

Web-based interactive analysis environment (Wise)

Tao Lin <lintao@ihep.ac.cn>

May 10th - 15th , 2018

JUNO workshop @WHU

Outline

- Introduction and design
- Jupyter
- SWAN
- Summaries and Plans

Introduction

- Physics analysis in JUNO is a challenge.
 - PB-scale data every year.
 - Time correlated events.
- Two steps in analysis procedures (general analysis model)
 - Using framework to process data containing Event Data Model.
 - Using standalone program to process user data, fit data, produce plots.
- For physics users, the second step is quite important. Need the ability to analyze data interactively.
 - Easy to access or get data.
 - Fitting data and tuning parameters.
 - Visualization, or interactive detector simulation.

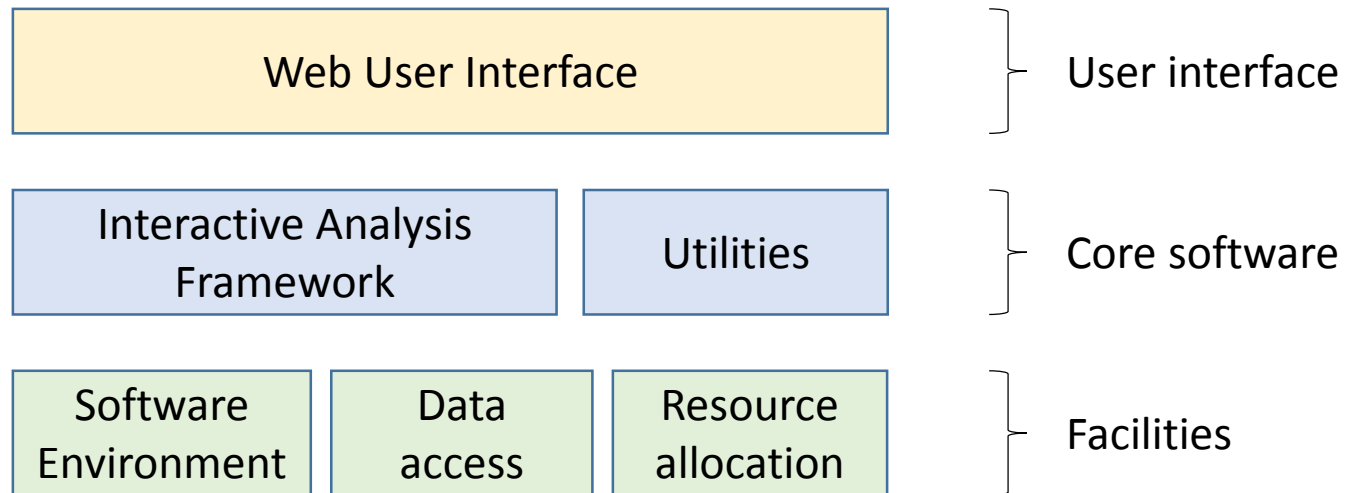
Challenges in interactive analysis

- Computing powers.
 - The current computing mode is that users develop in login node, and submit jobs to computing nodes.
 - The ability to do interactive analysis in login node is limited.
 - Not possible to use multiple nodes to accelerate data processing in interactive analysis.
 - Not easy to download the event data to local, run software locally.
- Software functionalities.
 - Not easy to access existing services in framework.
- New analysis technologies.
 - Machine learning and deep learning are popular.
 - Different software stacks. Such as CUDA, Python+TensorFlow.
 - How to help users?

“Analysis as a Service”

- To improve the abilities, “Analysis as a Service” is proposed by several communities. Such as
 - Jupyter and JupyterHub.
 - CERN’s SWAN (Service for Web based ANalysis)
- Adopt a cloud-based solution for JUNO analysis:
 - Elastic computing power and storage.
 - Flexible software configuration.
 - Integration with framework, new technologies.
 - Thin client. Just a web browser.
- Wise: Web-based interactive analysis environment
 - Simplify the analysis within this environment.
 - Not start from scratch, but based on existing Jupyter, SWAN and so on.

Designs



Need to consider:

- Web-based software and data
- Resource allocation and management
- Interactive analysis framework
- Monitoring and validation utilities

Based on following:

- Jupyter: UI and kernel
- SWAN: Architecture.
- ROOT6: Libraries.
- SNIPEr: Framework

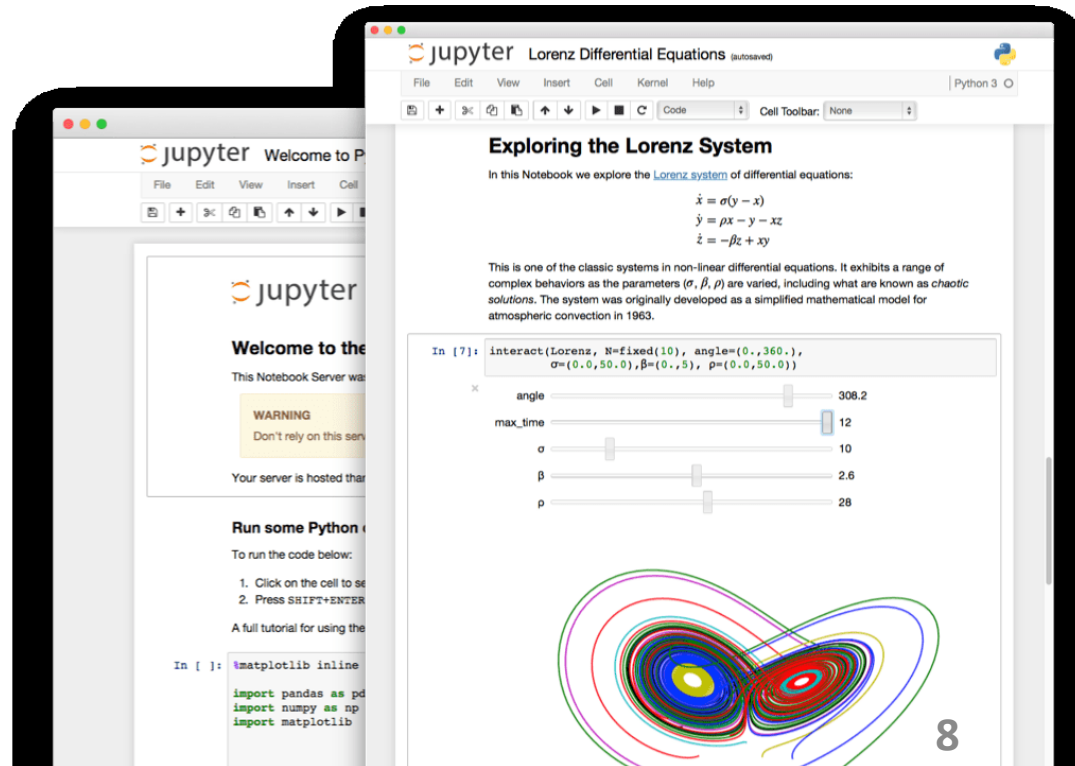


Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

<https://jupyter.org/>

Jupyter notebook interface

- Web-based.
- It combines code, text, figures in the same document.
 - Multiple programming language
 - Markdown support.
 - Big data integration
- Widely used in
 - Python community
 - Data analysis
 - Machine learning



Jupyter notebook

Currently in use at



Jupyter notebook

Simple spectral analysis

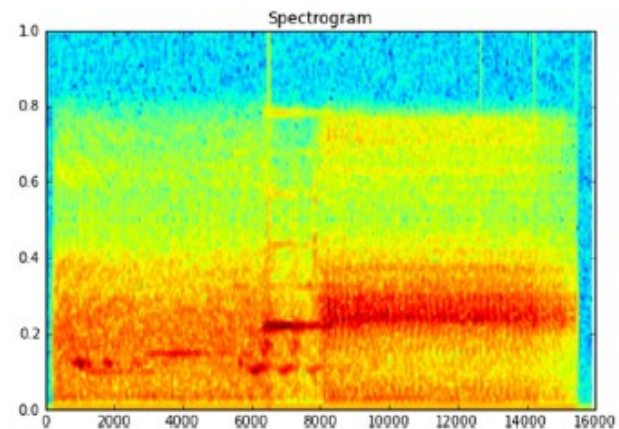
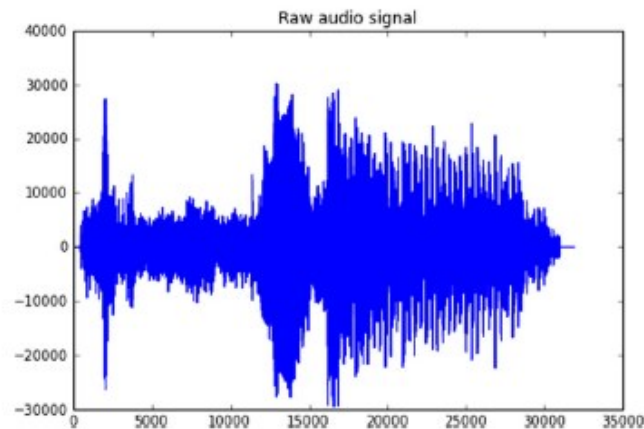
An illustration of the [Discrete Fourier Transform](#)

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(\frac{-2\pi i}{N} kn\right) \quad k = 0, \dots, N-1$$

```
In [2]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```



nbview: online viewer

nbviewer

A simple way to share Jupyter Notebooks

Enter the location of a Jupyter Notebook to have it rendered here:

URL, such as Github repo



NotebookPrimer / notebooks

NotebookPrimer

Name

← root-project's repository

data

notebooks

→ README.md

Name

← ..

images

1-Motivation-and-Introduction.ipynb

2-ROOT-Basics.ipynb

3-ROOT-Macros.ipynb

2 ROOT Basics

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

Now that you have installed ROOT, what is this interactive shell thing you're running? It is like this: that you can run from the command line or like other applications. But it is also an interactive shell that is extremely useful for debugging, quick hacking and testing. In the [notebook environment](#) you will have from your browser. Let us first have a look at some very simple examples.

2.1 ROOT as a calculator

You can even use the ROOT interactive shell instead of a calculator by launching the ROOT interactive shell

```
root
```

on your Linux box. The prompt should appear shortly. Below you will find some examples:

```
In [2]: 1+1
```

```
(int) 2
```



SWAN

SWAN (Service for Web based ANalysis) is a platform to perform interactive data analysis in the cloud.

<https://swan.web.cern.ch/>

SWAN[#]

- A platform to perform interactive data analysis in the cloud.
- Analyze data without the need to install any software
- Jupyter notebook interface as well as shell access from the browser
- Use CERNBox as your home directory and synchronize your local user storage with the cloud
- Access experiments' and user data in the CERN cloud
- Share your work with your colleagues thanks to CERNBox
- Document and preserve science - create catalogues of analyses: encourage reproducible studies and learning by example

Examples

Simple ROOTbook (Python)

```
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!

Open in SWAN

Simple I/O

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

Before reading the object, we can check from the commandline the content of the file with the `ls` utility:

```
In [8]: !ls -l output.root
```

```
TH1F Apr 12 14:06 theHisto "My Test Histogram"
```

We see that the file contains one object of type TH1F, the name of which is theHisto and the title of which is My Test Histogram. Let's now use the ROOT interface to read it and draw it:

```
In [10]: inputFile = ROOT.TFile("output.root")
         h = inputFile.theHisto
         c = ROOT.TCanvas()
         h.Draw()
         c.Draw()
```

Open in SWAN

3D Visualisation

```
In [2]: auto topVolum = gGeoManager->GetTopoVolume();
        topVolum->Draw();
```

Open in SWAN

TMVA Basics

Plot ROC Curve

```
In [8]: %jsroot
        TCanvas *c1=factory->GetROCCurve(loader1);
        c1->Draw();
```

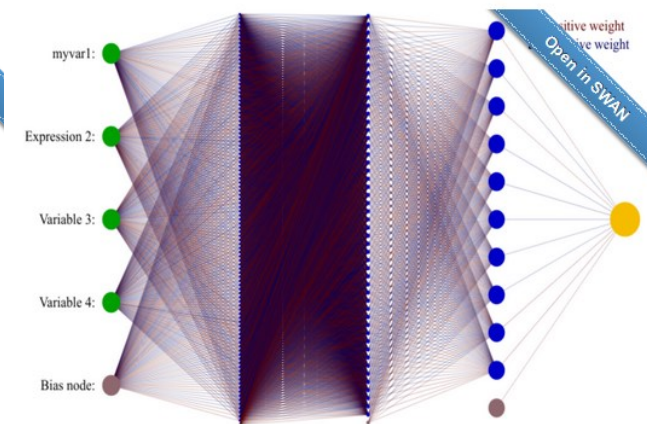
Open in SWAN

DNN

Background Rejection vs. Signal Efficiency

Open in SWAN

TMVA in Jupyter



Open in SWAN

demo: choose container

Spawner options

Select the contextualisation parameters for the container to spawn. See [the guide](#) for more details.

LCG release [more...](#)

84 SWAN1

Platform [more...](#)

x86_64-slc6-gcc49-opt

Environment script [more...](#)

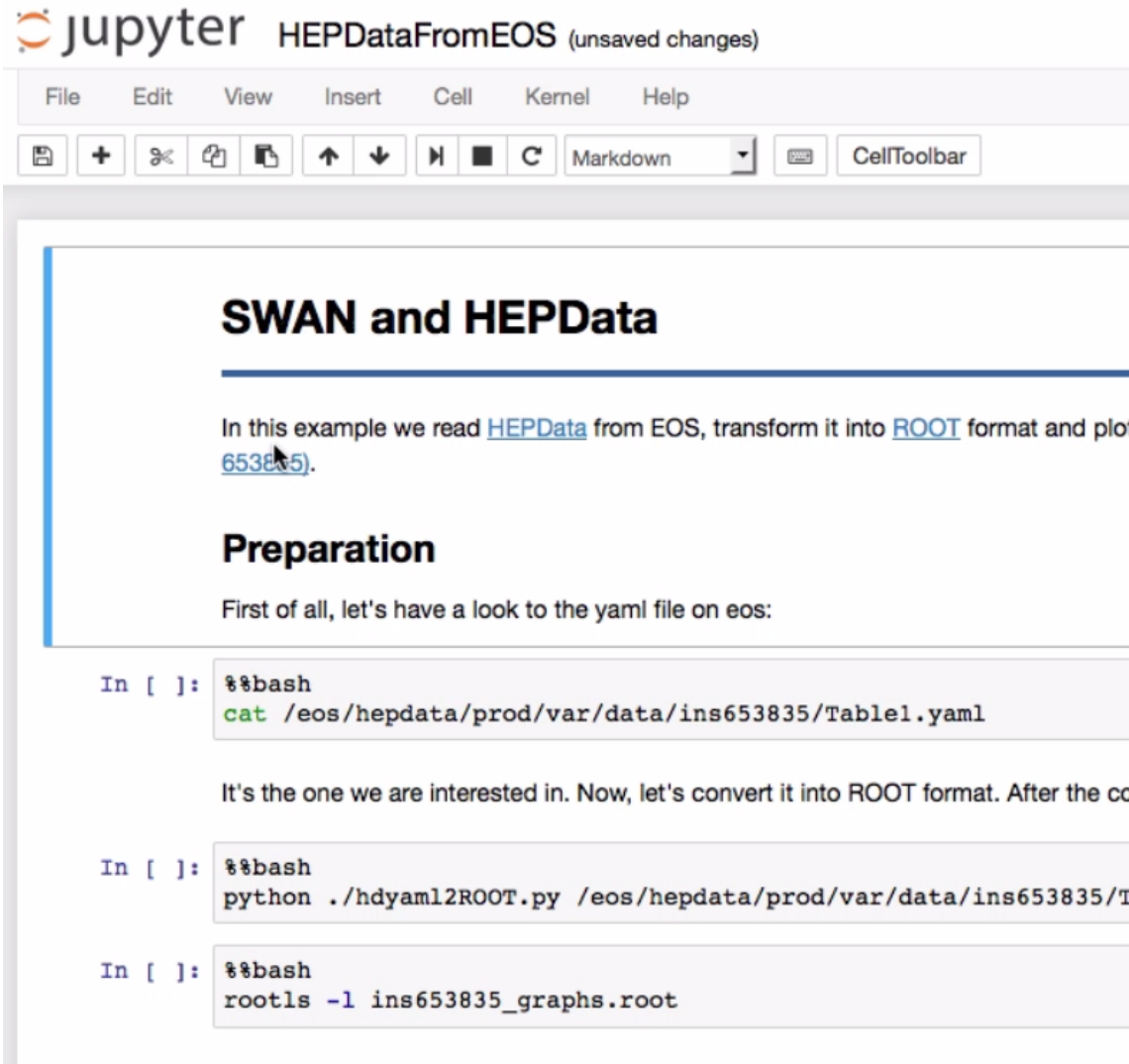
e.g. \$CERNBOX_HOME/MySWAN/myscript.sh

Number of cores [more...](#)

1

Spawn

demo: jupyter notebook



jupyter HEPDataFromEOS (unsaved changes)

File Edit View Insert Cell Kernel Help

📁 + 🔍 📄 ⬆️ ⬆️ ⏪ ⏹️ 🔄 Markdown 🗨️ CellToolbar

SWAN and HEPData

In this example we read [HEPData](#) from EOS, transform it into [ROOT](#) format and plot [653835](#).

Preparation

First of all, let's have a look to the yaml file on eos:

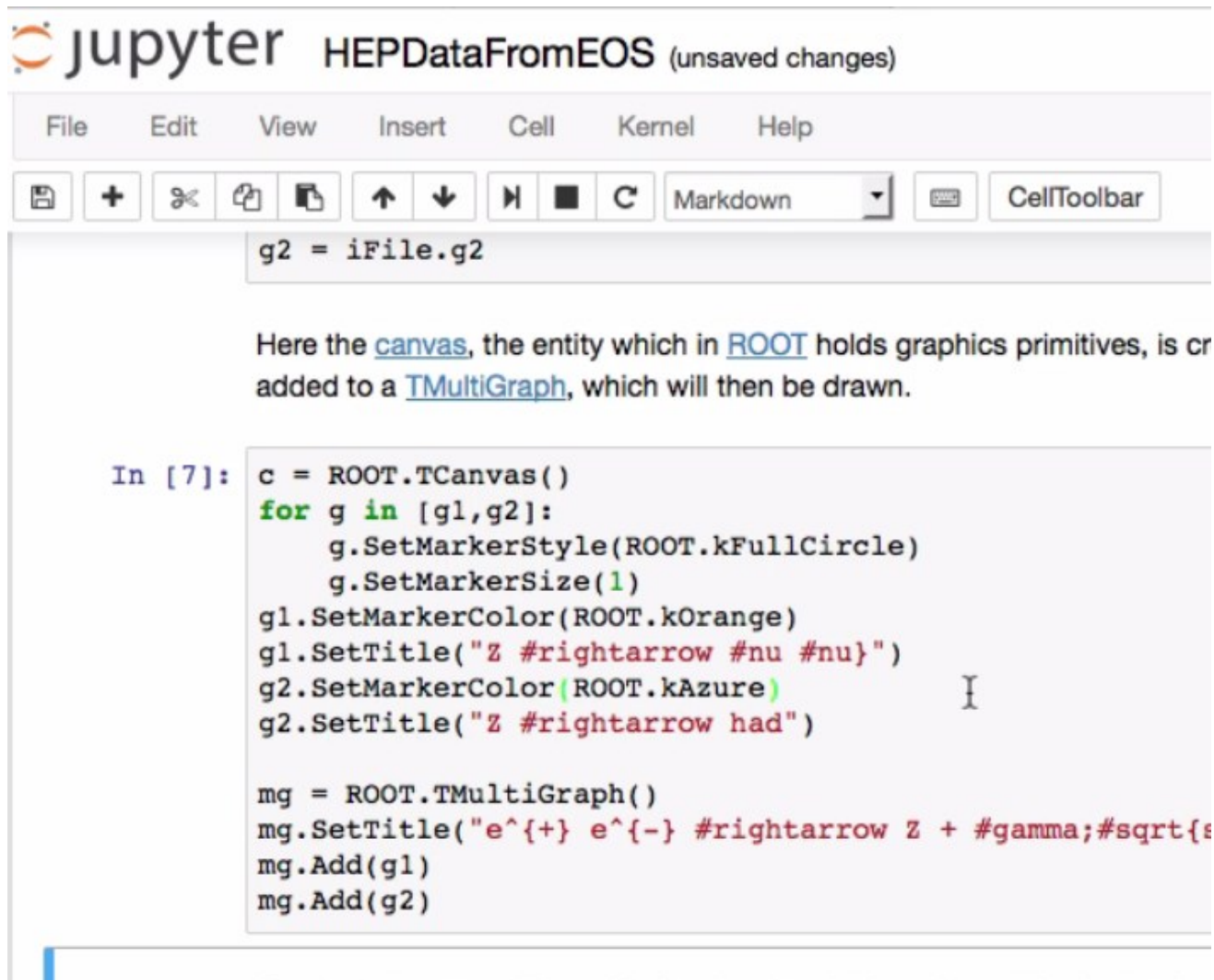
```
In [ ]: %%bash
cat /eos/hepdata/prod/var/data/ins653835/Table1.yaml
```

It's the one we are interested in. Now, let's convert it into ROOT format. After the co

```
In [ ]: %%bash
python ./hdyaml2ROOT.py /eos/hepdata/prod/var/data/ins653835/T
```

```
In [ ]: %%bash
rootls -l ins653835_graphs.root
```

demo: ROOT



The screenshot shows a Jupyter Notebook window titled "jupyter HEPDataFromEOS (unsaved changes)". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". Below the menu is a toolbar with icons for saving, adding cells, undo, redo, copy, paste, and navigation, along with a "Markdown" dropdown and a "CellToolbar" button.

The notebook contains two cells:

- A code cell with the following code:

```
g2 = iFile.g2
```
- A text cell containing the text: "Here the [canvas](#), the entity which in [ROOT](#) holds graphics primitives, is cr added to a [TMultiGraph](#), which will then be drawn."
- A code cell (labeled "In [7]:") with the following code:

```
c = ROOT.TCanvas()
for g in [g1,g2]:
    g.SetMarkerStyle(ROOT.kFullCircle)
    g.SetMarkerSize(1)
g1.SetMarkerColor(ROOT.kOrange)
g1.SetTitle("Z #rightarrow #nu #nu")
g2.SetMarkerColor(ROOT.kAzure)
g2.SetTitle("Z #rightarrow had")

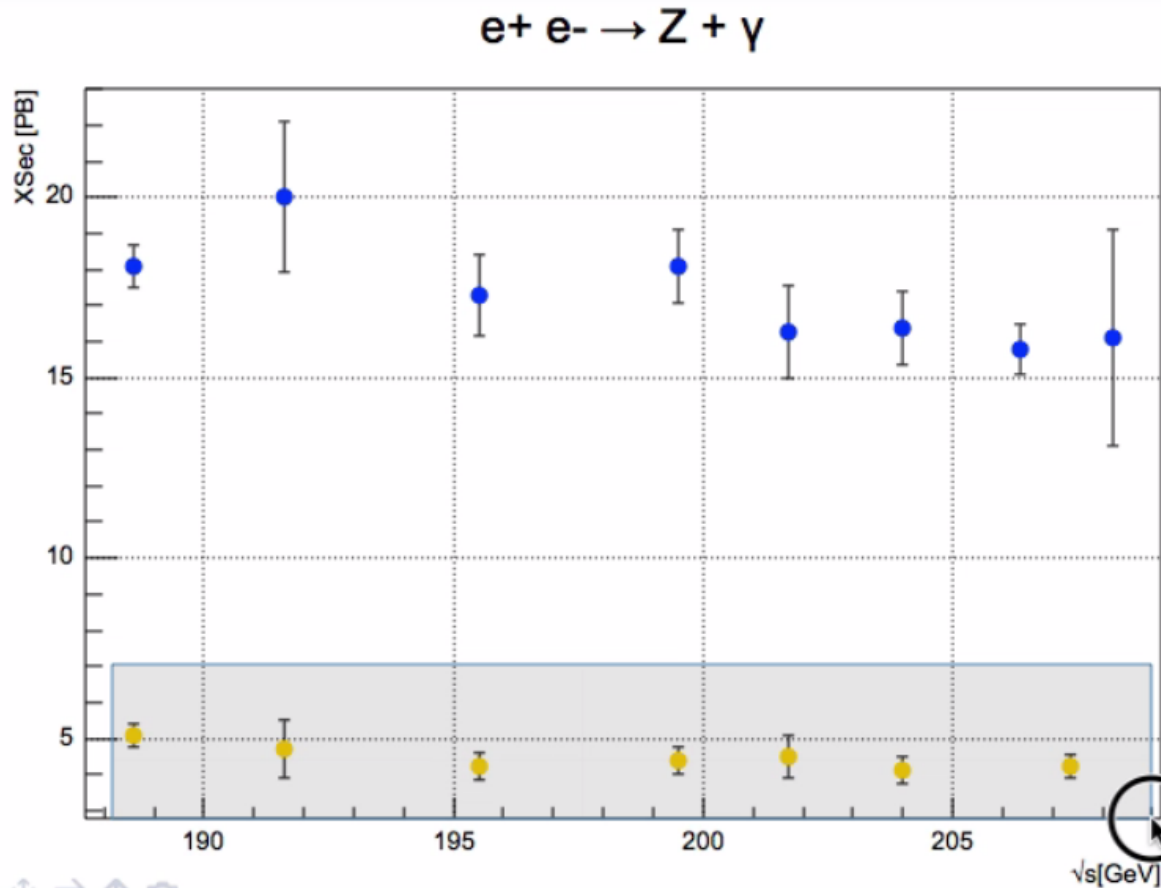
mg = ROOT.TMultiGraph()
mg.SetTitle("e^{+} e^{-} #rightarrow Z + #gamma;#sqrt{s}")
mg.Add(g1)
mg.Add(g2)
```

demo: interactive

jupyter HEPDataFromEOS (unsaved changes)

File Edit View Insert Cell Kernel Help

Markdown CellToolbar



demo: save results

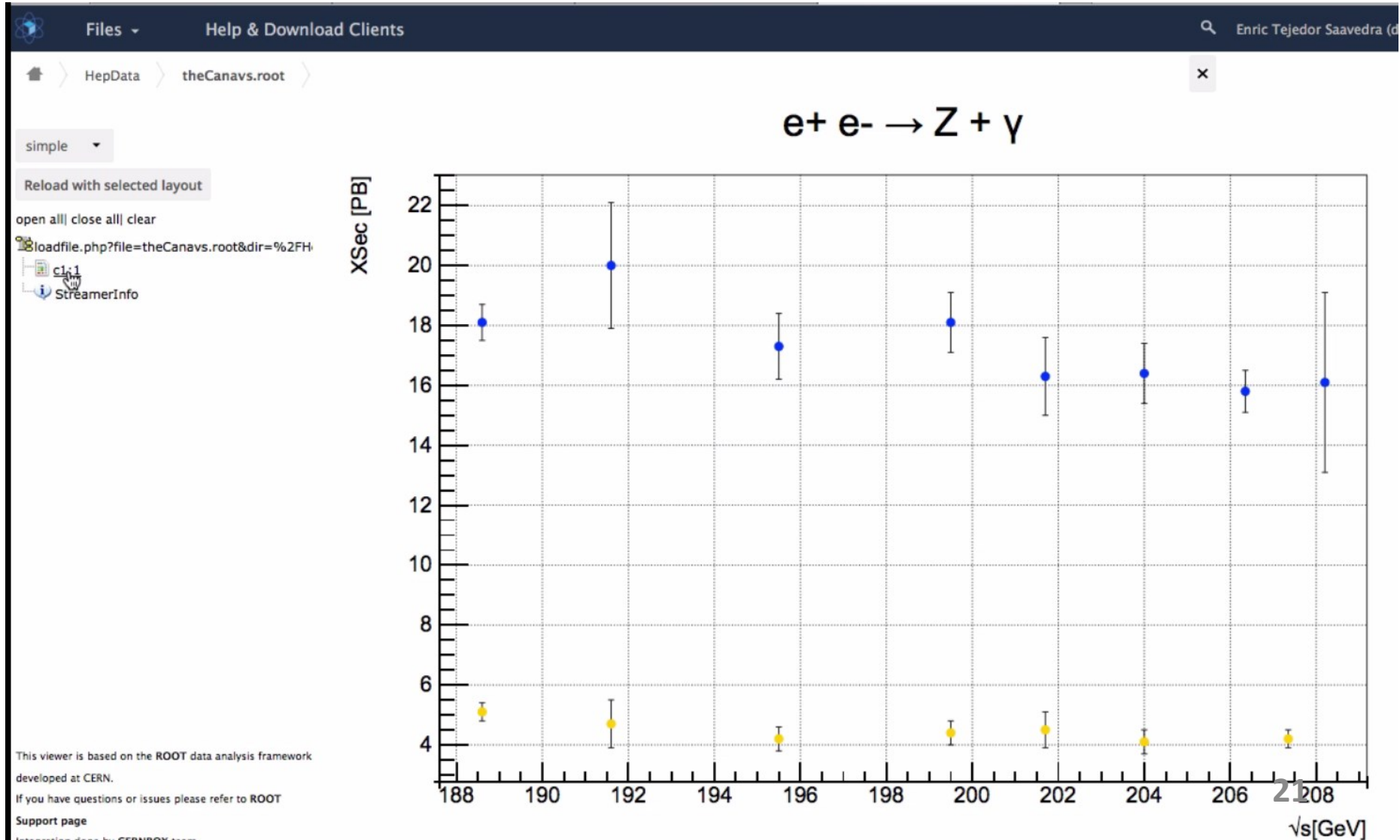
We now save our plot in pdf format on disk.

```
In [9]: c.Print("myStudy.pdf")
```

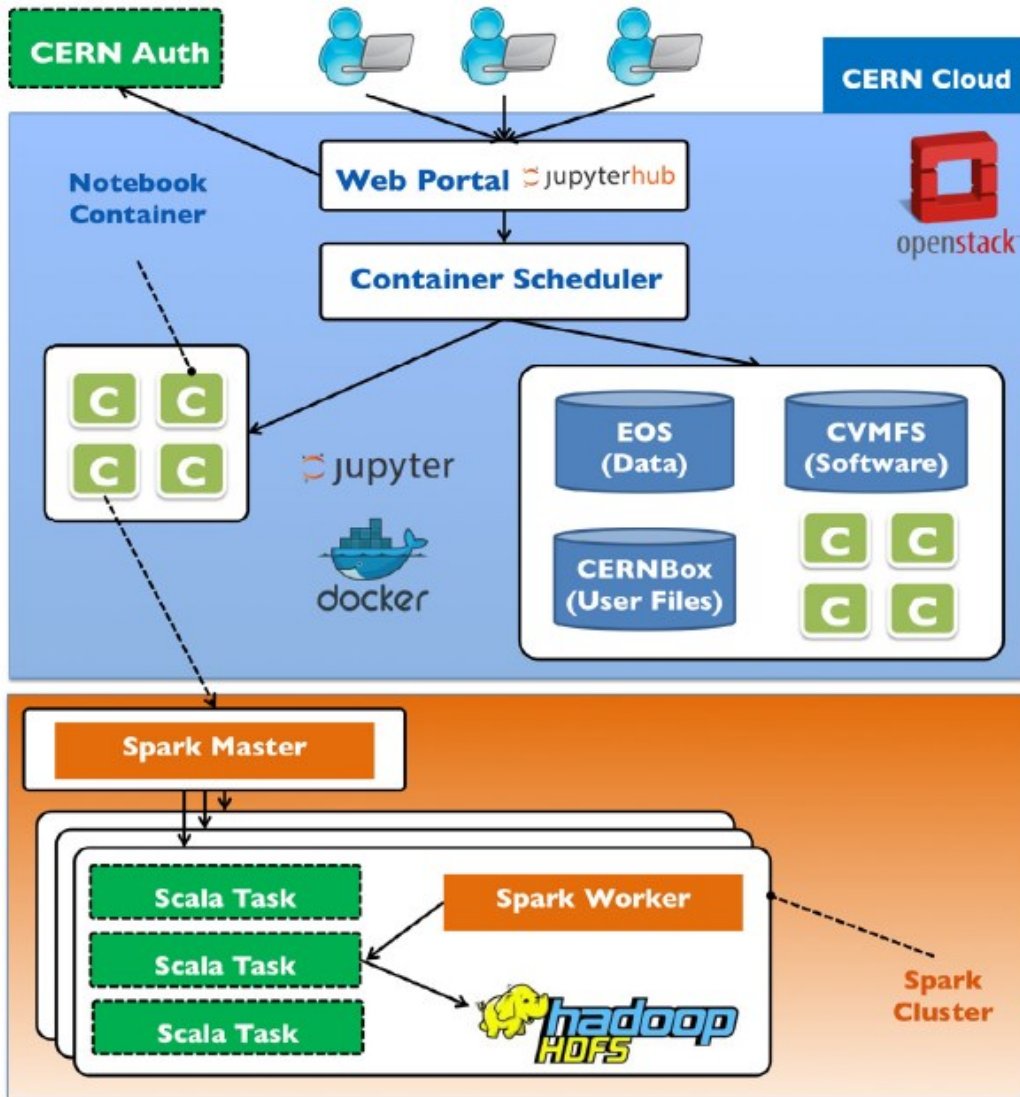
```
Info in <TCanvas::Print>: pdf file myStudy.pdf has been created
```

```
In [10]: f=ROOT.TFile("theCanavs.root", "RECREATE")  
         c.Write()  
         f.Close()
```

Demo: result in CERNBox



Architecture of SWAN



Authentication and security

Web portal

Virtualized infrastructure

Containers

Software distribution

Analysis software

Storage

Asynchronous data processing

- Spark
- HTCondor

Note: only available for CERN.

Summaries and Plans

- We propose a new project called Wise. It is “Analysis as a Service” for JUNO.
 - Based on Jupyter and SWAN.
 - Extend the software functionalities, such as SNIper framework support.
- A preliminary design
 - User interface is based on web.
 - A core software layer is used to integrate Jupyter and SNIper.
 - Underlying layer to supply software distribution, data access, computing resource allocation and so on.
- Plans
 - Setup a prototype before December in 2018.