ROOT in a Nutshell

- ROOT is a software framework with building blocks for:
  - **Data processing**
  - **Data analysis**
  - **Data visualisation**
  - **Data storage**
- ROOT is mainly written in C++ (C++11/17 standard)
  - Bindings for Python available as well ! (PyROOT)
- Adopted in High Energy Physics and other sciences and also industry
  - more than 1 Exabyte of data in ROOT format
  - Data analysis (machine learning), parameters estimations and discovery significances (e.g. the Higgs)
  - Thousands of ROOT plots in scientific publications

An Open Source Project
We are on github
github.com/root-project
All contributions are warmly welcome!

ROOT Components

ROOT can be seen as a collection of building blocks for various activities, like:

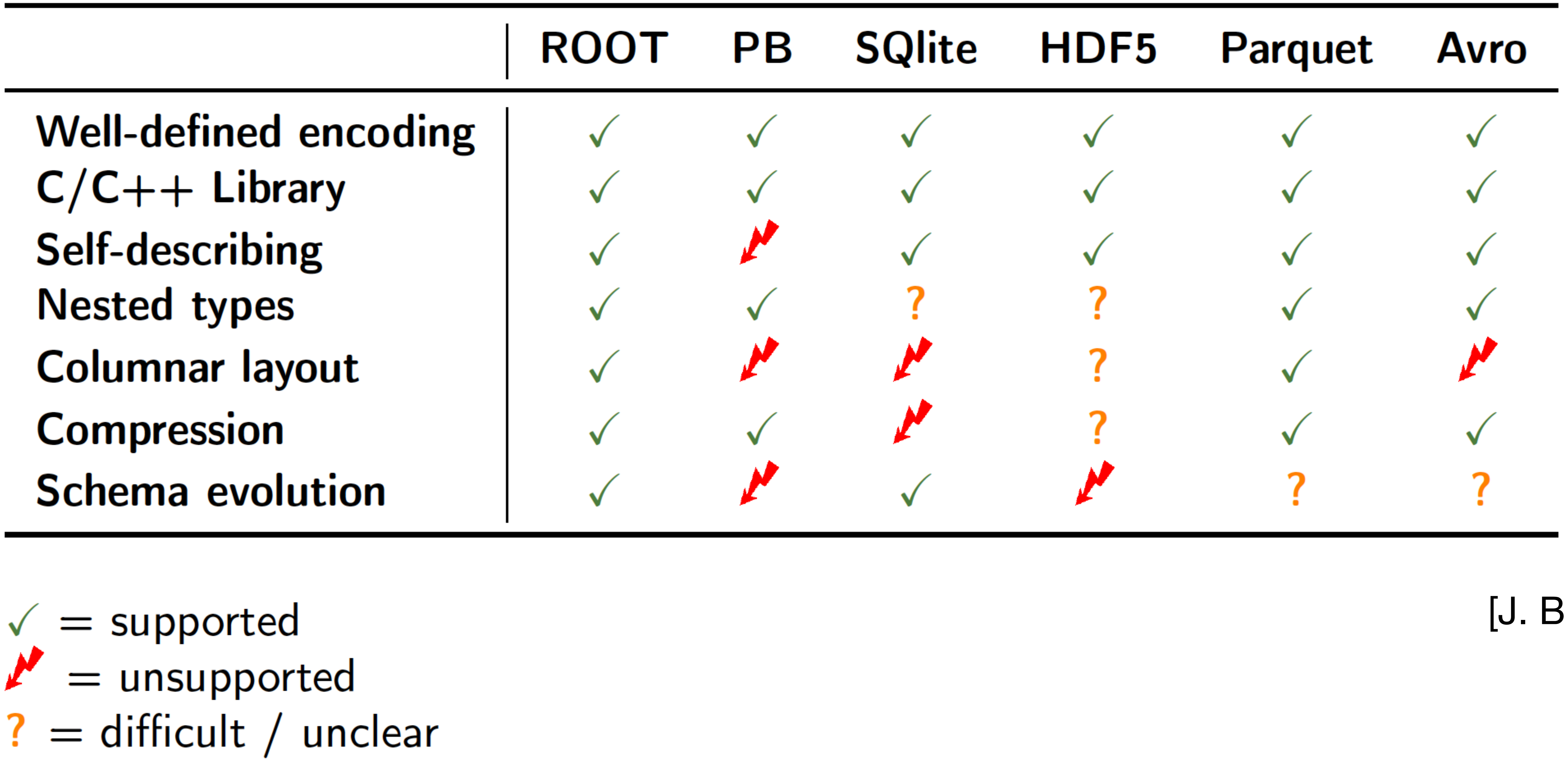
- **Data analysis**: histograms, graphs, functions
- **I/O**: row-wise, column-wise storage of any C++ object
- **Statistical tools**: rich modeling tools and statistical inference
- **Math**: math functions, linear algebra and minimisation algorithms
- **C++ interpretation**: full language compliance
- **Multivariate Analysis (TMVA)**: e.g. Boosted decision trees, Neural networks (including deep learning)
- **Advanced graphics** (2D, 3D, event display)
- **Declarative Analysis**: Data Frame for event filtering and selection
- And more: HTTP serving, JavaScript visualisation

ROOT I/O

- ROOT offers the possibility to write C++ objects into files
  - This is impossible with C++ alone
  - Used by the LHC detectors to write several petabytes per year
  - seamless C++ integration: unique feature of ROOT
- 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 simple method -

  ```
  file->WriteObject(pObj, "name");
  ```

I/O Feature Comparison



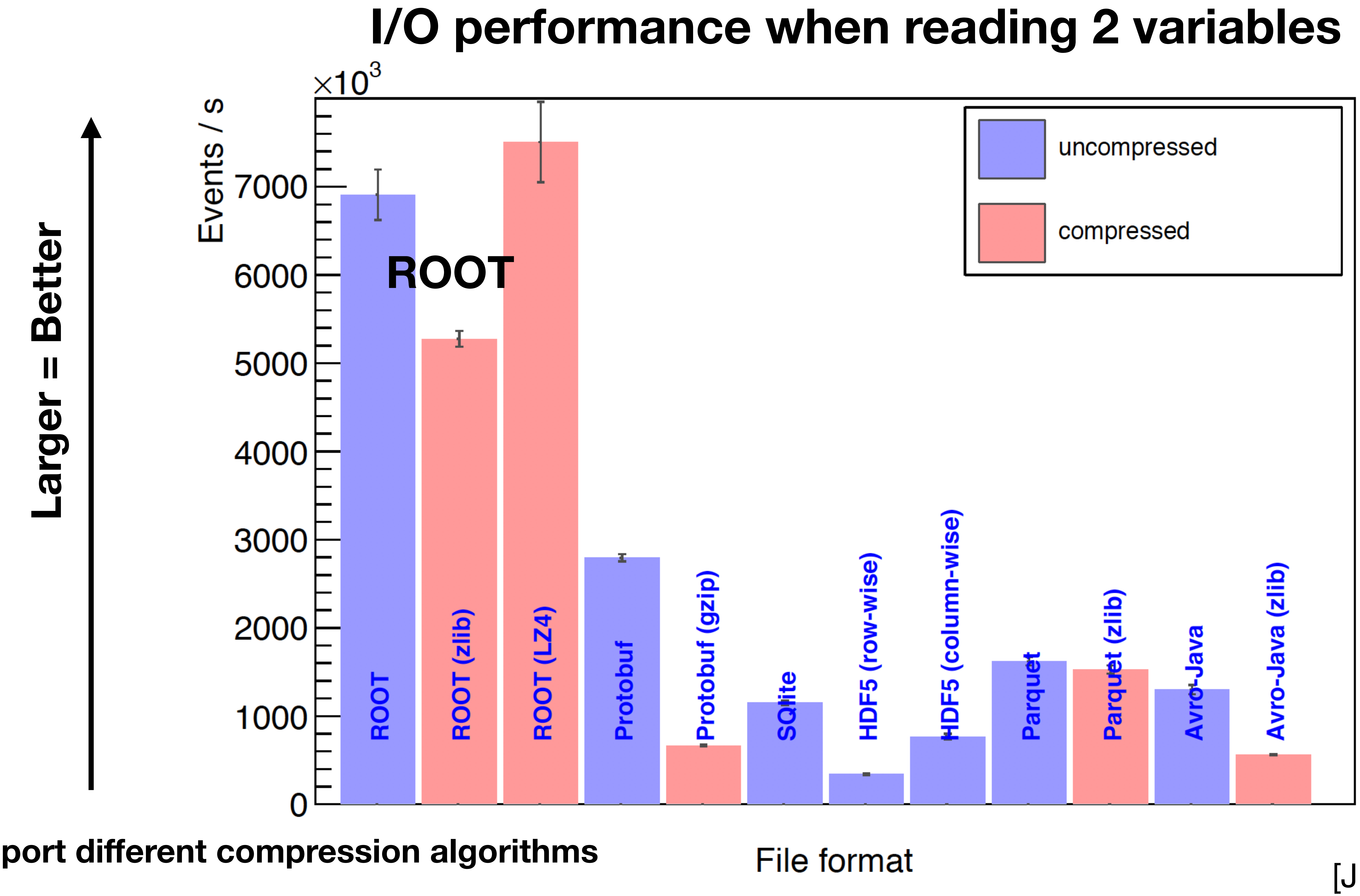
	ROOT	PB	SQlite	HDF5	Parquet	Avro
Well-defined encoding	✓	✓	✓	✓	✓	✓
C/C++ Library	✓	✓	✓	✓	✓	✓
Self-describing	✓	↯	✓	✓	✓	✓
Nested types	✓	✓	?	?	✓	✓
Columnar layout	✓	↯	↯	?	✓	↯
Compression	✓	✓	↯	?	✓	✓
Schema evolution	✓	↯	✓	↯	?	?

✓ = supported
↯ = unsupported
? = difficult / unclear

[J. Blomer, ACAT 2017]

- Unique capabilities of ROOT required for HEP data
- Superior performances when compared with others

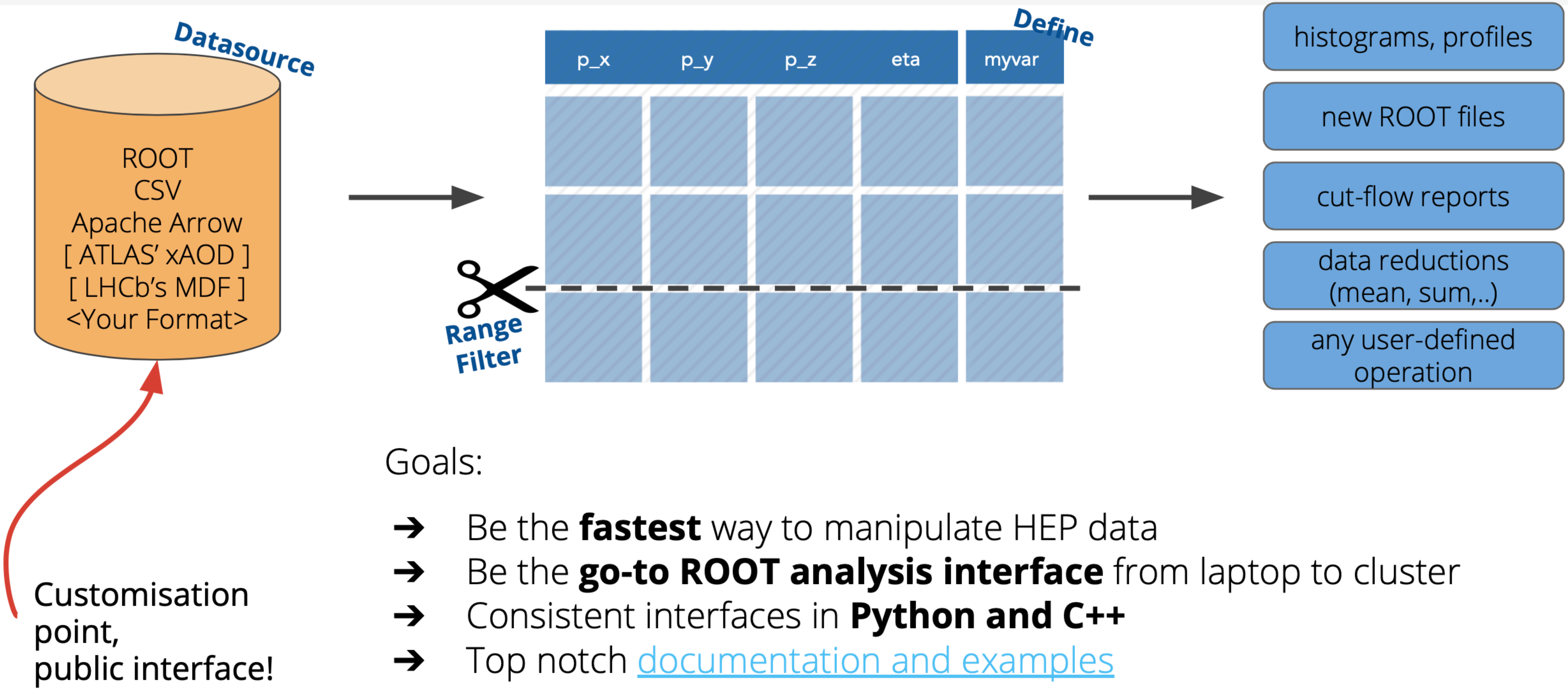
I/O Performance Comparison



Support different compression algorithms

[J. Blomer, ACAT 2017]

Event analysis: RDataFrame

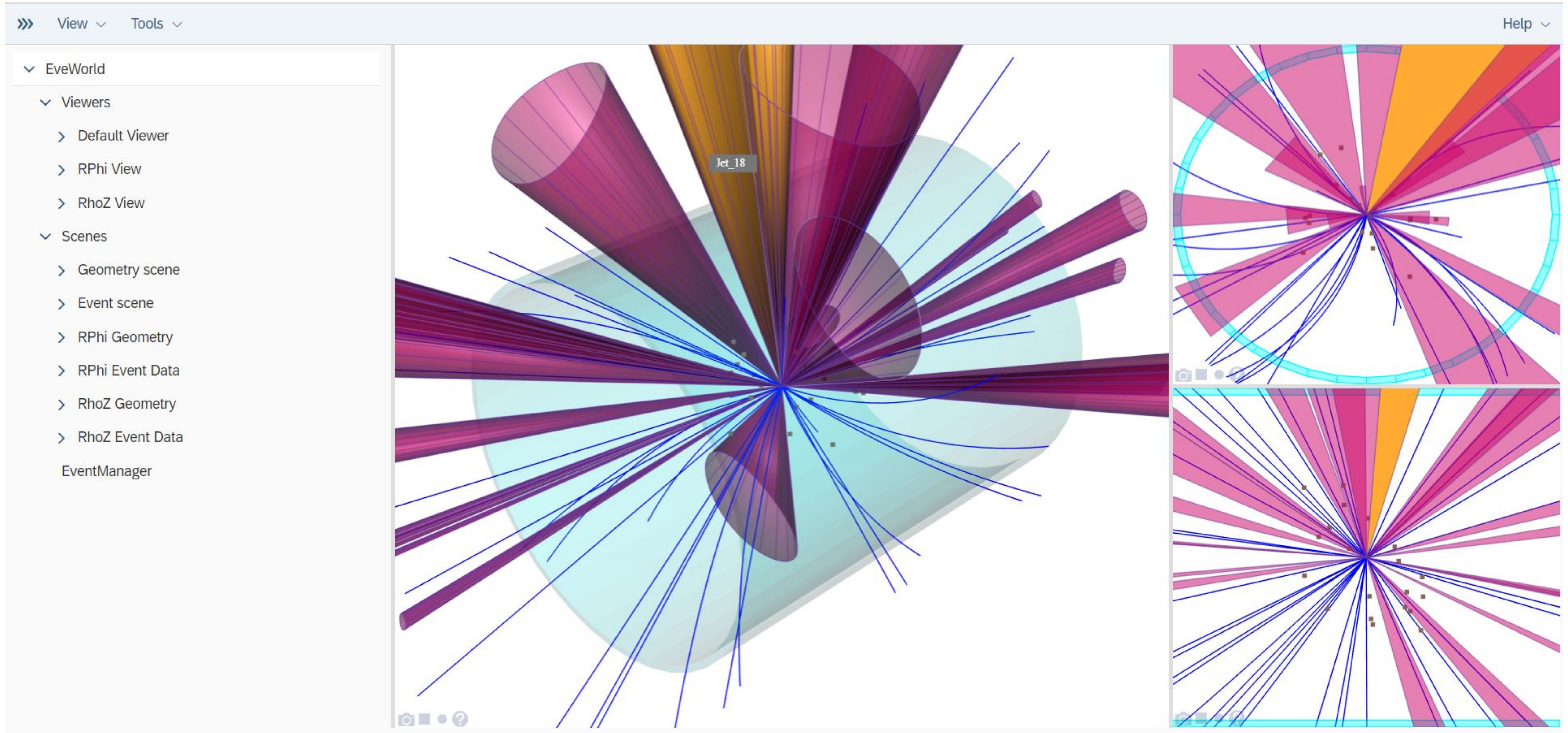


Goals:

- ➔ Be the **fastest** way to manipulate HEP data
- ➔ Be the **go-to ROOT analysis interface** from laptop to cluster
- ➔ Consistent interfaces in **Python and C++**
- ➔ Top notch documentation and examples

New ROOT Graphics

- New Graphics of ROOT uses Web based technologies
  - able to run in the Web browser



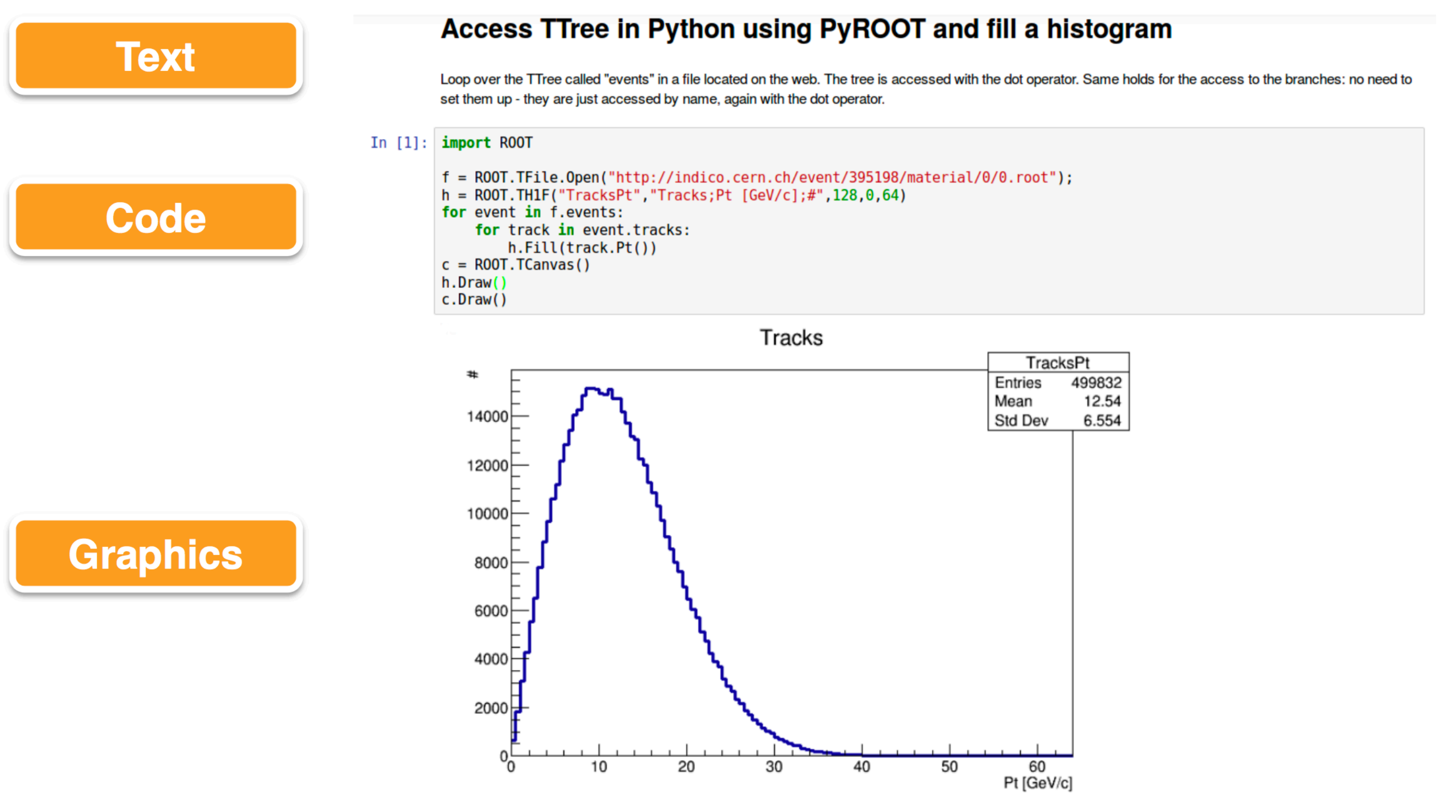
WebEve

can run as

- a standalone application
- an existing browser
- embedded in other Web based GUI's (e.g. Jupiter notebooks)

SWAN: Data Analysis as a Service

- *Interface:*
  - Jupyter Notebooks
- *Goals:*
  - Analysis only with a web browser
    - Platform independent ROOT-based data analysis
    - Calculations, input and results “in the Cloud”
    - Supports for deploying computations using GPU’s
  - Easy sharing of scientific results: plots, data, code
  - Integration with other analysis ecosystems: R, Python, ...



Access TTree in Python using PyROOT and fill a histogram

Loop over the TTree called "events" in a file located on the web. The tree is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

```
In [1]: import ROOT

        f = ROOT.TFile.Open("http://indico.cern.ch/event/395198/material/0/0.root");
        h = ROOT.TH1F("TracksPt","Tracks;Pt [GeV/c];#",128,0,64)
        for event in f.events:
            for track in event.tracks:
                h.Fill(track.Pt())
        c = ROOT.TCanvas()
        h.Draw()
        c.Draw()
```

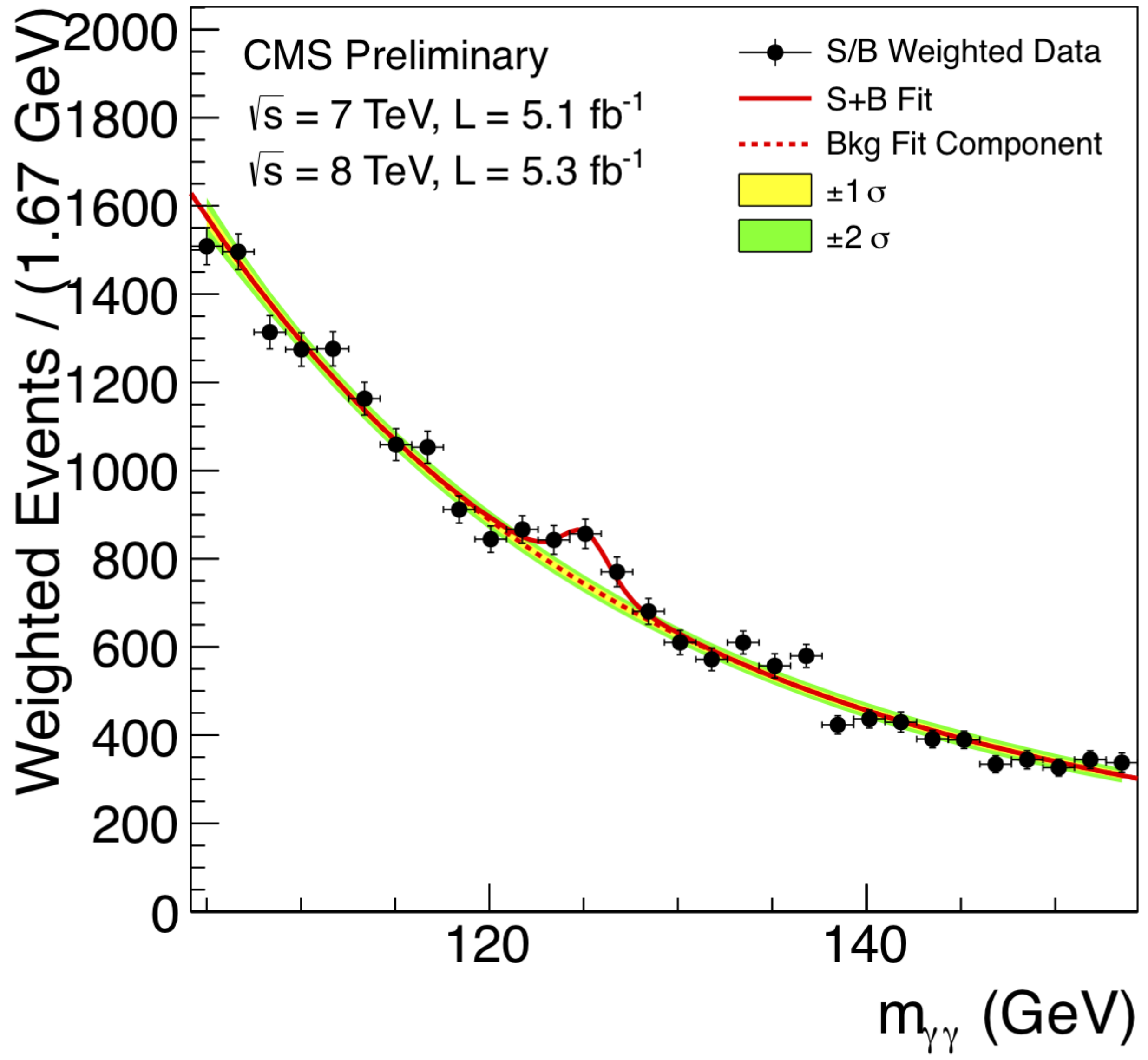
```
TracksPt
Entries    499832
Mean        12.54
Std Dev     6.554
```

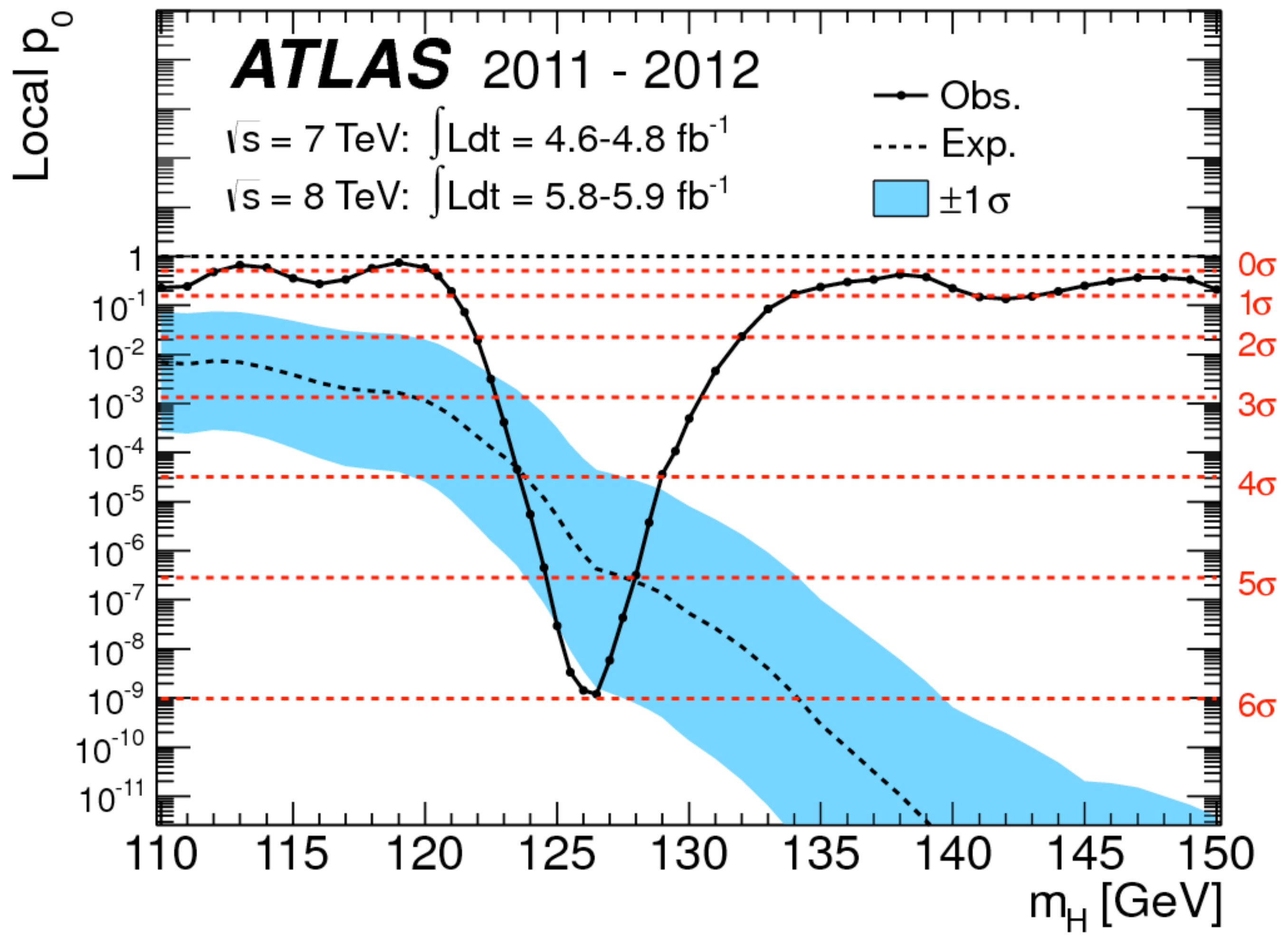
Modeling in Physics Analysis

- **Statistical modeling for physics parameter θ**
  - Estimate probability density functions p(**x** | **θ**)
    - typically using simulation (generative models)
  - parameter estimation (with uncertainties), hypothesis tests (frequentist) or Bayesian analysis to get p(**θ** |**x**$_{\text{obs}}$)
- **Discriminative modeling (classification, regression)**
  - Model the p(**y**|**x**) using a training sample {**x**,**y**}
  - methods: neural networks, decision trees (boosted trees, random forest), etc..

Statistics and Data modeling

- **RooFit**: toolkit for data modeling and fitting
  - functionality for building models
  - maximum likelihood fitting for parameter estimation
  - bootstrapping (sampling from models)
  - visualisation
- **RooStats**: advanced statistics tools
  - confidence interval estimation
  - hypothesis testing (discovery significance)
  - Bayesian analysis

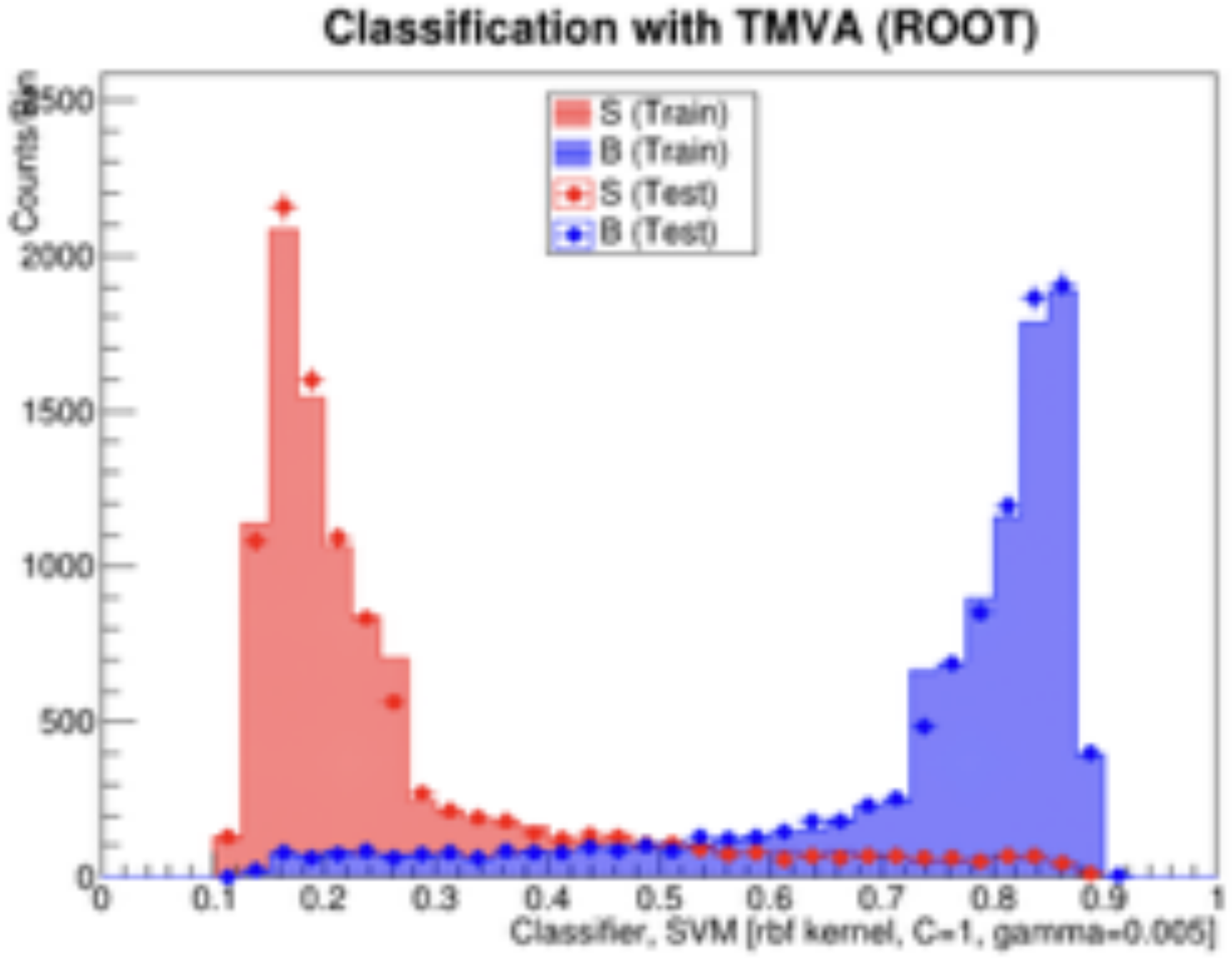




Machine Learning: TMVA

TMVA : Toolkit for Multi-Variate data Analysis in ROOT

- provides several built-in ML methods for HEP usage including:
  - **Boosted Decision Trees**
  - **Deep Neural Networks**
- and interfaces to external ML tools packages
  - scikit-learn, Keras (Tensorflow), R



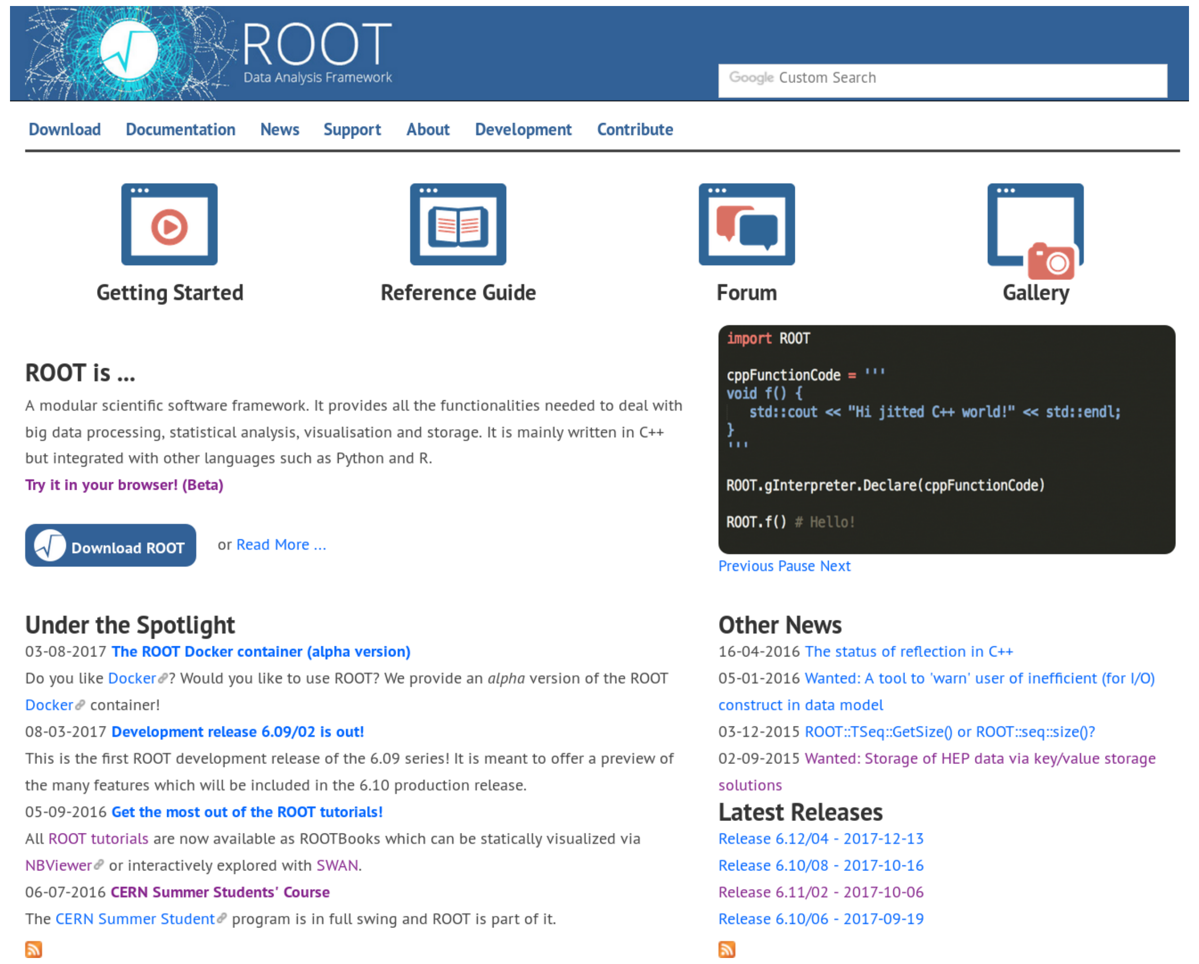
Summary

Some of the ROOT tools could be applied outside CERN domain

- Quality control and filtering:
  - usage of ROOT/IO
  - powerful filtering tools as input to ML algorithms
- Model predictions
  - statistical data modeling, ML algorithms for regression
- Time series classification and anomaly detection
  - ML algorithms for time dependent data (recurrent networks)
  - auto-encoders for anomaly detection
- Fast model evaluation
  - Optimize evaluation (inference) of ML models (DL and decision trees)
- Web data analysis in Jupyter notebooks with SWAN with GPU’s

ROOT

- Web page: https://root.cern
- Forum: https://root-forum.cern.ch
- github: https://github.com/root-project
- @root-project
- https://www.linkedin.com/groups/1826455
- root-dev@cern.ch



ROOT is ...

A modular scientific software framework. It provides all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage. It is mainly written in C++ but integrated with other languages such as Python and R.

Try it in your browser! (Beta)

or Read More ...

```
import ROOT

cppFunctionCode = '''
void f() {
   std::cout << "Hi jitted C++ world!" << std::endl;
}
'''

ROOT.gInterpreter.Declare(cppFunctionCode)

ROOT.f() # Hello!
```

Under the Spotlight

03-08-2017 The ROOT Docker container (alpha version)

Do you like Docker? Would you like to use ROOT? We provide an *alpha* version of the ROOT Docker container!

08-03-2017 Development release 6.09/02 is out!

This is the first ROOT development release of the 6.09 series! It is meant to offer a preview of the many features which will be included in the 6.10 production release.

05-09-2016 Get the most out of the ROOT tutorials!

All ROOT tutorials are now available as ROOTBooks which can be statically visualized via NBViewer or interactively explored with SWAN.

06-07-2016 CERN Summer Students' Course

The CERN Summer Student program is in full swing and ROOT is part of it.

Other News

16-04-2016 The status of reflection in C++

05-01-2016 Wanted: A tool to 'warn' user of inefficient (for I/O) construct in data model

03-12-2015 ROOT::TSeq::GetSize() or ROOT::seq::size()?

02-09-2015 Wanted: Storage of HEP data via key/value storage solutions

Latest Releases

Release 6.12/04 - 2017-12-13

Release 6.10/08 - 2017-10-16

Release 6.11/02 - 2017-10-06

Release 6.10/06 - 2017-09-19