Jupyter in HPC

Matthias Bussonnier

Feb 28th, 2018

About Me

Matthias Bussonnier – UC BIDS - @mbussonn/@carreau

- A Physicist
- Core developer of IPython/Jupyter since 2012
- Post doctoral Scholar on Jupyter

3 Parts

This webinar will be in 3 parts:

- Overview of what Jupyter is and typical use case
- Two case studies

Outline

- A bit of History (From IPython to Jupyter)
- What is Jupyter
- Why is Jupyter Popular
- What is Jupyter used for

From IPython to Jupyter

- 2001: Fernando Perez
  - Can replace bunch of C/C++/Make/Perl script with Python
  - Python REPL is pretty basic for Interactive use.
- Create IPython for Interactive Python.
  - prompt numbers.
  - gnuplot integration...

Two Programing "modes"

- Software engineer way:
  - Know what to write
  - Run it for long period of time
  - "Human" time small compared to CPU time.
- "Scientist" way:
  - Try add-hoc solution in a loop.
  - Update self-understanding of problem
  - repeat.
  - Human time greater than CPU

How Software Engineers see Scientists

I have no idea what I am doing

Exploratory programming

IPython was designed for exploratory programming, as a REPL (Read Eval Print Loop) and grew popular, especially among scientist who loved it to explore.

> IPython have weaponized the tab key
>
> – Fernando Pérez

Birth of the notebook

(Fast forward 2012)

- Decision to refactor IPython to make it "network enabled". Mature web technologies made it possible and attractive

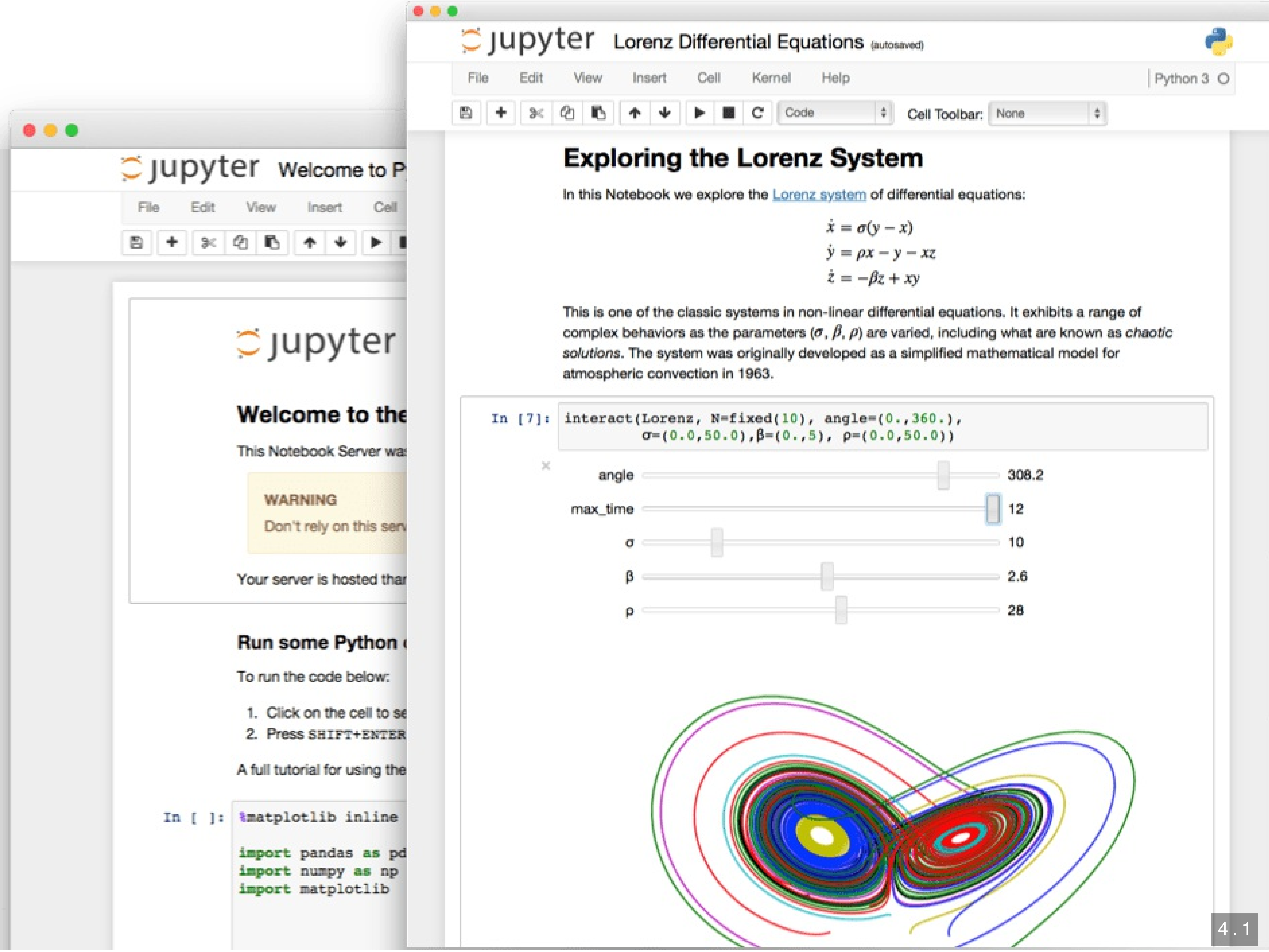
Multi Language

The "Protocol" spoken over the network can be implemented by many languages not just Python.

2013 - In about week we got a prototype of Julia kernel.

2014 - we renamed the Python-Agnostic part to Jupyter.

What is Jupyter
Mainly known for The Notebook



jupyter

Welcome to the

This Notebook Server was

WARNING

Don't rely on this serv

Your server is hosted thar

Run some Python c

To run the code below:

1. Click on the cell to se
2. Press `SHIFT+ENTER`

A full tutorial for using the

In [ ]:
```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

In [7]:
```
interact(Lorenz, N=fixed(10), angle=(0.,360.),
         σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

angle 308.2

max_time 12

σ 10

β 2.6

ρ 28

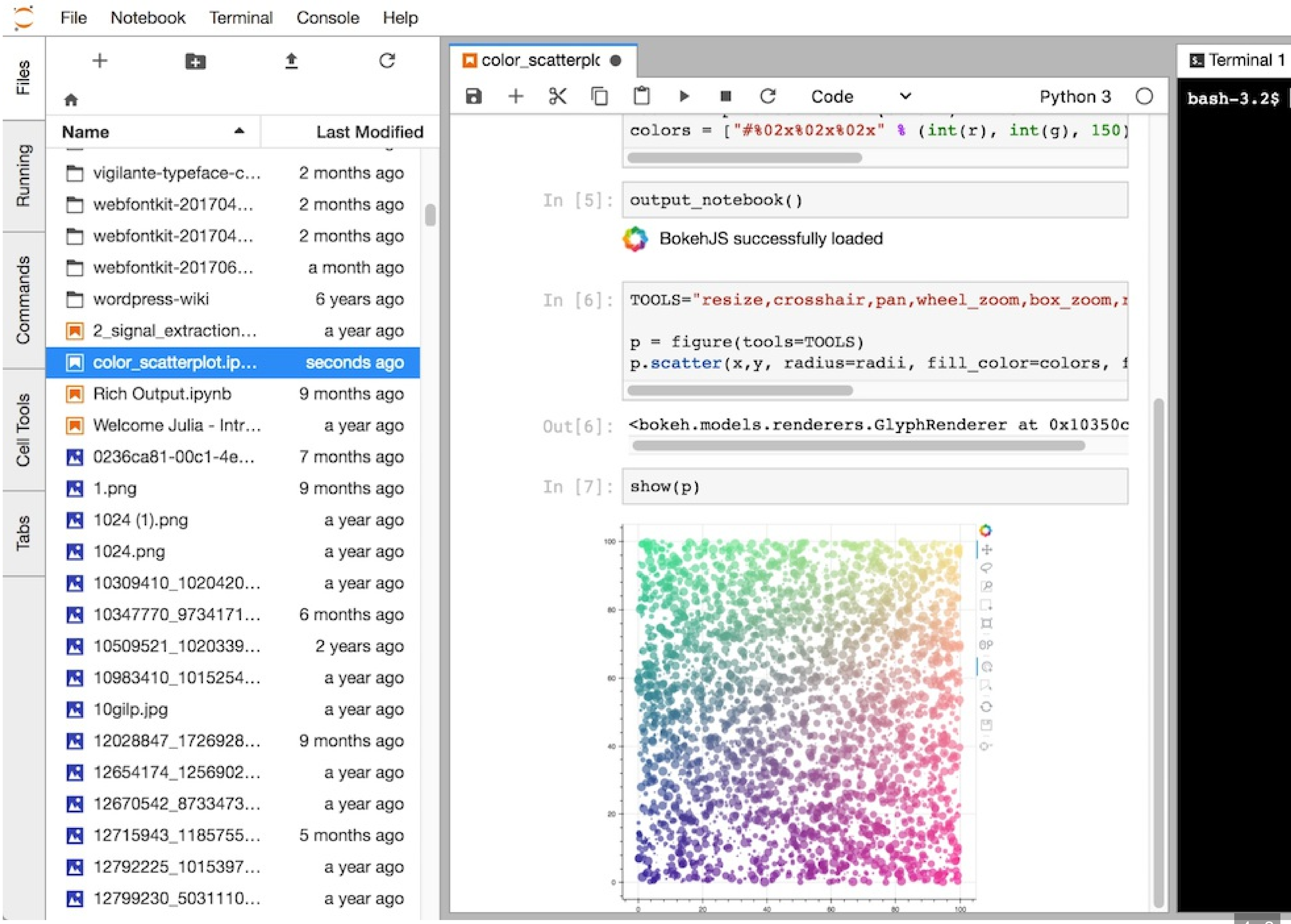
The Notebook

(Highly overloaded term)

- Web server (often local), with a web application that load .ipynb documents (json), that con contain both code, narrative (includes Math rendering) and results.
- Attached to a Kernel (often local) doing heavy computation.
- Results can be:
  - Static (Image)
  - Interactive (Pure Javascript side scoll/pan/brush)
  - Dynamic (Call back into Python if necessary)

JupyterLab

A couple of Days ago/ Soon should be release JupyterLab:



Name	Last Modified
vigilante-typeface-c...	2 months ago
webfontkit-201704...	2 months ago
webfontkit-201704...	2 months ago
webfontkit-201706...	a month ago
wordpress-wiki	6 years ago
2_signal_extraction...	a year ago
color_scatterplot.ip...	seconds ago
Rich Output.ipynb	9 months ago
Welcome Julia - Intr...	a year ago
0236ca81-00c1-4e...	7 months ago
1.png	9 months ago
1024 (1).png	a year ago
1024.png	a year ago
10309410_1020420...	a year ago
10347770_9734171...	6 months ago
10509521_1020339...	2 years ago
10983410_1015254...	a year ago
10gilp.jpg	a year ago
12028847_1726928...	9 months ago
12654174_1256902...	a year ago
12670542_8733473...	a year ago
12715943_1185755...	5 months ago
12792225_1015397...	a year ago
12799230_5031110...	a year ago

```
colors = ["#%02x%02x%02x" % (int(r), int(g), 150)

In [5]: output_notebook()

        BokehJS successfully loaded

In [6]: TOOLS="resize,crosshair,pan,wheel_zoom,box_zoom,r

        p = figure(tools=TOOLS)
        p.scatter(x,y, radius=radii, fill_color=colors, f

Out[6]: <bokeh.models.renderers.GlyphRenderer at 0x10350c

In [7]: show(p)
```

```
bash-3.2$
```

Extends the notebook interface with text editor, shell, ...etc

Protocols and Formats

Jupyter is also a set of Protocols and Formats that reduce the N-frontends x M-backends problem to a M-Frontends + N-backends,

- Open, Free and as simple as possible.
    - Json (almost) everywhere
- Thought for Science and Interactive use case.
    - Results embedded in documents no "Copy past" mistake.
    - Scale from Education to HPC jobs.

Ecosytem

Frontends: Notebook, JupyterLab, CLI, Vim, Emacs, Visual Studio Code, Atom, Nteract, Juno...

Kernels: Python, Julia, R, Haskell, Perl, Fortran, Ruby, Javascript, C/C++, Go, Scala, Elixir... 60+

Why the Popularity

Interactivity

Coding is not the full time Job of most of our users. A simple, single tool, with friendly interface helps.

Persisting kernel state allows to iterate only on part of an analysis.

Notebook interface give the interactivity of the REPL with the editability and linearity of a script with intermediate result. Aka "Literate Computing"

Separation of state

Computation, and visualisation/narrative/result are in different processes.

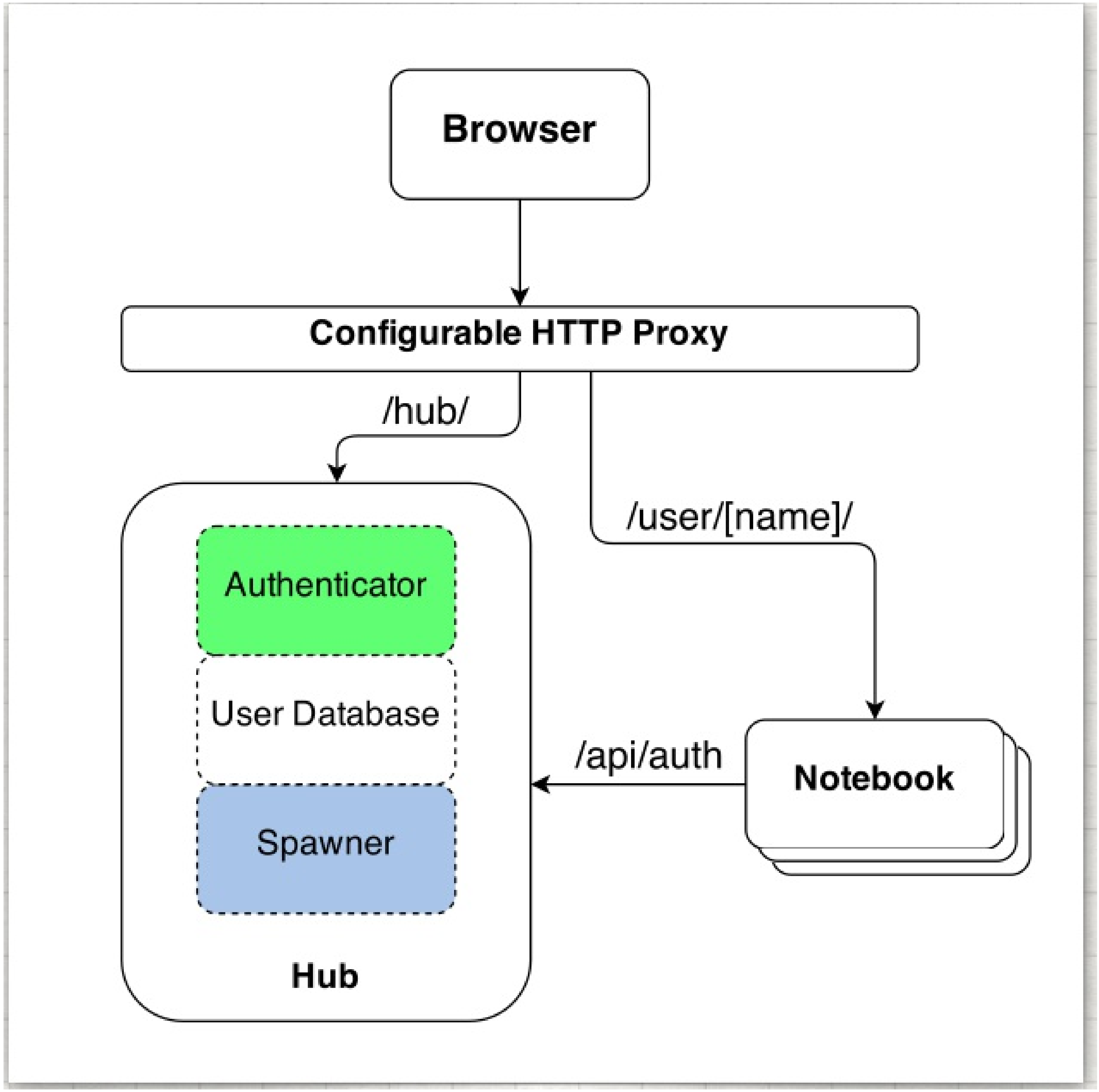
- Robust to crashes
- Can "Share" and analysis / notebook without having to "rerun" the all code. And more trustworthy (No copy-past issues).

Cons:

- Understanding that document/kernel can have different states can be challenging.

Network enabled / web based
User love fancy schmancy colors and things moving. Using D3 and other dynamic libraries are highly popular
Seamless transition to HPC: Kernel Menu > Restart on Cluster
Document persist if code crash.
Can be Zero-Installation (See JupyterHub).

JupyterHub



- Each user can get their own process/version(s)/configuration(s)
- Hooks into any Auth
- Only requires a browser

Use cases

Education

The format of the notebook is attractive for Education/Tutorial

Small Data analytics

AKA "Fit in memory", run on your laptop.

HPC

Batch Jobs

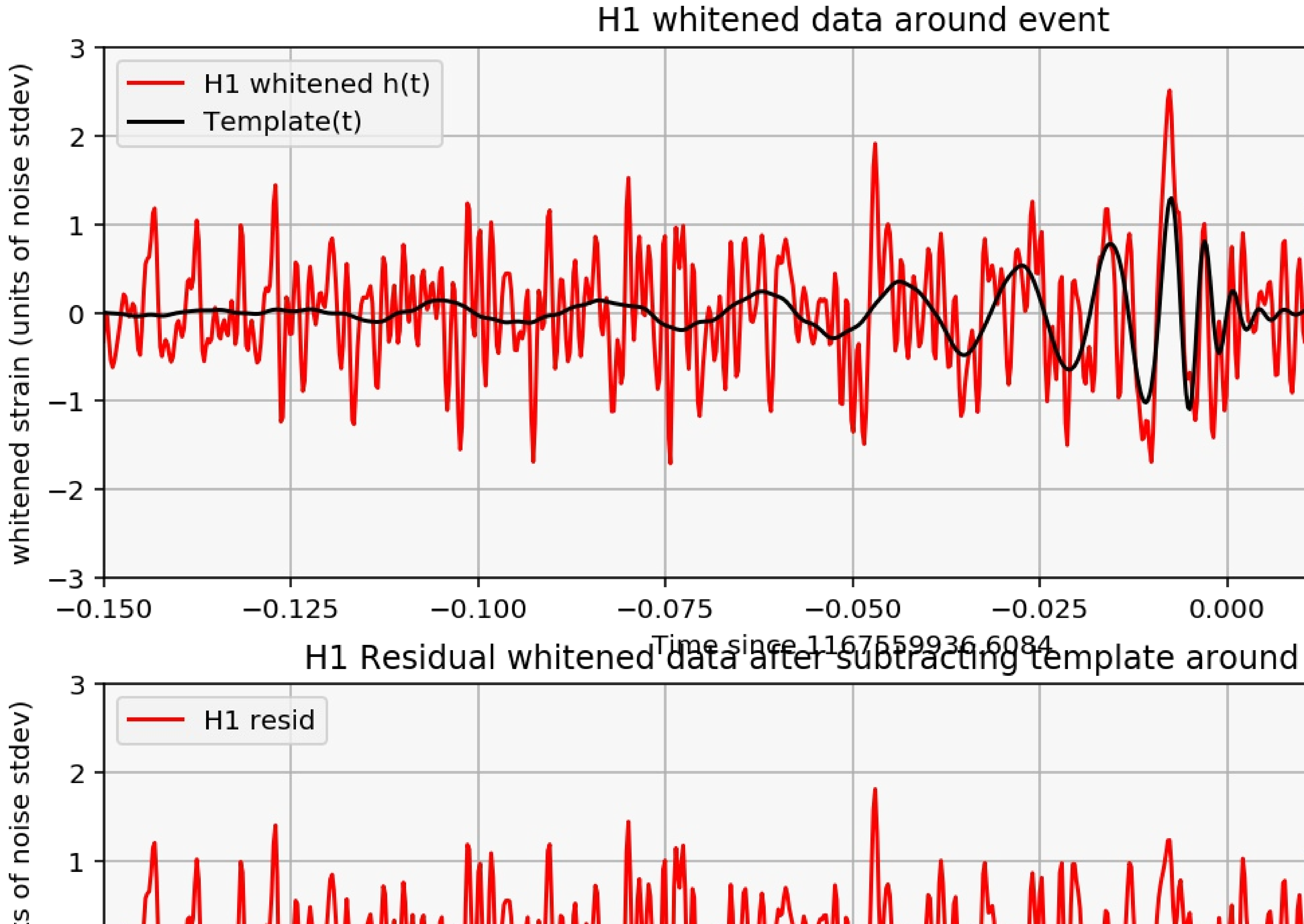
You can run notebook in a headless manner... but not the best usecase

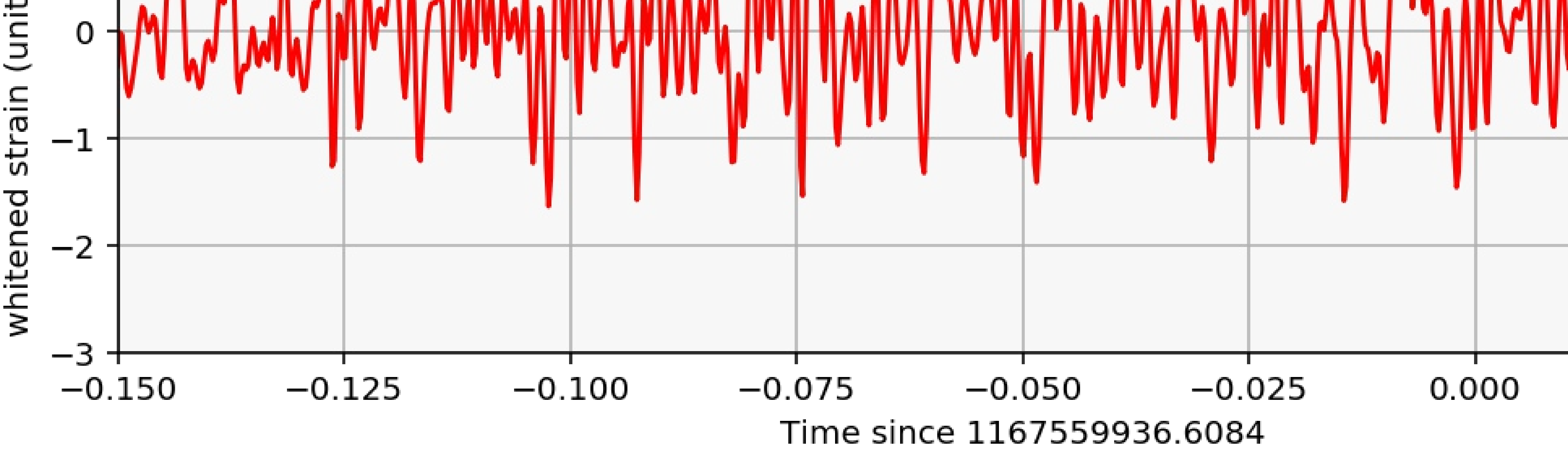
Interactive Cluster.

- Run a Hub (hook into LDAP/PAM...)
- Run notebook server on a Head node
- Run Kernels on head Node/fast queue
- Workers on Batch queue/cluster.

Example of Famous notebook workflow.

LIGO gravitaional wave





- Binder (data subset): https://github.com/minrk/ligo-binder

Pangeo
(GeoScience in the cloud)

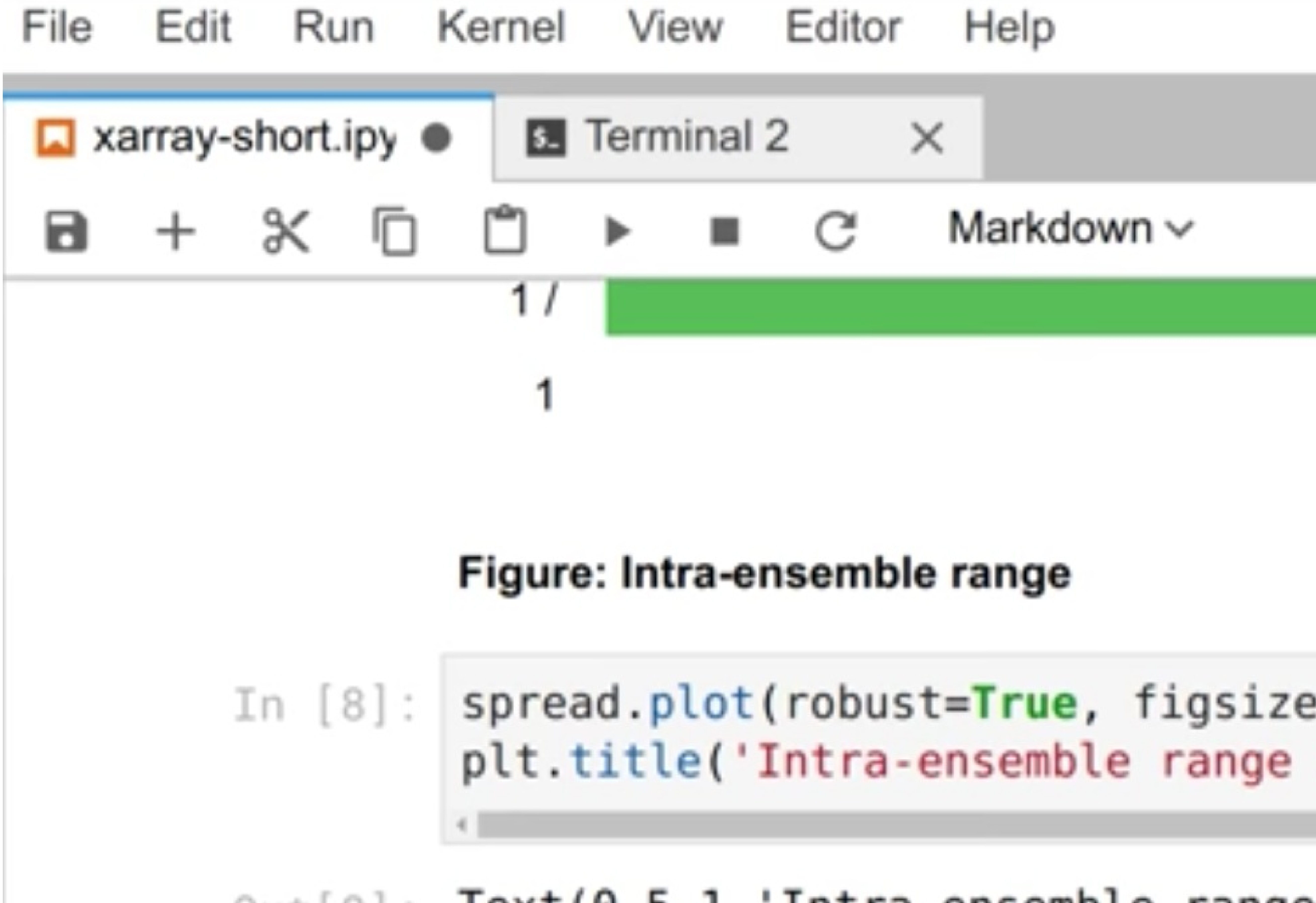
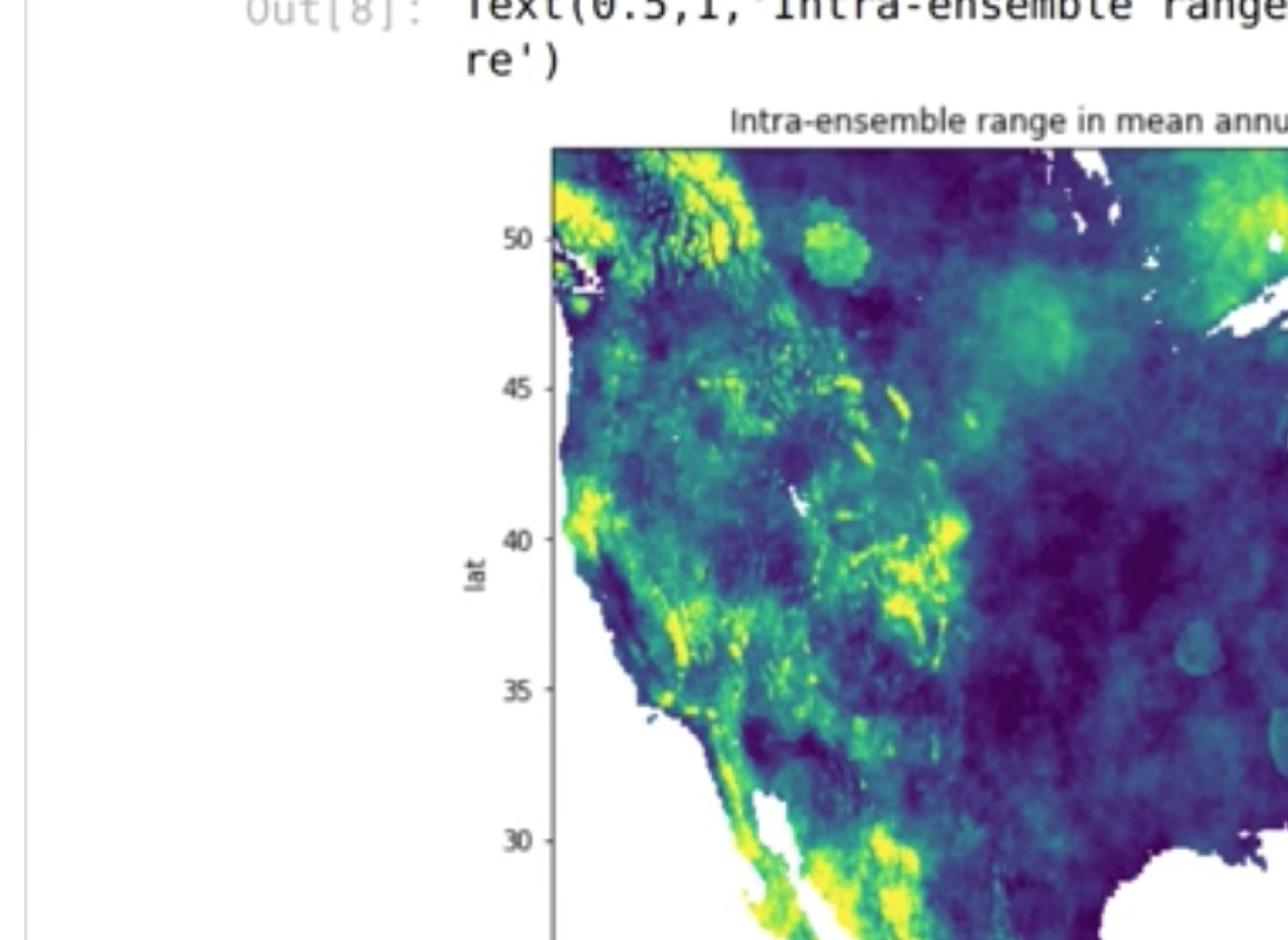


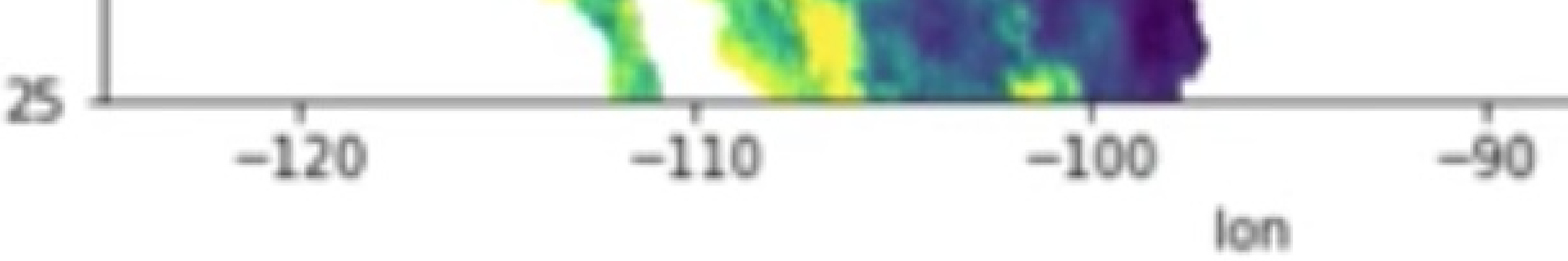
Figure: Intra-ensemble range

```
In [8]: spread.plot(robust=True, figsize
        plt.title('Intra-ensemble range
```

```
Out[8]: Text(0.5, 1, 'Intra-ensemble range
```

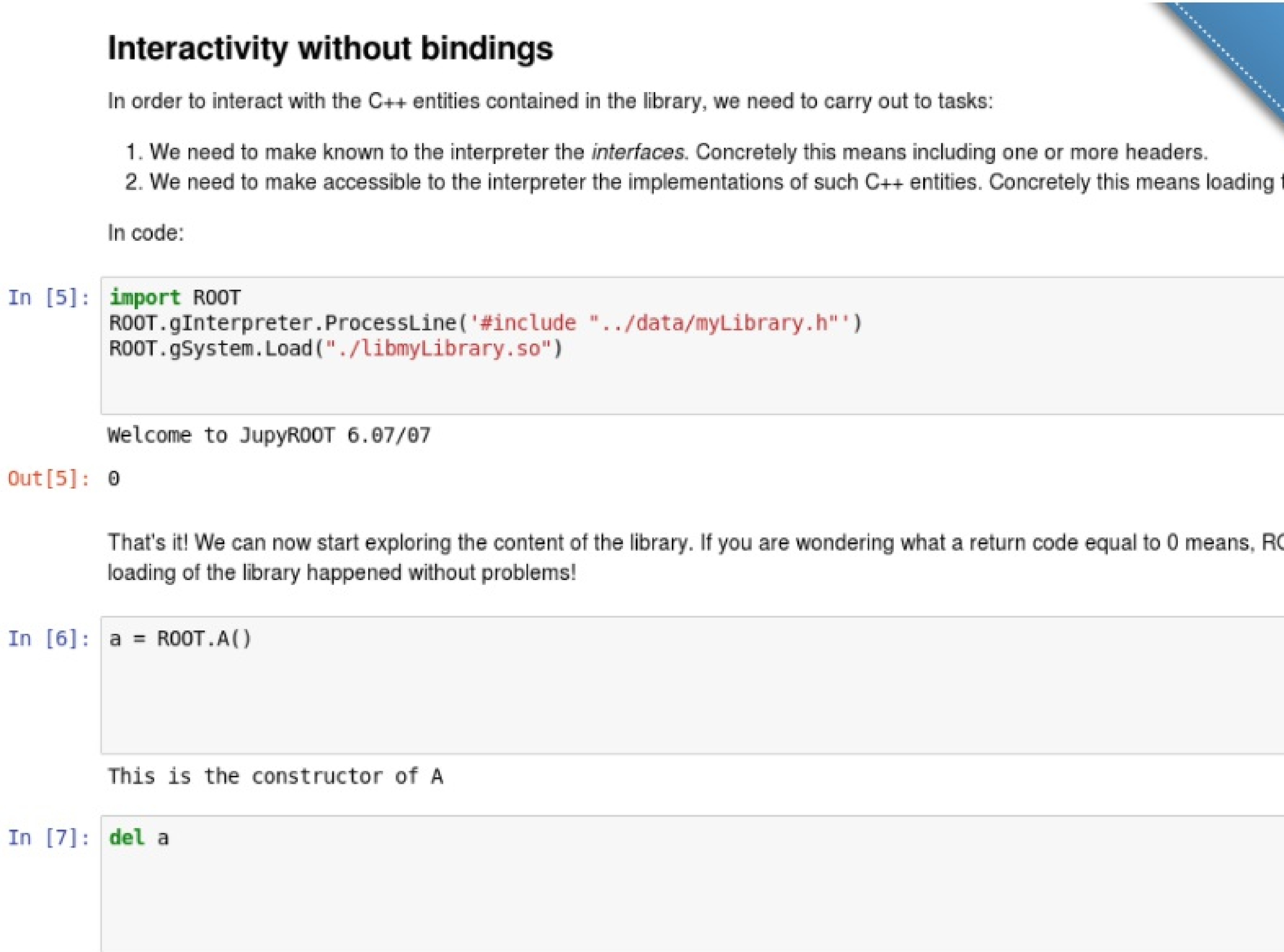


```
Out[8]: Text(0.5,1,'Intra-ensemble range
re')
```



- [JupyterHub, Dask, and XArray on the Cloud](http://pangeo.pydata.org/)
- http://pangeo.pydata.org/

C++ from Python w/o bindings



Interactivity without bindings

In order to interact with the C++ entities contained in the library, we need to carry out to tasks:

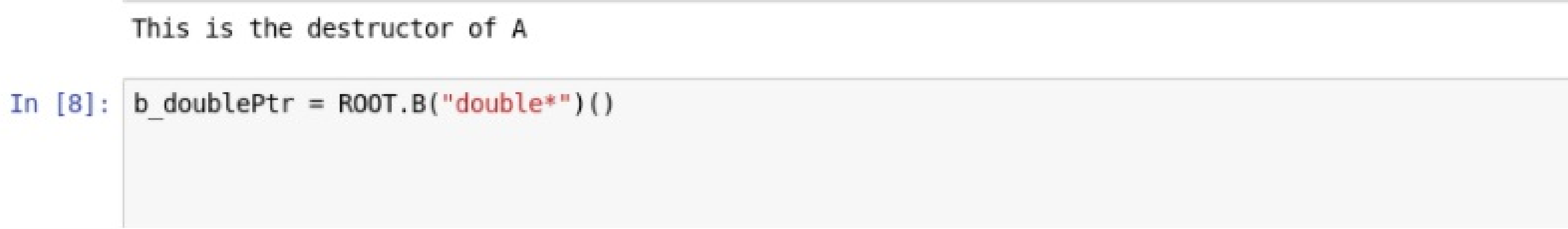
1. We need to make known to the interpreter the *interfaces*. Concretely this means including one or more headers.
2. We need to make accessible to the interpreter the implementations of such C++ entities. Concretely this means loading t

In code:

```
In [5]: import ROOT
        ROOT.gInterpreter.ProcessLine('#include "../data/myLibrary.h"')
        ROOT.gSystem.Load("./libmyLibrary.so")

        Welcome to JupyROOT 6.07/07

Out[5]: 0
```

That's it! We can now start exploring the content of the library. If you are wondering what a return code equal to 0 means, RC loading of the library happened without problems!

```
In [6]: a = ROOT.A()

        This is the constructor of A

In [7]: del a
```

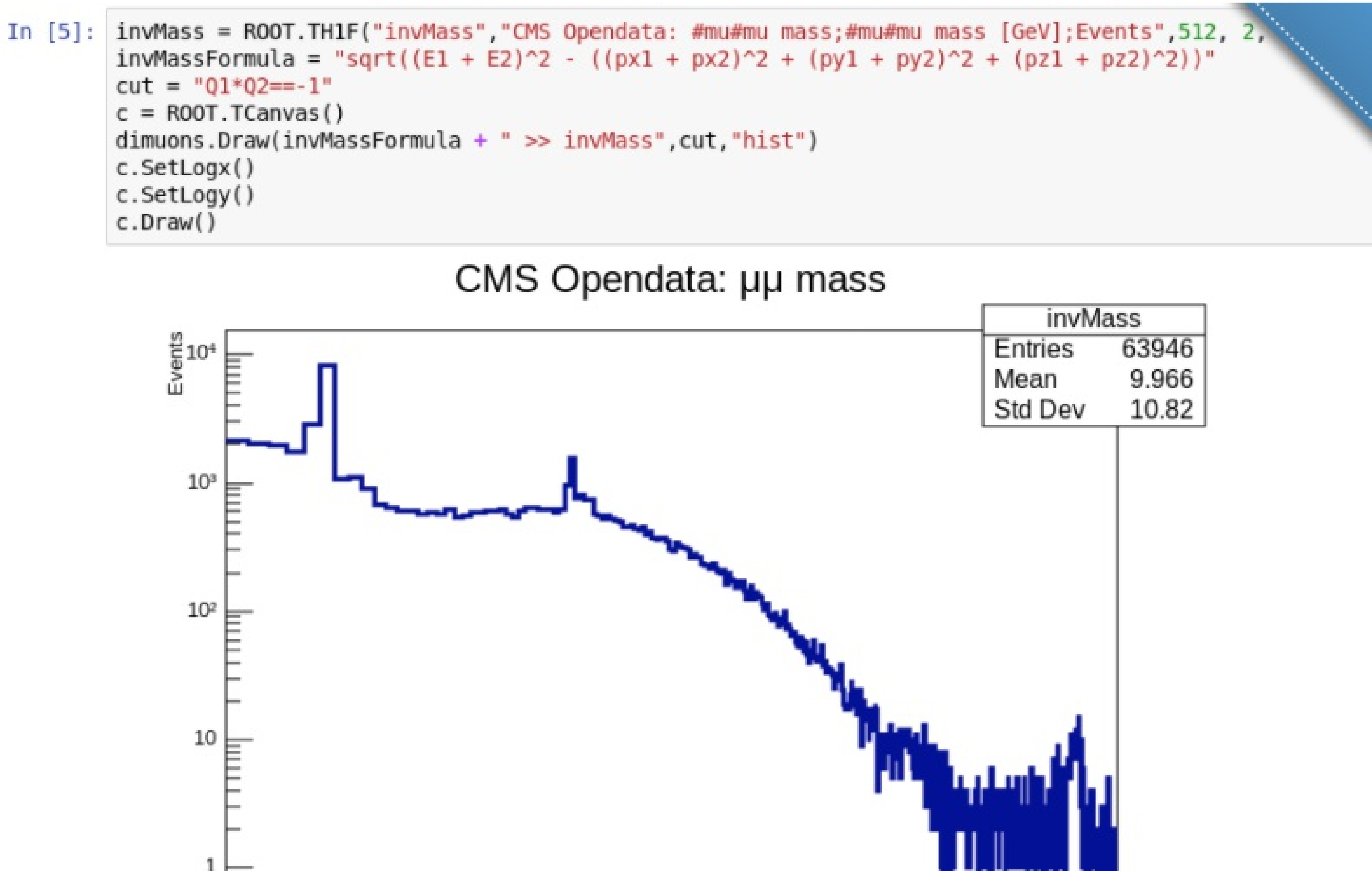


```
This is the destructor of A
```

In [8]:

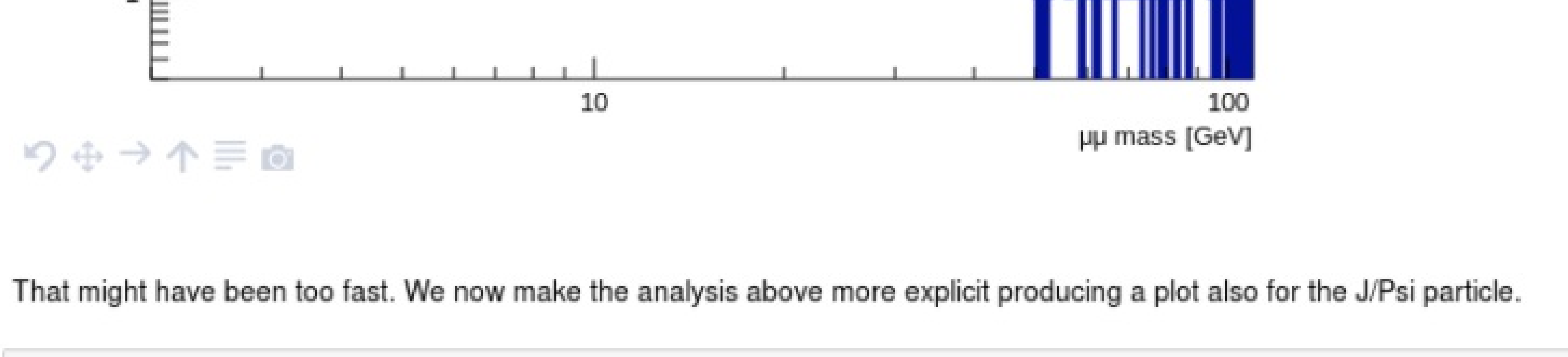
```
b_doublePtr = ROOT.B("double*")()
```

CMS Opendata: di-muon analysis



In [5]:

```
invMass = ROOT.TH1F("invMass","CMS Opendata: #mu#mu mass;#mu#mu mass [GeV];Events",512, 2,
invMassFormula = "sqrt((E1 + E2)^2 - ((px1 + px2)^2 + (py1 + py2)^2 + (pz1 + pz2)^2))"
cut = "Q1*Q2==-1"
c = ROOT.TCanvas()
dimuons.Draw(invMassFormula + " >> invMass",cut,"hist")
c.SetLogx()
c.SetLogy()
c.Draw()
```



That might have been too fast. We now make the analysis above more explicit producing a plot also for the J/Psi particle.

- https://swan.web.cern.ch/ http://jupyterhub.readthedocs.io/en/latest/gallery-jhub-deployments.html