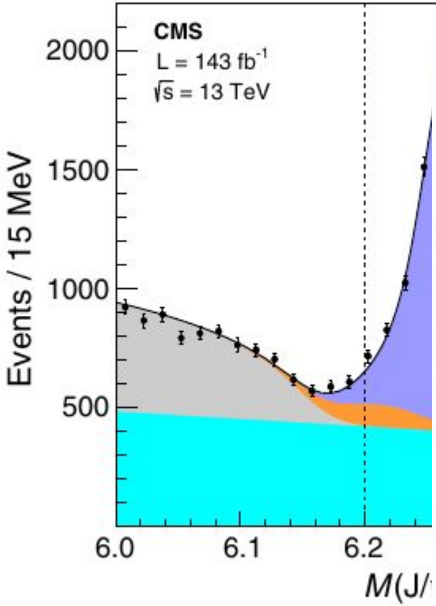
TMVA : Toolkit for Multi-Variate data Analysis in ROOT

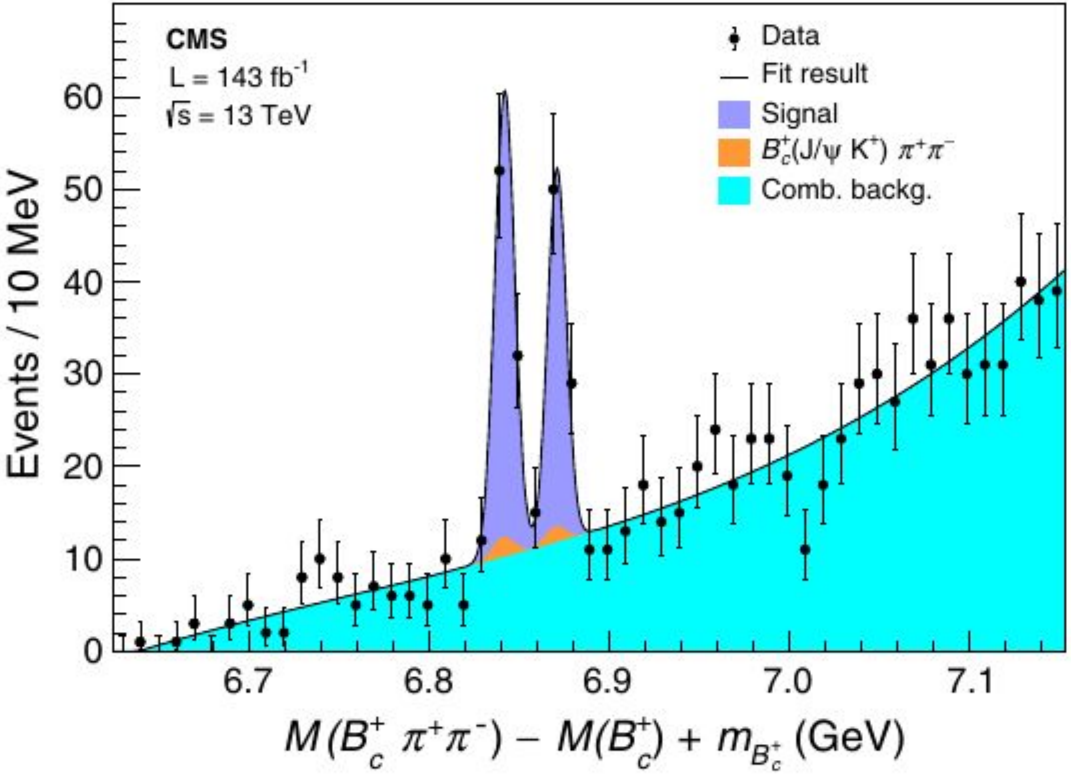
- provides several built-in ML methods including:
  - Boosted Decision Trees
  - Deep Neural Networks
  - Support Vector Machines



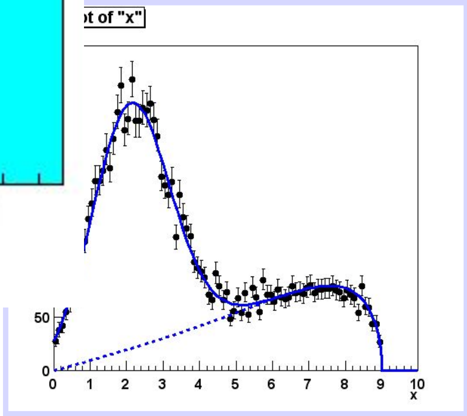
- and interfaces to external ML tools
  - scikit-learn, Keras (Theano/Tensorflow), R

RooFit





d.f.)

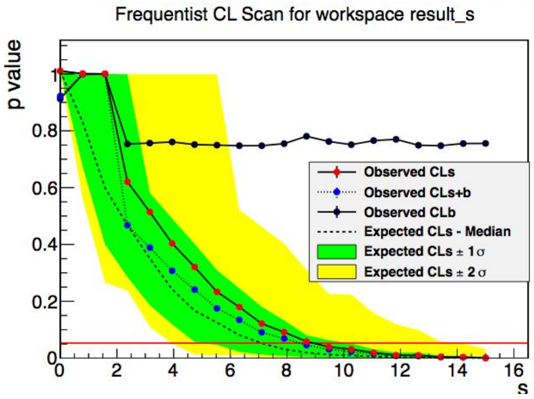


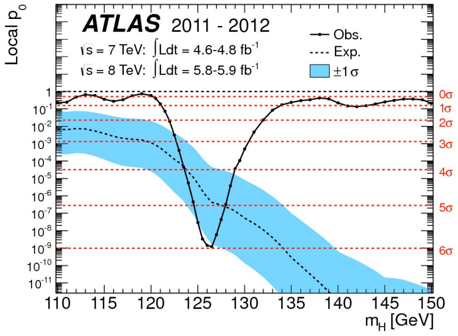
- visualizatic
- sharing an

[BPH-18-007, Phys. Rev. Lett. 122, 132001](#)

RooStats

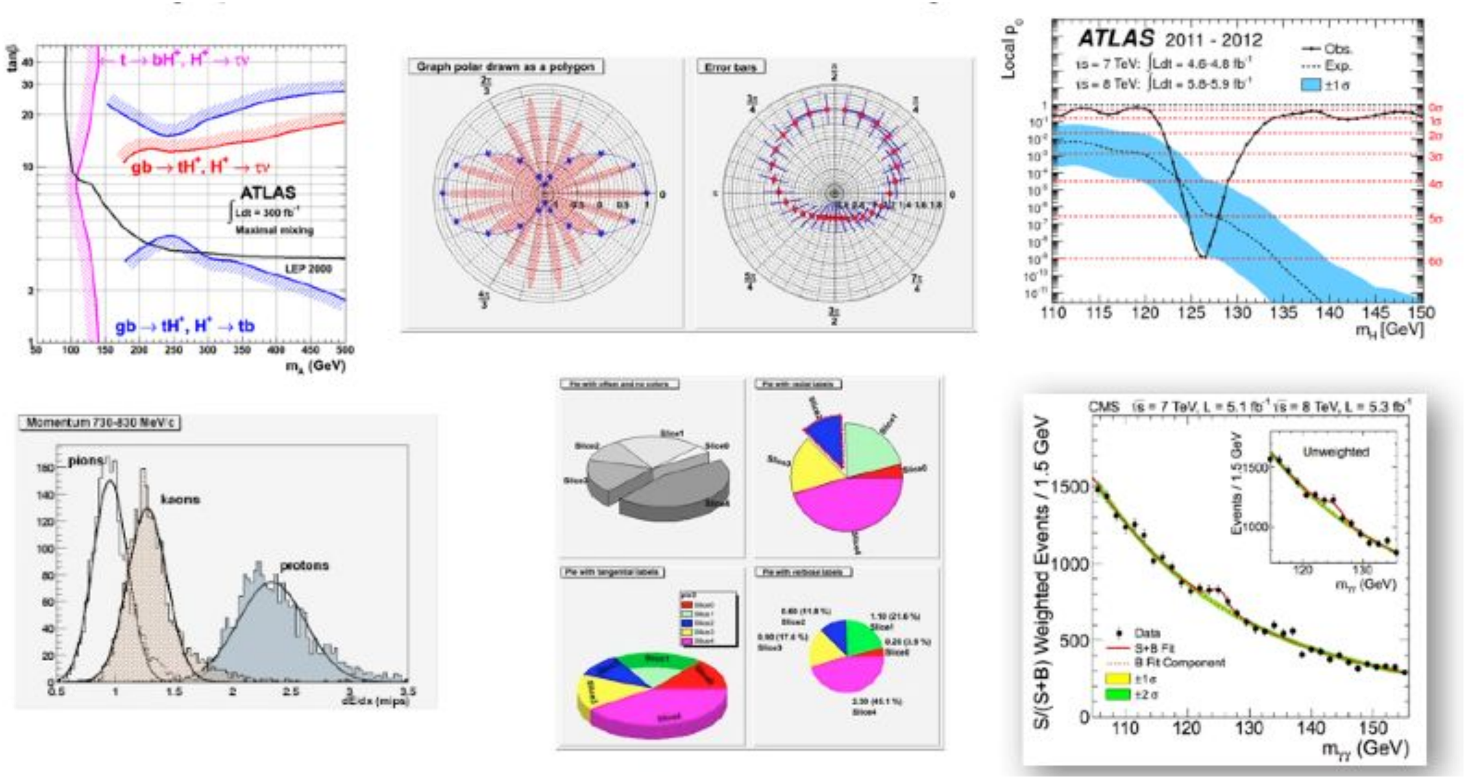
- Advanced Statistical Tools for HEP analysis. Used for :
  - estimation of Confidence/Credible intervals
  - hypotheses Tests
    - e.g. Estimation of Discovery significance
- Provides both Frequentist and Bayesian tools
- Facilitate combination of results



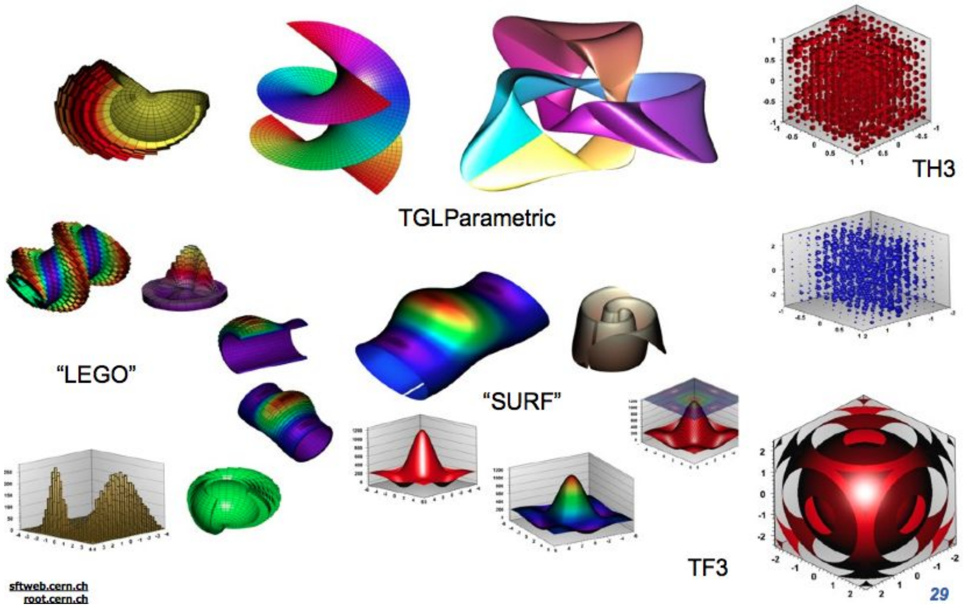


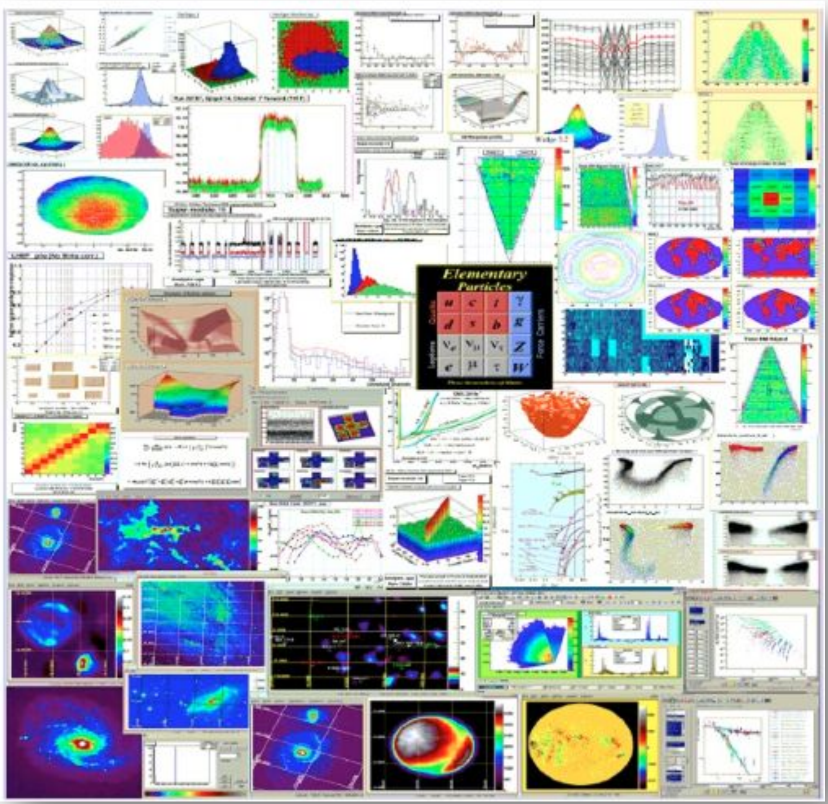
Graphics in ROOT

- Many formats for data analysis, and not only, plots



2D and 3D Graphics



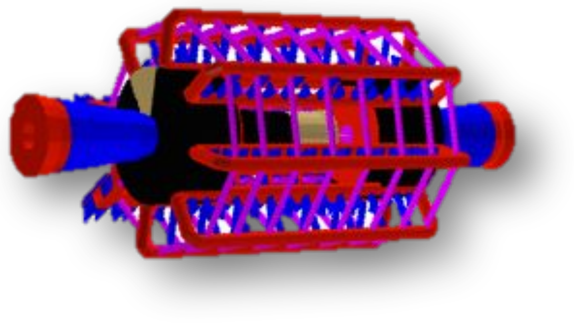


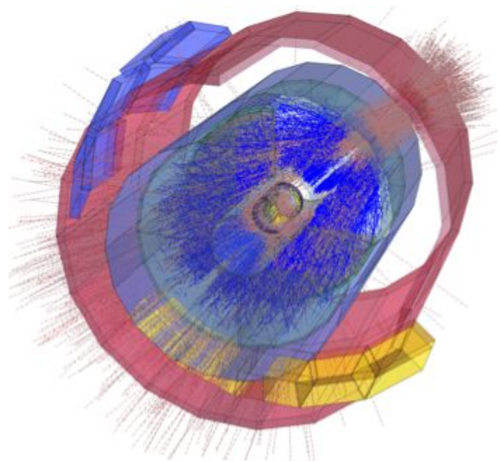
Can save graphics in many formats:
ps, pdf, svg, jpeg, LaTex, png, c, root …

Parallelism

- Many ongoing efforts to provide means for parallelisation in ROOT
- Explicit parallelism
  - **TThreadExecutor** and **TProcessExecutor**
  - Protection of resources
- Implicit parallelism
  - **TDataFrame**: functional chains
  - TTreeProcessor: process tree events in parallel
  - TTree::GetEntry: process of tree branches in parallel
- Parallelism is a fundamental element for tackling data analysis during LHC Run III and HL-LHC

Many More Features!





- Geometry Toolkit
  - Represent geometries as complex as LHC detectors
- Event Display (EVE)
  - Visualise particle collisions within detectors

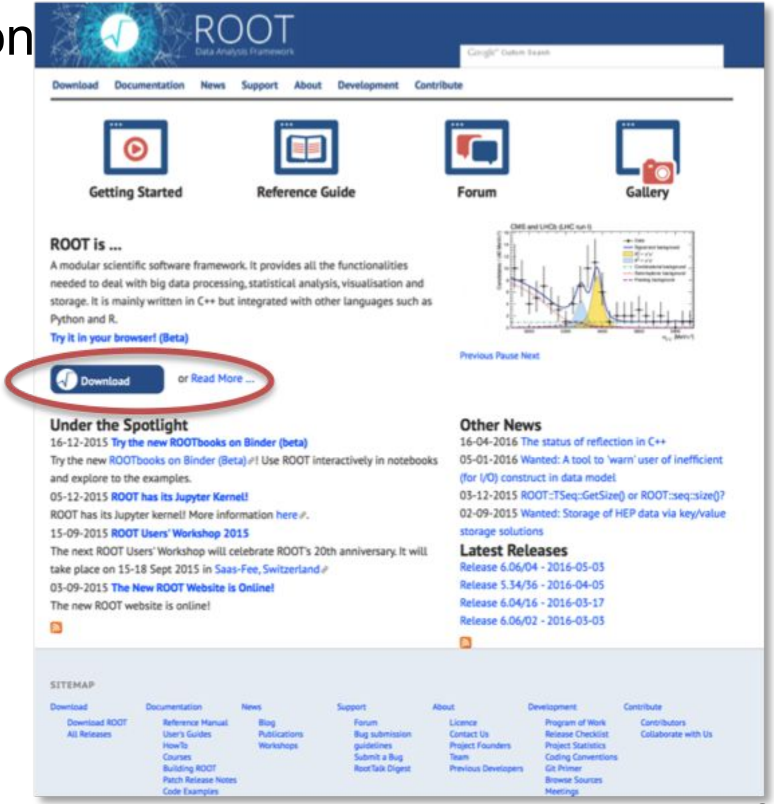
The SWAN Service

- **Data analysis with ROOT “as a service”**
- *Interface:* Jupyter Notebooks
- *Goals:*
  - Use ROOT only with a web browser
    - Platform independent ROOT-based data analysis
    - Calculations, input and results “in the Cloud”
  - Allow easy sharing of scientific results: plots, data, code
    - Through your CERNBox
  - Simplify teaching of data processing and programming

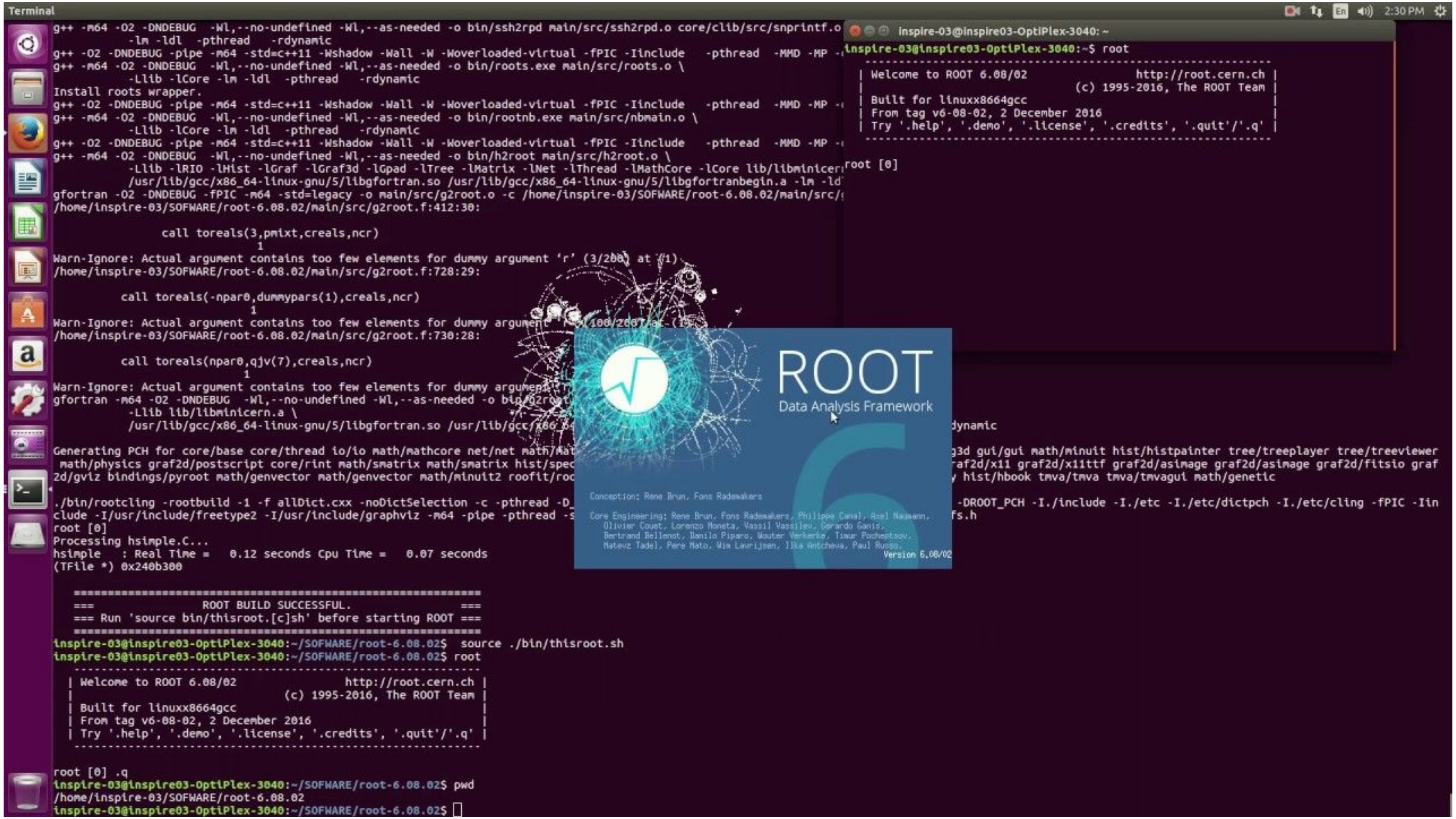
http://swan.web.cern.ch

https://root.cern

- ROOT web site: **the** source of information and help for ROOT users
  - For beginners and experts
  - Downloads, installation instructions
  - Documentation of all ROOT classes
  - Manuals, tutorials, presentations
  - Forum
  - ...



ROOT Basics



ROOT
Data Analysis Framework
ROOT BUILD SUCCESSFUL.
Run 'source bin/thisroot.[c]sh' before starting ROOT
Welcome to ROOT 6.08/02
http://root.cern.ch
(c) 1995-2016, The ROOT Team
Built for linuxx8664gcc
From tag v6-08-02, 2 December 2016
Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'

ROOT Basics: The ROOT Prompt

- C++ is a compiled language
  - A compiler is used to translate source code into machine instructions

- ROOT provides a C++ **interpreter**
  - Interactive C++, without the need of a compiler, like Python, Ruby, Haskell …
    - Code is **Just-in-Time compiled!**
  - Allows reflection (inspect at runtime layout of classes)
  - Is started with the command:

```
root
```

  - **The interactive shell is also called “ROOT prompt” or “ROOT interactive prompt”**

ROOT As a Calculator

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \ldots$$
$$= \sum_{n=0}^{\infty} x^n$$

```
root [0] double x=.5
(double) 0.5
root [1] int N=30
(int) 30
root [2] double gs=0;
```

Here we make a step forward.
We declare **variables** and use a ***for*** control structure.

```
root [3] for (int i=0;i<N;++i) gs += pow(x,i)
root [4] std::abs(gs - (1/(1-x)))
(Double_t) 1.86265e-09
```

Controlling ROOT

- Special commands which are not C++ can be typed at the prompt, they start with a "."

```
root [1] .<command>
```

- For example:
  - To quit root use **.q**
  - To issue a shell command use **.! <OS_command>**
  - To load a macro use **.L <file_name>** (see following slides about macros)
  - **.help** or **.?** gives the full list

Ex Tempore Exercise

- Fire up ROOT
- Verify it works as a calculator
- Inspect the help
- Quit

Exercise

- ROOT provides mathematical functions, for example the widely known and adopted Gaussian
- For x values of 0,1,10 and 20 check the difference of the value of a hand-made non-normalised Gaussian and the TMath::Gaus routine

For one input value

```
root [0] double x=0
root [1] exp(-x*x*.5) - TMath::Gaus(x)
[...]
```

Solution

- For x values of 0,1,10 and 20 check the difference of the value of a hand-made non-normalised Gaussian and the TMath::Gaus routine

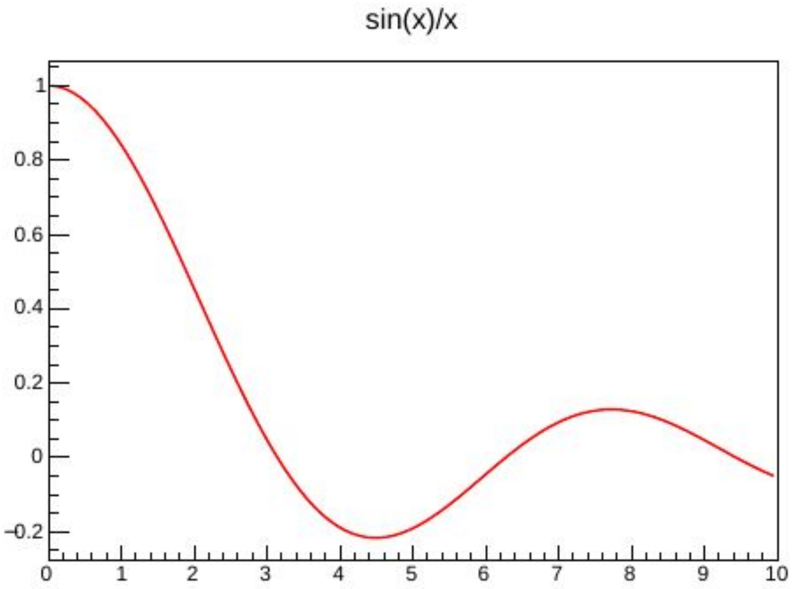
```
root [0] double x=0
root [1] exp(-x*x*.5) – TMath::Gaus(x)
[…]
```

- Many possible ways of solving this! E.g:

```
root [0] for (auto v : {0.,1.,10.,20.}) cout << v << “ ” <<
exp(-v*v*.5) – TMath::Gaus(v) << endl
```

my first function

```
root [0] auto fa1 = new TF1("fa1","sin(x)/x",0,10);
root [1] fa1->Draw();
[...]
```



Please, take a look at the "toolbar"

1) save as png,pdf,root,.....
2) change the line color, style
3) change the tf1 range
4) change the tf1 function (cos(x)+x)?
4) change the x-label name
6) continue to explore the options

my Error function

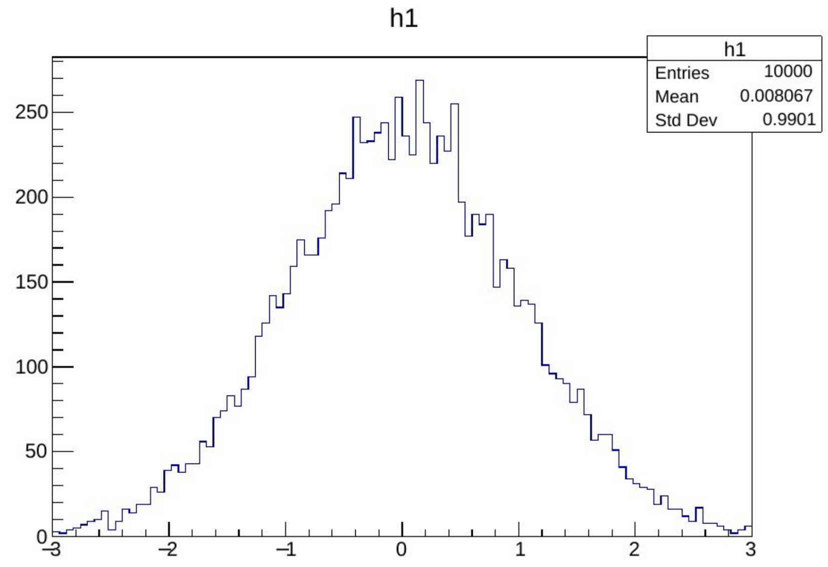
```
root [0] auto fa2 = new TF1("fa2","TMath::Erf(x)",0,2);
root [1] fa2->Draw();
[...]
```

[Trigger Effy](), https://en.wikipedia.org/wiki/Error_function,

```
root [13] auto fa2 = new TF1("fa2","TMath::Erf(x)",0,2);
root [14] fa2->Draw();
auto fa2 = new TF1("fa2","TMath::Erf(-x)",0,2);
root [15] fa2->Draw();
root [16] auto fa2 = new TF1("fa2","(TMath::Erf((-x+1.0)))",0,2);
root [17] fa2->Draw();
root [18] auto fa2 = new
TF1("fa2","(TMath::Erf((-x+1.0)/0.035))",0,2);
root [19] fa2->Draw();
```

my first histogram

```
root [0] TH1F h1("h1","h1",100,-3,3);
root [1] h1.FillRandom("gaus",10000);
root [2] h1->Draw();
ROOT_prompt_2:1:3: error: member reference type 'TH1F' is not a
pointer; did you mean to use '.'?
h1->Draw();
~~^~
  .
root [3] h1.Draw();
```



Please, take a look at the "toolbar", again.

User function

[a]*exp(-0.5*((x-[b])/[c])*((x-[b])/[c]))

ROOT Macros

- We have seen how to interactively type lines at the prompt
- The next step is to write “ROOT Macros” – lightweight programs
- The general structure for a macro stored in file *MacroName.C* is:

Function, no main, same name as the file

```
void MacroName() {
          <              ...
            your lines of C++ code
                         ...              >
}
```

Unnamed ROOT Macros

- Macros can also be defined with no name
- Cannot be called as functions!
    - See next slide :)

```
{
        <           ...
          your lines of C++ code
                    ...            >
}
```

Running a Macro

- A macro is executed at the system prompt by typing:

```
> root MacroName.C
```

- or executed at the ROOT prompt using .x:

```
> root
root [0] .x MacroName.C
```

- or it can be loaded into a ROOT session and then be run by typing:

```
root [0] .L MacroName.C
root [1] MacroName();
```

Time for Exercises

- Go back to the geometric series example we executed at the prompt
- Make a macro out of it and run it

Examples:
geometrica_macro.C
geometrica_macroName.C
¿Como volver esto un codigo puramente cpp?
¿geometrica_macroMain.C?

Examples:
myGaussfun.C
¿Como volver esto un codigo puramente cpp?
¿myGaussfunAlaC.C?

Interpretation and Compilation

- We have seen how ROOT interprets and “just in time compiles” code. ROOT also allows to compile code “traditionally”. At the ROOT prompt:

```
root [1] .L macro1.C+
root [2] macro1()
```

Generate shared library and execute function

- ROOT libraries can also be used to produce standalone, compiled applications:

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

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

Un paréntesis útil: Argumentos de línea de comandos

Por suerte, la interfaz para **transmitir argumentos a una función main() se ha estandarizado en C++**, así que la emisión y recepción de argumentos puede hacerse de manera casi mecánica.

Los argumentos transmitidos a main(), como todos los argumentos de función, **deben declararse como parte de la definición de la función**.

```
int main(int argc, char *argv[])
```

Sin importar cuántos argumentos se mecanografíen en la línea de comandos, main() sólo necesita las dos piezas de información estándares proporcionadas por **argc y argv**; el número de elementos en la línea de comandos y la lista de direcciones iniciales que indican dónde se almacena en la actualidad cada argumento.
argc (abreviatura para contador de argumento)
argv (abreviatura para valores de argumentos)

Example: Parentesis_ARG

Cualquier argumento mecanografiado en una línea de comandos se considera una cadena en C. Si se desea transmitir datos numéricos a main(), depende de usted convertir la cadena transmitida en su contraparte numérica

Macro myGaussfun

- Make a macro out of it and run it

Examples:
myGaussfun.C
¿Como volver esto un codigo puramente cpp?
¿myGaussfunAlaC_2.C?
g++ -o miname myGaussfunAlaC_2.C `root-config --cflags --libs`
./miname

My first Canvas

TCanvas

```
root[0] new TCanvas          ← "quick" creation of graphical window with generic properties
(class TCanvas*)0x2c1e5e0                 ← Notice: automatic name assignment ("c1")
root[1] c1->Set

root[1] c1->SetTitle("HelloCanvas")
root[2] c1->GetTitle()
(const char* 0x1557339)"HelloCanvas"
root[3] c1->ls ()
root[4] c1->Close()
root[5] TCanvas c2            ← We create a new window. Before we used pointer. Now – object.
root[6] c2.GetName()
(const char* 0x16632b1)"c1_n2"     ← Title is different than variable name!
root[7] TCanvas c3 (
                                                 Multitude of constructors
TCanvas TCanvas(Bool_t build = kTRUE)
TCanvas TCanvas(const char* name, const char* title = "", Int_t form = 1)
TCanvas TCanvas(const char* name, const char* title, Int_t ww, Int_t wh)
TCanvas TCanvas(const char* name, const char* title, Int_t wtopx, Int_t wtopy,
Int_t ww, Int_t wh)
TCanvas TCanvas(const char* name, Int_t ww, Int_t wh, Int_t winid)
root[7] TCanvas c3 ("c3canvas", "My canvas", 600, 400);
```

Proper name of object (within C++).

"Name" (identifier) (within ROOT).

Displayed title (just a c-string)

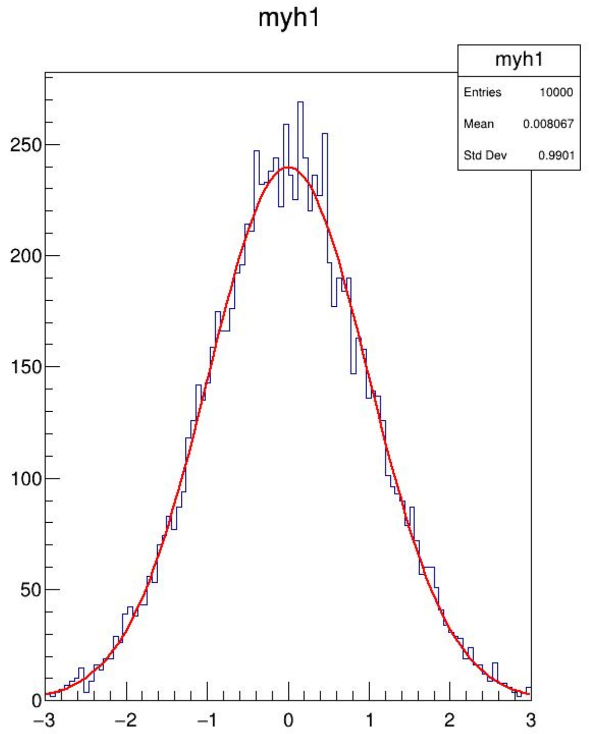
```
root[8] c3Canvas          ← Name as identifier or replacement of C++ name
(class TCanvas*)0x16964d0
root[9] TCanvas* c4 = new TCanvas ("c4canv", "2nd Canvas", 600, 400);
```

Dynamic allocation (we then use a pointer to an object)

My first Canvas

https://root.cern.ch/doc/master/classTCanvas.html

```
auto myc1 = new
TCanvas("myc1","myc1",600,800);
myc1->cd();
myc1->SetLeftMargin(0.15);
myc1->SetRightMargin(0.06);
myc1->SetTopMargin(0.09);
myc1->SetBottomMargin(0.14);
```



Please, take a look to this example:
https://root.cern/doc/master/canvas_8C.html

Example: myfirtsCanvas.C