The new ROOT interface: Jupyter Notebooks

E. Tejedor, D. Piparo, P. Mató

L. Mascetti, J. Moscicki, M. Lamanna

CERN

CHEP

/10/2016

Prelude: The Notebook

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

jupyter

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"

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

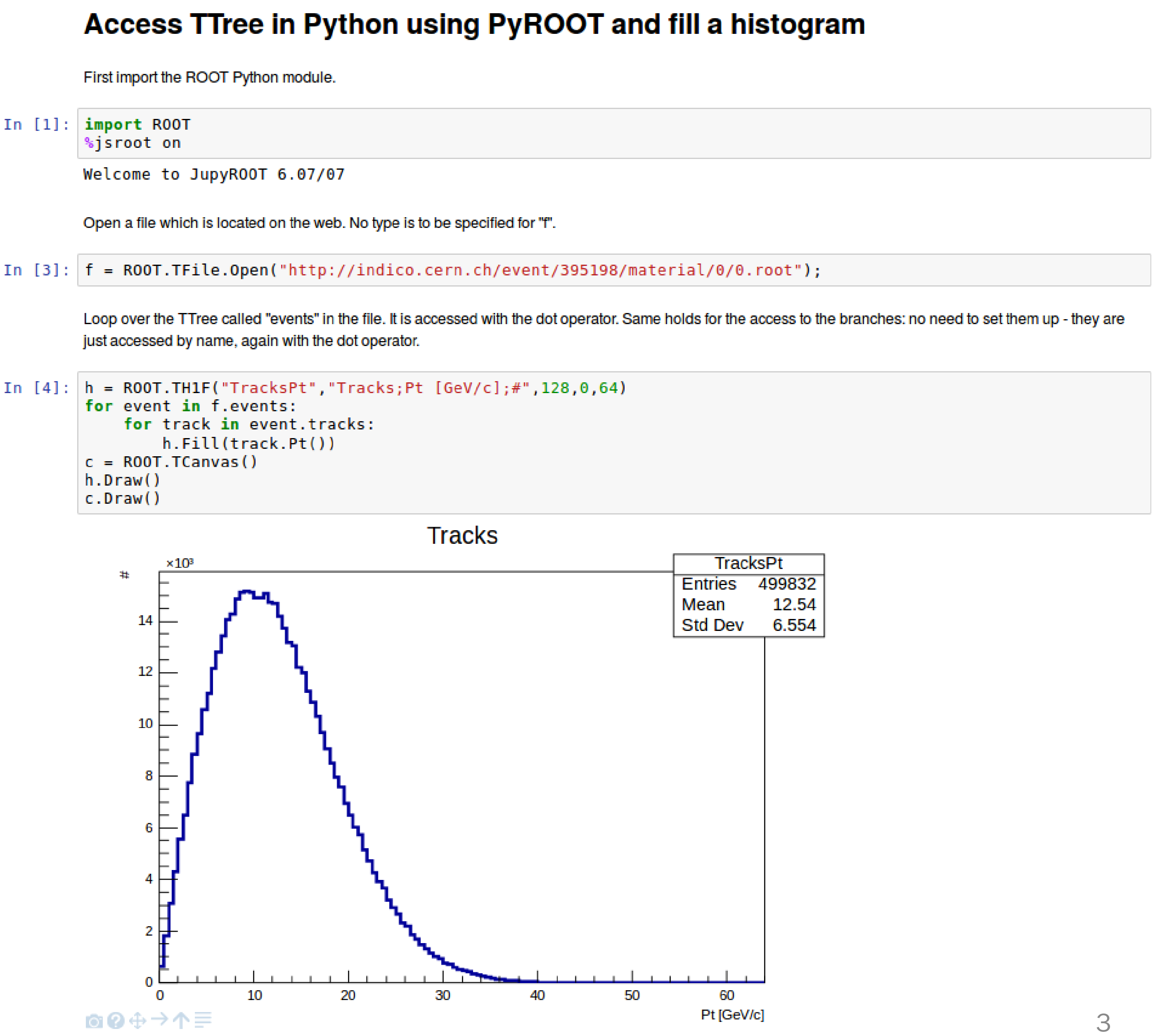
The Notebook: An Example

In a Browser

Text

Code

Graphics



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 "f".

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

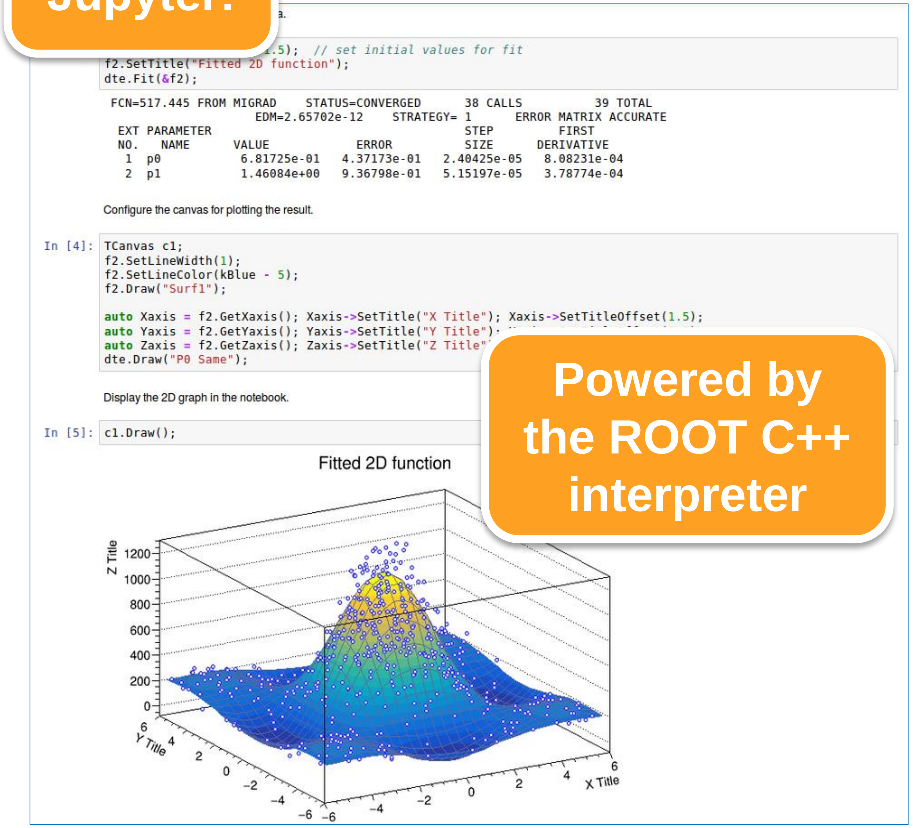
Loop over the TTree called "events" in the file. It is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

```
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()
```

ROOT flavours for Jupyter

- ROOT has been fully integrated with the Jupyter technology
- Two language flavours (a.k.a. kernels) are available:

New in Jupyter!



a.

```
.5);  // set initial values for fit
f2.SetTitle("Fitted 2D function");
dte.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.

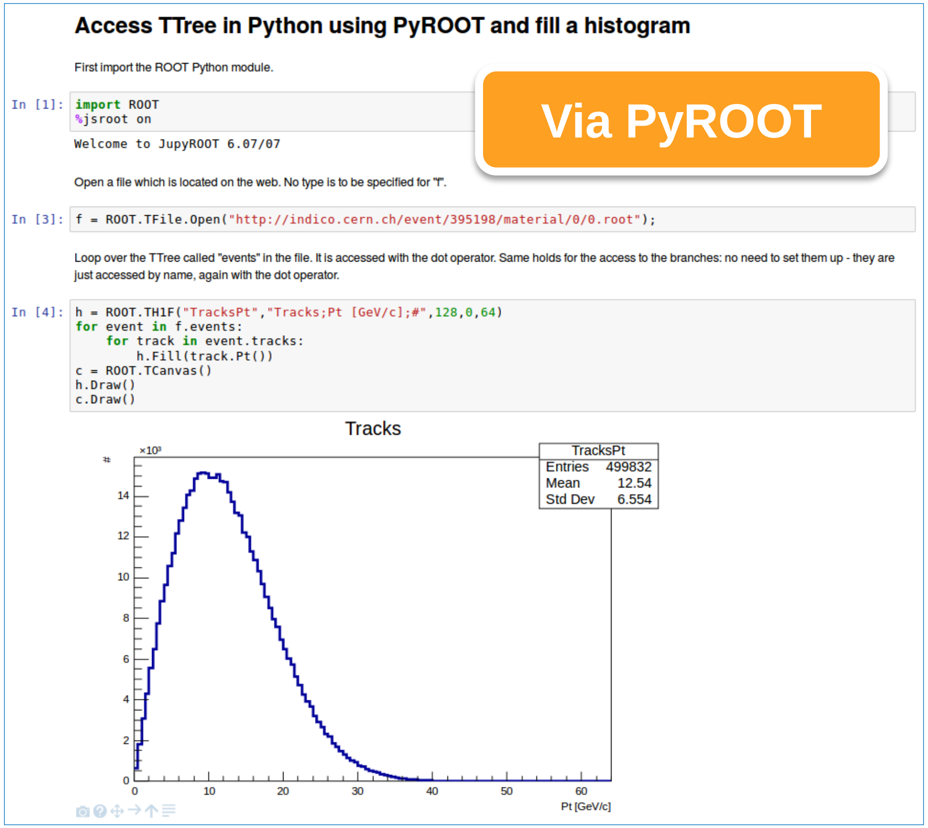
```
In [4]: TCanvas c1;
        f2.SetLineWidth(1);
        f2.SetLineColor(kBlue - 5);
        f2.Draw("Surf1");

        auto Xaxis = f2.GetXaxis(); Xaxis->SetTitle("X Title"); Xaxis->SetTitleOffset(1.5);
        auto Yaxis = f2.GetYaxis(); Yaxis->SetTitle("Y Title");
        auto Zaxis = f2.GetZaxis(); Zaxis->SetTitle("Z Title"
        dte.Draw("P0 Same");
```

Display the 2D graph in the notebook.

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

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 "f".

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

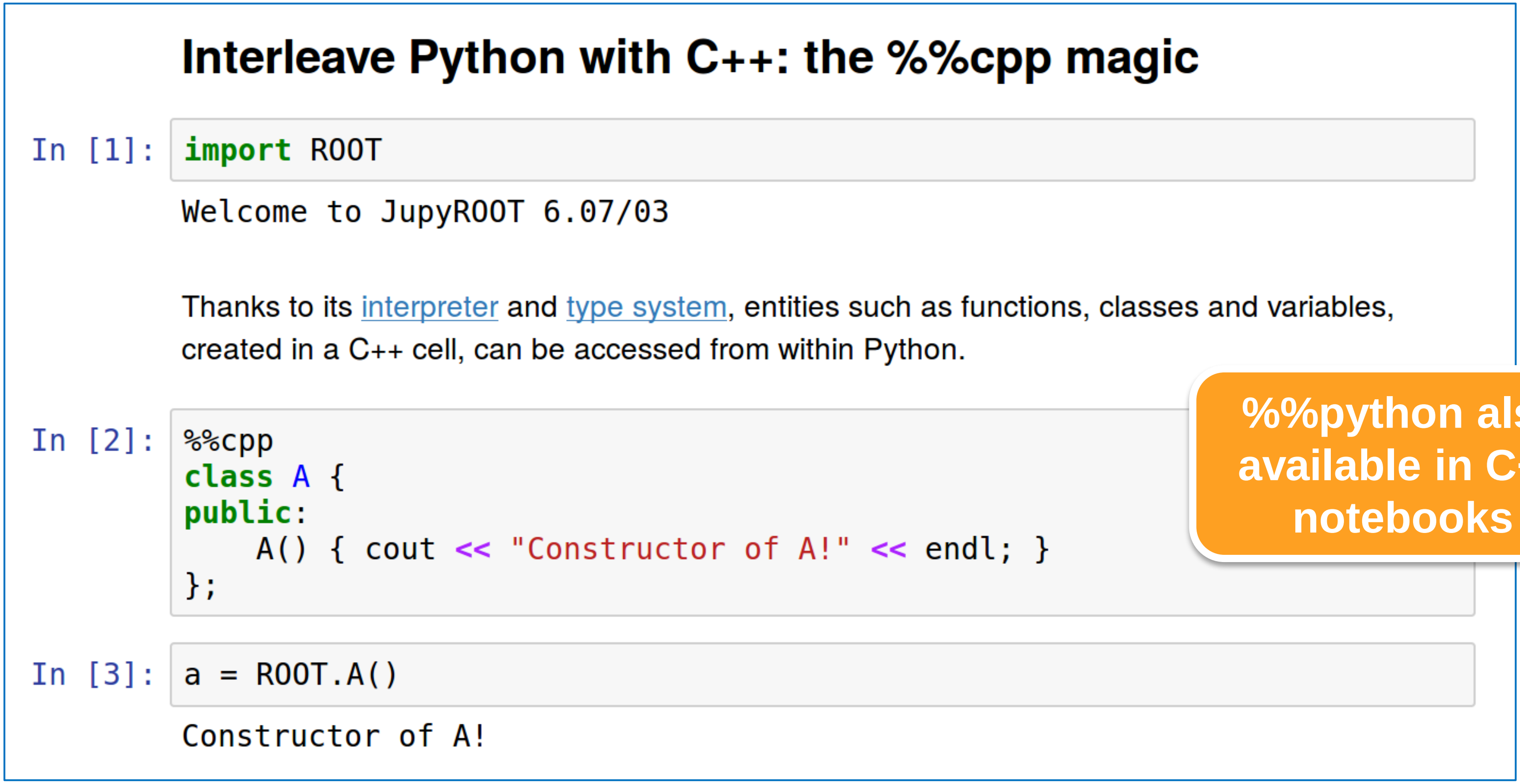
Loop over the TTree called "events" in the file. It is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

```
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()
```

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; }
        };
```

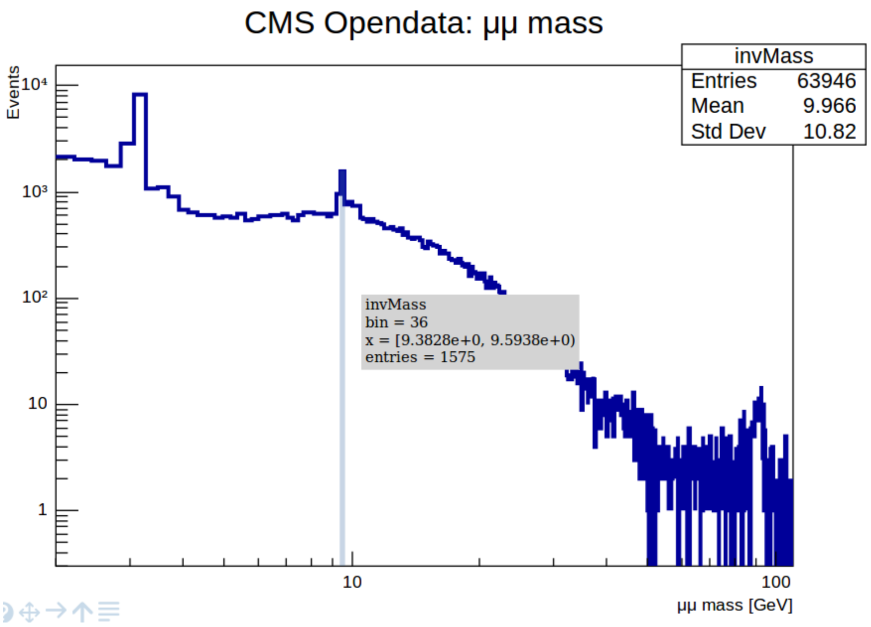
```
In [3]: a = ROOT.A()
```

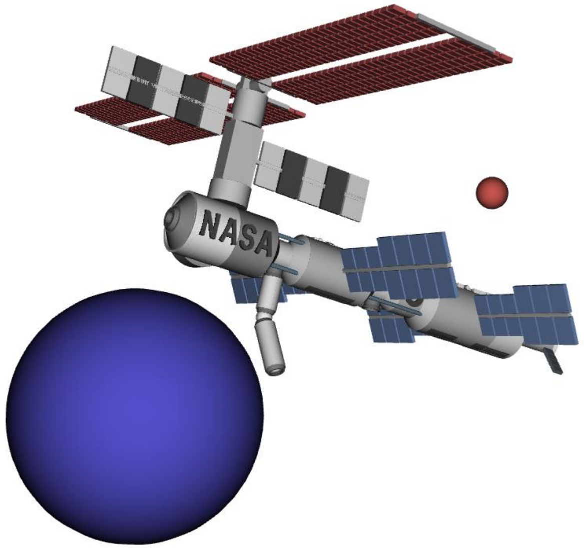
```
Constructor of A!
```

%%python also available in C++ notebooks

Interactive Graphics: JSROOT

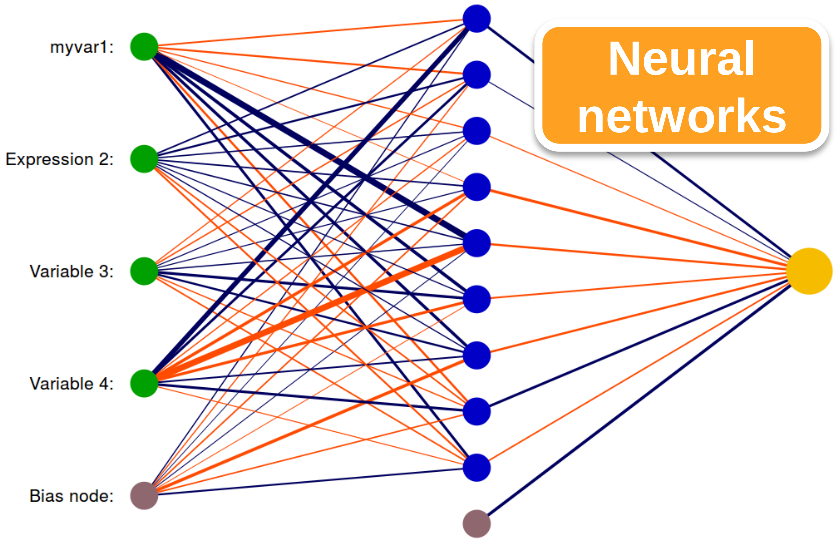
- Both of the presented flavours (C++, Python) allow to inline ROOT graphics in a notebook
- Two modes: static image and JavaScript visualisation
  - Activate JSROOT mode with **%jsroot on**
  - Interact with your plot: zoom, modify axis, inspect bins, …



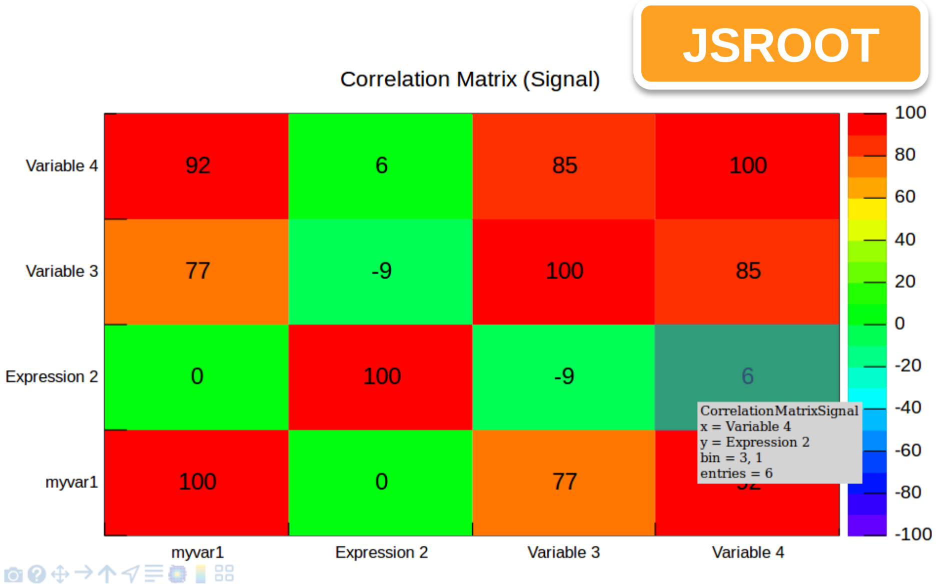


Interactive Machine Learning

- 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



Neural networks



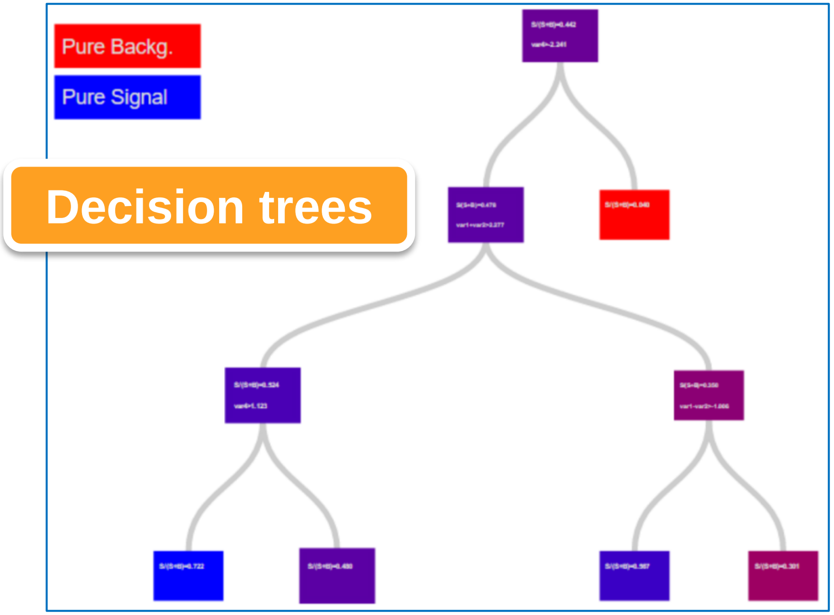
JSROOT

Correlation Matrix (Signal)

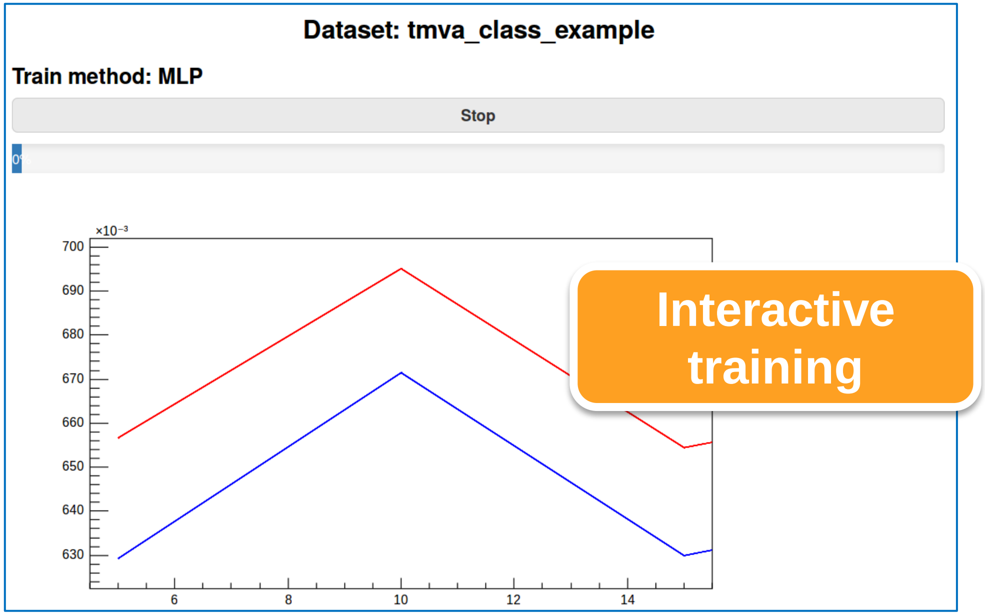
	myvar1	Expression 2	Variable 3	Variable 4
Variable 4	92	6	85	100
Variable 3	77	-9	100	85
Expression 2	0	100	-9	6
myvar1	100	0	77	92

CorrelationMatrixSignal
x = Variable 4
y = Expression 2
bin = 3, 1
entries = 6

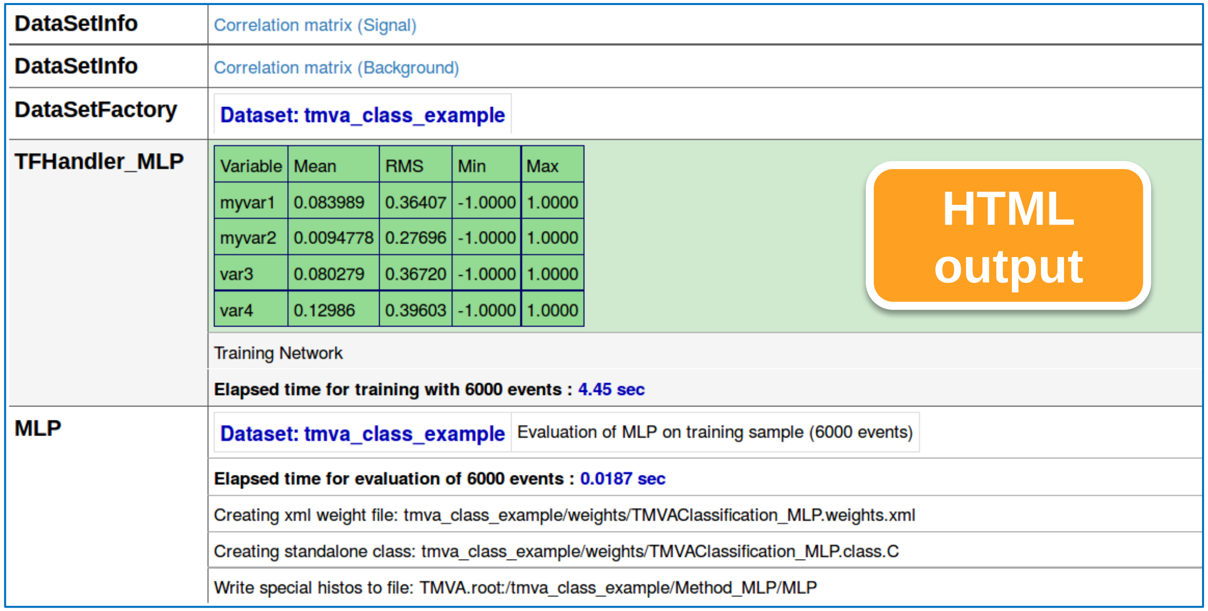
Interactive Machine Learning (II)



Decision trees



Interactive training



DataSetInfo	Correlation matrix (Signal)
DataSetInfo	Correlation matrix (Background)
DataSetFactory	**Dataset: tmva_class_example**
TFHandler_MLP	(variable table below) 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

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

HTML output

Try It Out! - Local Machine

Follow some simple instructions in:

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

and…

```
$ 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

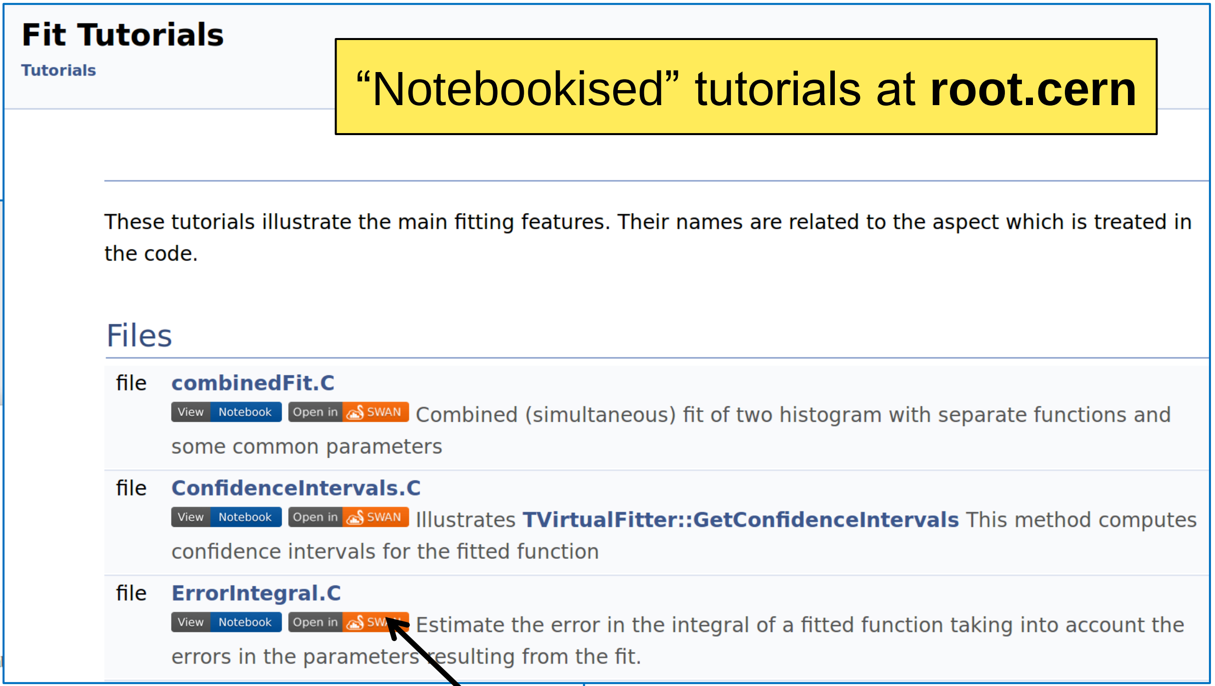
Goals:

- 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, …

Notebook Gallery, Tutorials

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

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



Fit Tutorials

Tutorials

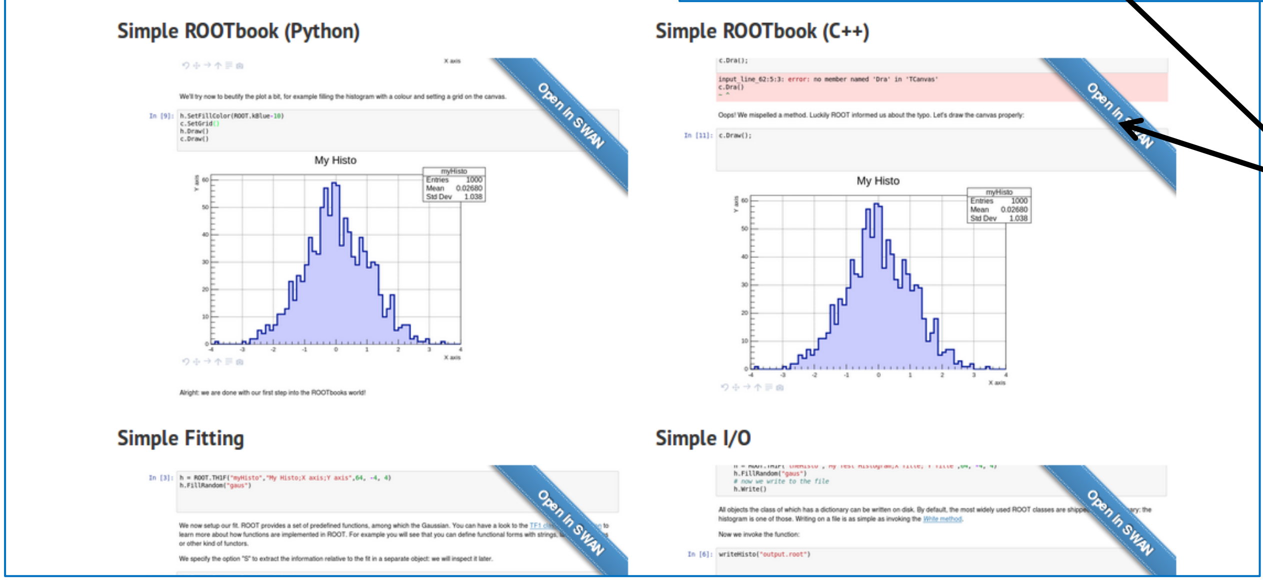
These tutorials illustrate the main fitting features. Their names are related to the aspect which is treated in the code.

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.

Basic Examples

This is a gallery of basic example notebooks: click on the images to inspect the underlying

Open in SWAN

Many of the notebooks are ROOTbooks, based on the ROOT framework. To know more a

Simple ROOTbook (Python)

We'll try now to beutify the plot a bit, for example filling the histogram with a colour and setting a grid on the canvas.

```
In [9]: h.SetFillColor(ROOT.kBlue-10)
        c.SetGrid()
        h.Draw()
        c.Draw()
```

Alright: we are done with our first step into the ROOTbooks world!

Simple ROOTbook (C++)

```
c.Dra();

input_line_62:5:3: error: no member named 'Dra' in 'TCanvas'
c.Dra()
~ ^
```

Oops! We mispelled a method. Luckily ROOT informed us about the typo. Let's draw the canvas properly:

```
In [11]: c.Draw();
```

Simple Fitting

```
In [3]: h = ROOT.TH1F("myHisto","My Histo;X axis;Y axis",64, -4, 4)
        h.FillRandom("gaus")
```

We now setup our fit. ROOT provides a set of predefined functions, among which the Gaussian. You can have a look to the TF1 cl[illegible] to learn more about how functions are implemented in ROOT. For example you will see that you can define functional forms with strings, [illegible] or other kind of functors.

We specify the option "S" to extract the information relative to the fit in a separate object: we will inspect it later.

Simple I/O

```
h.FillRandom("gaus")
# now we write to the file
h.Write()
```

All objects the class of which has a dictionary can be written on disk. By default, the most widely used ROOT classes are shippe[illegible]ary: the histogram is one of those. Writing on a file is as simple as invoking the Write method.

Now we invoke the function:

```
In [6]: writeHisto("output.root")
```

Click to open in SWAN!

Summary

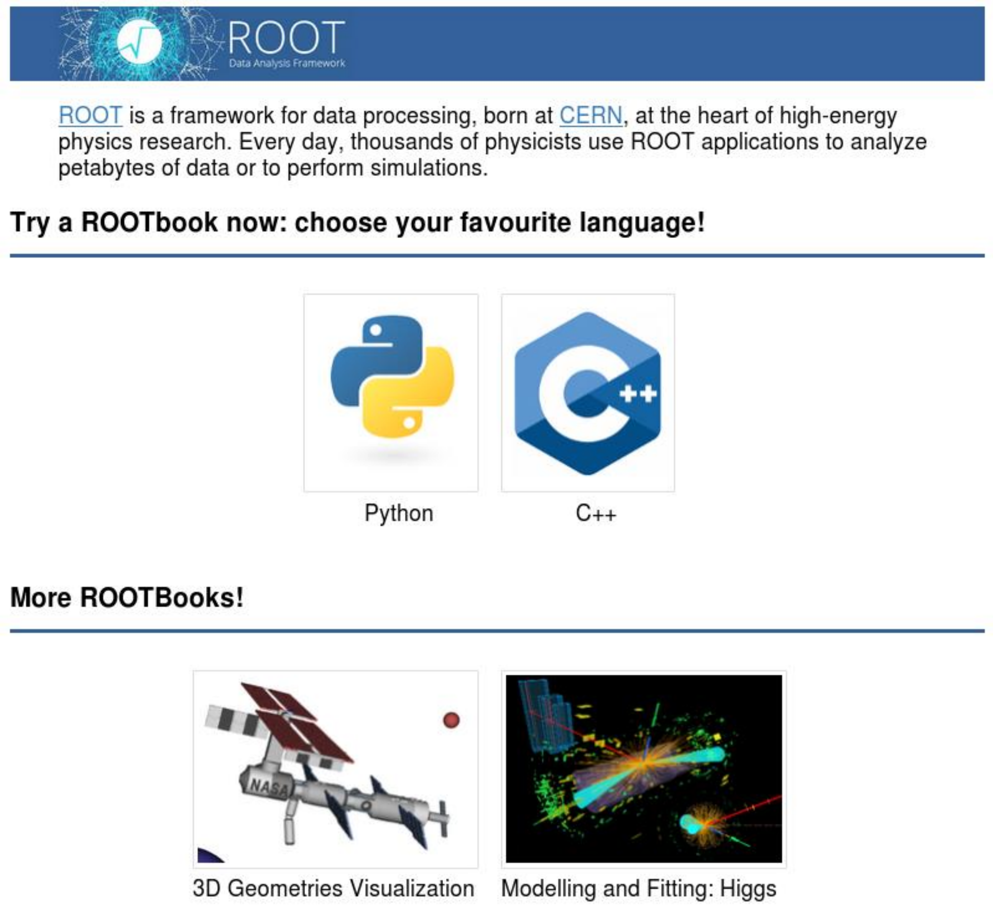
- ROOT integrated with Jupyter notebooks
  - C++ and Python notebook flavours
  - Inline graphics
  - JsROOT interactive visualisation
  - TMVA interactive features
  - Other goodies: tab completion, language mixing, …
- All available in the next ROOT release (6.08)
- Accessible online thanks to SWAN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

Backup

Try It Out! - ROOT Binder

binder

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



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!

More ROOTBooks!

Anonymous access

View, Create and Run ROOTbooks!