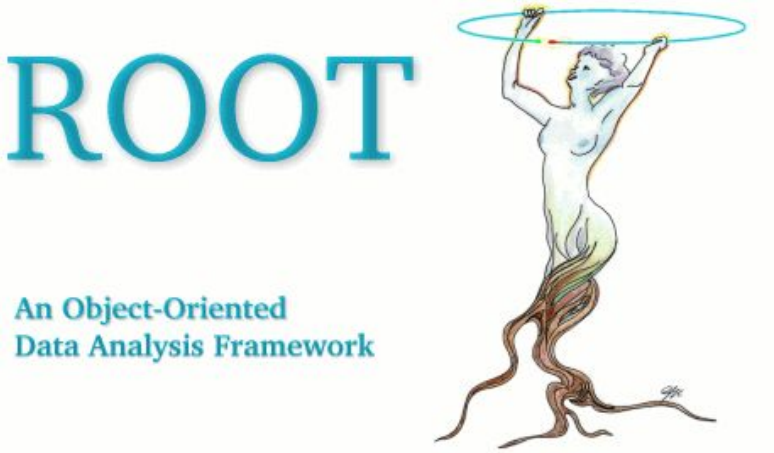
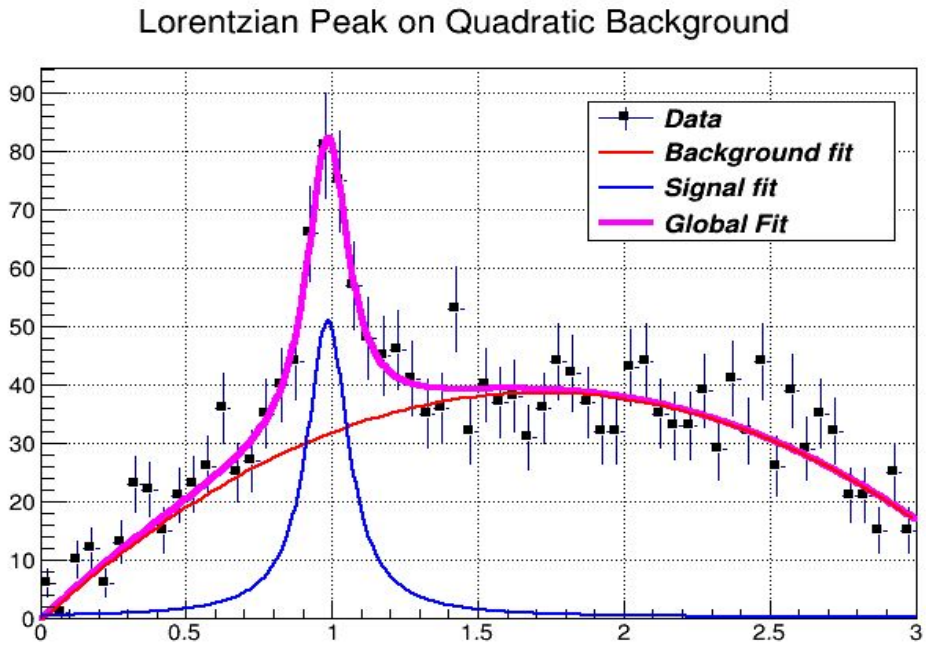
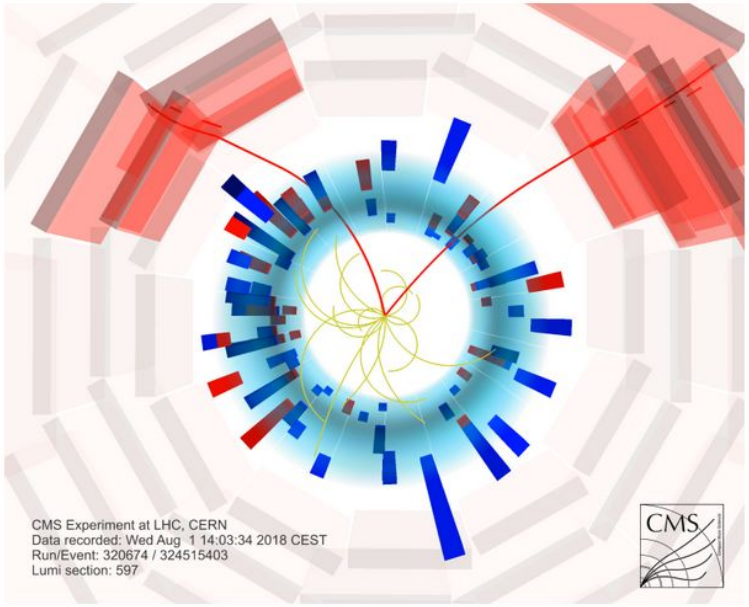



# Big data With ROOT CERN

**Jhovanny Andres Mejia Guisao**

Centro de Investigación y de Estudios Avanzados del IPN (Cinvestav)

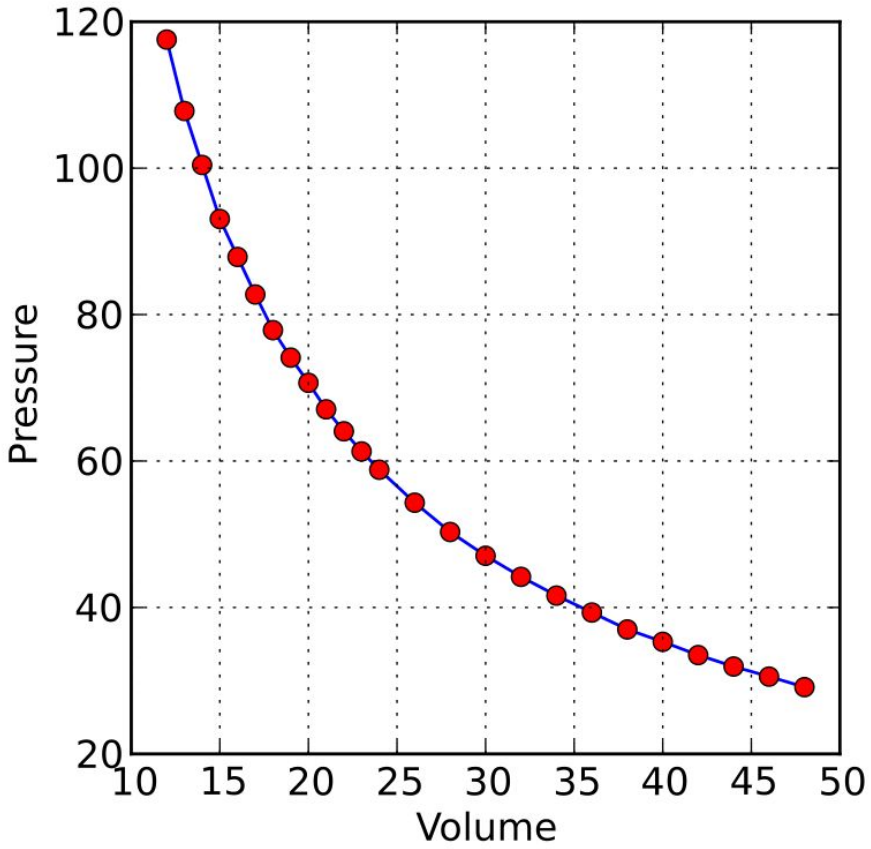
# Motivation

## 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

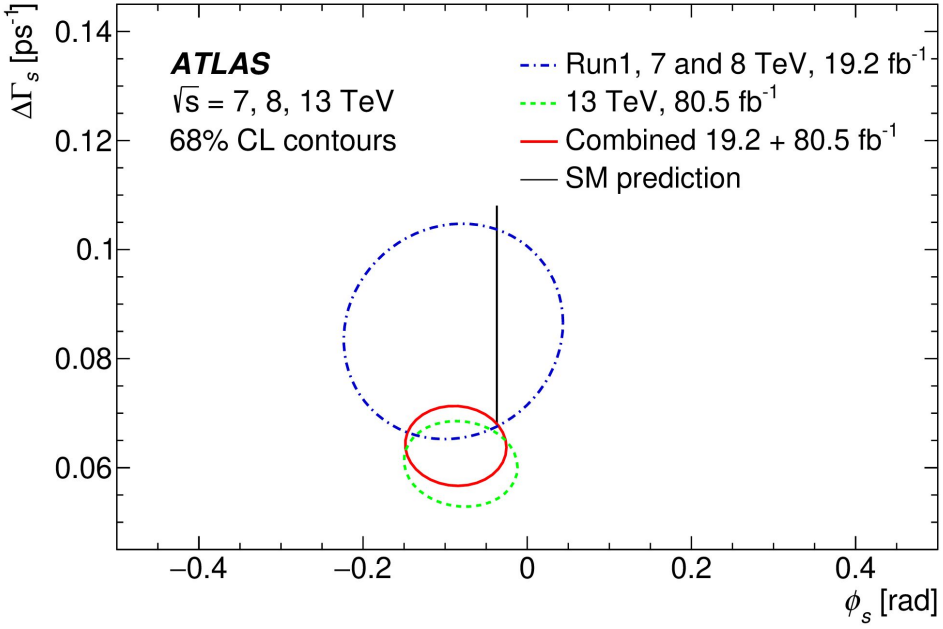


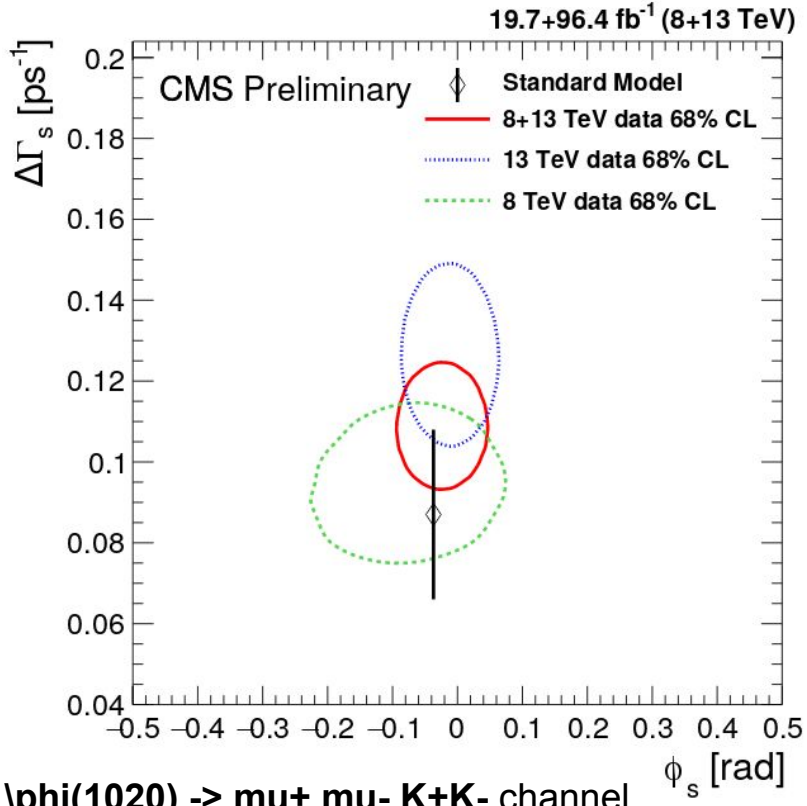

Para un gas a temperatura constante, el volumen es inversamente proporcional a la presión sobre éste[link]

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.

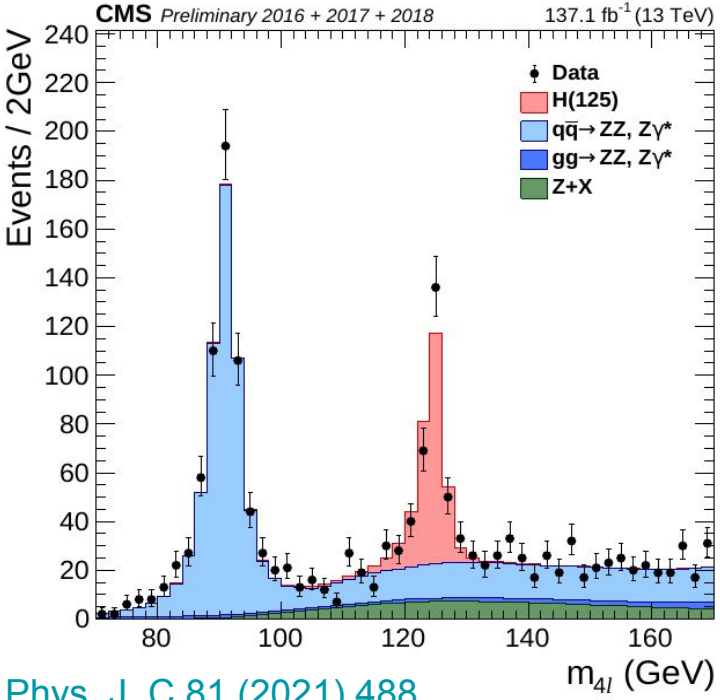
# Motivation

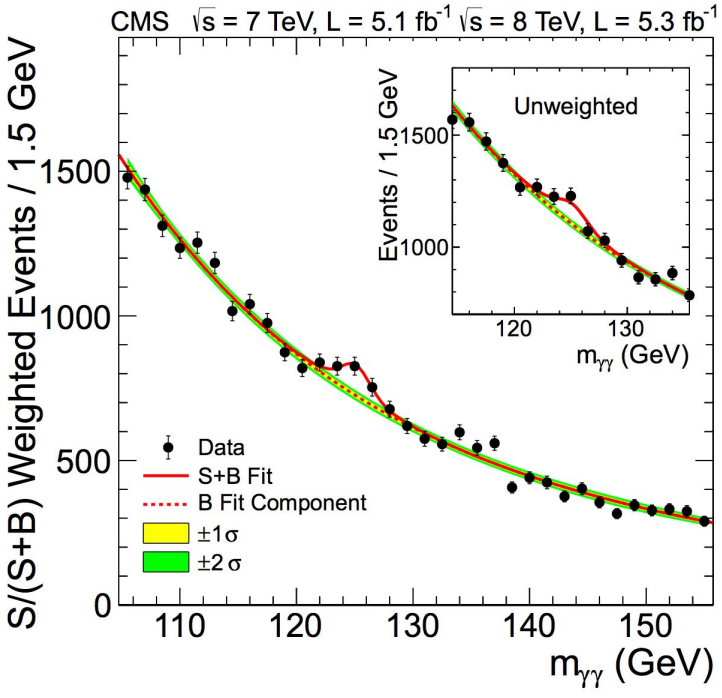



Measurement of the **CP violating phase** in the **Bs-> J/\psi \phi(1020) -> mu+ mu- K+K-** channel in proton-proton collisions at √s = 13 TeV (PLB 816 (2021) 136188)

# Motivation



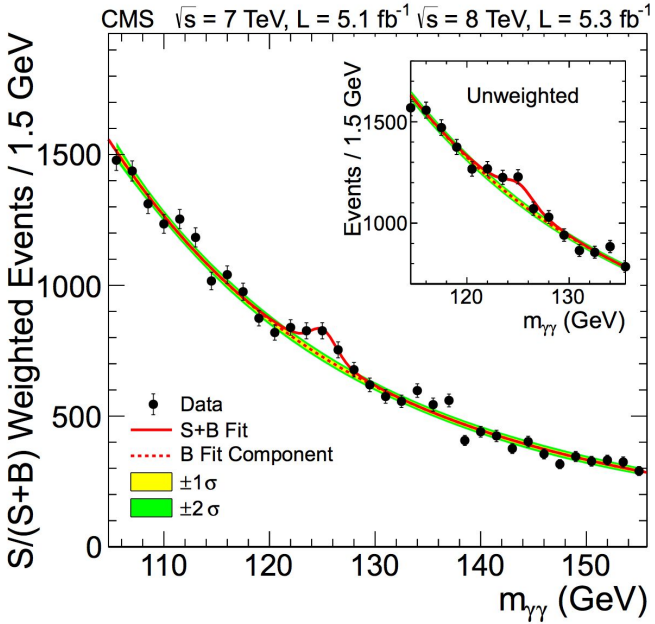




Eur. Phys. J. C 81 (2021) 488

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

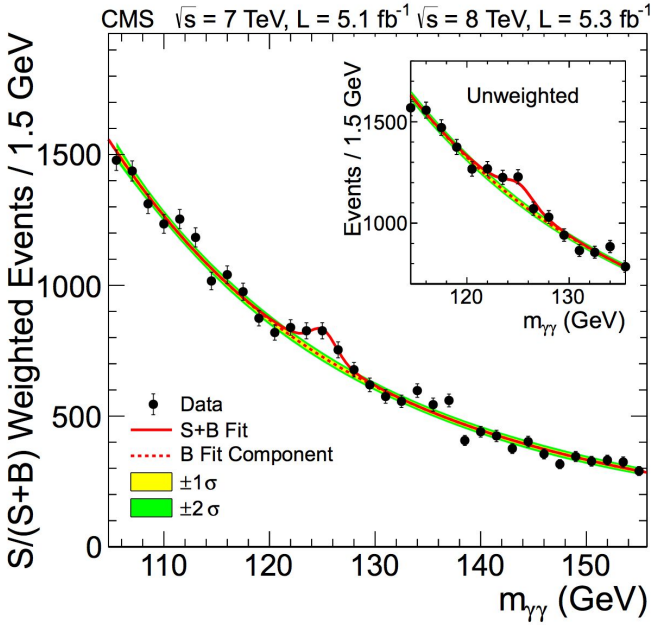
**Visualization of the data**

**Corrections or parameter transformations?**

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

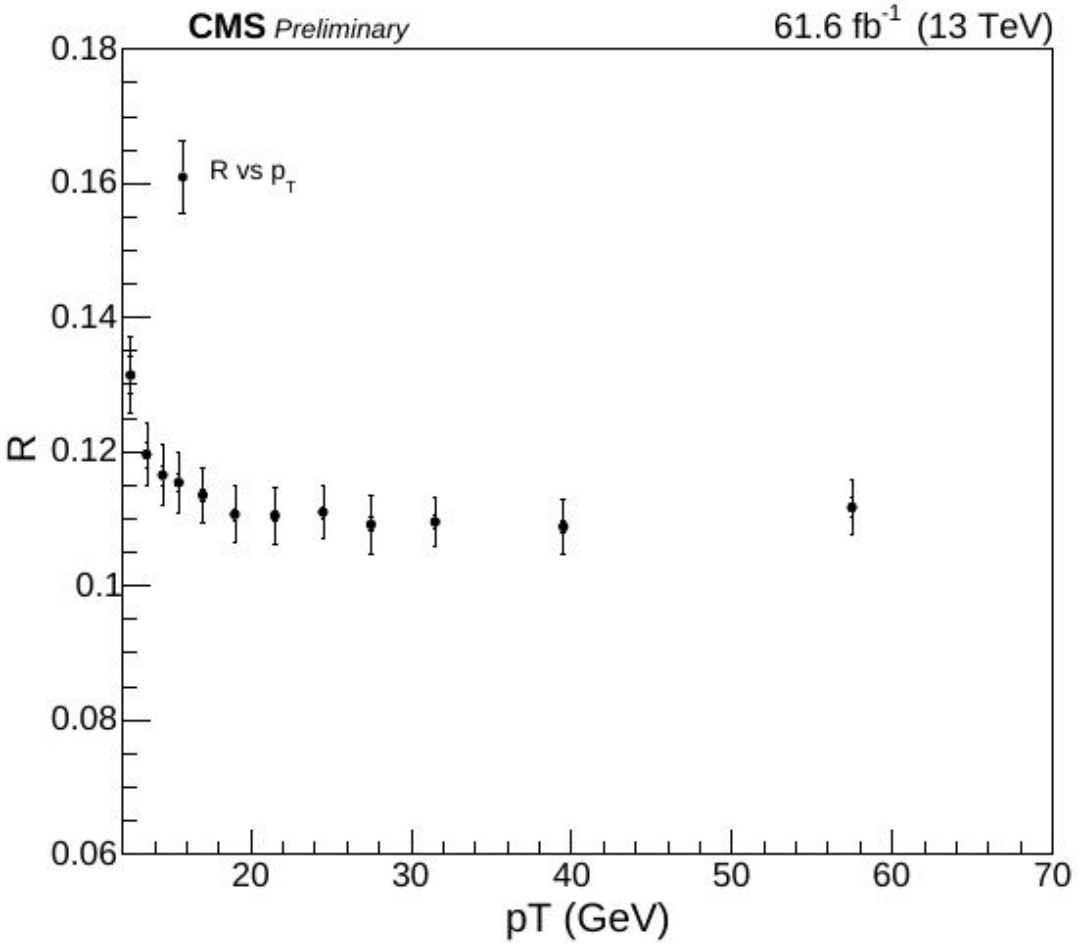
**The statistical nature of the errors must be handled properly.**

**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?**

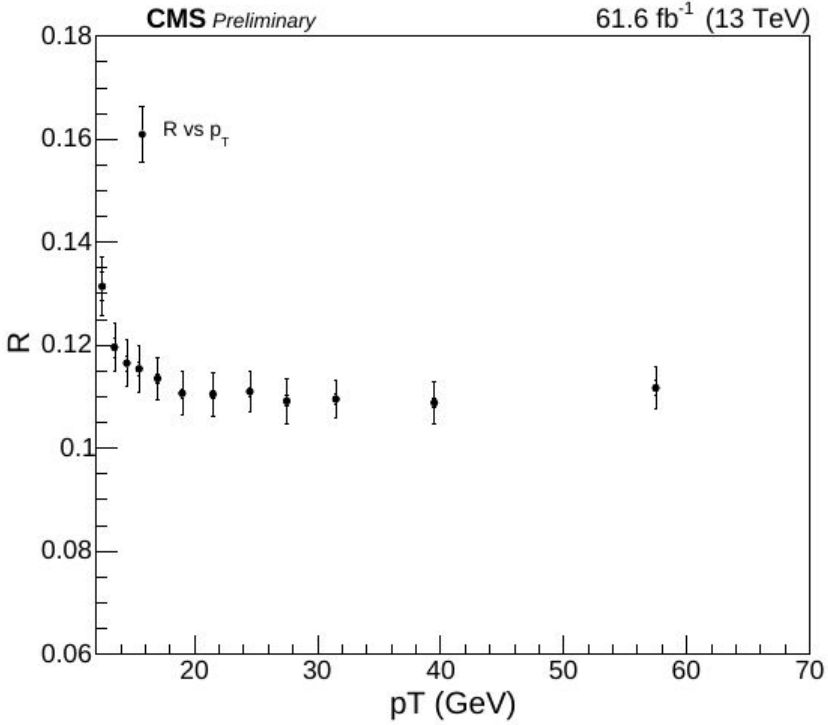
BPH-21-001

# Motivation

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

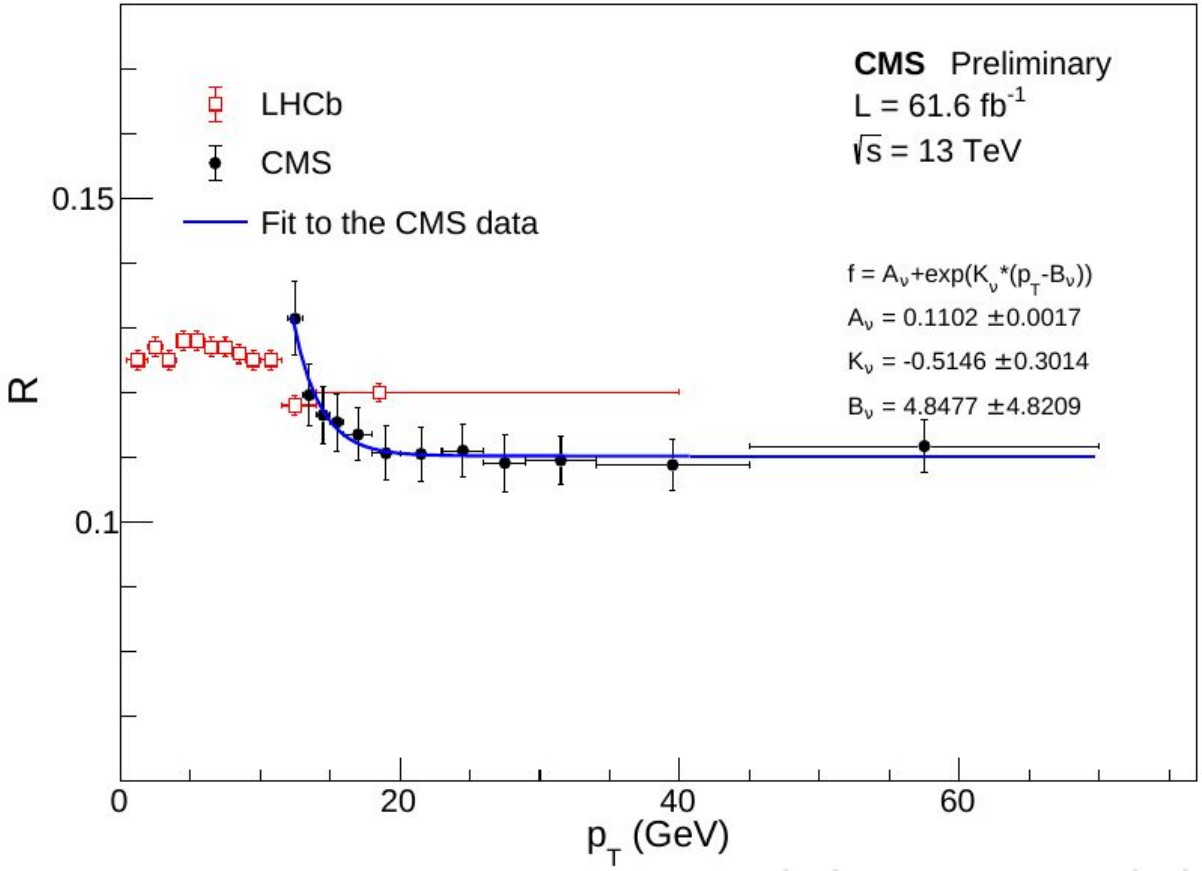
| $p_T$ (GeV) | $\mathcal{R}$ | $\sigma_{tot}$ | $\sigma_{stat}$ | $\sigma_{sys}^{uncor}$ | $\sigma_{sys}^{cor}$ |
|---|---|---|---|---|---|
| 12 – 13 | 0.1314 | 0.0058 | 0.0028 | 0.0033 | 0.0038 |
| 13 – 14 | 0.1196 | 0.0046 | 0.0019 | 0.0015 | 0.0039 |
| 14 – 15 | 0.1165 | 0.0041 | 0.0015 | 0.0016 | 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 |

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



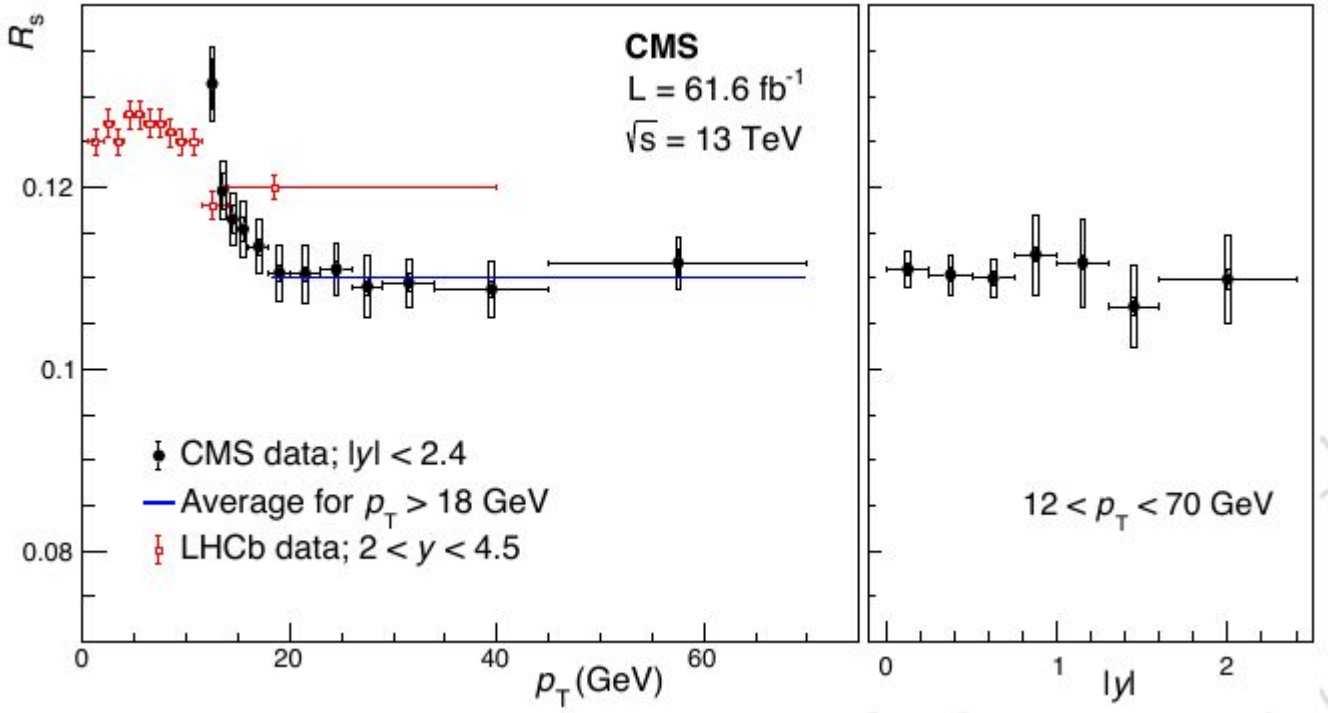

BPH-21-001

# 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, 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.**

# ROOT in a Nutshell

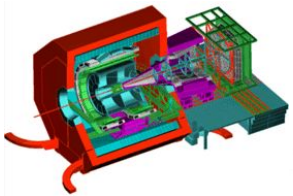
- ROOT is a software framework with building blocks for:
  - Data processing
  - Data analysis
  - Data visualisation
  - Data storage
- ROOT is written mainly in C++ (C++11/17 standard)
  - Bindings for Python available as well
- Adopted in High Energy Physics and other sciences (but also industry)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

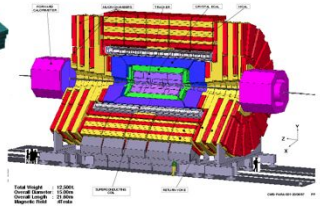
# ROOT in a Nutshell

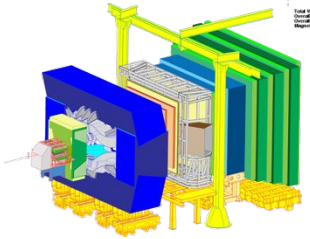
- ROOT can be seen as a collection of building blocks for various activities, like:
  - **Data analysis: histograms, graphs, functions**
  - **I/O: row-wise, column-wise** storage of any C++ object
  - **Statistical tools** (RooFit/RooStats): rich modeling and statistical inference
  - Math: **non trivial 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**

Data

**Analysis**

**Offline Processing**

Event Selection, statistical treatment …

Further processing, skimming

Reconstruction

Raw

Reco

…

Analysis Formats

Images

**Data Storage: Local, Network**

# 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

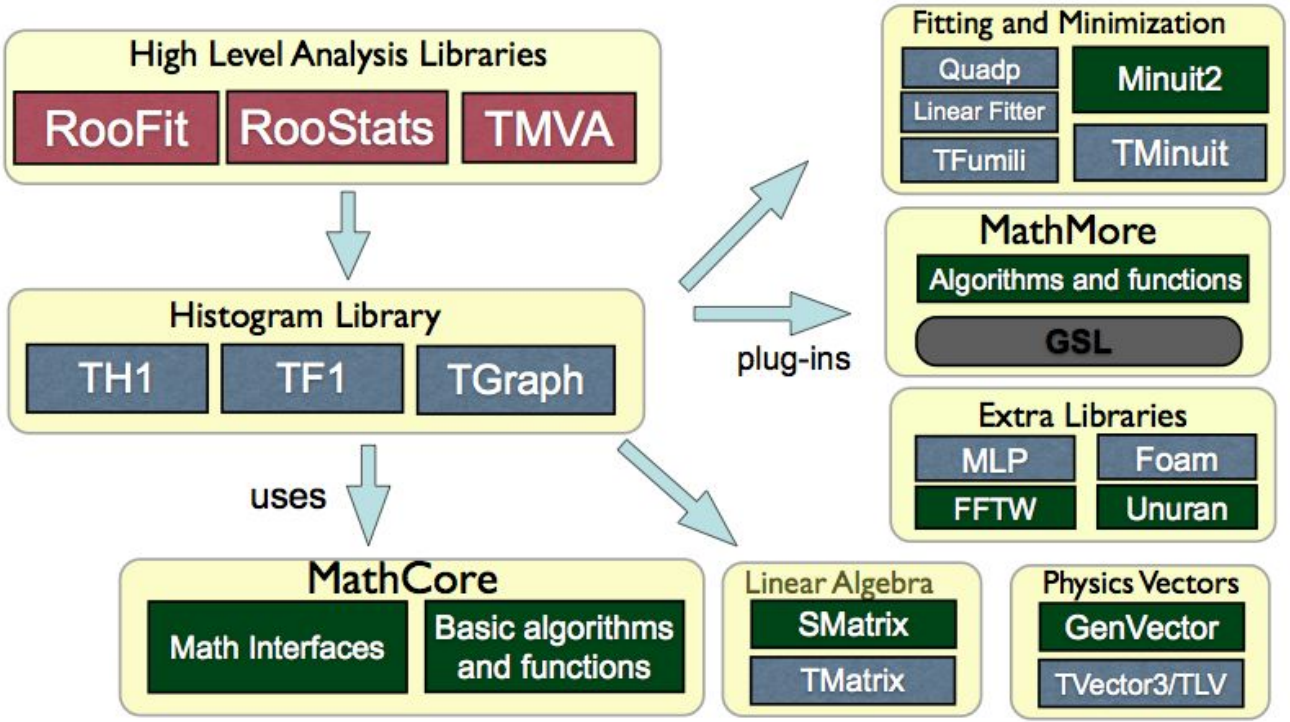
# LHC Data in ROOT Format

**1 EB**

**as of 2017**

# Mathematics and Statistics

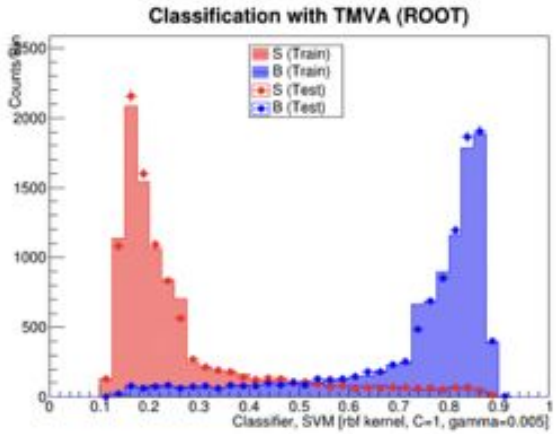
- ROOT provides a rich set of mathematical libraries and tools for sophisticated statistical data analysis



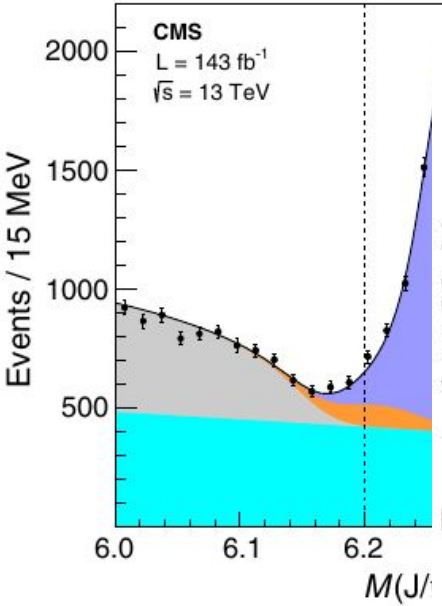
# Machine Learning: TMVA

## **TMVA** : Toolkit for Multi-Variate data Analysis in ROOT

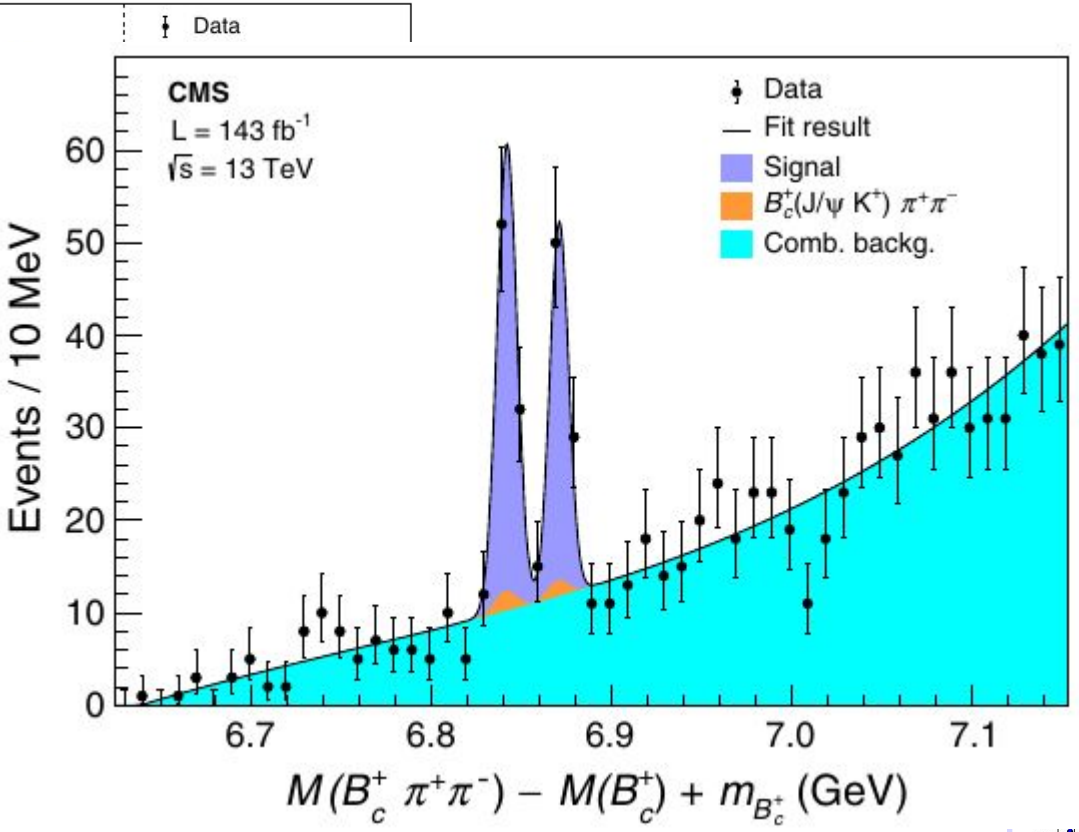
- provides several built-in ML methods including:
  - Boosted Decision Trees
  - Deep Neural Networks
  - Support Vector Machines
- and interfaces to external ML tools
  - scikit-learn, Keras (Theano/Tensorflow), R



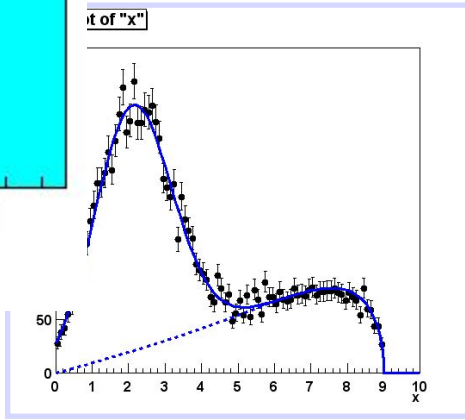
# RooFit




- visualizatic
- sharing an



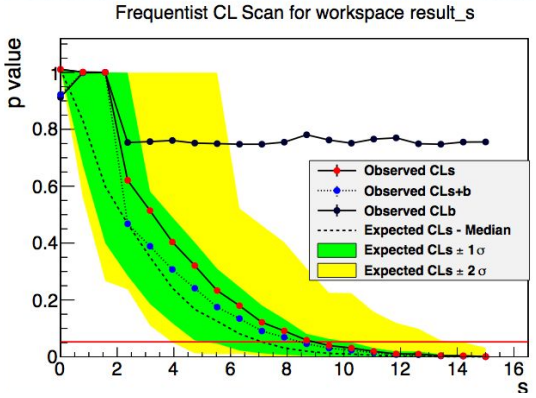

d.f.)

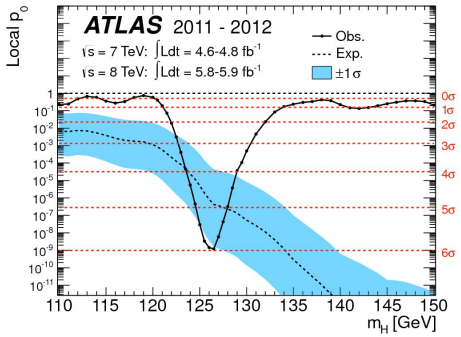



BPH-18-007, Phys. Rev. Lett. 122, 132001

# RooStats

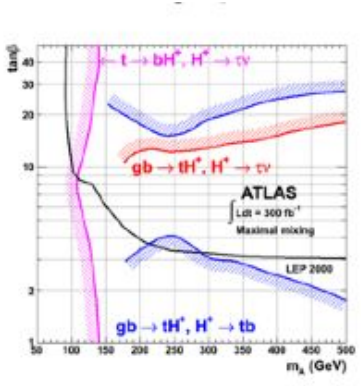
- Advanced Statistical Tools for HEP analysis. Used for :
  - estimation of Confidence/Credible intervals
  - hypotheses Tests
    - e.g. Estimation of Discovery significance
- Provides both Frequentist and Bayesian tools
- Facilitate combination of results

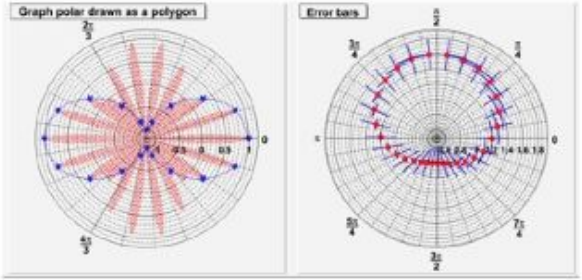


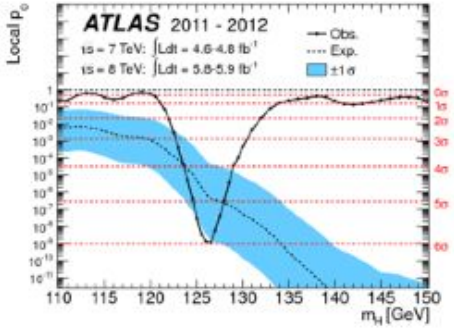



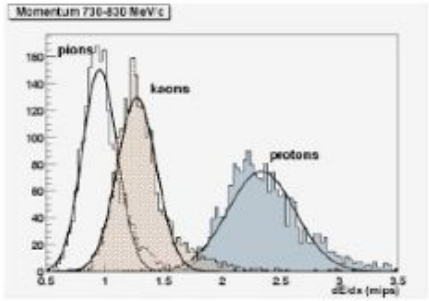
# Graphics in ROOT

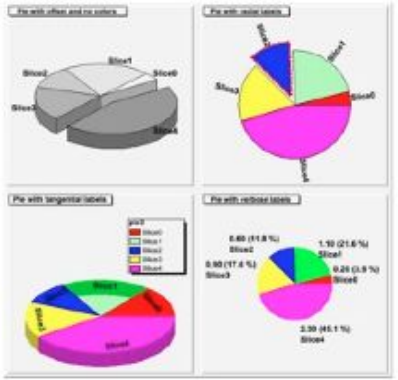
- Many formats for data analysis, and not only, plots

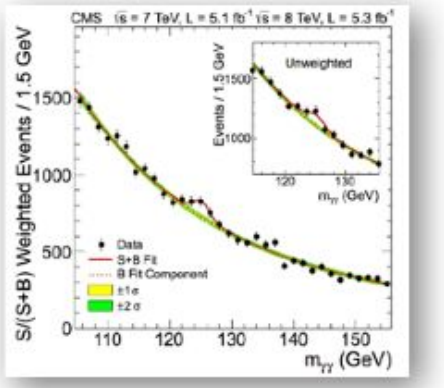


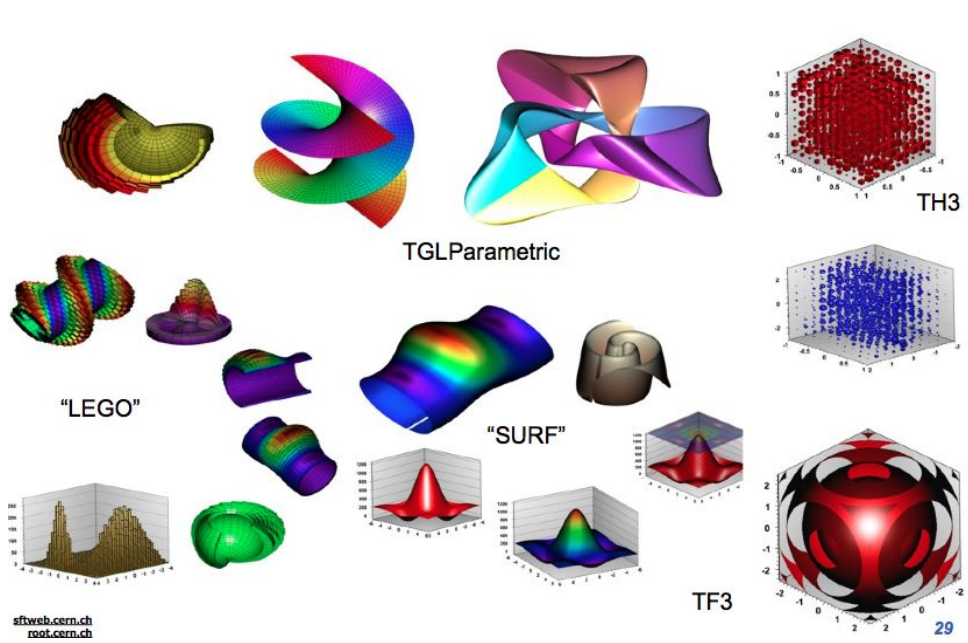



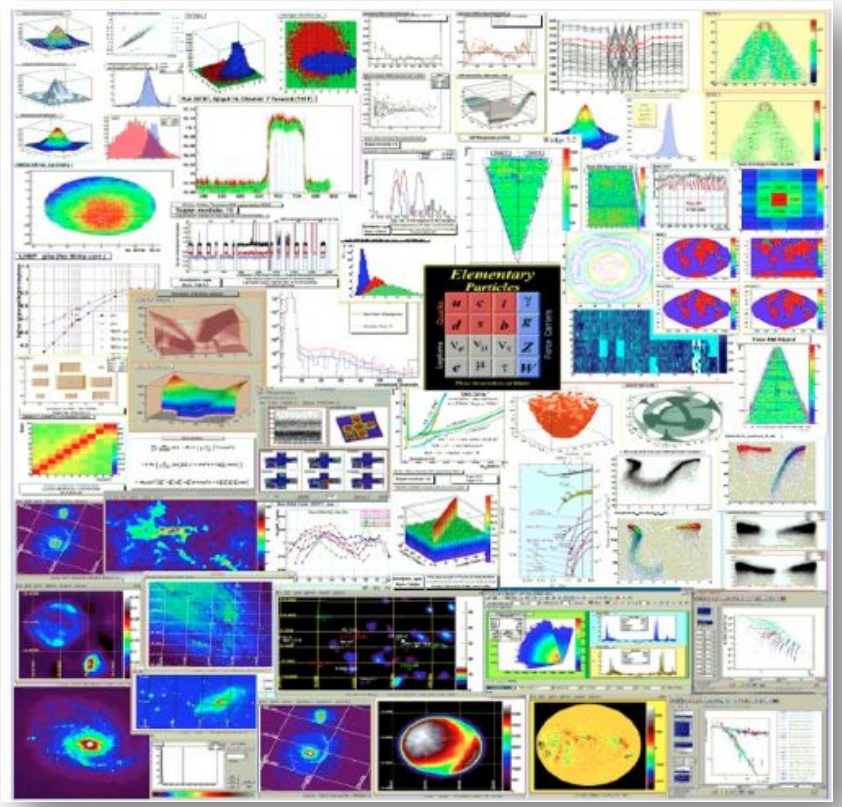





# 2D and 3D Graphics



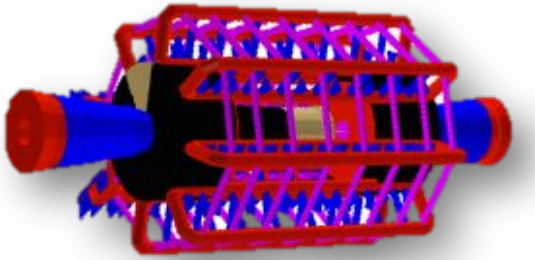



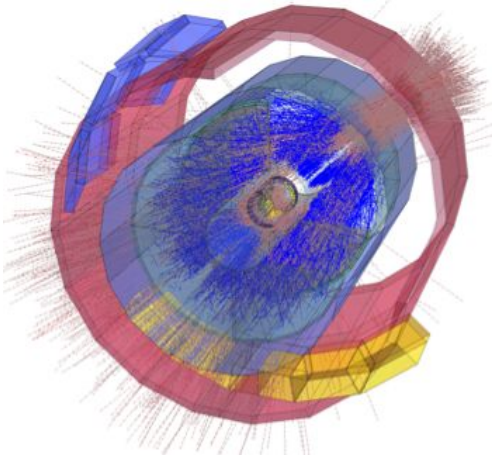

Can save graphics in many formats:
*ps, pdf, svg, jpeg, LaTex, png, c, root …*

# 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

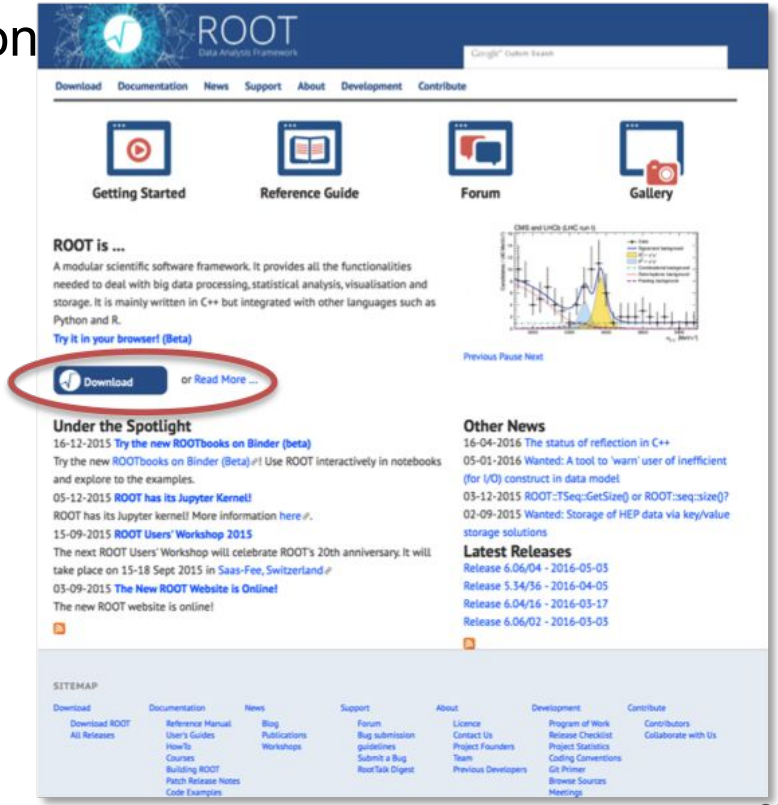
- **Data analysis with ROOT “as a service”**
- *Interface:* Jupyter Notebooks
- *Goals:*
  - Use ROOT only with a web browser
    - Platform independent ROOT-based data analysis
    - Calculations, input and results “in the Cloud”
  - Allow easy sharing of scientific results: plots, data, code
    -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



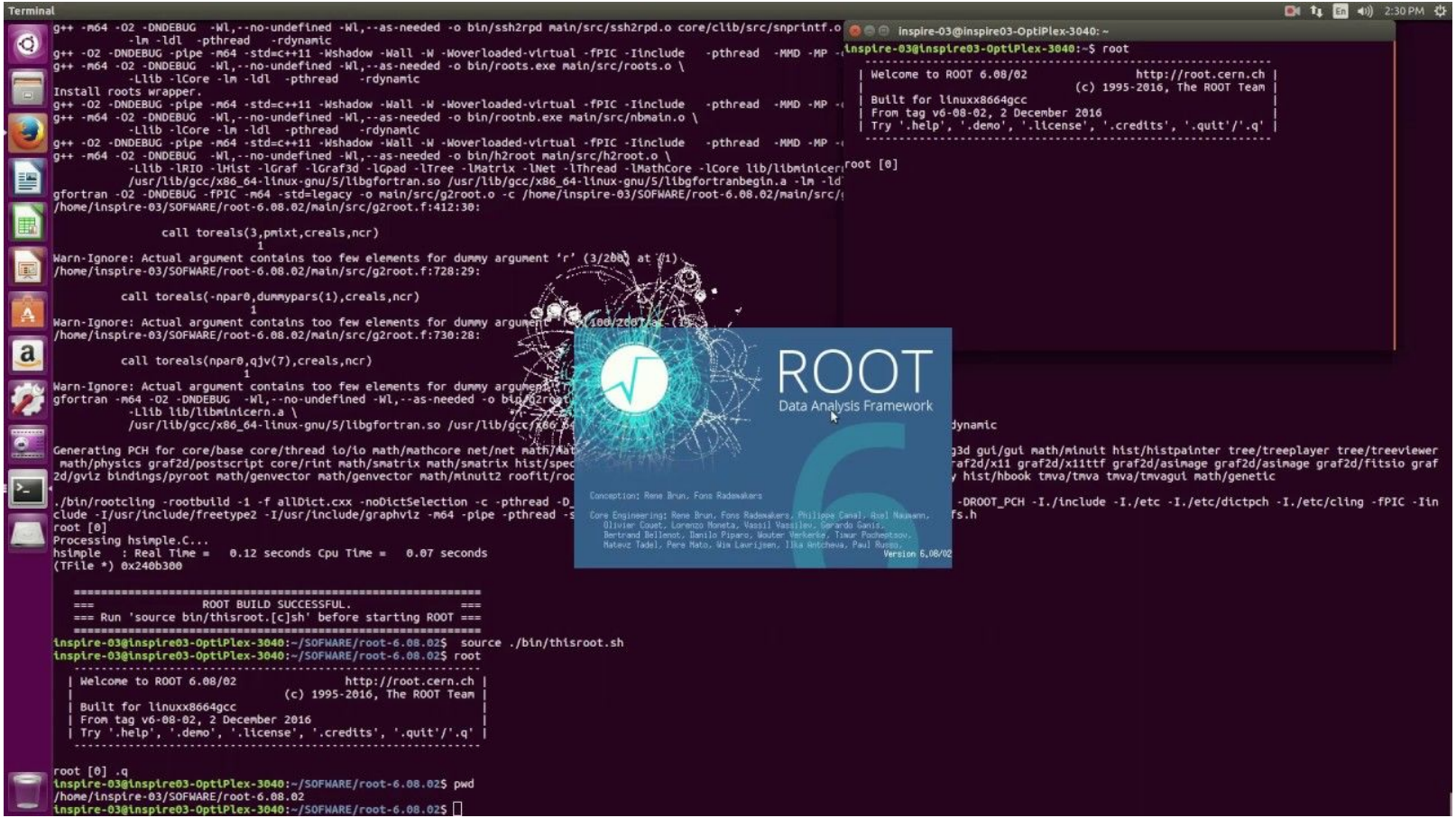

http://swan.web.cern.ch

# 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


Terminal
inspire-03@inspire03-OptiPlex-3040: ~
ROOT
Data Analysis Framework
Conception: Rene Brun, Fons Rademakers
Version 6.08/02

# Programación, diseño y complejidad.

- **¿cual es el objetivo del software, resolver un problema particular?**
  cálculo de problemas numéricos, mantenimiento de una base de datos organizada de información, búsqueda de nuevas partículas, análisis de imágenes…

- **En las últimas décadas, el crecimiento del poder computacional nos ha permitido abordar problemas cada vez más complejos**

- **Como consecuencia, el software también se ha vuelto más poderoso y complejo.**
  Física no es la excepción:  La colección de paquetes de software para la reconstrucción / análisis del experimento **BaBar es ~ 6.4M líneas de C++**.
  "**CMSSW , athena are written in C++ and Python, and has several million lines of code**".

- **¿Cómo lidiamos con una complejidad tan creciente?**

# Filosofías de programación

- La **clave** para codificar con éxito sistemas complejos es descomponer el código en **módulos más pequeños** y **minimizar las dependencias** entre estos módulos.
- **Los lenguajes de programación tradicionales (Fortran, Pascal, C) logran esto mediante los procedimientos orientados.**

  -La modularidad y la estructura del software giran en torno a algoritmos encapsulados en "**funciones**"

  -Aunque las funciones son una herramienta importante en la estructuración de software, dejan algunos dolores de cabeza de diseño importantes

- **Los lenguajes orientados a objetos (C++, Java, python ...) llevan estos pasos más allá**

  Agrupando datos y funciones asociadas en **objetos.**

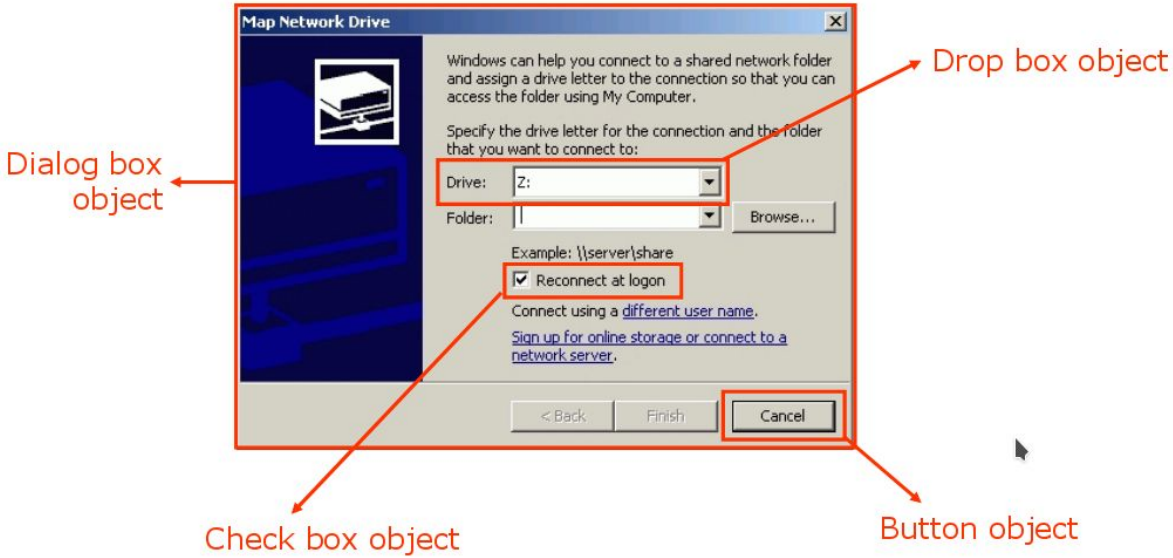


Identificar objetos concretos es relativamente sencillo, sin embargo identificar objetos abstractos requiere un poco de práctica e intuición como por ejemplo, transacción de una cuenta bancaria, detalle de una factura, que son objetos no tan evidentes, pero que son los que en la mayoría de casos serán los objetos que deberán ser implementados en programas, ya que interactuarán con los objetos más evidentes como la factura, o la cuenta bancaria en el caso de la transacción.

# ¿Qué son los objetos?

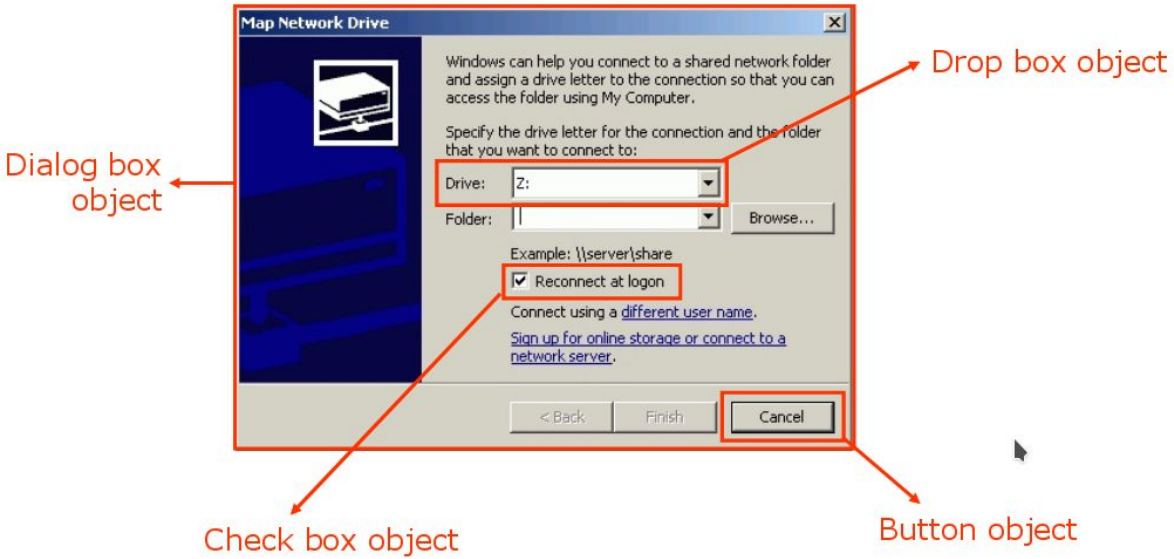
- **‘Software objects’** a menudo se encuentran de forma natural en problemas de la vida cotidiana.

- **Programación orientada a objetos (POO)** → Encontrar estos objetos y cual es su papel en su problema.



# ¿Qué son los objetos?

- Un objeto tiene:
  **Propiedades**: posición, forma, etiqueta de texto
  **Comportamiento**: si hace clic en el ‘Cancel button’, se produce una acción definida




**El análisis y diseño orientado a objetos busca la relación entre objetos**

- ➔ "Es-Un" Relacion (un objeto "PushButton" es un objeto Clickiable).
- ➔ ‘Tiene-Un’ Relacion (a DialogBox Tiene Un CheckBox)

# Beneficios de la programación orientada a objetos

- Beneficios de la programación orientada a objetos
  - **Reutilización del código existente** - los objetos pueden representar problemas genéricos.
  -**Mantenimiento mejorado** - los objetos son más autónomos que las "subrutinas", por lo que el código está menos enredado.
  -**A menudo, una forma "natural" de describir un sistema** - podemos ver el ejemplo anterior del "Dialog box"

- Pero….
  -El modelado orientado a objetos no sustituye al pensamiento sólido.
  -La POO no garantiza un alto rendimiento, pero tampoco se interpone en su camino.

- **Sin embargo**
  -POO es actualmente la mejor forma en que sabemos describir sistemas complejos.

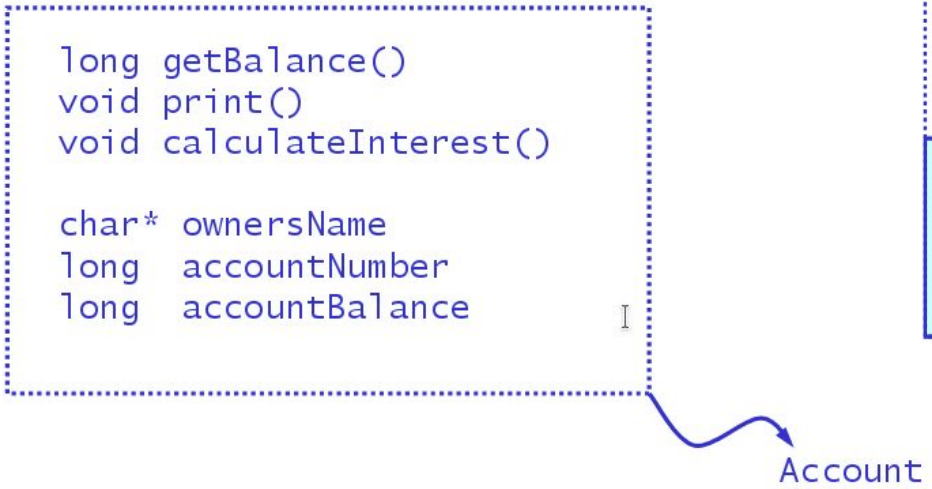
# Técnicas para lograr la abstracción

1. Modularidad 2. Encapsulación 3. Herencia 4. Polimorfismo.

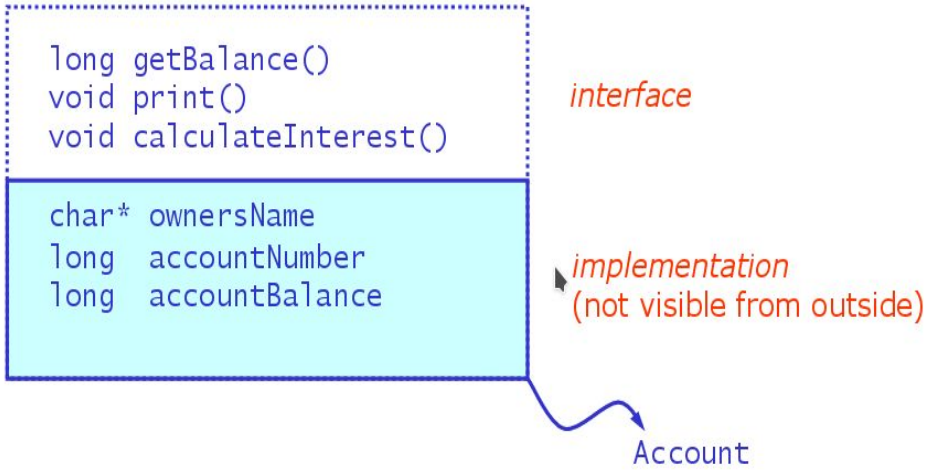
1.Descomponga su problema de forma logica en unidades independientes

–Minimizar dependencias entre unidades - Acoplamiento suelto

- Agrupar cosas que tengan una conexión lógica - Fuerte cohesión




2. Separar la interfaz y la implementación y proteja la implementación de los "usuarios" del objeto.



# Técnicas para lograr la abstracción

1. Modularidad 2. Encapsulación 3. Herencia 4. Polimorfismo.

## 3.Herencia

Es el pilar más fuerte que asegura la reutilización de código, ya que a partir de esta característica es posible reutilizar (heredar) las características y comportamientos de una clase superior llamada clase padre, a sus clases hijas, denominadas clases derivadas.


## 4. Polimorfismo

A través de esta característica es posible definir varios métodos o comportamientos de un objeto bajo un mismo nombre, de forma tal que es posible modificar los parámetros del método, o reescribir su funcionamiento, o incrementar más funcionalidades a un método.