The New ROOT Interface: Jupyter Notebooks

Enric Tejedor for the ROOT team
CERN

CHEP
10/10/2016

Prelude: The Notebook

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

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

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

The Notebook: An Example

Text

Code

Graphics

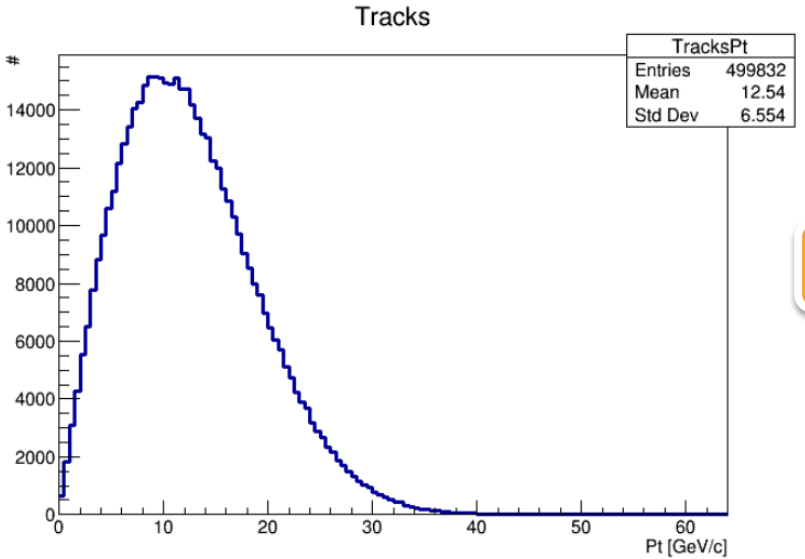
In a Browser

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()
```



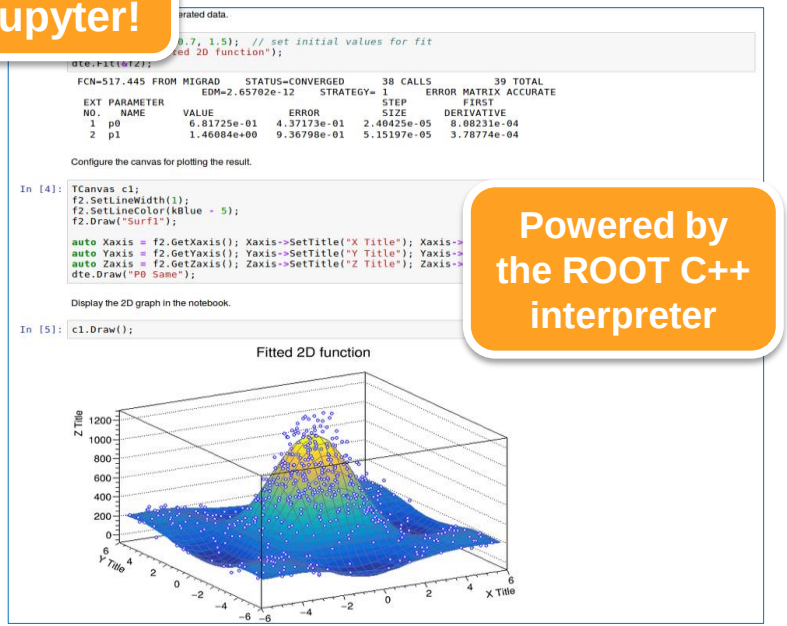
ROOT & Jupyter - Motivation

- ROOT has been fully integrated with Jupyter notebooks
- Notebook features appealing to ROOT:
  - Sharing: scientists can share their results (code, plots, text) in the form of notebooks
  - Teaching: runnable tutorials and exercises, combining code and explanations
  - Reproducibility: a notebook contains results and the code that led to them

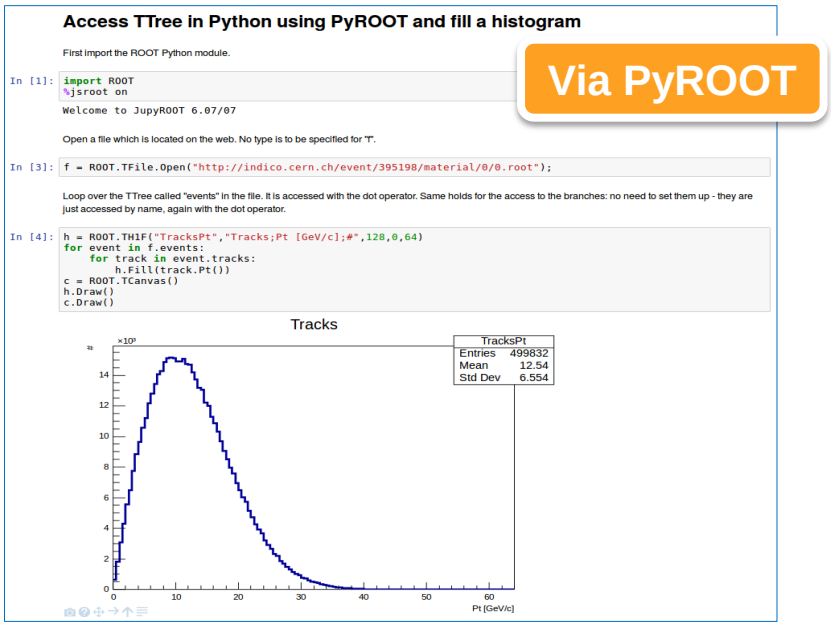
ROOT flavours for Jupyter

- Two language flavours (a.k.a. kernels) are available:

New in Jupyter!



Powered by the ROOT C++ interpreter



Via PyROOT

ROOT flavours for Jupyter (II)

- C++ and Python can be mixed in the same notebook
  - Thanks to the ROOT type system

Interleave Python with C++: the %%cpp magic

In [1]:
```
import ROOT
```
```
Welcome to JupyROOT 6.07/03
```

Thanks to its interpreter and type system, entities such as functions, classes and variables, created in a C++ cell, can be accessed from within Python.

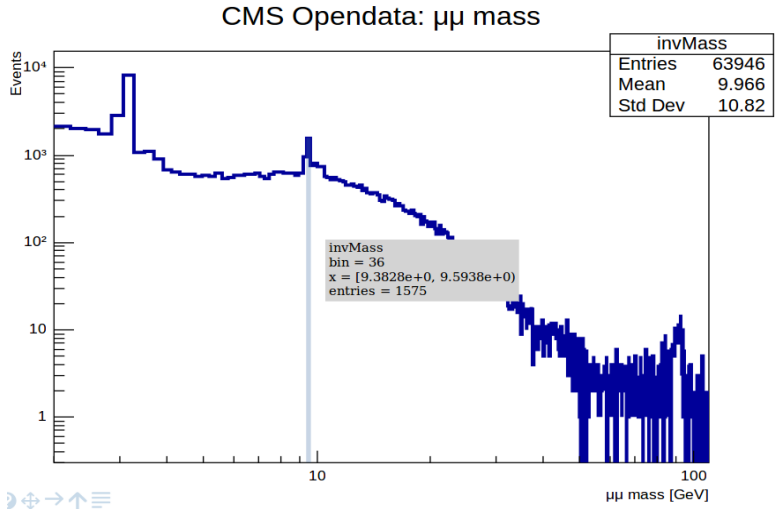
In [2]:
```
%%cpp
class A {
public:
    A() { cout << "Constructor of A!" << endl; }
};
```

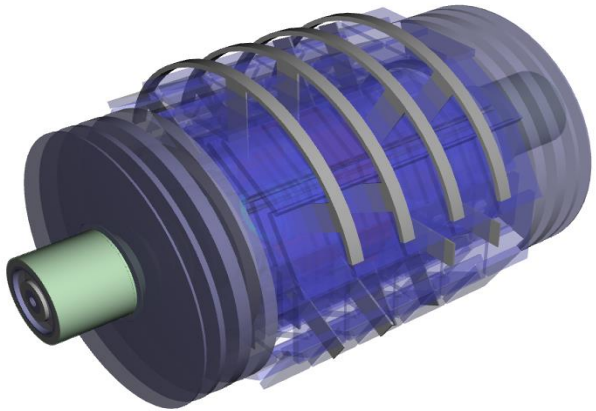
%%python also available in C++ notebooks

In [3]:
```
a = ROOT.A()
```
```
Constructor of A!
```

Interactive Graphics: JSROOT

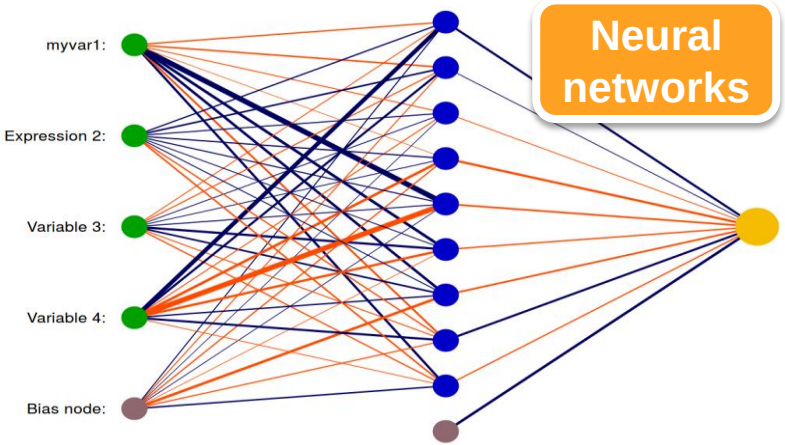
- Both of the presented flavours (C++, Python) allow to inline ROOT graphics in a notebook
- Two modes: static image and JavaScript visualisation
  - Activate JSROOT mode with **%jsroot on**
  - Interact with your plot: zoom, modify axis, inspect bins, …

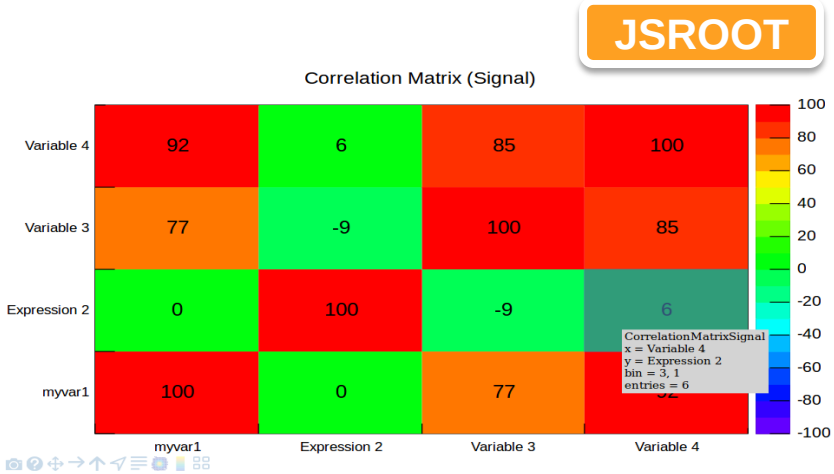




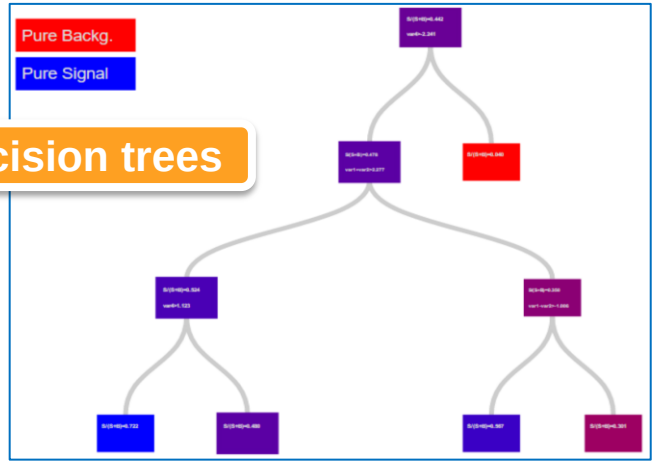
Interactive Machine Learning

- TMVA: machine learning toolkit in ROOT
  - Recently integrated with Jupyter as well: **%jsmva on**
  - JSROOT plots for input variables
  - Visualisation of neural networks and decision trees, DNN designer
  - Interactive training: stop a server computation
  - HTML output formatting

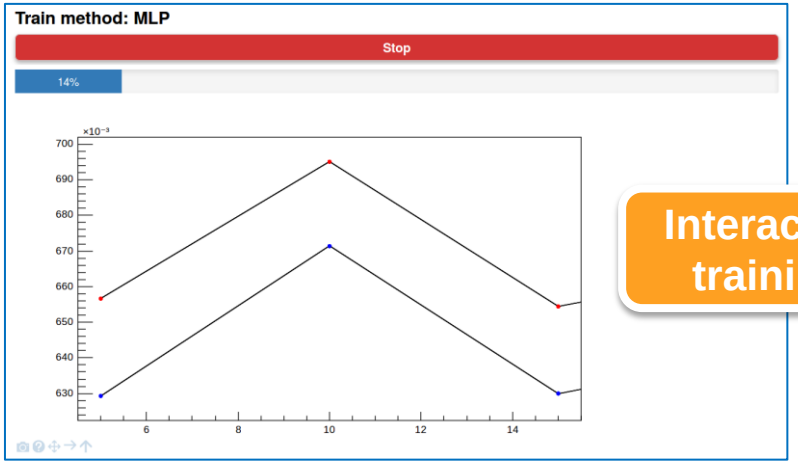




Interactive Machine Learning (II)



Decision trees



Interactive training

DataSetInfo	Correlation matrix (Signal)
DataSetInfo	Correlation matrix (Background)
DataSetFactory	**Dataset: tmva_class_example**
TFHandler_MLP	Variable | Mean | RMS | Min | Max myvar1 | 0.083989 | 0.36407 | -1.0000 | 1.0000 myvar2 | 0.0094778 | 0.27696 | -1.0000 | 1.0000 var3 | 0.080279 | 0.36720 | -1.0000 | 1.0000 var4 | 0.12986 | 0.39603 | -1.0000 | 1.0000
	Training Network
	Elapsed time for training with 6000 events : 4.45 sec
MLP	**Dataset: tmva_class_example** Evaluation of MLP on training sample (6000 events)
	Elapsed time for evaluation of 6000 events : 0.0187 sec
	Creating xml weight file: tmva_class_example/weights/TMVAClassification_MLP.weights.xml
	Creating standalone class: tmva_class_example/weights/TMVAClassification_MLP.class.C
	Write special histos to file: TMVA.root:/tmva_class_example/Method_MLP/MLP

HTML output

Try It Out! - Local Machine

Follow some simple instructions in:

https://root.cern.ch/how/how-create-rootbook

and…

Since 6.06

```
$ root --notebook
```

This command:

1. Starts a local notebook server
2. Connects to it via the browser

Provides a ROOT C++ kernel and the rest of ROOTbook goodies

Try It Out! - SWAN

SWAN: Data analysis “as a service”

https://swan.cern.ch

Interface: Jupyter Notebooks

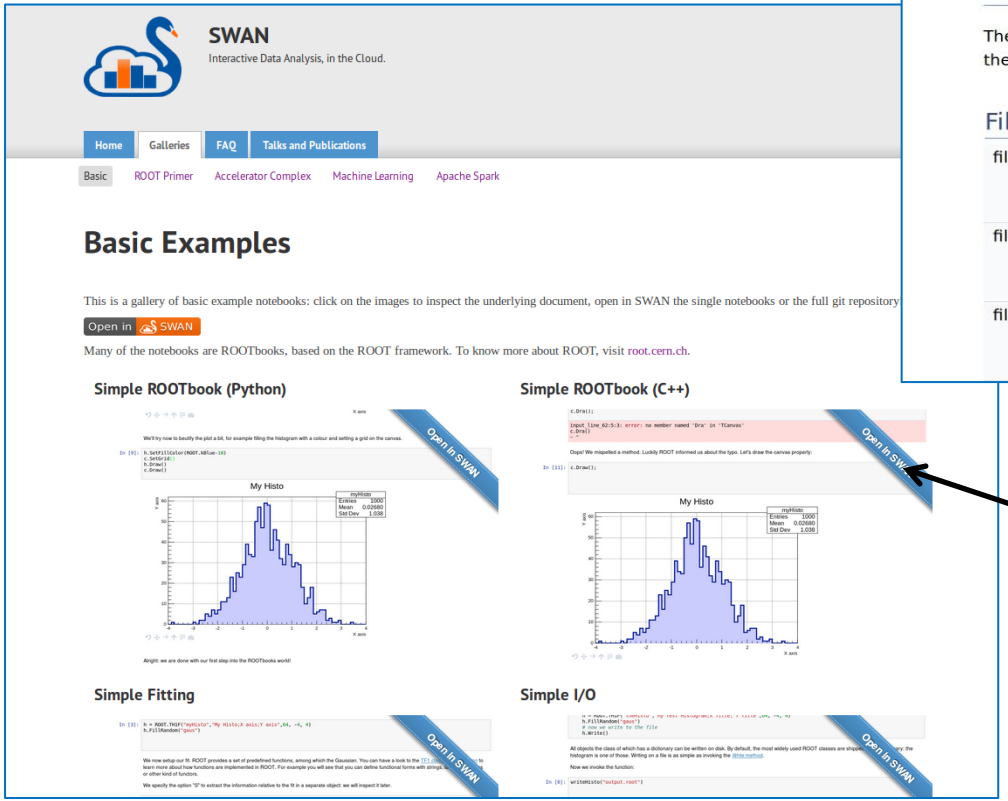
Goals:

- Analysis only with a web browser
  - Platform independent ROOT-based data analysis
- Calculations, input and results “in the Cloud”
  - Easy sharing of scientific results: plots, data, code
- Centrally-distributed software: CVMFS
  - Integration with other analysis ecosystems: R, Python, …

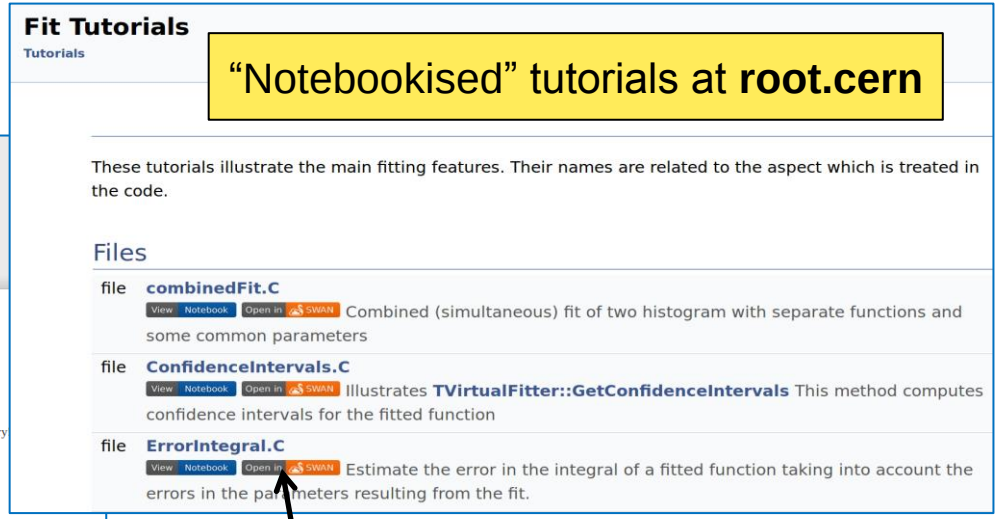
12/10, 11:45 – *E. Tejedor*
SWAN: a Service for Web-based Data Analysis in the Cloud

Notebook Gallery, Tutorials

Gallery of notebooks at **swan.web.cern.ch**



“Notebookised” tutorials at **root.cern**



Click to open in SWAN!

Summary

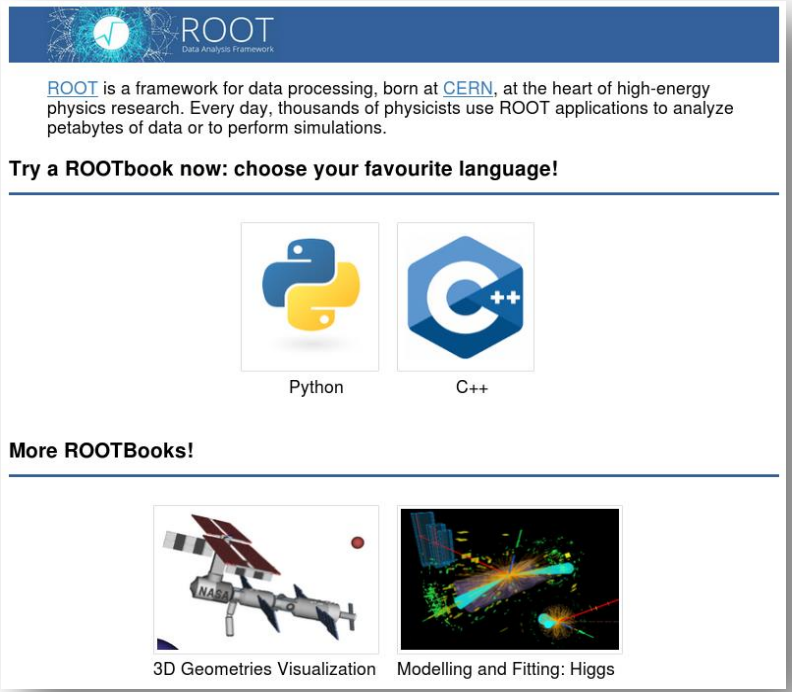
- ROOT integrated with Jupyter notebooks
  - C++ and Python notebook flavours
  - Inline graphics
  - JSROOT interactive visualisation
  - TMVA interactive features
  - Other goodies: tab completion, language mixing, …
- All available in the next ROOT release (6.08)
- Accessible online thanks to SWAN
  - https://swan.cern.ch

Backup

Try It Out! - ROOT Binder

binder

http://mybinder.org/repo/cernphsft/rootbinder



Anonymous access

View, Create and Run ROOTbooks!