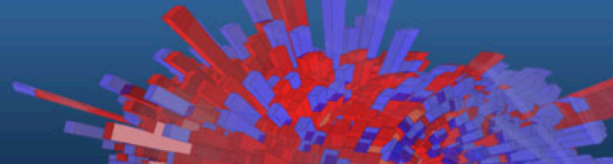
# Python and ROOT: Effective and Interactive Analysis of Big Data

https://root.cern.ch

D. Piparo, E. Tejedor for the ROOT Team

2nd Developers@CERN Forum

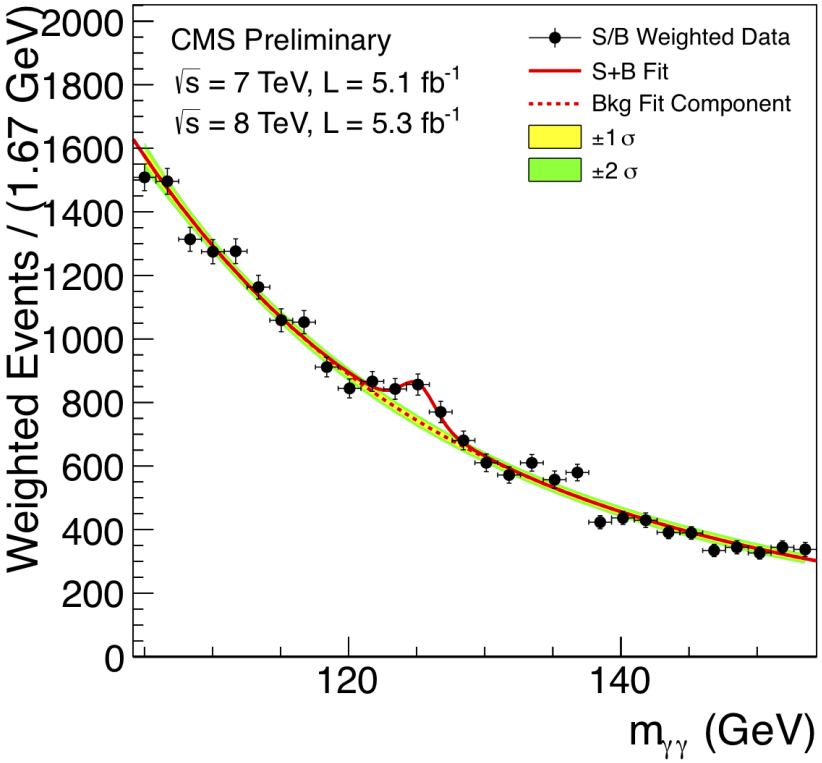

# Summary



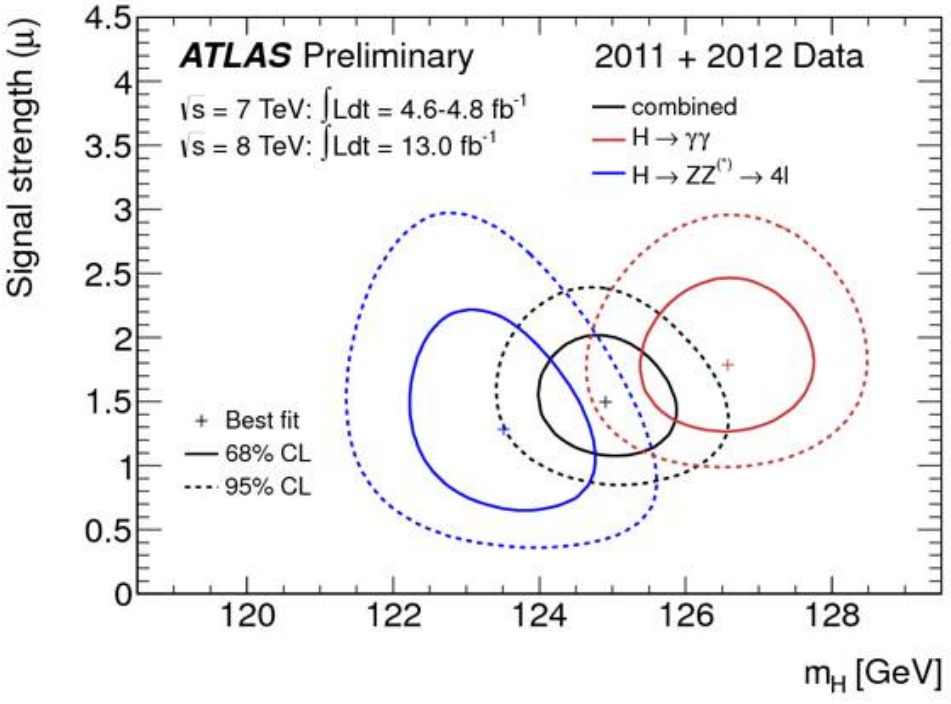
- **ROOT** in a Nutshell
- Python + ROOT (PyROOT) examples at CERN
- JuPyROOT: Integration of **Jupyter** notebooks and PyROOT
- **SWAN**: a **S**ervice for **W**eb based **AN**alysis

# ROOT in a Nutshell

# What can you do with ROOT?




**LHC collision in CMS:**
**event display, also done with ROOT!**



# ROOT in a Nutshell

ROOT is a software toolkit which provides building blocks for

- Data processing
- Data analysis
- Data visualisation
- Data storage

**An Open Source Project**
*All contributions are warmly welcome!* **LGPL2**

ROOT is written mainly in C++ (C++11 standard)

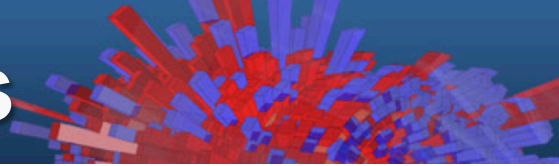
- Bindings for Python and other languages* provided


Adopted in High Energy Physics and other sciences (but also industry)

- ~0.5 ExaBytes of data in ROOT format on the LHC Computing Grid
- Fits and parameters' estimations for discoveries (e.g. the Higgs)
- Thousands of ROOT plots in scientific publications

* Atm only Python for the 6 series

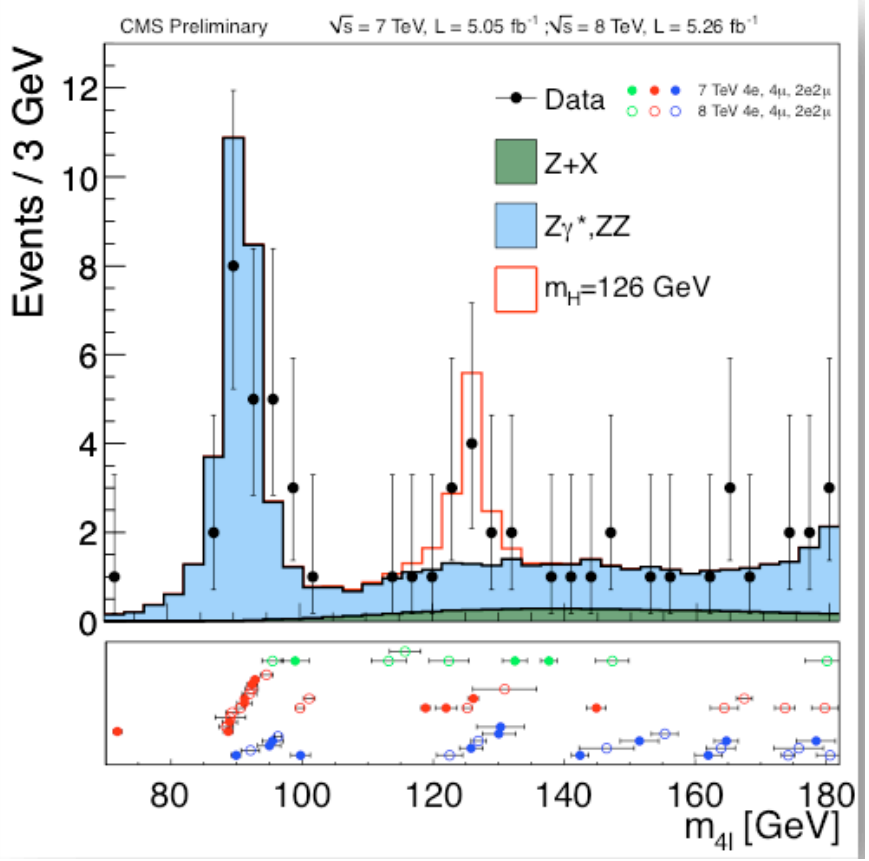
# Some of the Functionalities



ROOT a framework for a variety of activities, for example:

- Data analysis: histograms, graphs, trees
- I/O: row-wise, column-wise storage of **any** C++ object
- Multiprocessing sub-framework (map, map-reduce …)
- Statistical tools (RooFit/RooStats): rich modeling and statistical inference
- Math: non trivial functions (e.g. Erf, Bessel), optimised math functions (VDT), linear algebra
- **C++ interpretation and JIT**: fully C++11 (and optionally C++14) compliant
- Machine Learning (TMVA): e.g. Boosted decision trees, neural networks
- And more: HTTP server, JavaScript visualisation, advanced graphics (2D, 3D, event display).

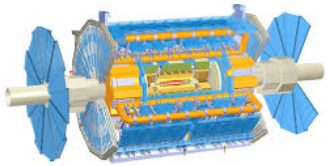
# Prestigious Discoveries with ROOT

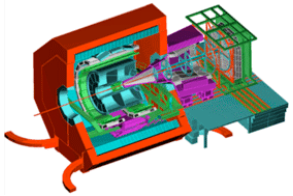


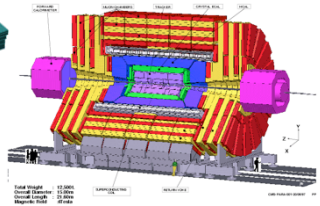
# … guess who is also using it …

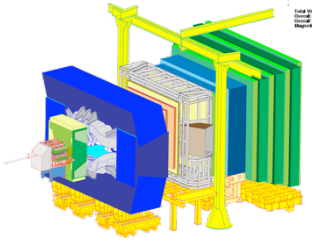
# ROOT Application Domains

A selection of the experiments adopting ROOT









Event Filtering

Data

Analysis

Offline Processing

Reconstruction

Further processing, skimming

Event Selection, statistical treatment …

Raw

Reco

…

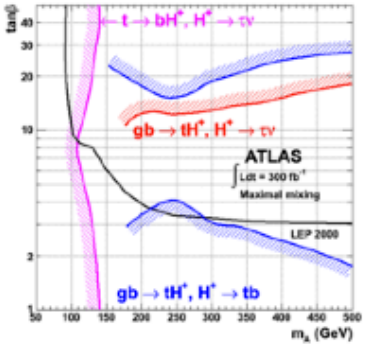
Analysis Formats

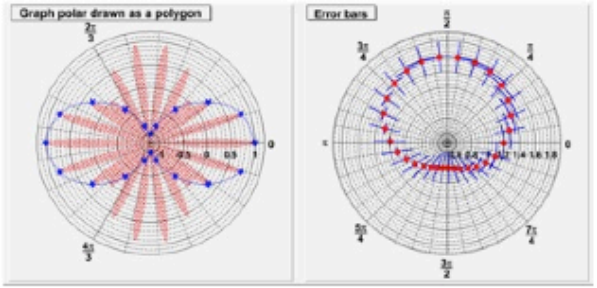
Images

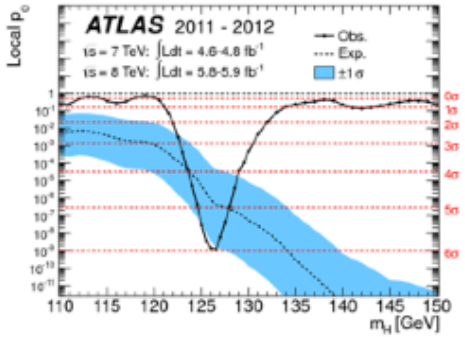
Data Storage: Local, Network

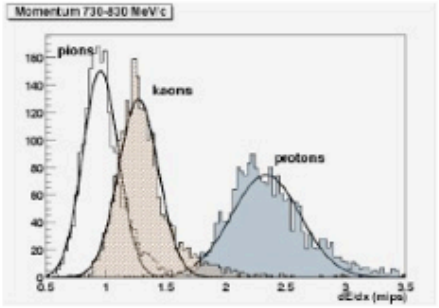
# Graphics In ROOT

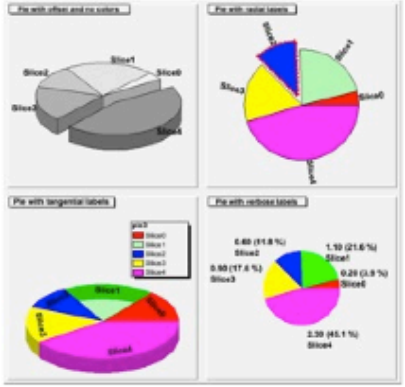
Many formats for data analysis, and not only, plots

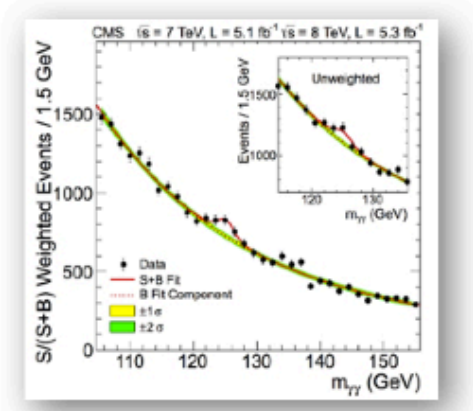


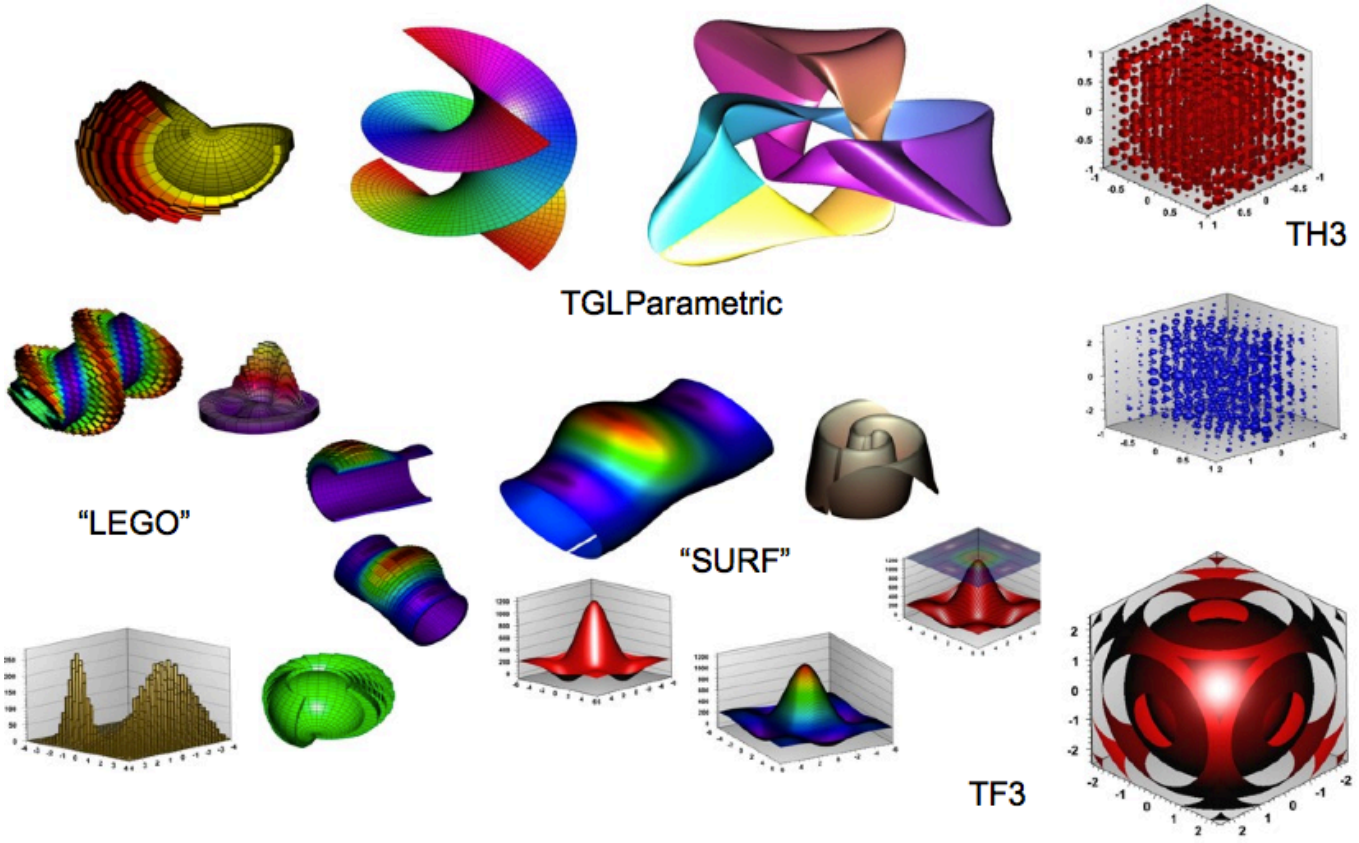





# 3D Graphics



# Interpreter

ROOT is shipped with a C++ interpreter, CLING

- C++ interpretation: non trivial, not foreseen by the language!
- One of its kind: Just In Time (JIT) compilation
- A C++ interactive shell

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

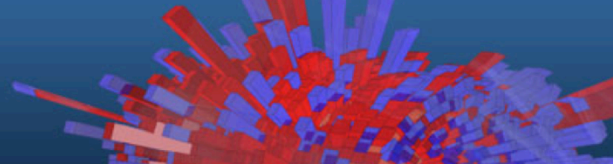
Can interpret "macros" (non compiled programs)

- Rapid prototyping possible

ROOT provides also a Python interface:

- Can use Python interpreter directly after a simple *import ROOT*
- Possible to "mix" the two languages (see more in the following slides!)

# C++ Interpretation



**Many analyses steered with interpreted C++ macros!**

```
~-> root
root [0] i = 4
(int) 4
root [1] TH1F h("myHisto","Histogram Title", 64, -4, 4)
(TH1F &) Name: myHisto Title: Histogram Title NbinsX: 64
root [2] h.FillRandom("gaus")
root [3] h.GetSkewness()
(Double_t) 0.00699916
…
```

# The Python Interface

```
$> python
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import ROOT
>>> h = ROOT.TH1F('myHisto','Histogram Title', 64, -4, 4)
>>> h.FillRandom('gaus')
>>> h.GetSkewness()
0.00699916
…
```

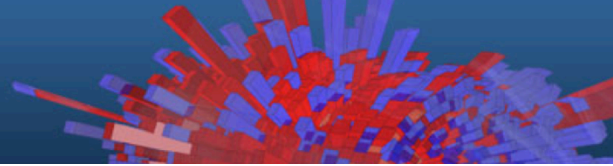
# Interactivity without Bindings

```
>>> import ROOT
>>> ROOT.gInterpreter.ProcessLine('#include "a.h"')
0L
>>> o=ROOT.A("PyName!")
>>> o.printName()
PyName!
>>> ROOT.dummy()
42
```

```
#include <iostream>
class A{
 public:
  A(const char* n):m_name(n){};
  void printName(){std::cout << m_name
                             << std::endl;}
private:
const std::string m_name;};
int dummy(){return 42;}
```

Superior interpreter and typesystem: no need of hand made bindings for interactivity

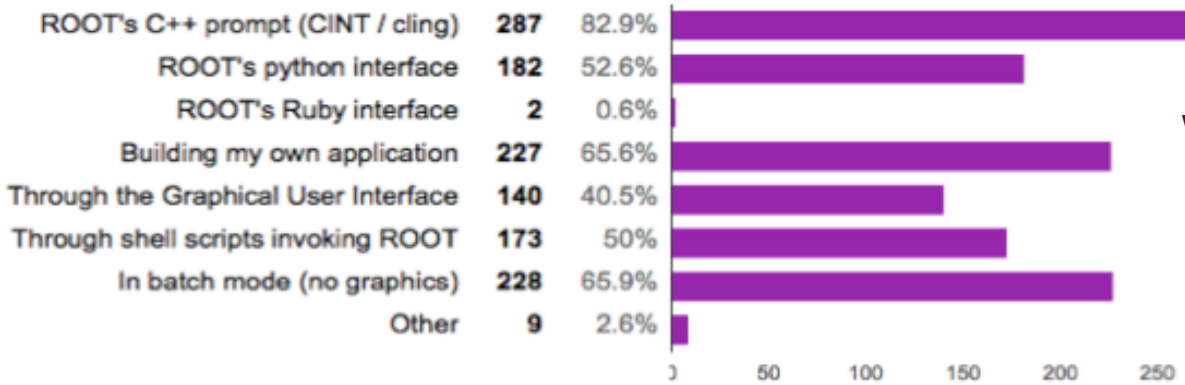
# Melting Python and C++



See the **Seamless Melting of Python and C++** contribution for all the details:

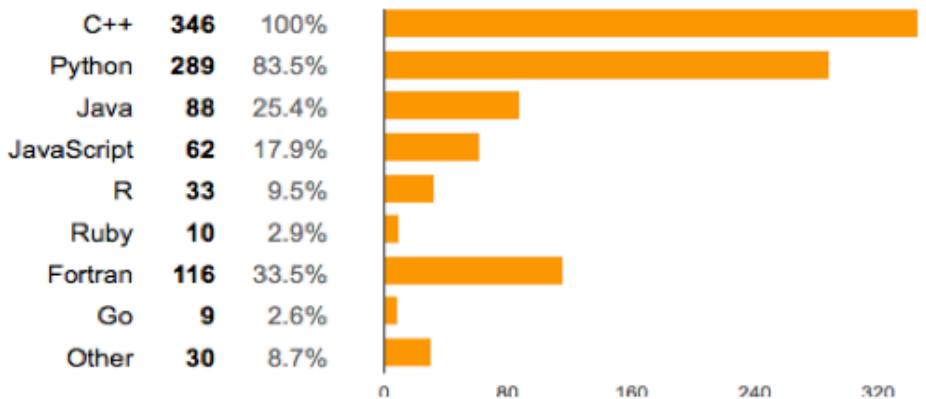
https://indico.cern.ch/event/487416/contributions/2174917

# Python and C++ usage



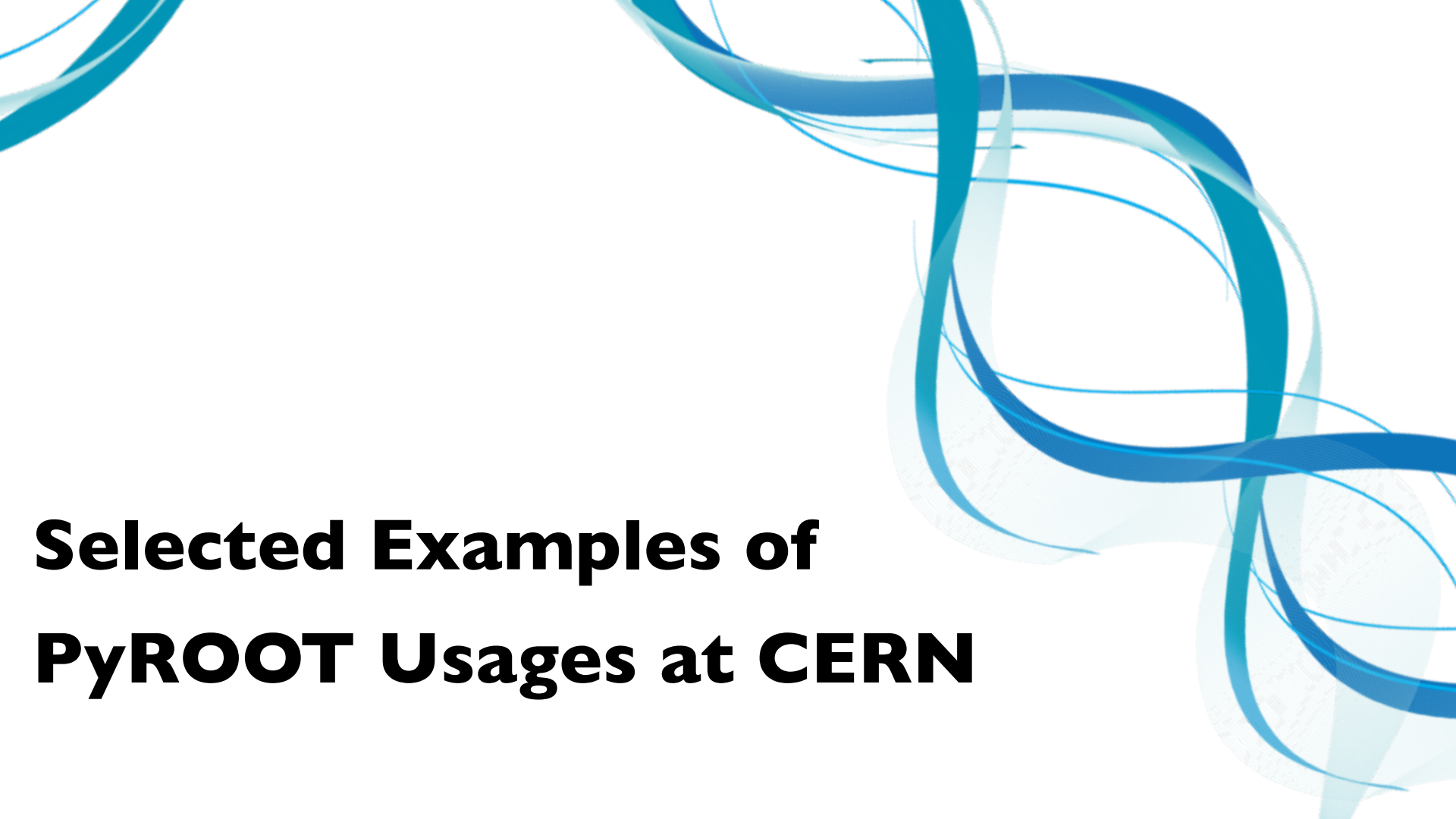

Which ROOT interfaces do you use?

What languages do you know?




Source: ROOT Users' Workshop 2015

http://indico.cern.ch/event/349459/contributions/822821

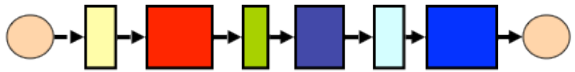


# Selected Examples of PyROOT Usages at CERN

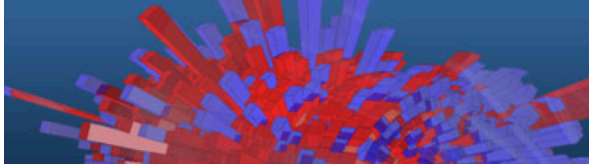
# HEP Experiments Frameworks

- Data processing of experiments managed by a *Framework*
  - *E.g.* AliROOT, Basf2, Gaudi, CMSSW
- Software bus model to orchestrate the data processing algorithms + ancillary components (e.g. Services to interface to condition DBs, I/O functionalities)
- Same C++ code (factored in libraries) configured by **many parameters**
  - E.g. (long) list of algorithms to execute, selection criteria to be applied at the High Level Trigger, input and output location, algorithm specific non-data inputs

Two possibilities:



1) Invent a DSL to express parameters
2) Rely on something that exists



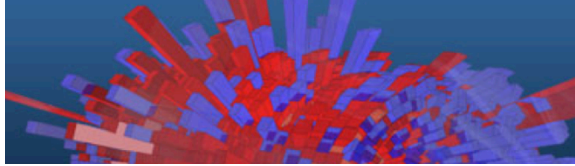
**AlgSequencer.opts**

```
//##############################################################
// Job options file
//==============================================================
#include "Common.opts"

//--------------------------------------------------------------
// Private Application Configuration options
//--------------------------------------------------------------
ApplicationMgr.TopAlg  = { "ParentAlg" };

// Set output level threshold (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL )
MessageSvc.OutputLevel  = 3;

//--------------------------------------------------------------
// Event related parameters
//--------------------------------------------------------------
ApplicationMgr.EvtMax    = 10;   // events to be processed (default is 10)
ApplicationMgr.EvtSel    = "NONE"; // do not use any event input


//--------------------------------------------------------------
// Testing scheduling stops in Event loop processing
//--------------------------------------------------------------
ApplicationMgr.TopAlg  += { "StopperAlg" };
StopperAlg.StopCount = 20;
```

**Gaudi configuration**:

- Configuration **was** expressed by “job option files”
- Domain Specific Language

https://gitlab.cern.ch/gaudi



```
##############################################################
# Job options file
#==============================================================

from Gaudi.Configuration import *
from Configurables import ParentAlg, StopperAlg, Prescaler, HelloWorld, TimingAuditor

from Configurables import GaudiExamplesCommonConf
GaudiExamplesCommonConf()

#--------------------------------------------------------------
# Testing Sequencers
#--------------------------------------------------------------
p1  = Prescaler('Prescaler1', PercentPass = 50., OutputLevel = WARNING )
p2  = Prescaler('Prescaler2', PercentPass = 10., OutputLevel = WARNING )
h   = HelloWorld( OutputLevel = DEBUG  )
c1  = EventCounter('Counter1')
c2  = EventCounter('Counter2')
s1  = Sequencer('Sequence1', Members = [p1, h, c1] )
s2  = Sequencer('Sequence2', Members = [p2, h, c2] )
top = Sequencer('TopSequence', Members = [s1, s2], StopOverride = True )

#--------------------------------------------------------------
# Testing the new GaudiSequencer
#--------------------------------------------------------------
sand = GaudiSequencer( 'ANDSequence',
                       Members = [ HelloWorld('AND'), EventCounter('ANDCounter') ],
                       MeasureTime = 1 )
```

**AlgSequencer.py**

**Gaudi configuration**:

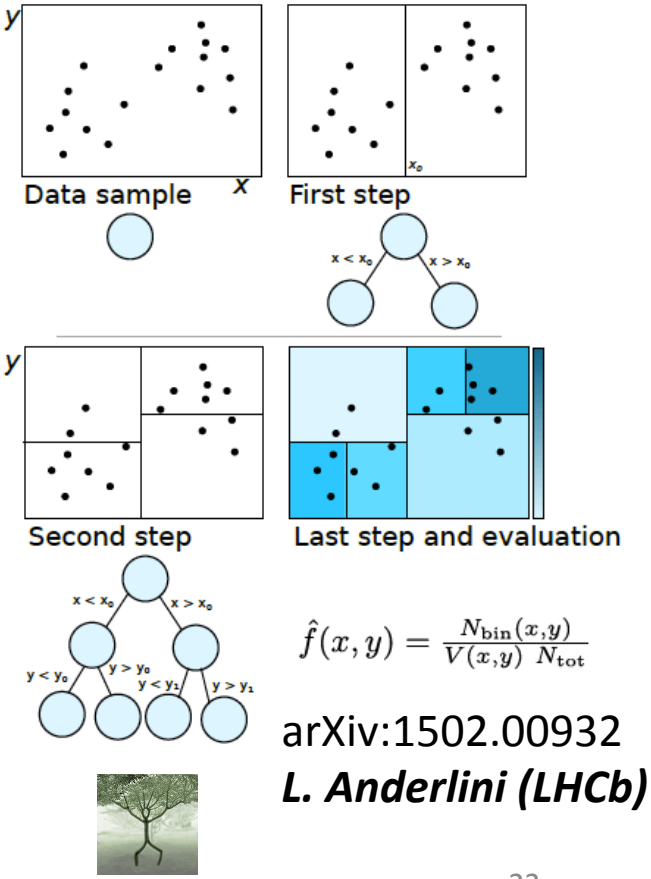
- Now Python and PyROOT based
- Rely on existing syntax
- Scripting, pickling, management of jobs: more scalability and clarity.

# Users' Code Too!

Density Estimation Trees: exploratory data analysis for multi-dimensional data models of large samples.

```
import ROOT

## Setup the PyROOT environment
ROOT.gROOT.SetBatch()
## Defines the Density Estimation Trees library
ROOT.gSystem.Load ( "libDet" )

from ROOT import RooRealVar, RooDataSet, RooArgSet, RooArgList
from ROOT import RooDetPdfMultiCore as RooDetPdf
from ROOT import DetInterfaces, TH2D, RooRandom

from DensityEstimationTrees import Report, Graphics, SampleGraph
from DensityEstimationTrees.RooFitCommands import *
print "Done."

# define axis variables
x = RooRealVar ( "x", "x", 0, 10 )
y = RooRealVar ( "y", "y", 0, 10 )

# define template histogram
h2_axes = TH2D ( "axes", ";x;y", 500, 0, 10, 500, 0, 10)
```

https://gitlab.cern.ch/landerli/density-estimation-trees




$$\hat{f}(x,y) = \frac{N_{\text{bin}}(x,y)}{V(x,y)\ N_{\text{tot}}}$$



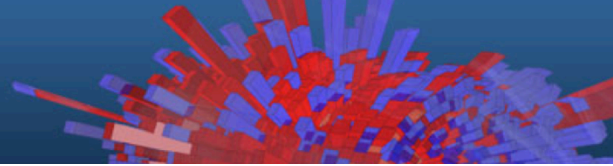
arXiv:1502.00932

***L. Anderlini (LHCb)***

# Data Quality Monitoring (DQM)

- Experiment's software: million of lines of code, O(100) libraries, constantly developed and improved
- Need to assess continuously quality of final Physics results
  - Traditional unit and integration testing not enough
  - Produce and compare with references thousands of histos of Physics quantities (E.g. energy of electrons, charge of muons, tracking efficiency..)
- Many releases of experiments software, many Physical quantities: need automated way to statistically compare large sets of histograms!

# An Approach from CMS



Concretely these histogram comparisons mean:

- Fetch histograms from central histos DB (ROOT format),
- Navigate file system like structure in ROOT files
- Apply statistical tests
- Produce plots
- Produce web pages with aggregated results

PyROOT comes to rescue: the RelMon tool

https://twiki.cern.ch/twiki/bin/view/CMSPublic/RelMon

Yes, *many* histograms!

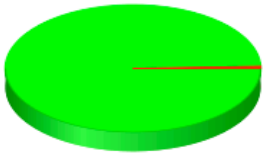
RelMon Global Report: FullSimReport

Summary

*769882 COMPARISONS:*

- SUCCESS: 99.4% (765607)
- FAIL: 0.6% (4275)

To the DQM GUI...



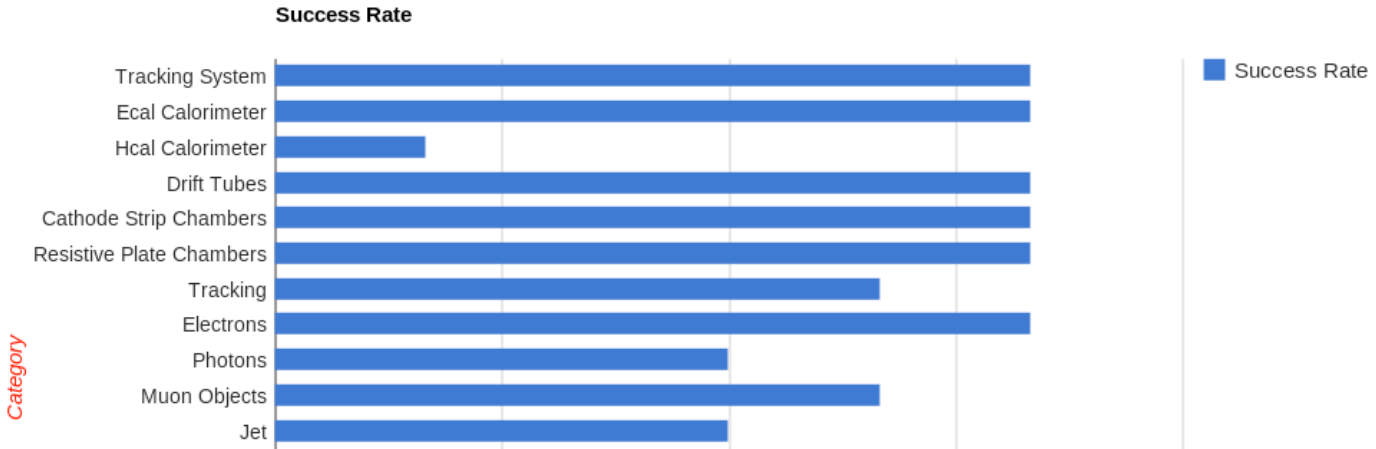
Releases:

- CMSSW_4_4_0_pre2
- CMSSW_4_4_0_pre3

Statistical Test (Threshold):

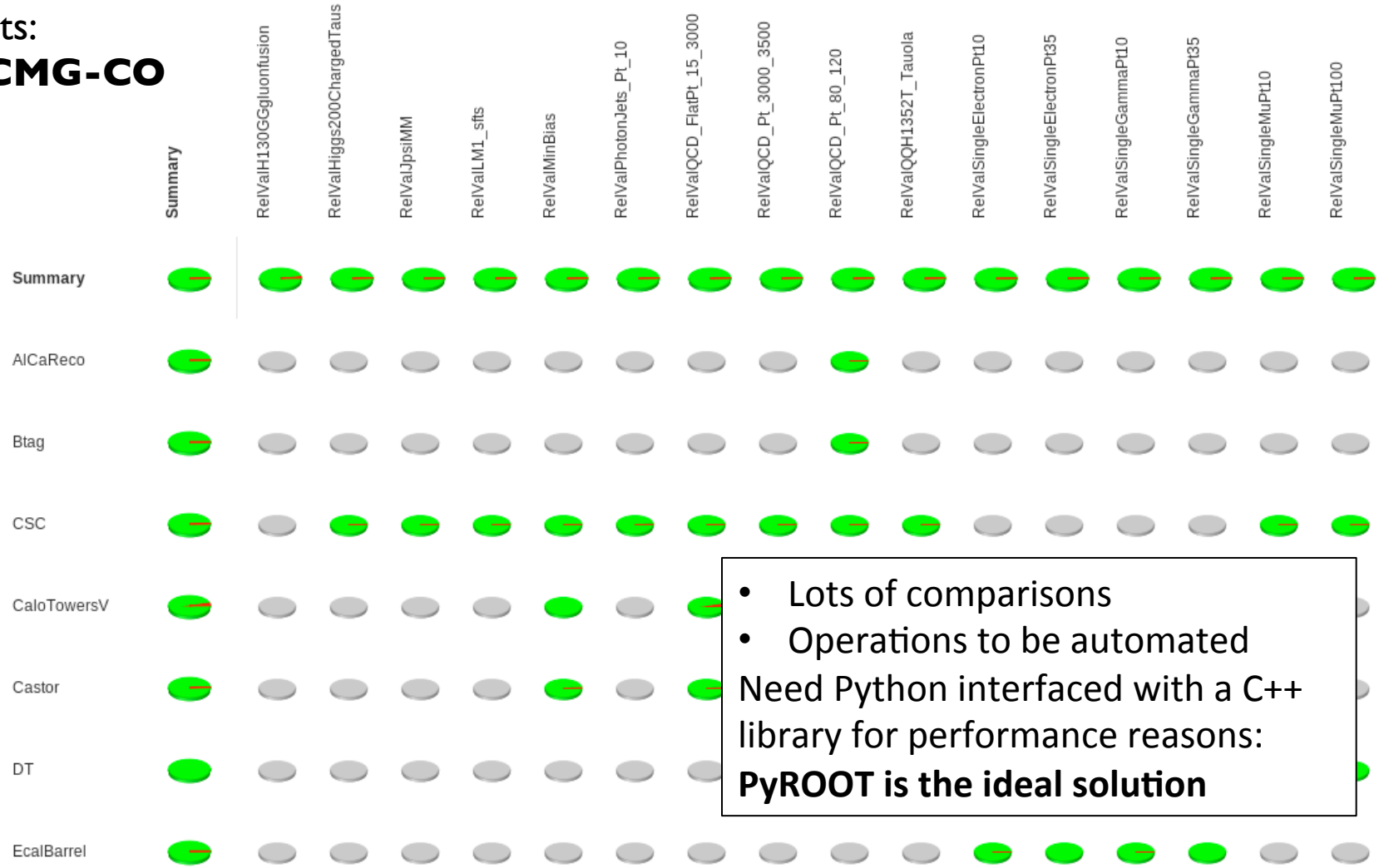
- CHI2 (0.00)

Regression between two software releases




Credits:
**EP/CMG-CO**

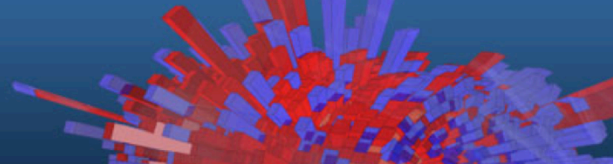
Credits:
**EP/CMG-CO**




- Lots of comparisons
- Operations to be automated

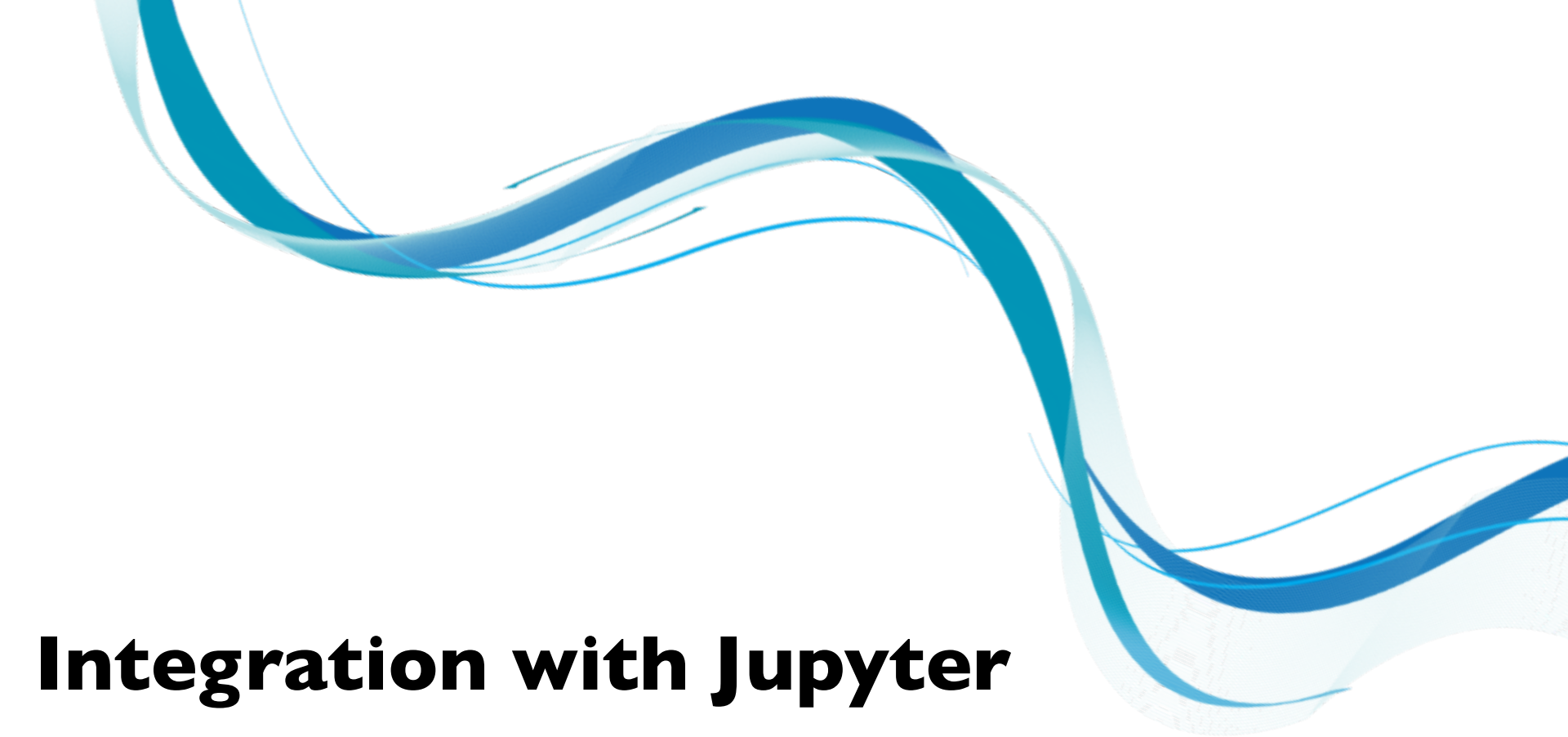
Need Python interfaced with a C++ library for performance reasons:
**PyROOT is the ideal solution**

# PyCOOL



More interesting details in the contribution of A. Valassi **Python bindings for C++ using PyROOT/cppyy: the experience from PyCool in COOL**

http://indico.cern.ch/event/487416/contributions/2174905



# Integration with Jupyter

# Interlude: The Notebook

A web-based interactive computing interface and platform that combines code, equations, text and visualisations.


http://www.jupyter.org

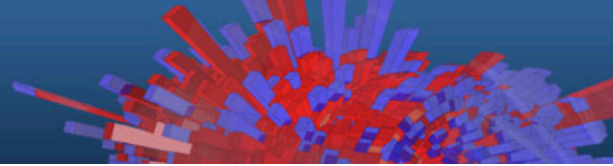
Many supported languages: Python, Haskell, Julia, R … One generally speaks about a "kernel" for a specific language

**No excuses possible when it comes to describe all steps in an analysis!**

In a nutshell: an "interactive shell opened within the browser"

Also called:
"Jupyter Notebook" or "IPython Notebook"

# JuPyROOT

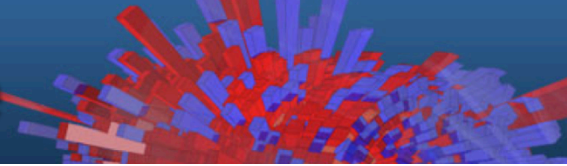


Full integration of PyROOT with Jupyter Notebooks

- "import ROOT": only action required to activate all goodies!
- Inlining of plots as images or JavaScript interactive graphics
- *Magics* to JIT or compile C++ code for acceleration
  - Immediately usable in Python!
- Tab completion for name and methods of classes known to ROOT
- Capturing of output from C++ libraries

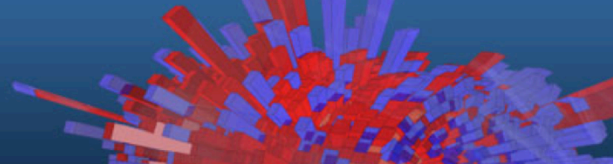
"Like before, but better"

# Notebook Examples Online



# A Service for Web based ANalysis at CERN: SWAN

# SWAN



A service for **analysing data** in the **Cloud** only with a **web browser**, using the **CERN software suite** and relying on existing **CERN services in production**

**https://swan.web.cern.ch**

Promotes CERN software suite and services, proposes widely adopted analysis ecosystems.

# Data Analysis as a Service

- Platform independent: only with a web browser
  - Analyse data via the Notebook web interface
  - No need to install and configure software
- Calculations, input and results "in the cloud"
- Allow easy sharing of scientific results: plots, data, code
  - Storage is crucial, both mass (EOS) and synchronised (CERNBox)
- Simplify teaching of data processing and programming
- Eases analysis reproducibility and documentation
- C++, Python and other languages or analysis "ecosystems"
  - Interfaced to widely adopted scientific libraries
  - e.g. Pandas, Numpy, ROOT, matplotlib, …

# Novel Idea, Production Technologies

SWAN relies on production technologies at CERN. In a nutshell:

- CPU: CERN OpenStack instance

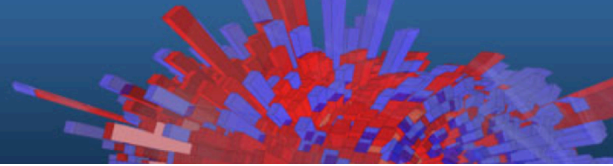
- CERN Credentials (SSO)

- Software: LCG releases on CVMFS
  - ~200 packages: fastjet, ROOT, Geant4, Pythia, matplotlib…
- Storage, in two flavours:

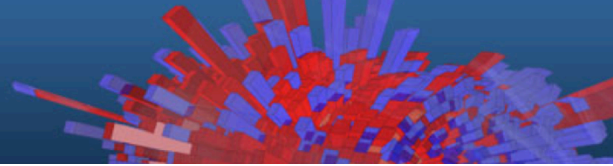

  - Mass: EOS (the storage backend of HEPData too!)
  - Synchronised: CERNBox (the "home directory" in SWAN)

- Same storage backend of **user and experiments data**.
- **Software building blocks** for data analysis
- **Share** code, results and derived data with colleagues

# SWAN In Action

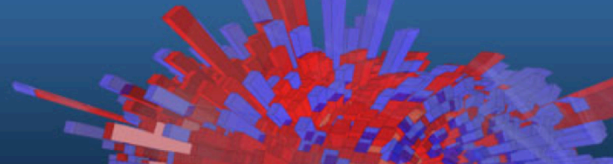


# Conclusions



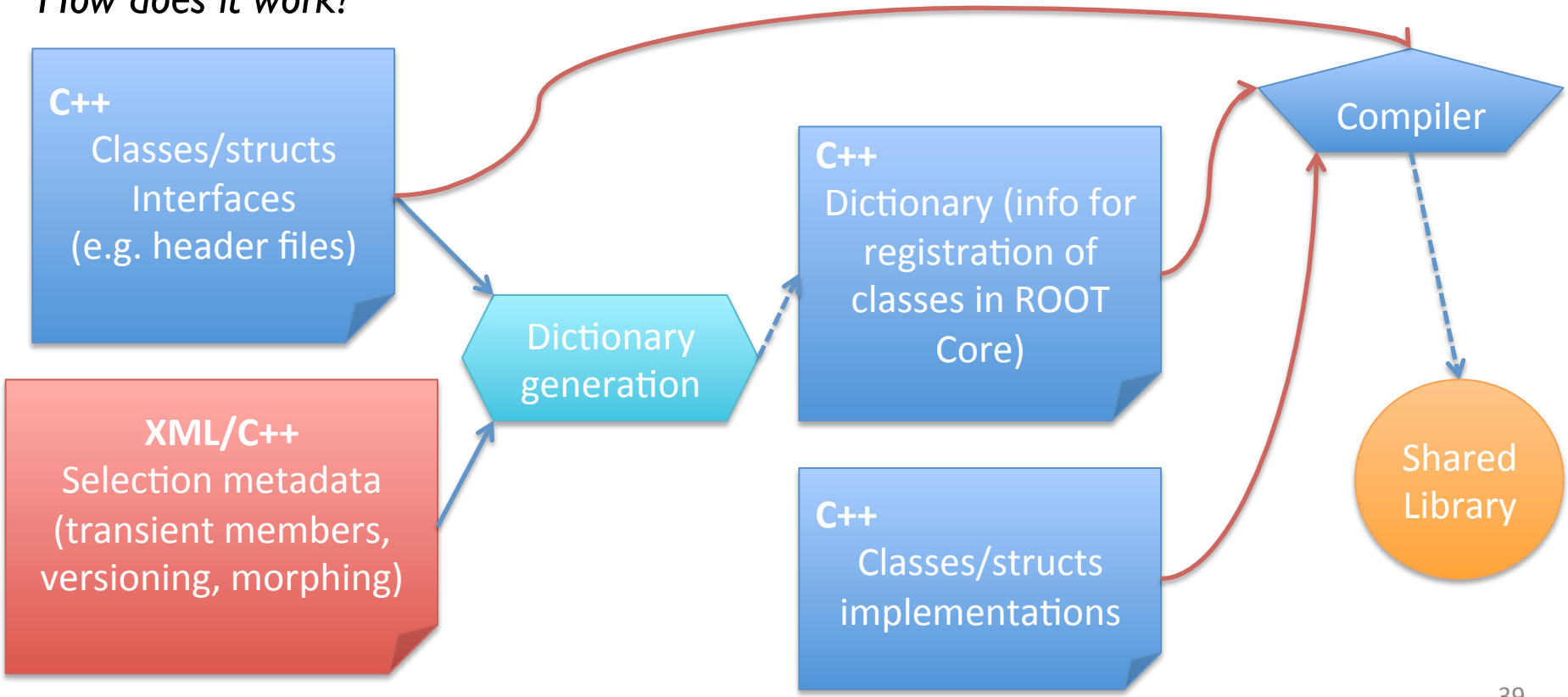
- ROOT, a widely used, production grade framework for HEP data processing can be used via its Python interface, PyROOT
  - Other C++ libraries can be then used too, w/o bindings! Free and open source!
- PyROOT: widely adopted by the experiments at LHC (and not only)
  - E.g. onfiguration management, validation and in users' analysis setups
- Full integration with Jupyter technology
  - Notebooks, interactive graphics, reproducibility, documentation
- SWAN: a CERN Service for Web based Analysis
  - PyROOT only with your browser
  - CERNBox, EOS, CERN and external software available with a click
  - Promotes CERN Services and Software assets within the Organisation

# Backup

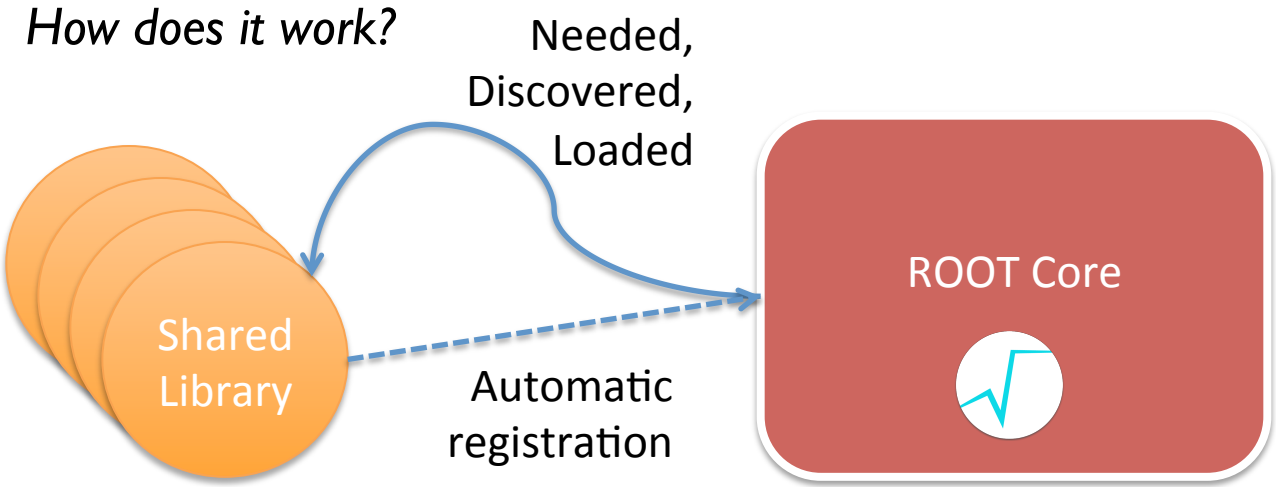


# Persistency (I/O)

*How does it work?*



# Persistency (I/O)



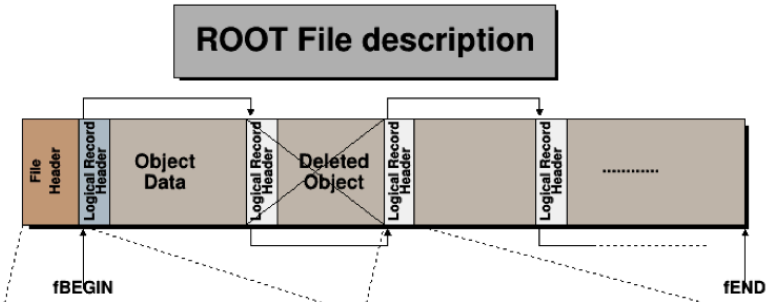

Now ROOT “knows” how to serialise the instances implemented in the library (series of data members, type, transiency) and to write them on disk in row or column format.

# The ROOT file Format

**The ROOT file format is the format of HEP**

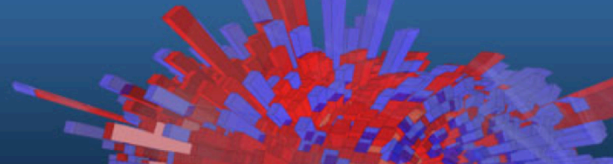
- Some sporadic exception in very specialised and controlled environments (e.g. Raw data of LHCb)

This is its complete description.




| Byte Range | Record Name | Description |
|---|---|---|
| 1->4 | "root" | Root file identifier |
| 5->8 | fVersion | File format version |
| 9->12 | fBEGIN | Pointer to first data record |
| 13->16 [13->20] | fEND | Pointer to first free word at the EOF |
| 17->20 [21->28] | fSeekFree | Pointer to FREE data record |
| 21->24 [29->32] | fNbytesFree | Number of bytes in FREE data record |
| 25->28 [33->36] | nfree | Number of free data records |
| 29->32 [37->40] | fNbytesName | Number of bytes in TNamed at creation time |
| 33->33 [41->41] | fUnits | Number of bytes for file pointers |
| 34->37 [42->45] | fCompress | Compression level and algorithm |
| 38->41 [46->53] | fSeekInfo | Pointer to TStreamerInfo record |
| 42->45 [54->57] | fNbytesInfo | Number of bytes in TStreamerInfo record |
| 46->63 [58->75] | fUUID | Universal Unique ID |

# The ROOT file Format



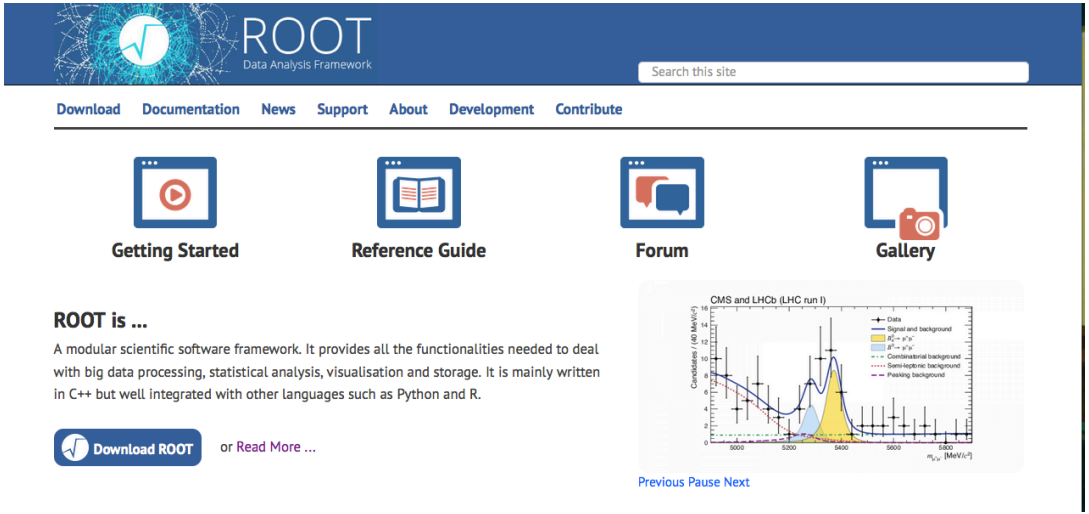
Provided and desirable features

- Simple: “keys” on disk, easily described and open source
- Demonstrated reliability: easy recovery, self describing (description of data stored with the data)
- Efficient: compressed data, options available for compression algorithms
- Remote reading over the network over already available protocols
- Backward/forward compatibility

# https://root.cern.ch

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 and more
- Forum
- …



# Preemptive Trouble Shooting

? *What could be the advantage of learning this software technology?*

! **1.** Batteries included: you have all the tools to process, store, analyse and visualise data in one single kit.

! **2.** You join a huge community, O($10^4$) users + a very supportive team of core developers

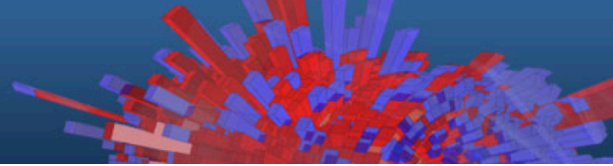
? *Why C++ and not a scripting language?!*

! Performance. Support for languages like Python

? *Why prompt and libraries instead of a GUI?*

! ROOT is a programming framework, not an office suite.

# Parallelism in ROOT



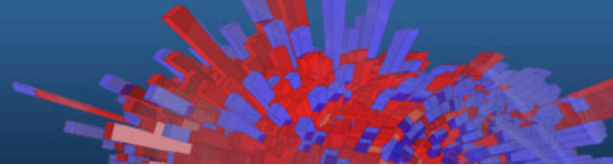
Data Parallelism (vectorisation):

- VC Library
- VDT Math Library

Task Parallelism - Multithreading and Multiprocessing

- Renovated thread awareness ★
- Production grade Proof
- Novel MultiProc facility ★
- Implicit parallelism ★

# At the Core



ROOT: modular, with synergy among its components

Slim Core ("orchestration framework") containing

- Management of global configuration settings
- A **type system**: classes, functions, methods, enumerators known to ROOT (highly non C++ standard!)
- Plugin registration mechanism: no need to load all libraries at start-up (link), just if needed, at runtime