

# 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

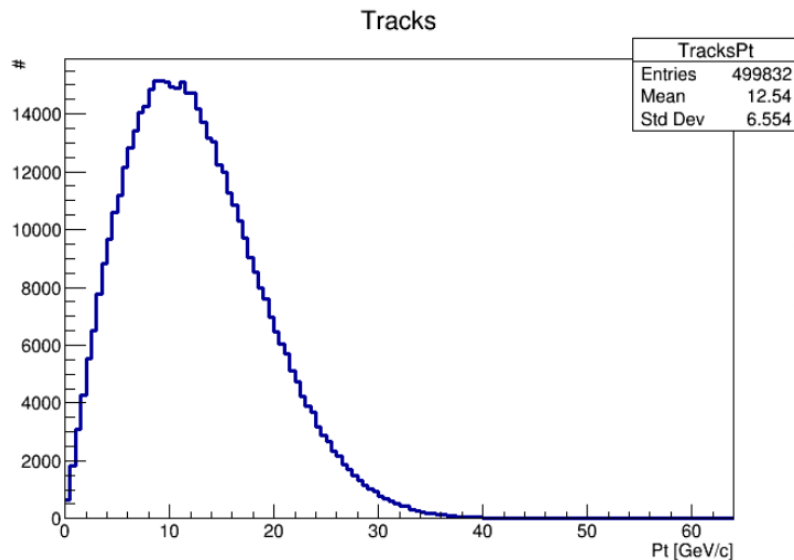
Graphics

## 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()
```



In a Browser



# 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!



```
...
// set initial values for fit
// set 2D function");
f2.Fit("f2");
FCN=517.445 FROM MIGRAD STATUS=CONVERGED 38 CALLS 39 TOTAL
EDM=2.65702e-12 STRATEGY= 1 ERROR MATRIX ACCURATE
EXT PARAMETER STEP FIRST
NO. NAME VALUE ERROR SIZE DERIVATIVE
1 p0 6.81725e-01 4.37173e-01 2.40425e-05 8.08231e-04
2 p1 1.46084e+00 9.36798e-01 5.15197e-05 3.78774e-04
```

Configure the canvas for plotting the result.

```
In [4]: TCanvas c1;
f2.SetLineWidth(1);
f2.SetLineColor(kBlue - 5);
f2.Draw("Surf1");

auto Xaxis = f2.GetAxis(); Xaxis->SetTitle("X Title"); Xaxis->
auto Yaxis = f2.GetAxis(); Yaxis->SetTitle("Y Title"); Yaxis->
auto Zaxis = f2.GetAxis(); Zaxis->SetTitle("Z Title"); Zaxis->
dte.Draw("PO Same");
```

Display the 2D graph in the notebook.

```
In [5]: c1.Draw();
```

Fitted 2D function

Powered by the ROOT C++ interpreter



## Access TTree in Python using PyROOT and fill a histogram

First import the ROOT Python module.

```
In [1]: import ROOT
%jsroot on
```

Welcome to JupyROOT 6.07/07

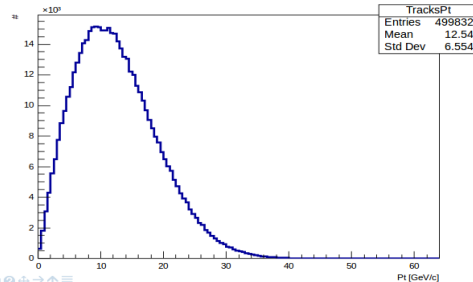
Open a file which is located on the web. No type is to be specified for "T".

```
In [3]: f = ROOT.TFile.Open("http://indico.cern.ch/event/395198/material/0/0.root");
```

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

```
In [4]: 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()
```

Tracks



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

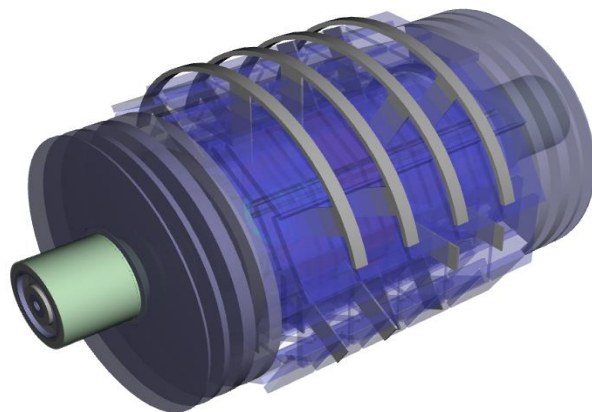
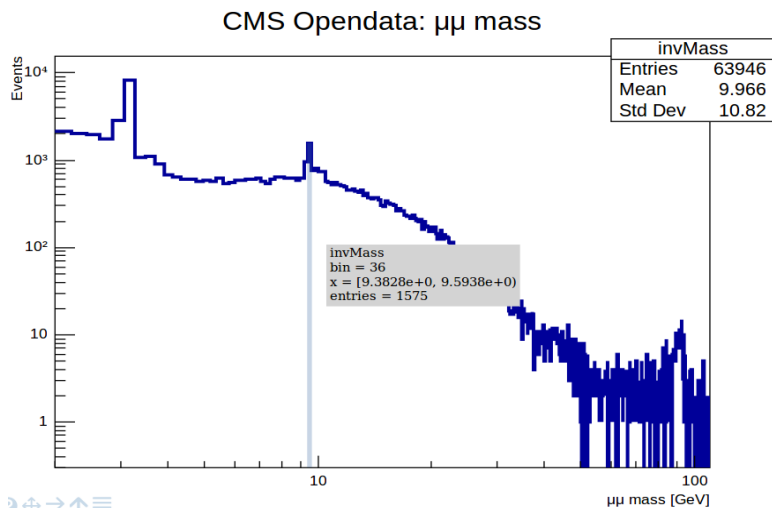
```
In [3]: a = ROOT.A()
```

Constructor of A!



# Interactive Graphics: JSROOT

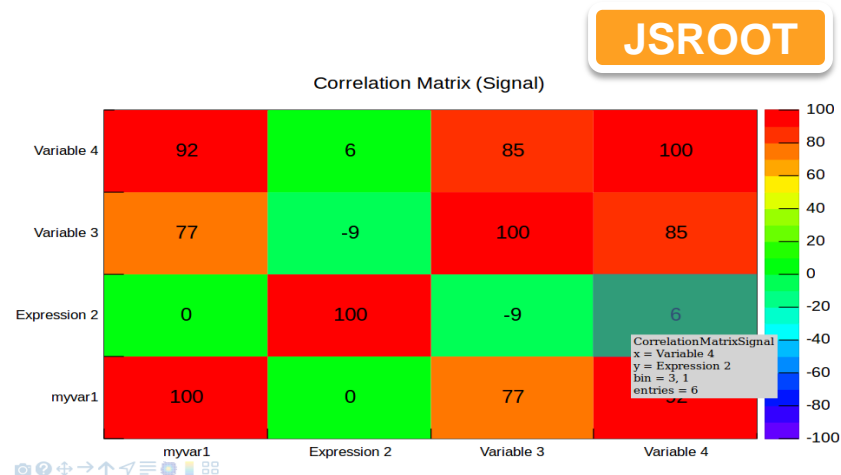
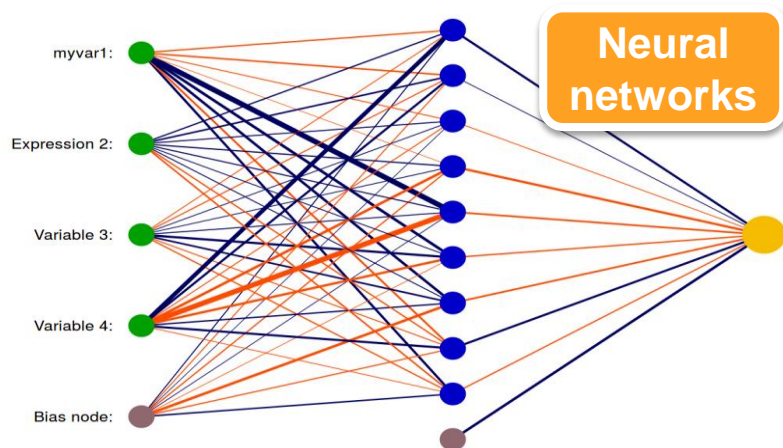
- Both of the presented flavours (C++, Python) allow to **inline ROOT graphics** in a notebook
- Two modes: static image and **JavaScript visualisation**
  - Activate **JSROOT** mode with **%jsroot on**
  - Interact with your plot: zoom, modify axis, inspect bins, ...





# 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

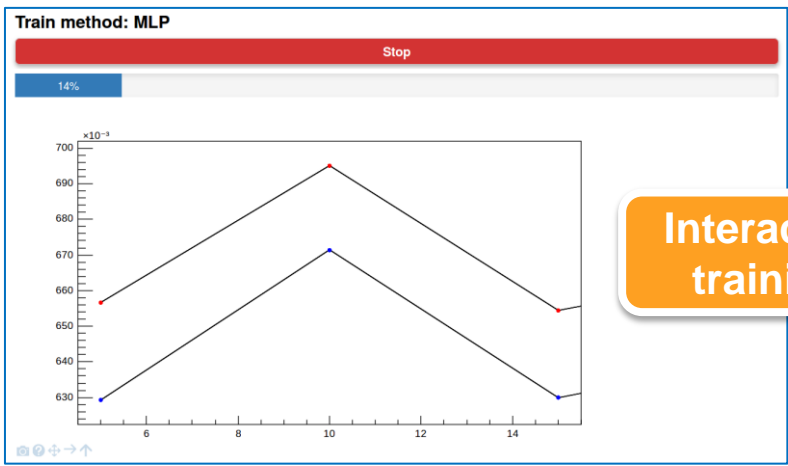
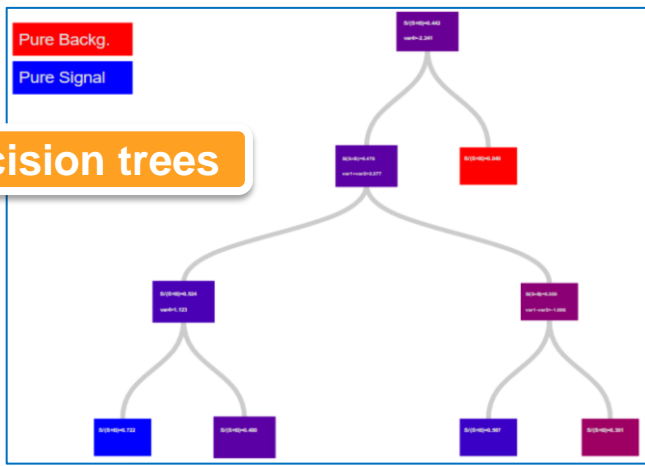






# Interactive Machine Learning (II)

Decision trees



Interactive training

<b>DataSetInfo</b>	Correlation matrix (Signal)																									
<b>DataSetInfo</b>	Correlation matrix (Background)																									
<b>DataSetFactory</b>	Dataset: <a href="#">tmva_class_example</a>																									
<b>TFHandler_MLP</b>	<table border="1"> <thead> <tr> <th>Variable</th> <th>Mean</th> <th>RMS</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>myvar1</td> <td>0.083989</td> <td>0.36407</td> <td>-1.0000</td> <td>1.0000</td> </tr> <tr> <td>myvar2</td> <td>0.0094778</td> <td>0.27696</td> <td>-1.0000</td> <td>1.0000</td> </tr> <tr> <td>var3</td> <td>0.080279</td> <td>0.36720</td> <td>-1.0000</td> <td>1.0000</td> </tr> <tr> <td>var4</td> <td>0.12986</td> <td>0.39603</td> <td>-1.0000</td> <td>1.0000</td> </tr> </tbody> </table> <p>Training Network            Elapsed time for training with 6000 events : 4.45 sec</p>	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
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																						
<b>MLP</b>	Dataset: <a href="#">tmva_class_example</a> 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



## 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](http://swan.web.cern.ch)

“Notebookised” tutorials at [root.cern](http://root.cern)

## Fit Tutorials

Tutorials

These tutorials illustrate the main fitting features. Their names are related to the aspect which is treated in the code.

### Files

file **combinedFit.C**

[View Notebook](#) [Open in SWAN](#) Combined (simultaneous) fit of two histogram with separate functions and some common parameters

file **ConfidenceIntervals.C**

[View Notebook](#) [Open in SWAN](#) Illustrates **TVirtualFitter::GetConfidenceIntervals** This method computes confidence intervals for the fitted function

file **ErrorIntegral.C**

[View Notebook](#) [Open in SWAN](#) Estimate the error in the integral of a fitted function taking into account the errors in the parameters resulting from the fit.



SWAN

Interactive Data Analysis, in the Cloud.

[Home](#) [Galleries](#) [FAQ](#) [Talks and Publications](#)

[Basic](#) [ROOT Primer](#) [Accelerator Complex](#) [Machine Learning](#) [Apache Spark](#)

## Basic Examples

This is a gallery of basic example notebooks: click on the images to inspect the underlying document, open in SWAN the single notebooks or the full git repository

[Open in SWAN](#)

Many of the notebooks are ROOTbooks, based on the ROOT framework. To know more about ROOT, visit [root.cern.ch](http://root.cern.ch).

### Simple ROOTbook (Python)

### Simple ROOTbook (C++)

### Simple Fitting

### Simple I/O

Click to open in SWAN!



- 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



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




**ROOT**  
Data Analysis Framework


ROOT is a framework for data processing, born at [CERN](#), at the heart of high-energy physics research. Every day, thousands of physicists use ROOT applications to analyze petabytes of data or to perform simulations.

**Try a ROOTbook now: choose your favourite language!**

---




Python



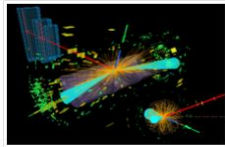
C++

**More ROOTBooks!**

---



3D Geometries Visualization



Modelling and Fitting: Higgs

**Anonymous  
access**

**View, Create and Run  
ROOTbooks!**