



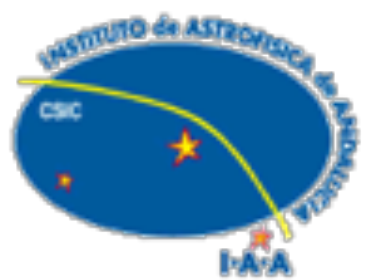
notebooks **ON STERoids**



MPIK Heidelberg

José Enrique Ruiz

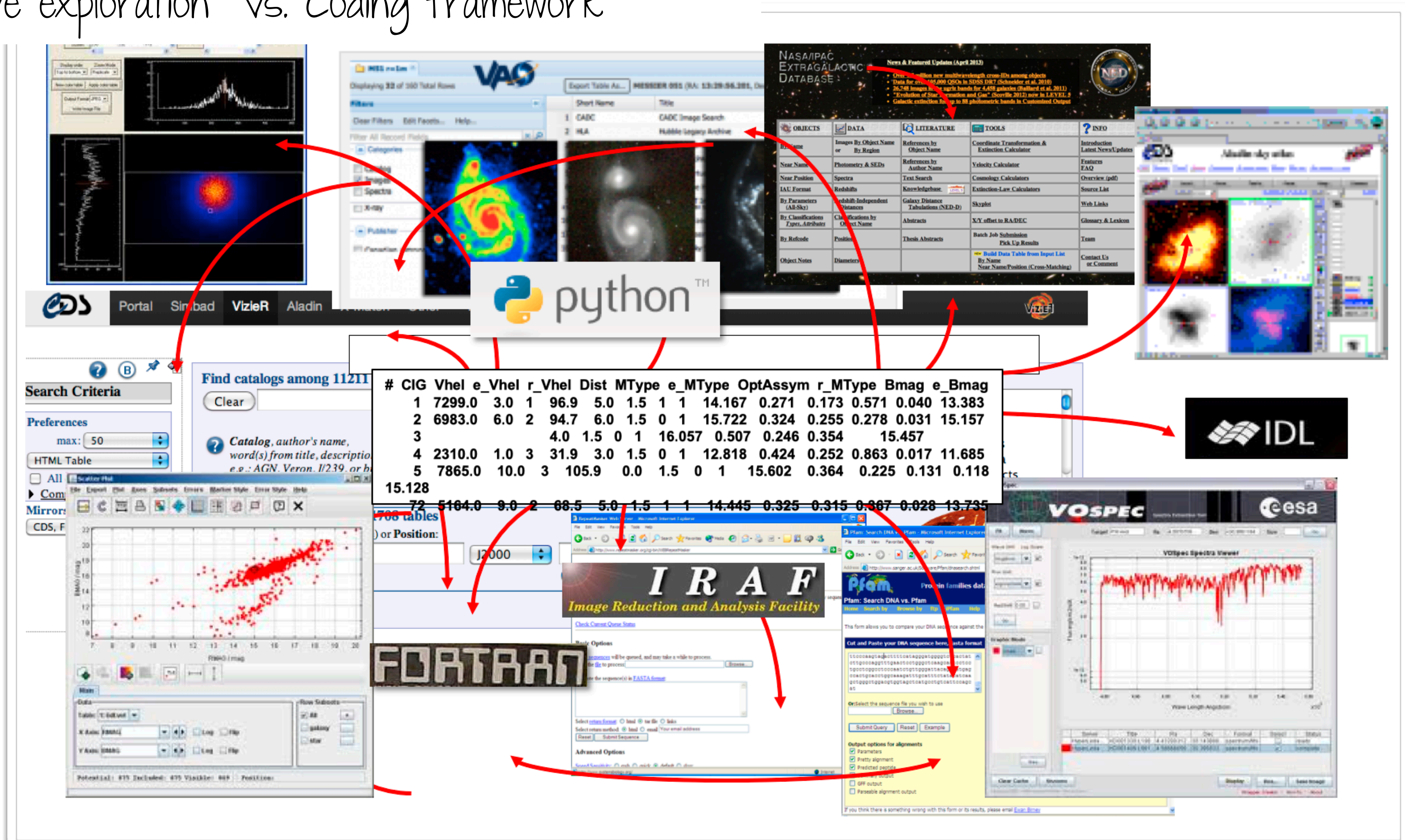
Instituto de Astrofísica de Andalucía - CSIC



@bultako 

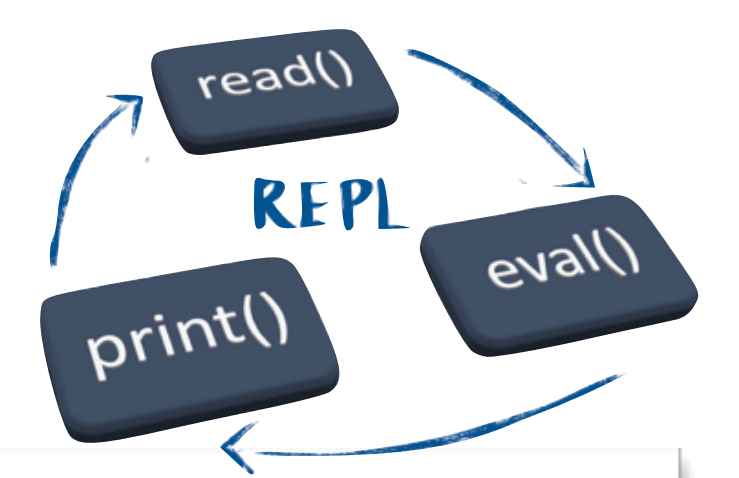
# The exploratory research workflow

Interactive exploration vs. Coding framework



# The read-eval-print loop

IP[y]: IPython  
Interactive Computing



A read-eval-print loop (REPL), also termed an interactive top-level or language shell, is a simple, **interactive environment** that takes single user inputs (i.e., single expressions), evaluates them, and returns the result to the user.

IPython 0.01

<https://gist.github.com/fperez/1579699>

2001

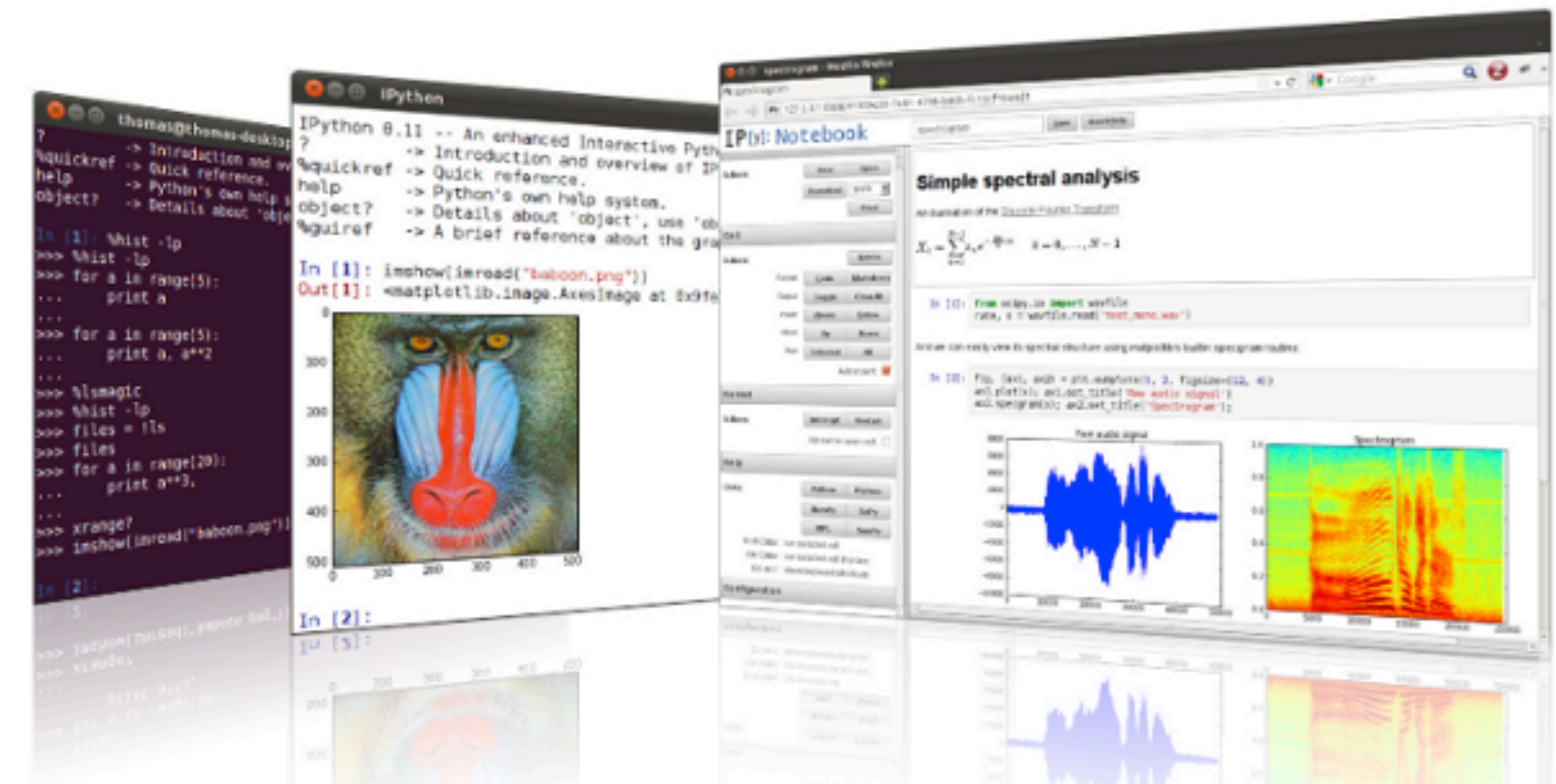


Photo credit: Adriana Restrepo. Fernando Pérez and Brian Granger discuss the architecture of Project Jupyter.

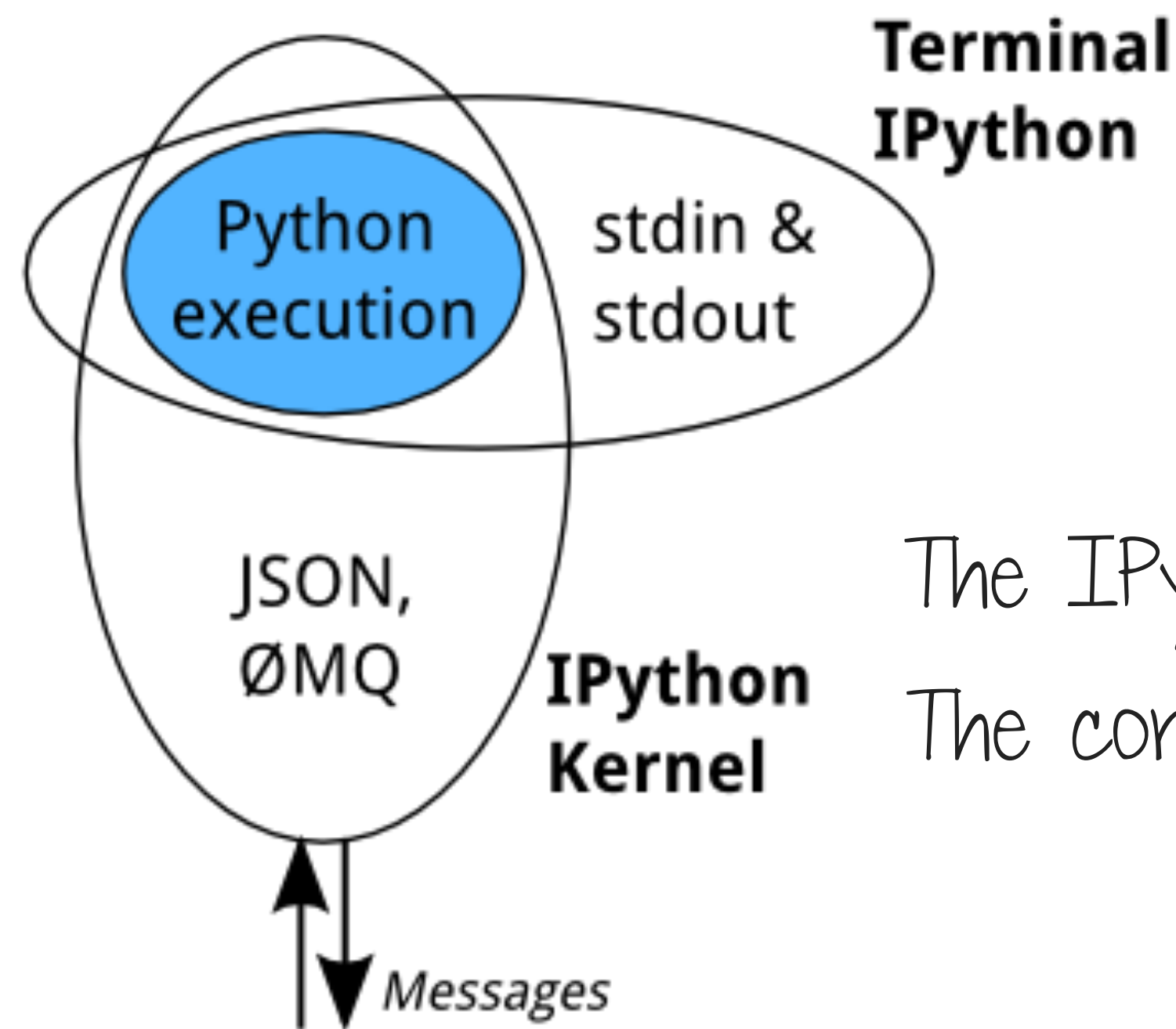
IPython 0.12

2011

IPython 0.12 was released on 18 December 2011. The major new feature with this release is the **IPython Notebook**, an interactive Python interface running in the browser. [Download](#) it now, or read more about [what's new](#).



# IPython internals



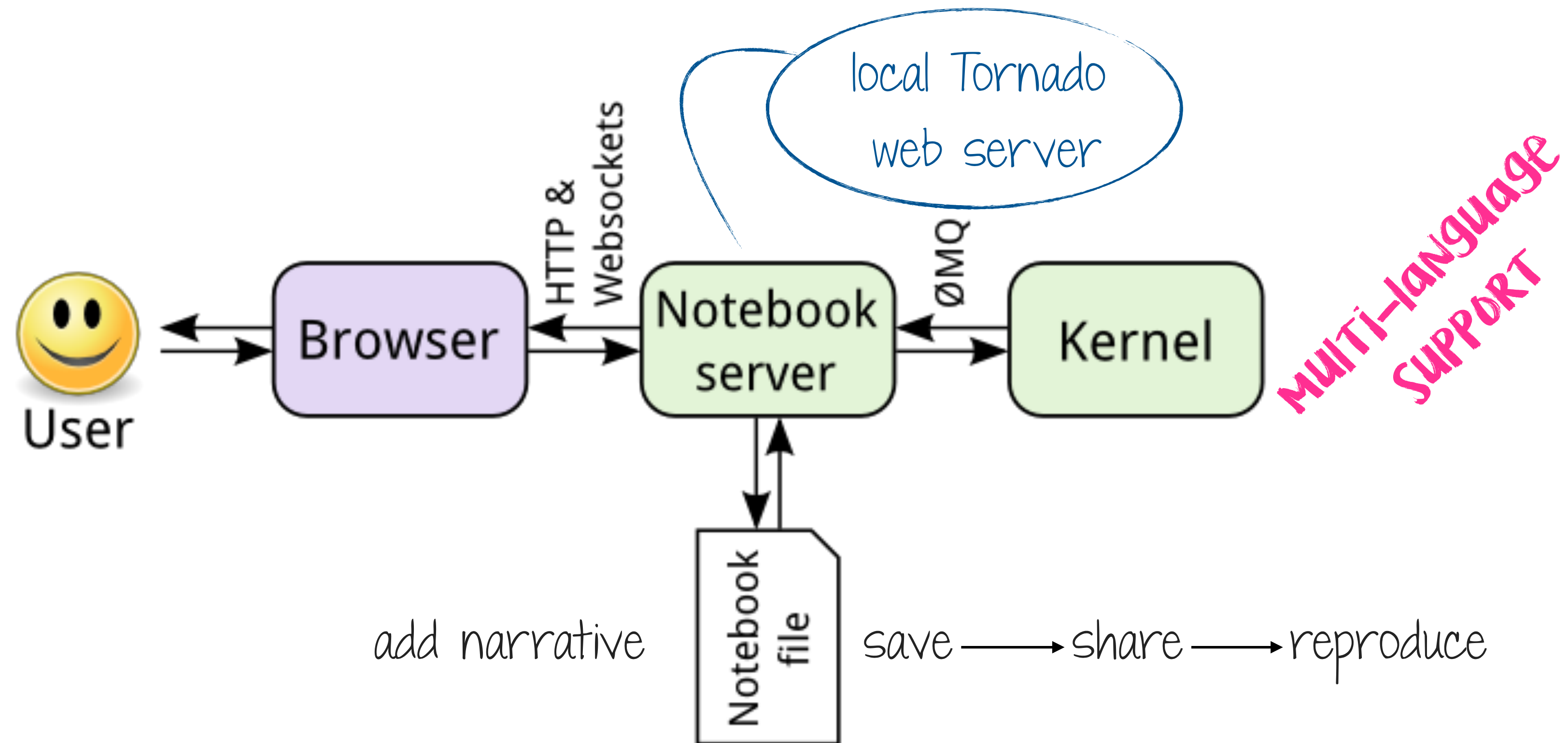
The terminal IPython is the REPL core, dealing with user **inputs and outputs** issued from their execution.

The IPython Kernel is the responsible for **executing** the user input.

The core execution machinery for the kernel is shared with terminal IPython.

Frontends, like the notebook or the Qt console, can **communicate** with the IPython Kernel using JSON messages sent over ZeroMQ sockets. The ZeroMQ library provides the low-level transport layer over which these messages are sent.

# IPython notebook internals



The Notebook frontend, stores code and output, together with markdown notes, in an editable document called a notebook. When you save it, this is sent from your browser to the notebook server, which saves it on disk as a JSON file with a .ipynb extension.

# The web analogy

Web execution server

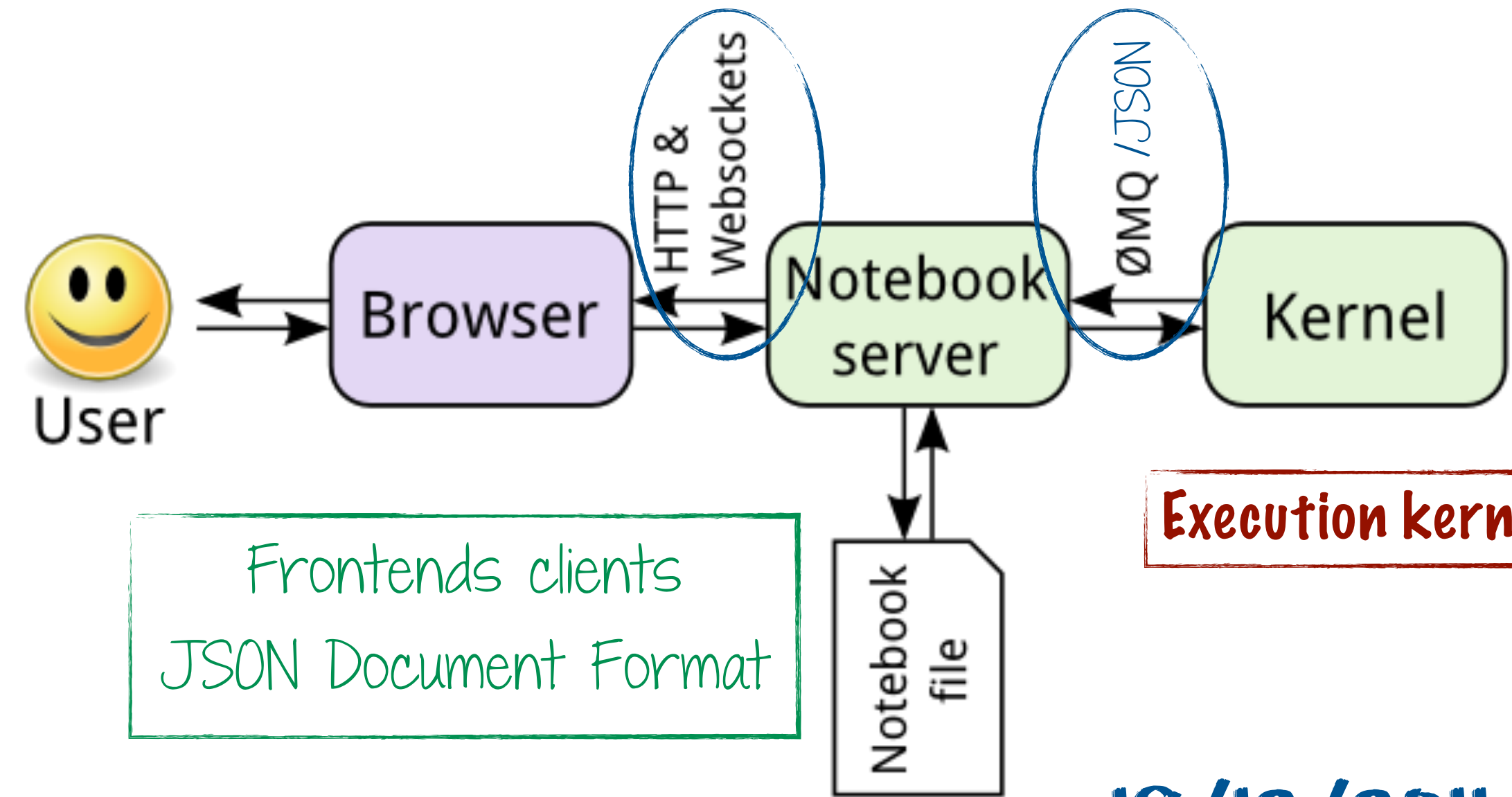


HTTP Transfer Protocol



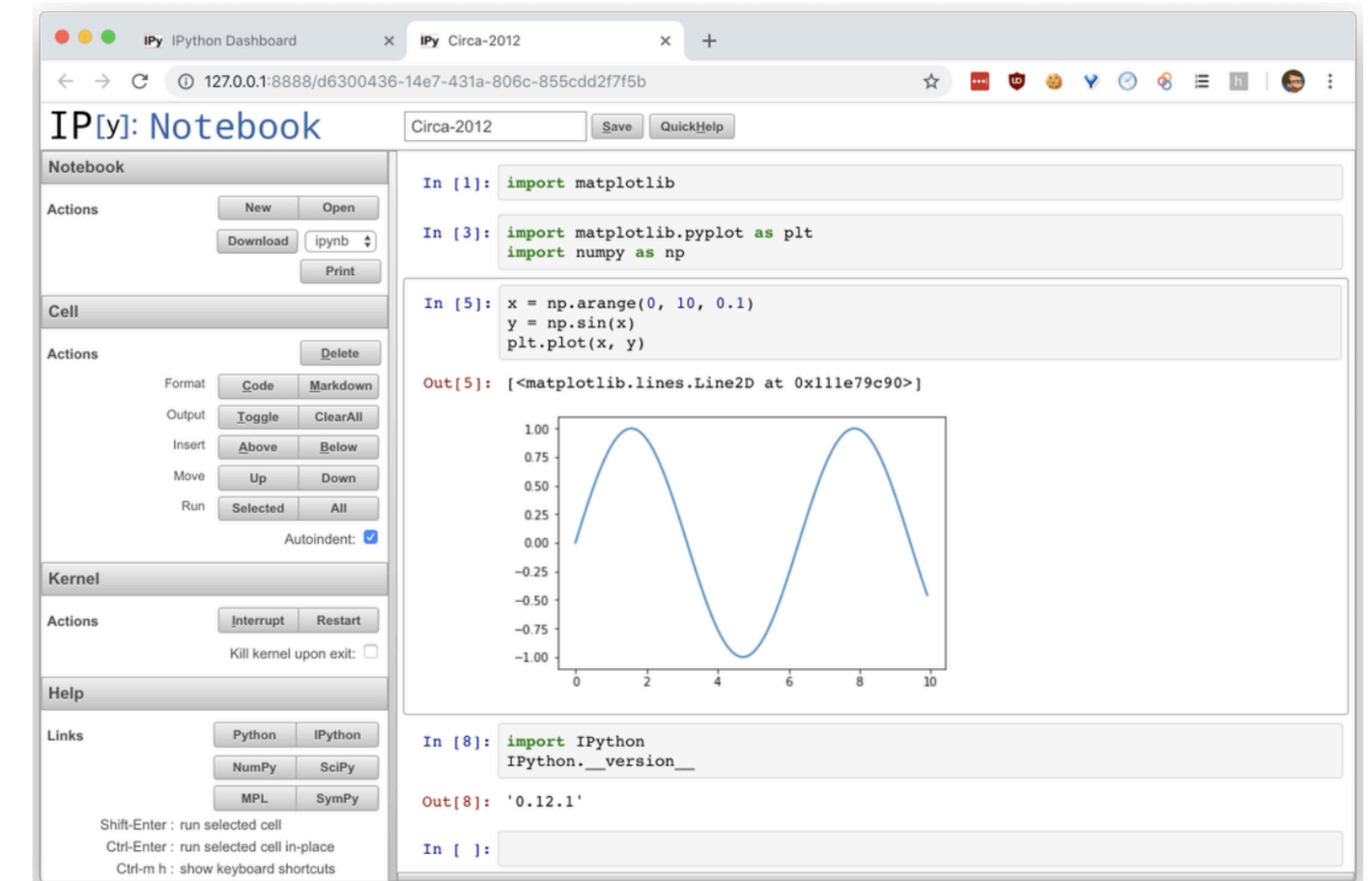
Frontends clients  
HTML Document Format

# Interactive Computing Messaging Protocol



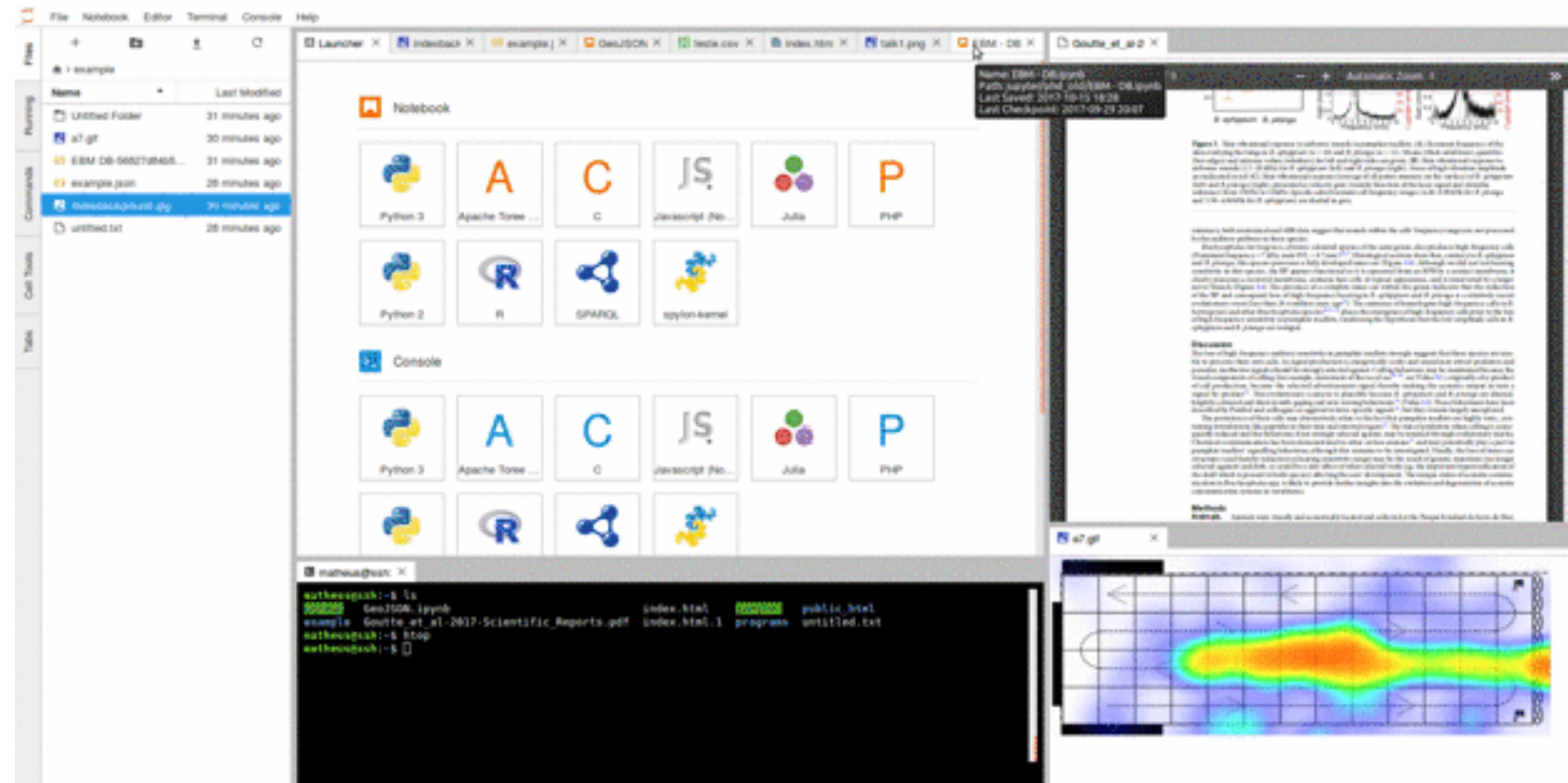
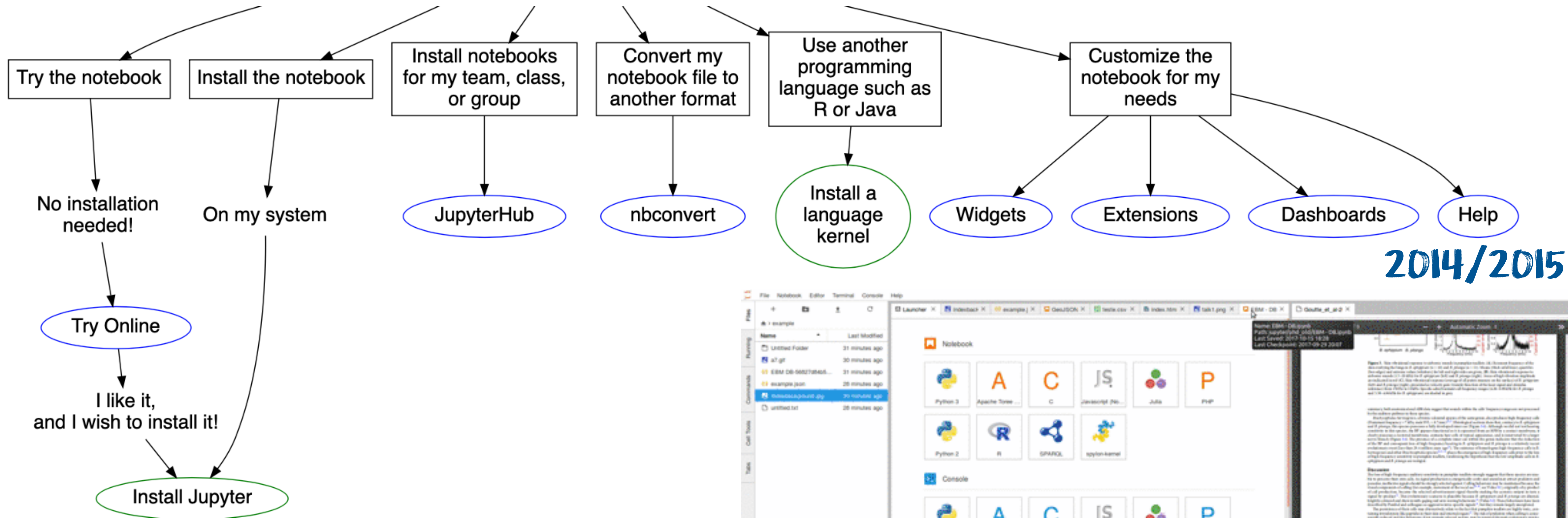
Frontends clients  
JSON Document Format

18/12/2011



# The jupyter project ecosystem

The interactive exploratory environment in the browser



# Jupyter Notebook <sup>2011</sup> and JupyterLab <sup>2018</sup>

- JupyterLab is (not only) another **user interface** for notebooks.
- JupyterLab is the **evolution** of the Jupyter Notebook user interface into a flexible unified platform.

FileBrowser   Terminal   TextEditor   Widgets   Extensions



- JupyterLab is a **major internal refactoring** of the frontend:
  - Clean Model-View separation -> multiple views / high interactivity.
  - Modern JavaScript -> TypeScript, npm/yarn webpack packaging, react, phosphor.js
  - Fully extensible by third parties -> everything is an extension + differentiated private/public APIs.
  - Higher performance - especially with big tabular datasets !
- There are **no changes in the notebooks, messaging protocol or the kernels.**
  - The interactive messaging protocol and JSON notebooks format is kept.
  - GitHub and any other existing front-ends will continue working with JupyterLab notebooks.
- User functionalities are focused towards an **IDE-like / integrated exploratory research desktop.**
  - Open windows to the local desktop and data formats.
  - Higher interactivity between any kind of element involved in the exploratory process.

## User Interfaces

JupyterLab

Jupyter Notebook

Jupyter Console

Qt Console

<https://jupyter-notebook.readthedocs.io>

<https://jupyterlab.readthedocs.io>



# JupyterLab Live Demo



# IPython magics

You may also access the shell with !

!shell command

<https://ipython.readthedocs.io/en/stable/interactive/magics.html>

Line magics act on one line %

Cell magics act on the entire cell %%

```
In [1]: %lsmagic
```

```
Out[1]: Available line magics:
```

```
%alias %alias_magic %autoawait %autocall %automagic %autosave %bookmark %cat %cd  
%clear %colors %conda %config %connect_info %cp %debug %dhist %dirs %doctest_mode  
%ed %edit %env %gui %hist %history %killbgscripts %ldir %less %lf %lk %ll %load  
%load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %lx  
%macro %magic %man %matplotlib %mkdir %more %mv %notebook %page %pastebin %pdb  
%pdef %pdoc %pfile %pinfo %pinfo2 %pip %popd %pprint %precision %prun %psearch  
%psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_  
ext %rep %rerun %reset %reset_selective %rm %rmdir %run %save %sc %set_env %sto  
re %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel  
%xmode
```

%matplotlib notebook for interactive plots

```
Available cell magics:
```

```
%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%javascript %%js %%late  
x %%markdown %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %  
%sh %%svg %%sx %%system %%time %%timeit %%writefile
```

```
Automagic is ON, % prefix IS NOT needed for line magics.
```

```
In [2]: %alias? use ?? to print the code and ? to print the docstring
```

# Interactive widgets

<https://ipywidgets.readthedocs.io>

ipywidgets

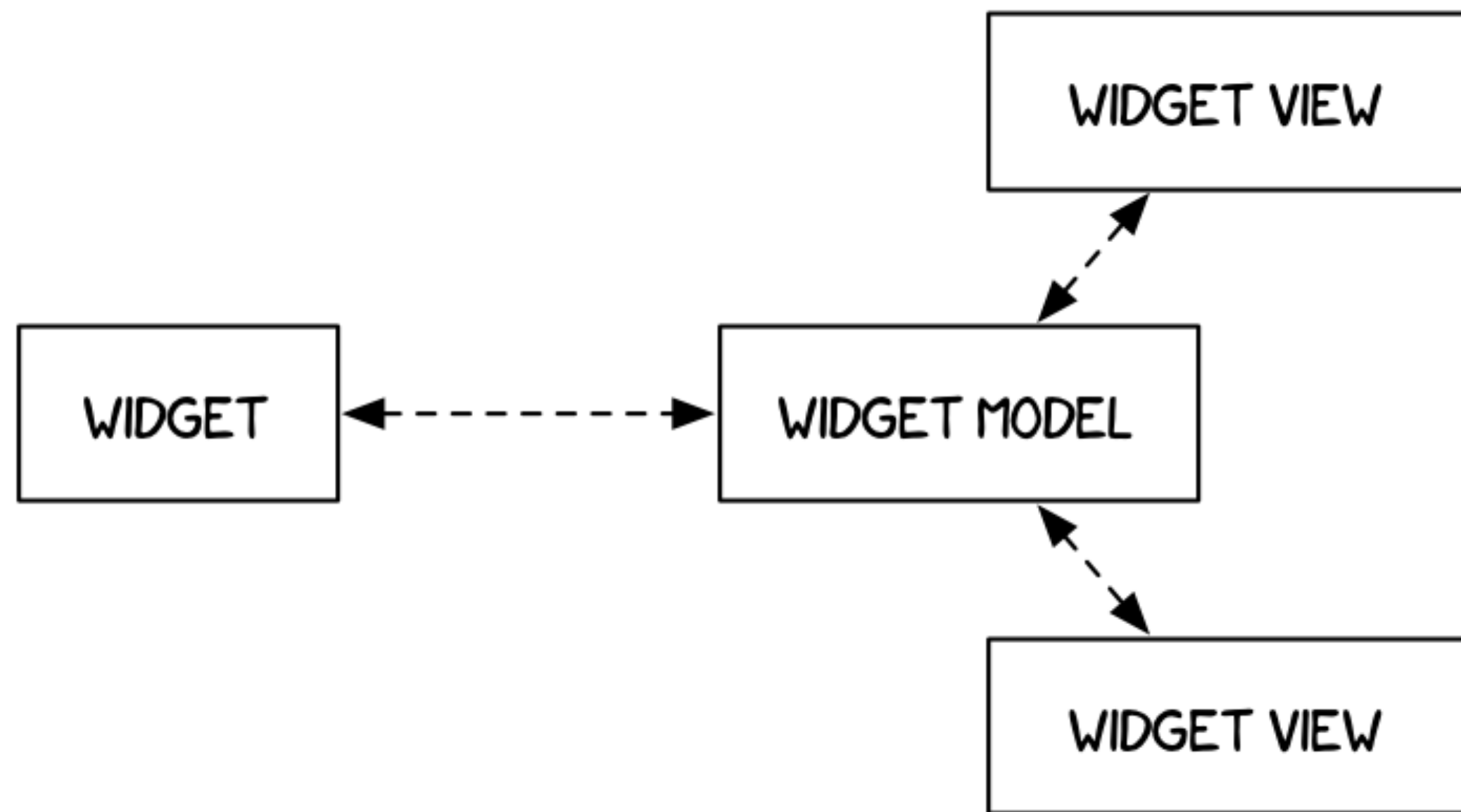
Build small interactive GUIs that can communicate with the kernel and with other widgets.

Mostly used for enhanced visualization purposes.

ipywidgets is required by many extensions.

KERNEL (PYTHON)

FRONTEND (HTML/JAVASCRIPT)



Gammapi @gammapiST · 28 Nov 2018

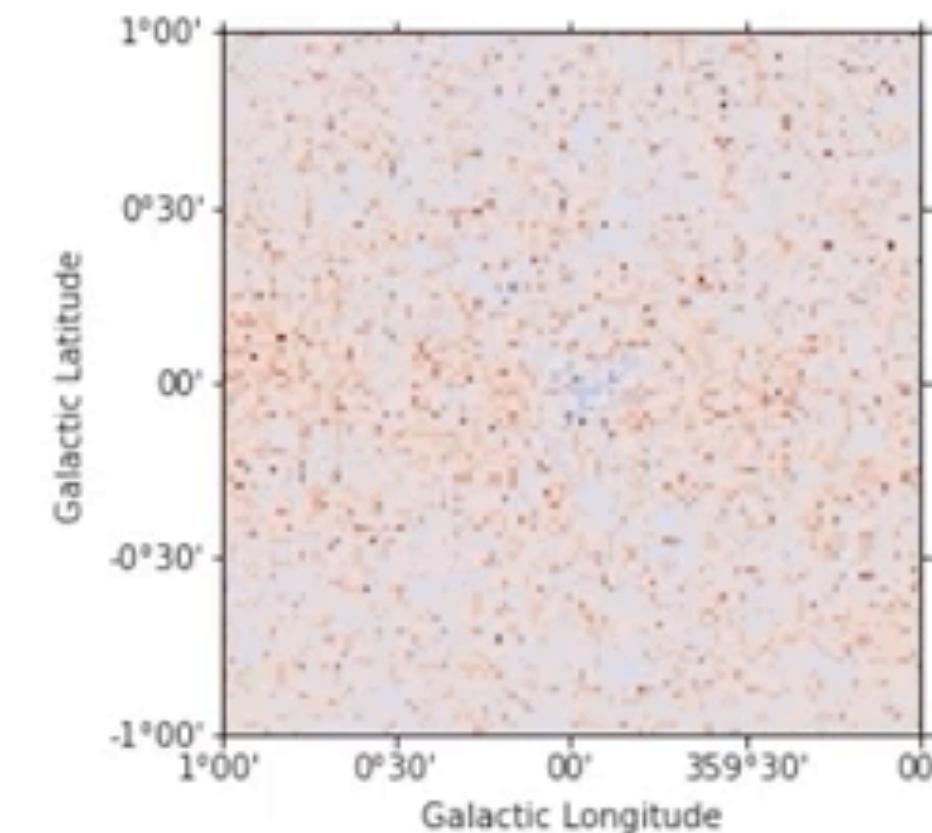
Let's have a look at the residuals over the energy axis with the `Map.plot_interactive` method.

[docs.gammapi.org/0.8/api/gammapi...](https://docs.gammapi.org/0.8/api/gammapi...)

```
In [43]: residual.plot_interactive(cmap="coolwarm")
```

Select energy:

Select stretch:  linear  
 sqrt  
 log



# Jupyter Notebook Extensions

<https://jupyter-contrib-nbextensions.readthedocs.io>

## Configurable nbextensions

disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter:

- |  |  |  |  |
|--|--|--|--|
| <input type="checkbox"/> (some) LaTeX environments for Jupyter   | <input type="checkbox"/> 2to3 Converter                        | <input type="checkbox"/> AddBefore                                 | <input type="checkbox"/> Autopep8                                |
| <input type="checkbox"/> AutoSaveTime                            | <input type="checkbox"/> Autoscroll                            | <input type="checkbox"/> Cell Filter                               | <input type="checkbox"/> Code Font Size                          |
| <input checked="" type="checkbox"/> Code prettify                | <input checked="" type="checkbox"/> Codefolding                | <input type="checkbox"/> Codefolding in Editor                     | <input type="checkbox"/> CodeMirror mode extensions              |
| <input type="checkbox"/> Collapsible Headings                    | <input type="checkbox"/> Comment/Uncomment Hotkey              | <input checked="" type="checkbox"/> contrib_nbextensions_help_item | <input type="checkbox"/> datestamper                             |
| <input type="checkbox"/> Equation Auto Numbering                 | <input type="checkbox"/> ExecuteTime                           | <input type="checkbox"/> Execution Dependencies                    | <input type="checkbox"/> Exercise                                |
| <input type="checkbox"/> Exercise2                               | <input type="checkbox"/> Export Embedded HTML                  | <input type="checkbox"/> Freeze                                    | <input type="checkbox"/> Gist-it                                 |
| <input type="checkbox"/> Help panel                              | <input type="checkbox"/> Hide Header                           | <input type="checkbox"/> Hide input                                | <input type="checkbox"/> Hide input all                          |
| <input type="checkbox"/> Highlight selected word                 | <input checked="" type="checkbox"/> highlighter                | <input type="checkbox"/> Hinterland                                | <input type="checkbox"/> Initialization cells                    |
| <input checked="" type="checkbox"/> ipyvolum/extension           | <input type="checkbox"/> isort formatter                       | <input type="checkbox"/> jupyter-datawidgets/extension             | <input checked="" type="checkbox"/> jupyter-js-widgets/extension |
| <input checked="" type="checkbox"/> jupyter-matplotlib/extension | <input type="checkbox"/> jupyter-threejs/extension             | <input type="checkbox"/> jupyter-webrtc/extension                  | <input checked="" type="checkbox"/> jupyter_boilerplate/main     |
| <input type="checkbox"/> Keyboard shortcut editor                | <input type="checkbox"/> Launch QTConsole                      | <input type="checkbox"/> Limit Output                              | <input type="checkbox"/> Live Markdown Preview                   |
| <input type="checkbox"/> Load TeX macros                         | <input type="checkbox"/> Move selected cells                   | <input type="checkbox"/> Navigation-Hotkeys                        | <input type="checkbox"/> nb_conda/main                           |
| <input type="checkbox"/> nb_conda/tree                           | <input checked="" type="checkbox"/> Nbextensions dashboard tab | <input checked="" type="checkbox"/> Nbextensions edit menu item    | <input type="checkbox"/> nbTranslate                             |
| <input type="checkbox"/> Notify                                  | <input type="checkbox"/> Printview                             | <input type="checkbox"/> Python Markdown                           | <input checked="" type="checkbox"/> qgrid/extension              |
| <input type="checkbox"/> Rubberband                              | <input type="checkbox"/> Ruler                                 | <input type="checkbox"/> Ruler in Editor                           | <input type="checkbox"/> Runtools                                |
| <input checked="" type="checkbox"/> Scratchpad                   | <input type="checkbox"/> ScrollDown                            | <input type="checkbox"/> Select CodeMirror Keymap                  | <input type="checkbox"/> SKILL Syntax                            |
| <input type="checkbox"/> Skip-Traceback                          | <input type="checkbox"/> Snippets                              | <input type="checkbox"/> Snippets                                  | <input checked="" type="checkbox"/> Snippets Menu                |
| <input type="checkbox"/> spellchecker                            | <input type="checkbox"/> Split Cells Notebook                  | <input checked="" type="checkbox"/> Table of Contents (2)          | <input type="checkbox"/> table_beautifier                        |
| <input type="checkbox"/> Toggle all line numbers                 | <input type="checkbox"/> Tree Filter                           | <input checked="" type="checkbox"/> Variable Inspector             | <input type="checkbox"/> zenmode                                 |

# Jupyter Notebook Extensions

<https://github.com/rasbt/watermark>

```
%watermark
```

```
: %load_ext watermark
```

```
: %watermark
```

```
2019-03-15T19:32:54+01:00
```

```
CPython 3.6.8  
IPython 7.3.0
```

```
compiler   : GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)  
system     : Darwin  
release    : 18.2.0  
machine    : x86_64  
processor  : i386  
CPU cores  : 4  
interpreter: 64bit
```

```
%watermark -v -m -p numpy,scipy -g
```

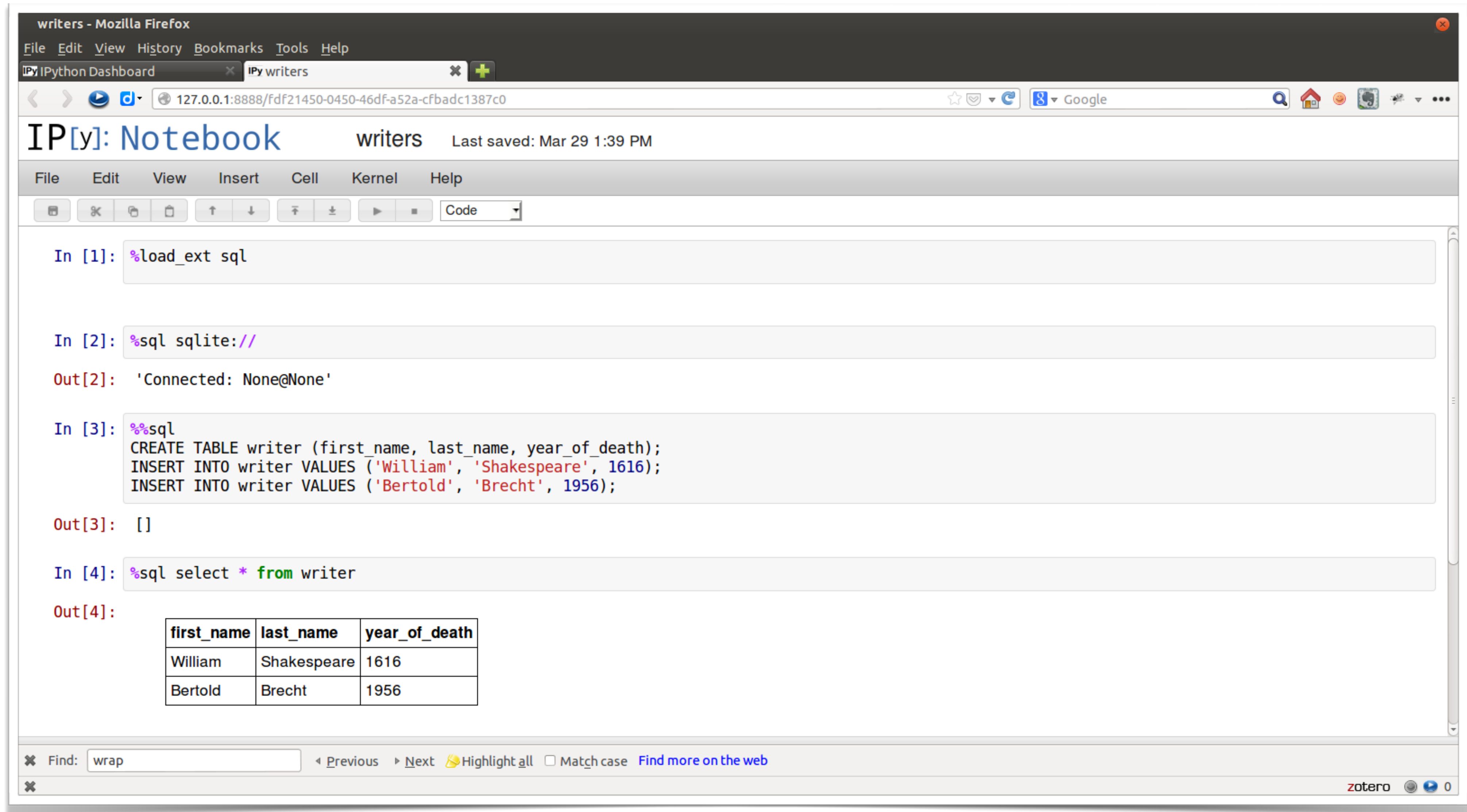
```
CPython 3.5.1  
IPython 4.0.3
```

```
numpy 1.10.2  
scipy 0.16.1
```

```
compiler   : GCC 4.2.1 (Apple Inc. build 5577)  
system     : Darwin  
release    : 15.3.0  
machine    : x86_64  
processor  : i386  
CPU cores  : 4  
interpreter: 64bit  
Git hash   : fldc669eff571603495747a1d99aeef5bc46dfb5
```

# Jupyter Notebook Extensions

<https://github.com/catherinedevlin/ipython-sql>



writers - Mozilla Firefox

File Edit View History Bookmarks Tools Help

IPython Dashboard IPy writers

127.0.0.1:8888/fdf21450-0450-46df-a52a-cfbadc1387c0

IP[y]: Notebook writers Last saved: Mar 29 1:39 PM

File Edit View Insert Cell Kernel Help

In [1]: `%load_ext sql`

In [2]: `%sql sqlite://`

Out[2]: 'Connected: None@None'

In [3]: `%%sql`  
`CREATE TABLE writer (first_name, last_name, year_of_death);`  
`INSERT INTO writer VALUES ('William', 'Shakespeare', 1616);`  
`INSERT INTO writer VALUES ('Bertold', 'Brecht', 1956);`

Out[3]: []

In [4]: `%sql select * from writer`

Out[4]:

first_name	last_name	year_of_death
William	Shakespeare	1616
Bertold	Brecht	1956

Find: wrap Previous Next Highlight all Match case Find more on the web

zotero 0

# Jupyter Notebook Extensions

<https://github.com/quantopian/qgrid>



```
In [2]: import qgrid
qgrid_widget = qgrid.show_grid(df, show_toolbar=True)
qgrid_widget
```

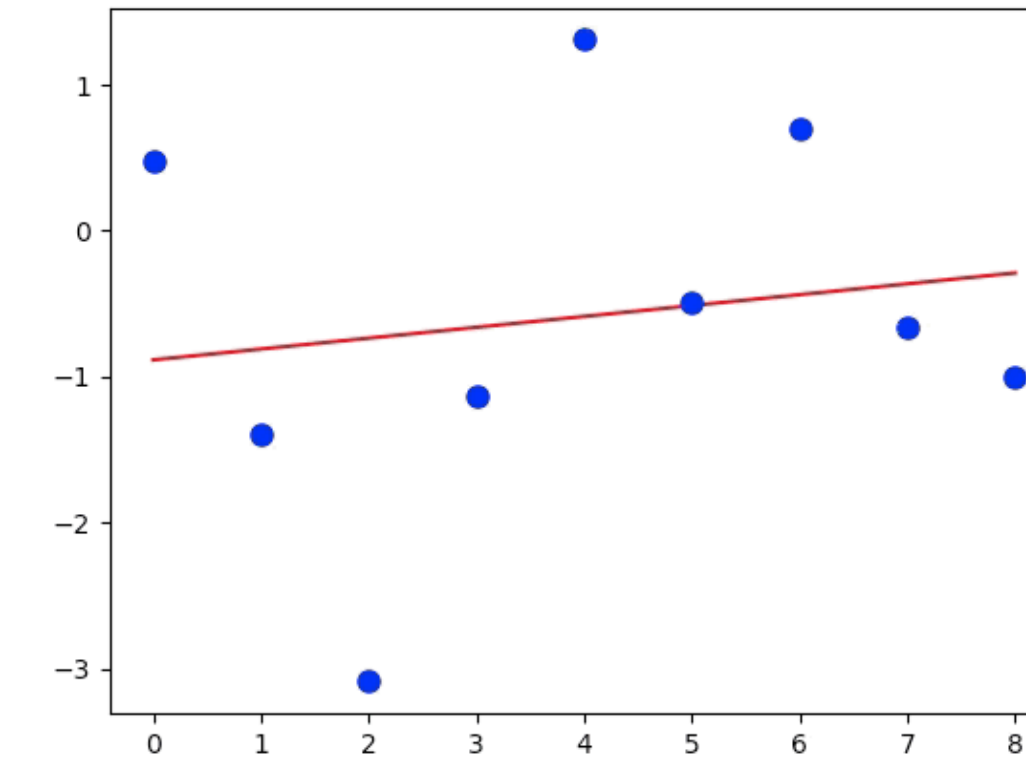
	Add Row	Remove Row									
index	T	A	T	B	T	C	T	D	T	E	T
0		2013-01-01		-2.0773		washington		foo		✓	
1		2013-01-02		-0.55236		adams		bar			
2		2013-01-03		-1.46247		washington		buzz			
3		2013-01-04		0.85559		madison		bippity			
4		2013-01-05		-1.66466		lincoln		boppity			
5		2013-01-06		0.85659		jefferson		foo		✓	
6		2013-01-07		1.23665		hamilton		foo		✓	
7		2013-01-08		-0.36515		roosevelt		bar			
8		2013-01-09		0.89955		kennedy		zoo			

```
In [3]: qgrid_widget.get_changed_df()
```

Out[3]:

	A	B	C	D	E
0	2013-01-01	-2.077295	washington	foo	True
1	2013-01-02	-0.552359	adams	bar	False
2	2013-01-03	-1.462471	washington	buzz	False
3	2013-01-04	0.855593	madison	bippity	False
4	2013-01-05	-1.664660	lincoln	boppity	False
5	2013-01-06	0.856594	jefferson	foo	True
6	2013-01-07	1.236655	hamilton	foo	True
7	2013-01-08	-0.365152	roosevelt	bar	False
8	2013-01-09	0.899548	kennedy	zoo	False

Figure 1



```
In [3]: qgrid_widget # render the qgrid widget again so we don't have to scroll to see the changes in the scatter plot
```

	Add Row	Remove Row									
index	T	A	T	B	T	C	T	D	T	E	T
0		2013-01-01		0.48146		washington		foo		✓	
1		2013-01-02		-1.39376		adams		bar			
2		2013-01-03		-3.08918		washington		buzz			
3		2013-01-04		-1.13064		madison		bippity			
4		2013-01-05		1.30978		lincoln		boppity			
5		2013-01-06		-0.49786		jefferson		foo		✓	
6		2013-01-07		0.69652		hamilton		foo		✓	
7		2013-01-08		-0.65856		roosevelt		bar			
8		2013-01-09		-1.00344		kennedy		zoo			

# Jupyter Notebook Extensions

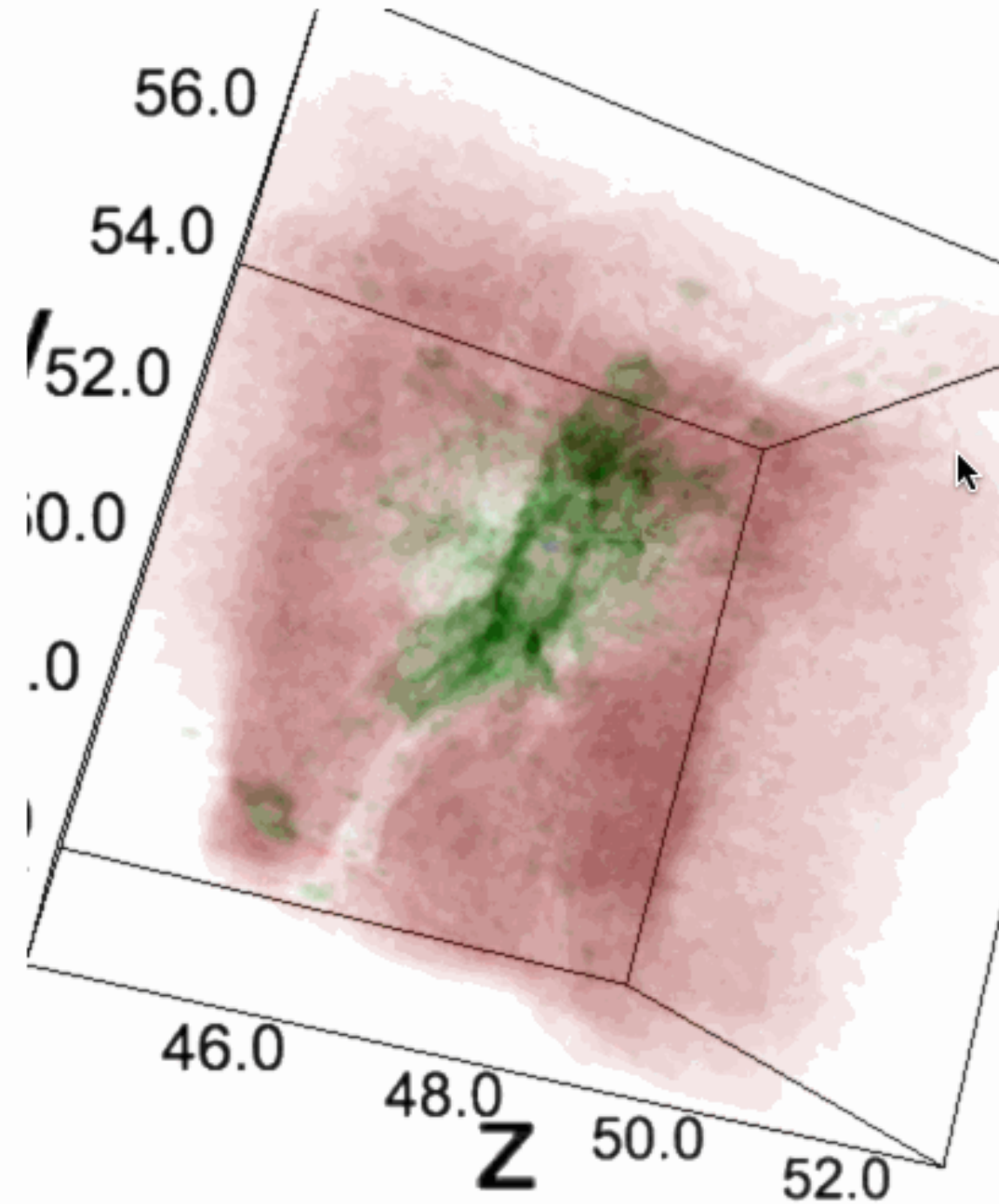
<https://ipyvolume.readthedocs.io>

Ipyvolume

```
import vaex
ds = vaex.open('/Users/maartenbreddels/datasets/aquarius/Aq-A-2-999-shuffled.hdf5')
ds.set_active_fraction(0.2)
ds.plot_widget('x', 'y', 'z', f='log', extent=[[40, 60]]*3, backend='ipyvolume', sha...
```

le...  0.10  0.50  0.1

opac...  0.01  0.05  0.1



opacity  1.00 brightness  1.00

progress

Astronomical data cube: Radio observations

jupyter head Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [13]: fig = ipv.figure()
vol = ipv.volshow(head, extent=[[0,]])
ipv.show()
```

lev...  0.23  0.37  0.49

opaci...  0.04  0.09  0.20

lev...  0.26  0.50  0.90

opaci...  0.00  0.05  0.10

opacity  0.251 brightness  1.58

opacity  0.0100 brightness  1.26

```
In [146]: vol.rendering_method = 'NORMAL'
vaq.rendering_method = 'MAX_INTENSITY'
vaq.rendering_method = 'NORMAL'
```

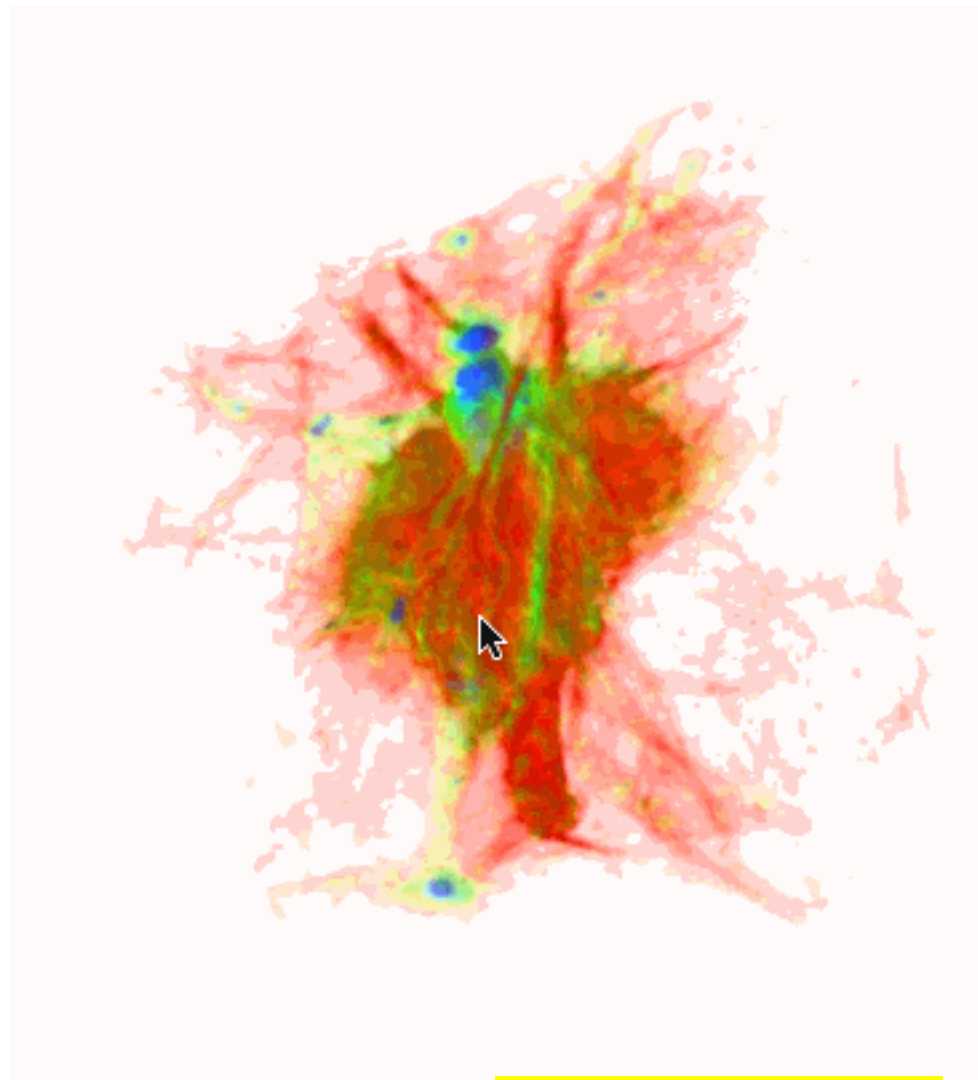
Medical data cube: scan of a male head



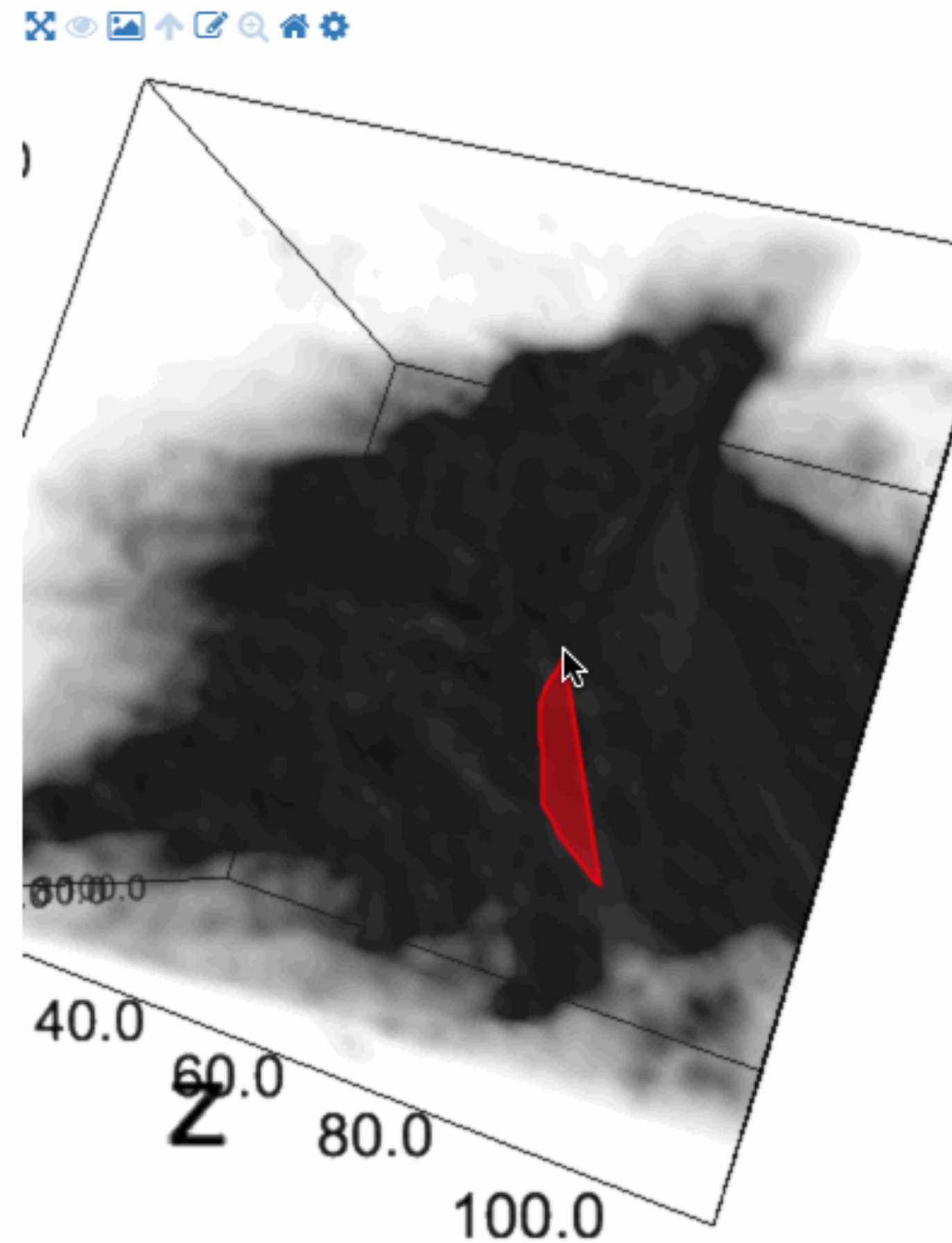
# Jupyter Notebook Extensions

<https://ipyvolume.readthedocs.io>

Ipyvolume



Exploring 100+ million rows by volume rendering a 3d histogram



General volume

Mode: lasso+ circle

rectangle

Show axes

Show movie maker

Glue-Jupyter uses ipyvolume for 3D rendering.

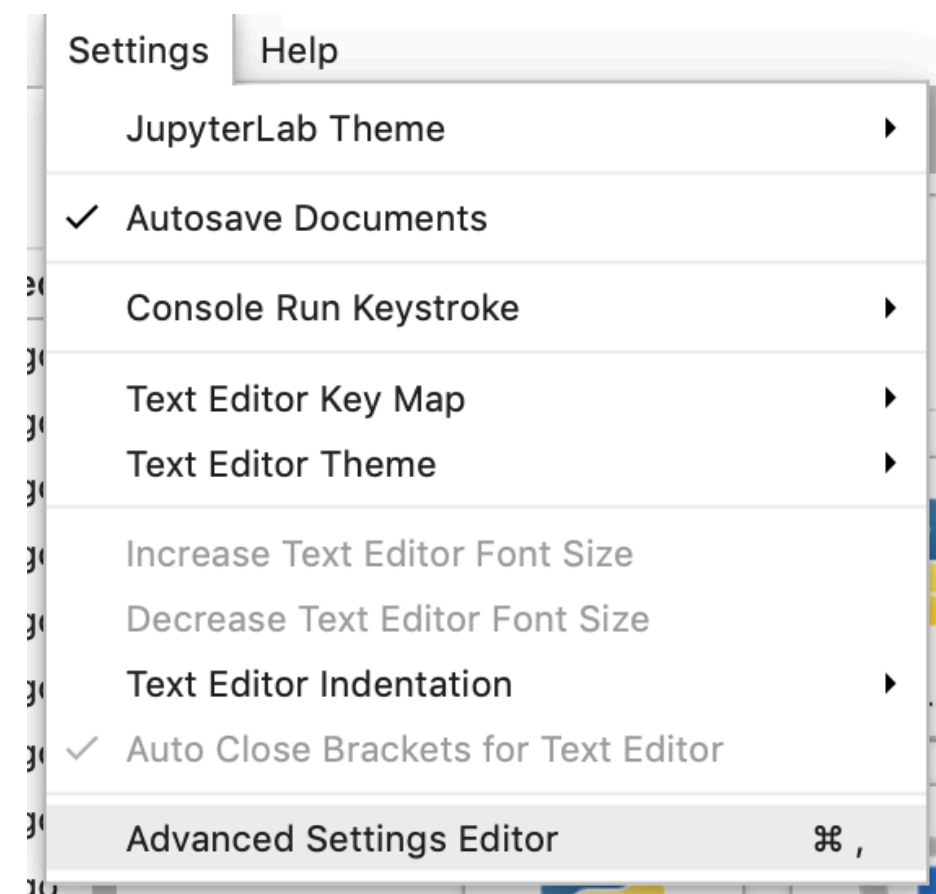
<https://github.com/glue-viz/glue-jupyter>

# JupyterLab Extensions

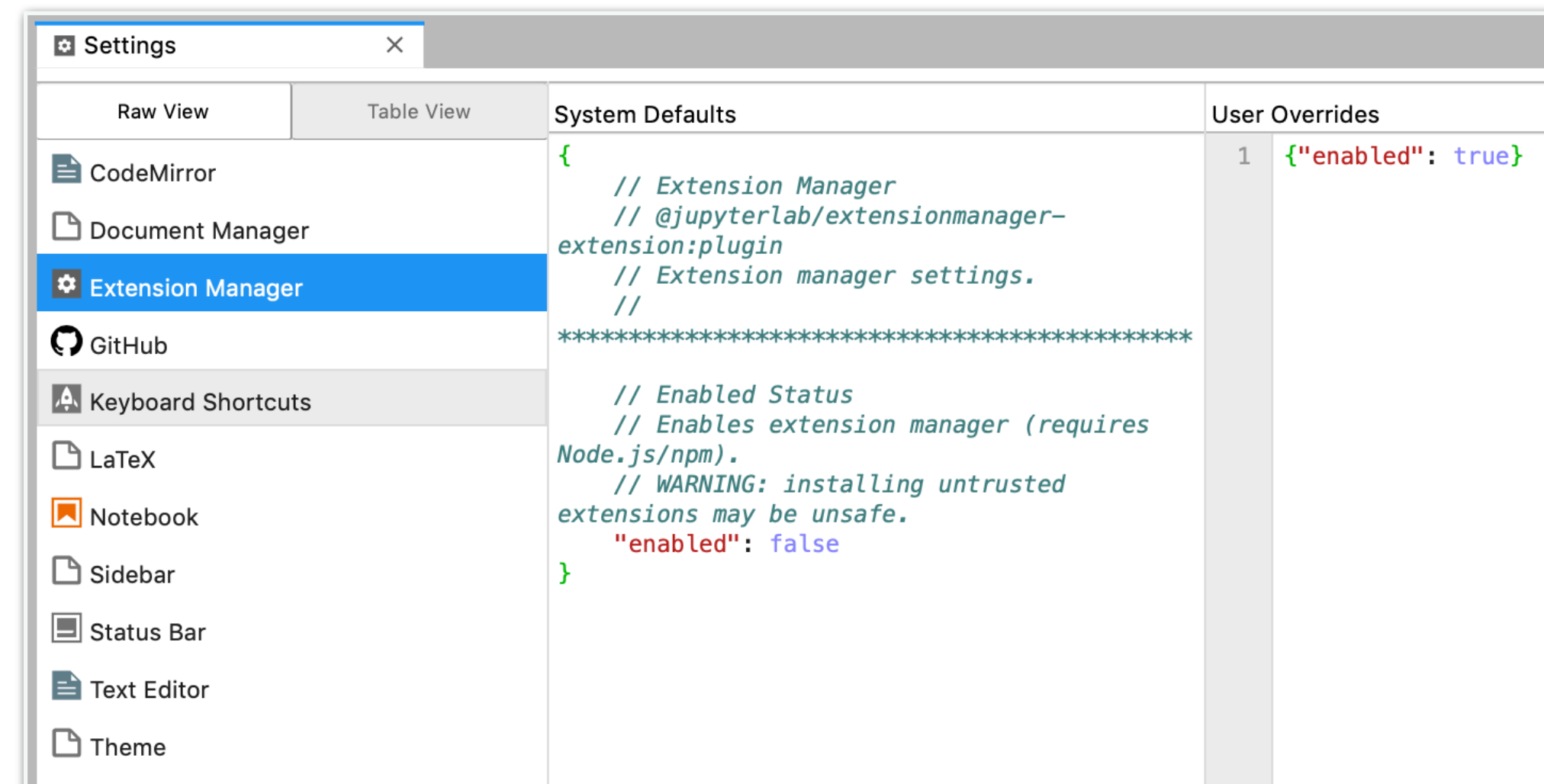
<https://jupyterlab.readthedocs.io/en/stable/user/extensions.html>

Use the extension manager

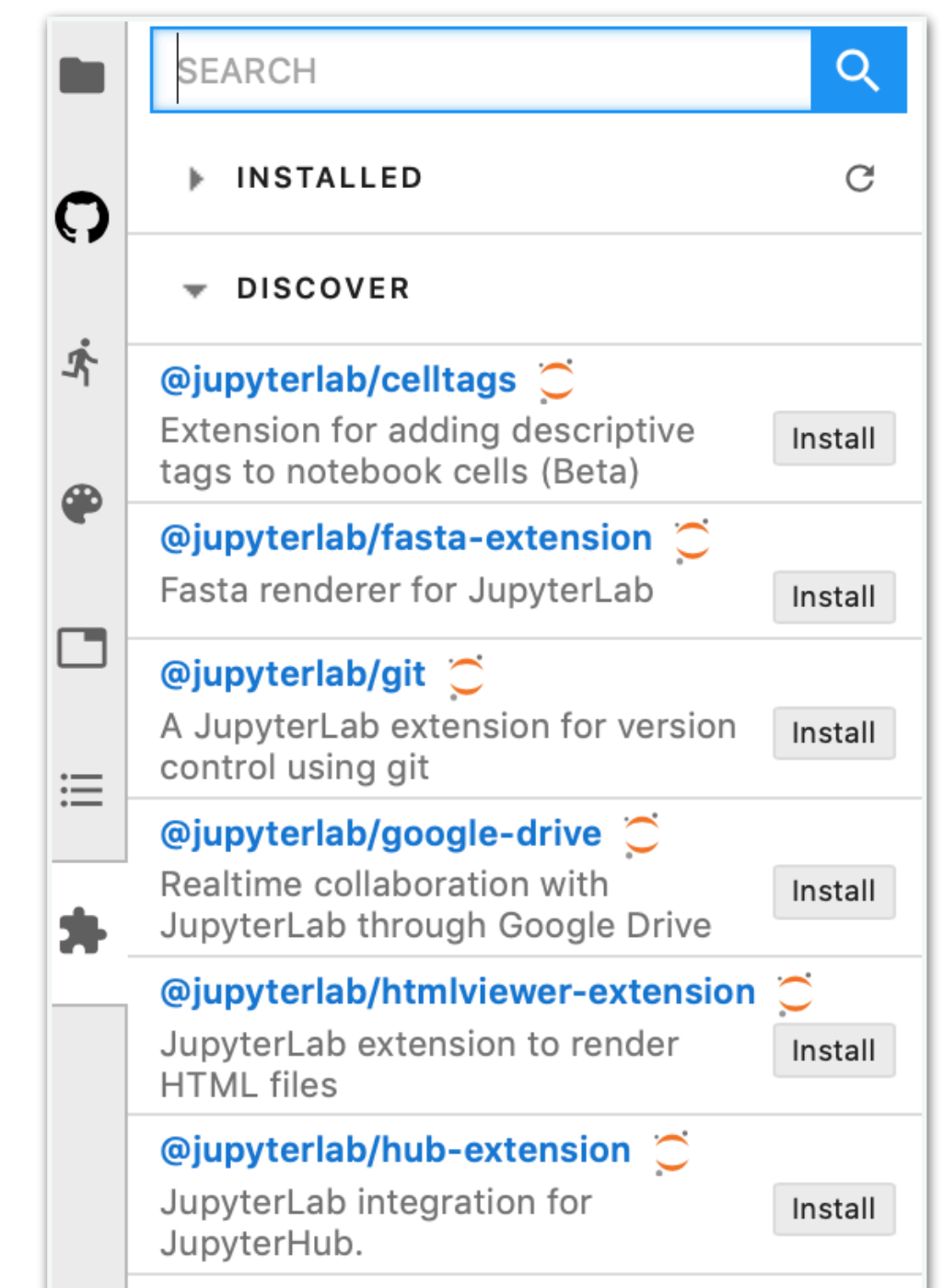
1. Go to Advance Settings



2. Enable Extension Manager



3. Discover /install extensions



# JupyterLab Workspaces

<https://jupyterlab.readthedocs.io/en/stable/user/urls.html>

JupyterLab sessions reside in workspaces, which contain the state of JupyterLab: the files that are currently open, the layout of the application areas and tabs, etc. When the page is refreshed or JupyterLab is re-started the workspace is restored.

Import/Export Workspaces with CLI

```
$ # Exports the default JupyterLab workspace
$ jupyter lab workspaces export
{"data": {}, "metadata": {"id": "/lab"}}
$
$ # Exports the workspaces named `foo`
$ jupyter lab workspaces export foo
{"data": {}, "metadata": {"id": "/lab/workspaces/foo"}}
$
$ # Exports the workspace named `foo` into a file called `file_name.json`
$ jupyter lab workspaces export foo > file_name.json
$
$ # Imports the workspace file `file_name.json`.
$ jupyter lab workspaces import file_name.json
Saved workspace: <workspaces-directory>/labworkspacesfoo-54d5.jupyterlab-workspace
```

Workspaces are stored in the web server.

Import/Export with URL params API

```
>/lab/workspaces/bar?clone=foo
```

```
>/lab/workspaces/foo/tree/path/to/notebook.ipynb?clone=bar
```

```
/lab/workspaces/foo?reset
```

```
>/lab/tree/path/to/notebook.ipynb?reset
```

# Multi-user environments



A **multi-user** version of the notebook designed for companies, classrooms and research labs.

**<https://jupyter.org/hub>**

- Sharing resources and computing environment.
- Users are not burdened with installation and maintenance tasks.
- Customizable and scalable, suitable for small and large large-scale infrastructures.
- Deployed anywhere i.e. commercial cloud providers, virtual machines, or even your own laptop hardware.
- Authentication is pluggable, supporting a number of authentication protocols (such as OAuth and GitHub).
- Users can get their work done in their own workspaces.

**<https://tljh.jupyter.org>**

ON UBUNTU

**<https://zero-to-jupyterhub.readthedocs.io>**

ON KUBERNETES

**a collaborative environment is not fully achieved since there is no easy sharing of notebooks**

JupyterLab work in progress GoogleDrive extension for Real Time Collaboration

Based on jupyter-drive extension to share notebooks on Google Drive **<https://github.com/jupyter/jupyter-drive>**

# Multi-language support

There exist Jupyter kernels for nearly any scripting language.

There exist extensions implemented as IPython magic commands to build **polyglot notebooks**.

These allow variable exchange across the different languages running in the same kernel.

We may see these magics commands as pipes dispatching the syntax and variables to new subprocesses.

This could also be done in pure Python and is possible due to the nature of Python as a glue language.

Available cell magics:

```
%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%javascript %%js %%late  
x %%markdown %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %  
%sh %%svg %%sx %%system %%time %%timeit %%writefile
```

`%%language` arguments

A block:

containing expressions and statement

from another:

language

may be options to  
exchange vars



[https://github.com/mgaitan/fortran\\_magic](https://github.com/mgaitan/fortran_magic)



<https://github.com/ebellm/ipython-idlmagic>



<https://pypi.org/project/idlmagic>



[https://pypi.org/project/ffi\\_magic](https://pypi.org/project/ffi_magic)



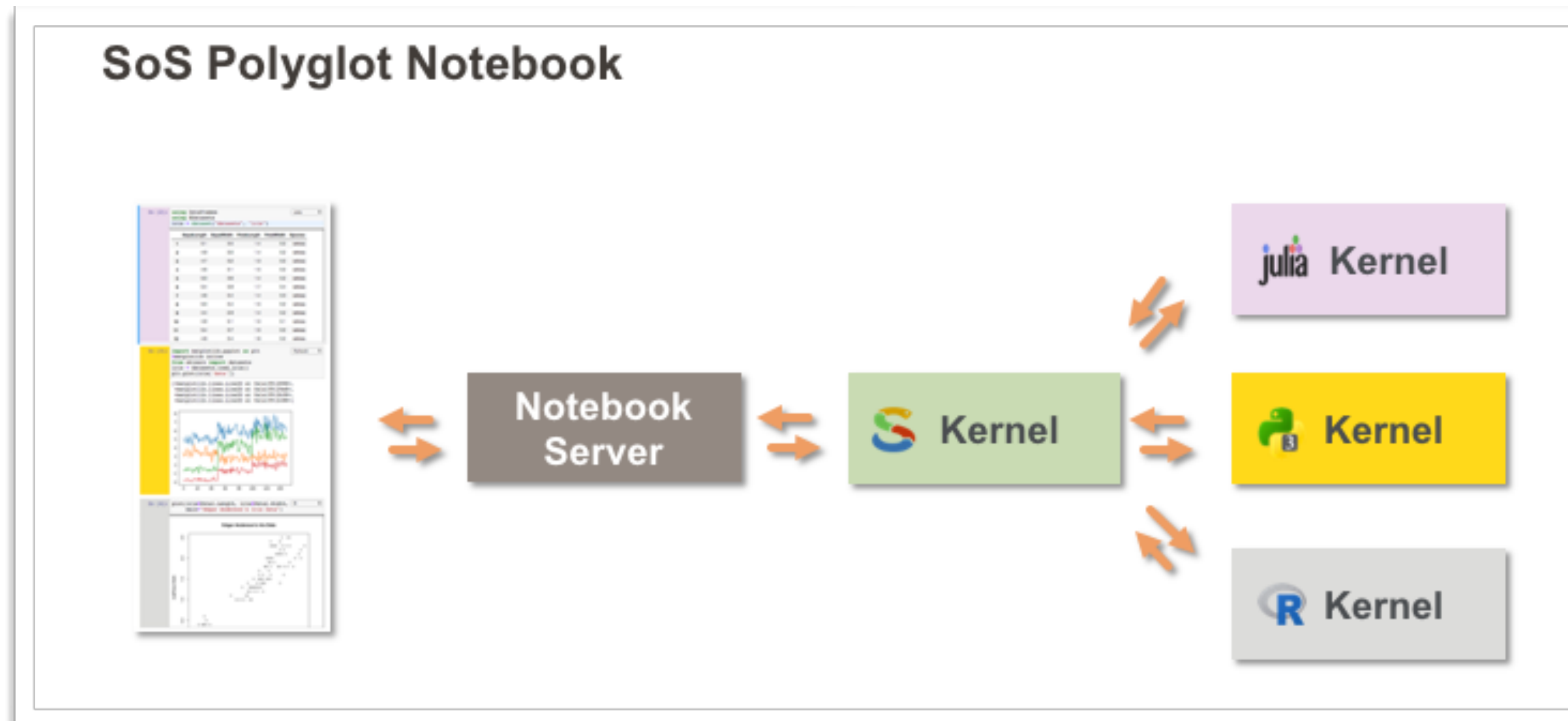
<https://rpy2.readthedocs.io>

ANY LANGUAGE

# Multi-language support

SoS Script of Scripts uses **multiple kernels** in one notebook.

[https://vatlab.github.io/sos-docs/doc/user\\_guide/multi\\_kernel\\_notebook.html](https://vatlab.github.io/sos-docs/doc/user_guide/multi_kernel_notebook.html)



The screenshot shows a web browser window displaying a SoS notebook interface. The browser address bar shows a URL ending in "/lab". The notebook interface includes a "Help" menu and a "Code" dropdown menu. The main content area shows three code cells:

```
[1]: %preview data
import pandas as pd
import numpy as np
data = pd.DataFrame(np.random.randn(4, 4), columns=list('ABCF'))
```

The output of the first cell is a table with columns A, B, C, and F:

	A	B	C	F
0	1.6571427	1.7836676	0.47313846	0.56679818
1	1.6608940	1.0082484	1.74608035	-0.02226743
2	-0.8151321	3.8990871	1.16560971	1.38280293
3	2.8252684	0.1767269	0.02486119	3.30067212

```
[2]: %get data
data = data + 1
data
```

The output of the second cell is the same table as above, but with the values in columns A, B, and C incremented by 1.

```
[3]: %get data --from R
data
```

The output of the third cell is a JavaScript object representing the data:

```
{ '0':
  { A: 1.6571426806,
    B: 1.7836676276,
    C: 0.4731384552,
```

# Scalability SCIENCE PORTALS

<https://jupyter-enterprise-gateway.readthedocs.io>

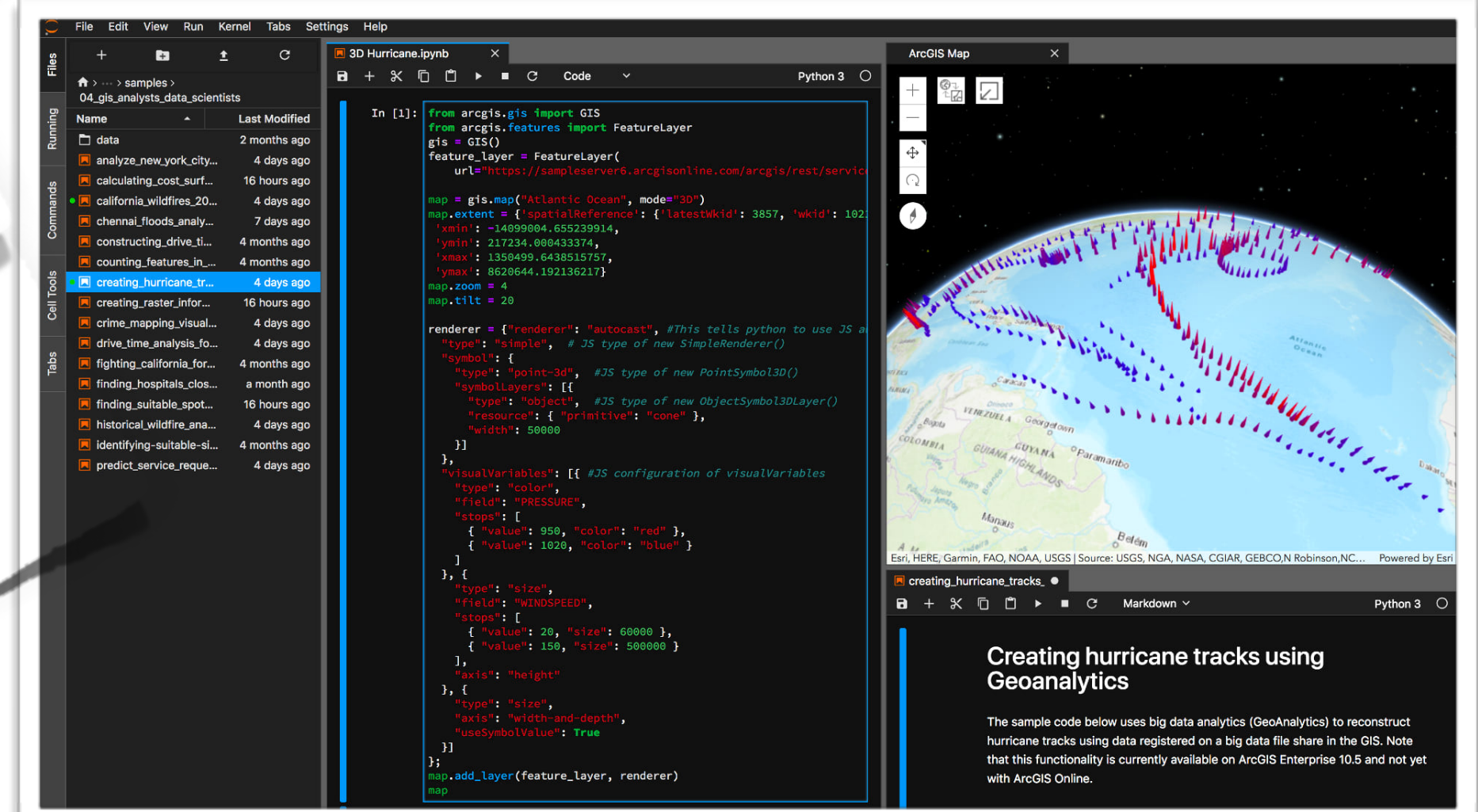
<https://github.com/jupyter-incubator/sparkmagic>

<https://github.com/jupyterhub/docker-spawner>

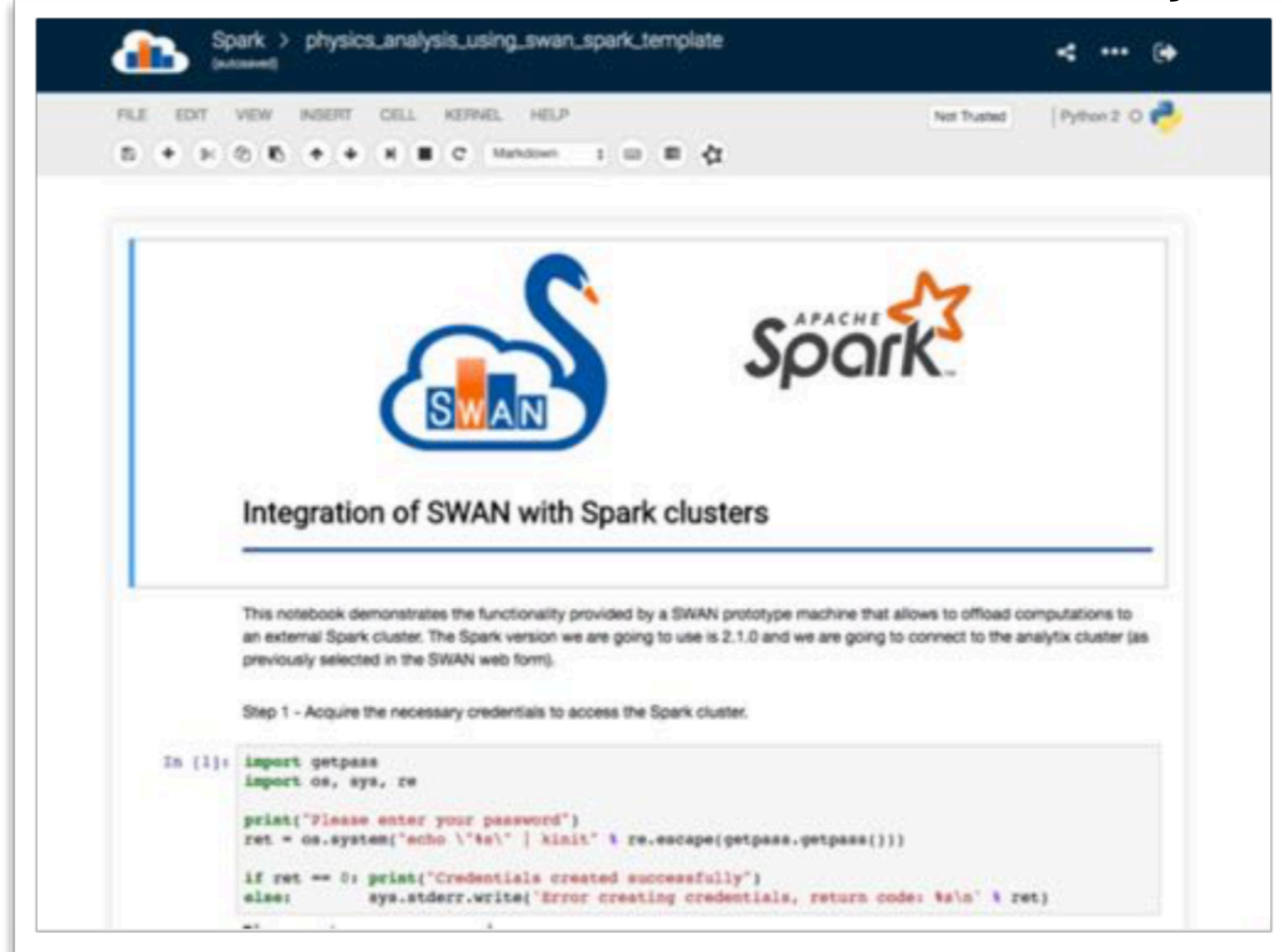
[https://github.com/jupyter-attic/kernel\\_gateway\\_bundlers](https://github.com/jupyter-attic/kernel_gateway_bundlers)

access to distributed computation  
 dockerized notebook servers  
 for each user  
 single notebooks deployed  
 as dockerized http  
 microservices

## ESRI Environmental Systems Research Institute

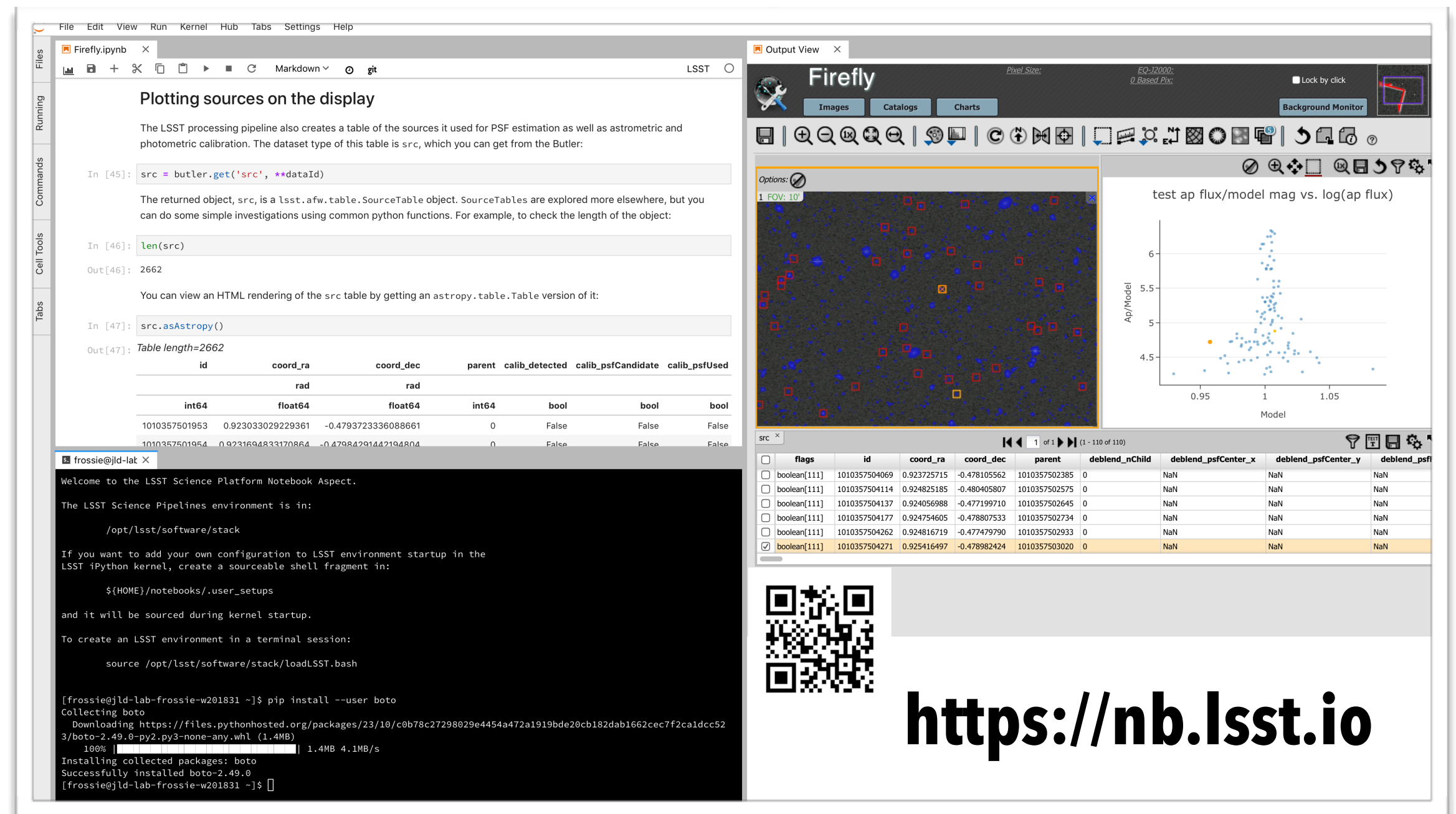


## CERN – SWAN Service for Web-based Analysis



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

## LSST Science Platform Aspect



<https://nb.lsst.io>

# Modularity

The easy solution

```
[1]: %%capture
      %run my_imported_notebook.ipynb
```

The sophisticated solution

<https://github.com/jupyter-incubator/contentmanagement>

```
In [1]: %load_ext jupyter_cms
```

```
In [2]: import mywb.cookbooks_demo.sklearn_cookbook as skcook
import mywb.cookbooks_demo.api_cookbook as apicook
```

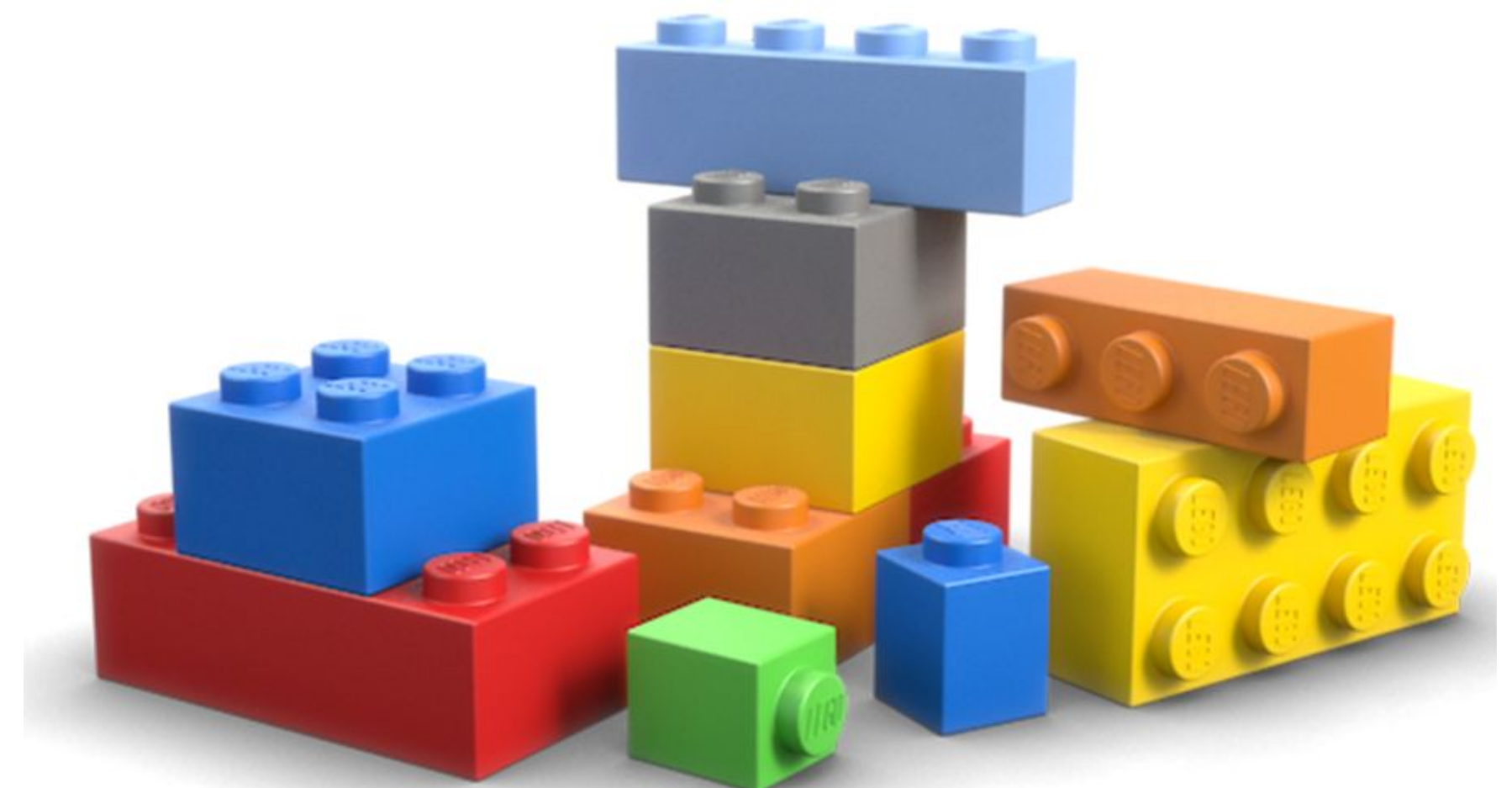
```
In [3]: from jupyter_cms.loader import load_notebook
```

```
In [4]: apicook = load_notebook('./api_cookbook.ipynb')
```

The very complex solution

<https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Importing%20Notebooks.html>

Transform the "very long notebook" into a short one using other notebooks as modules, which enables re-use and avoids duplication





# Automation

Papermill: parametrized notebooks



<https://github.com/nteract/papermill>

1. Declare some cells as a parameters cells.

2. Declare parameters values in YAML file.

```
In [ ]: 1 msgs = ["Hello, world!"]

In [ ]: 1 for msg in msgs:
        2     print(msg)
```

```
X:
  - 0.0
  - 1.0
  - 2.0
  - 3.0
linear_function:
  slope: 3.0
  intercept: 1.0
```

3. Run papermill with Python API.

3. Run papermill with CLI.

```
import papermill as pm

pm.execute_notebook(
    'path/to/input.ipynb',
    'path/to/output.ipynb',
    parameters = dict(alpha=0.6, ratio=0.1)
)
```

```
$ papermill local/input.ipynb s3://bkt/output.ipynb -f parameters.yaml
```



# Automation

Papermill + Scrapbook: workflows with parametrized notebooks

<https://nteract-scrapbook.readthedocs.io>  scrapbook

- Prototype automated tasks
- Reports building
- Testing and logging
- Benchmarking

Composing workflows based on the orchestration of several notebooks and their linked outputs.

## ∴ SHORT LINEAR NOTEBOOKS WITH SMALL CODE CELLS

### Some guidelines to make happy automated notebooks

- Low Branching Factor: Keep notebooks fairly linear. Not many conditionals or potential execution paths.
- Library Functions in Libraries: If you do end up with complex functions which you might reuse or refactor independently, these are good candidates for a coding library rather than in a notebook.
- Short and Simple is Better: A notebook which generates lots of useful outputs and visuals with a few simple cells is better than a ten page manual.

# Frontends / Viewers

nteract

<https://nteract.io>

Desktop app shipping node.js, R and Python as kernels. It can also access your locally defined kernels. Fires up notebook files from the desktop.

DataExplorer feature for enhanced visualization.

ipnb-quicklook / nbviewer app

<https://github.com/tuxu/nbviewer-app>

<https://github.com/tuxu/ipynb-quicklook>

Quicklook read-only viewers for MacOS X

Voila

<https://github.com/QuantStack/voila>

Renders read-only notebooks with interactive widgets. Execution of arbitrary code is disabled by default. The code cells are stripped by default producing a kind of documented interactive GUI from a notebook, that communicates with a dedicated kernel

Juno

<https://juno.sh>

iOS app for mobile platforms acting as a Jupyter frontend. It needs to connect to a remote notebook server (i.e. PC, AWS node, Azure notebooks, etc.)

# Publication

Please, make a small effort and transform exploratory into **explanatory** notebooks.

Publish in GitHub



Leading open code repository rendering notebooks very well indexed by search engines.  
Open collaborative community driven and linked to services (i.e. Collab, Binder, Zenodo)

Publish in HTML

**REPRODUCIBILITY** **VISIBILITY**



nbsphinx is a Sphinx extension that provides a source parser for \*.ipynb files.

<https://nbsphinx.readthedocs.io>

**nbinteract**

python package to generate interactive widgets in HTML pages by using Binder.

<https://www.nbinteract.com>

Publish an executable book



<https://github.com/jakevdp/PythonDataScienceHandbook>

<https://github.com/jakevdp/WhirlwindTourOfPython>



# Reproducibility

Make a Binder

<https://mybinder.org>

## Define your environment

- requirements.txt
- environment.yml
- Dockerfile



2 We build a Docker image of your repository

Binder will search for a dependency file, such as requirements.txt or environment.yml, in the repository's root directory ([more details on more complex dependencies in documentation](#)). The dependency files will be used to build a Docker image. If an image has already been built for the given repository, it will not be rebuilt. If a new commit has been made, the image will automatically be rebuilt.

3 Interact with your notebooks in a live environment!

A [JupyterHub](#) server will host your repository's contents. We offer you a reusable link and badge to your live repository that you can easily share with others.

Add Python kernels of different conda environments



```
conda activate myenv
conda install ipykernel
python -m ipykernel install --name myenv
jupyter kernelspec list
```

LOCALLY

Built-in browser

<https://alpha.iodide.io> BUILT-IN



Python is compiled to run on WebAssembly.

1 Enter your repository information  
Provide in the above form a URL or a GitHub repository that contains Jupyter notebooks, as well as a branch, tag, or commit hash. Launch will build your Binder repository. If you specify a path to a notebook file, the notebook will be opened in your browser after building.

Share on Google Colaboratory

<http://colab.research.google.com>



## Turn a GitHub repo into a collection of interactive notebooks

a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

### Build and launch a repository

GitHub repository name or URL

GitHub repository name or URL  GitHub

Git branch, tag, or commit

Git branch, tag, or commit

Path to a notebook file (optional)

Path to a notebook file (optional)  File

launch

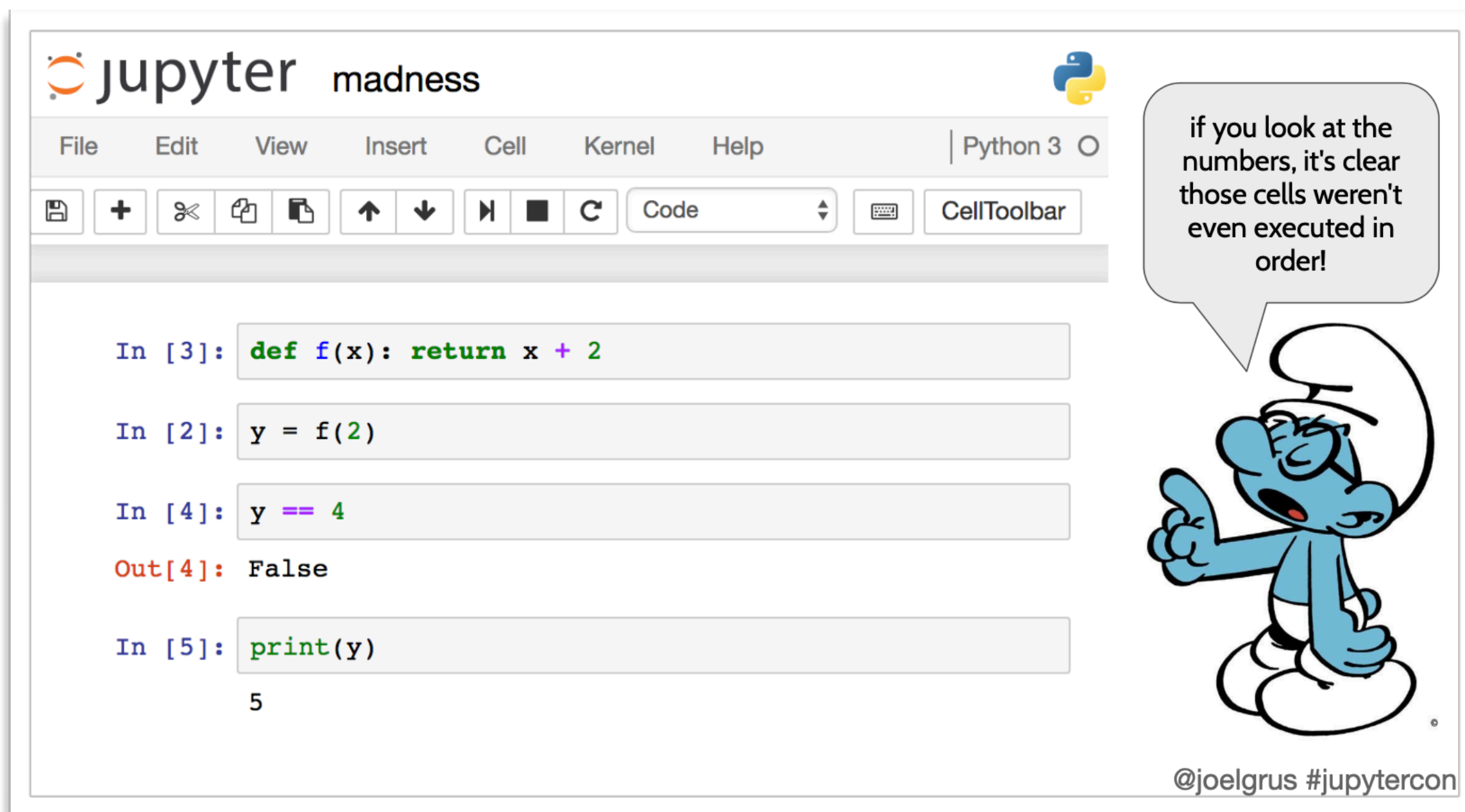
Copy the URL below and share your Binder with others:

Fill in the fields to see a URL for sharing your Binder.

Copy the text below, then paste into your README to show a binder badge: [launch binder](#)

# Reproducibility

Interactivity leads to complex/hidden state in non-linear notebooks



jupyter madness Python 3

File Edit View Insert Cell Kernel Help

Code CellToolbar

```
In [3]: def f(x): return x + 2
```

```
In [2]: y = f(2)
```

```
In [4]: y == 4
```

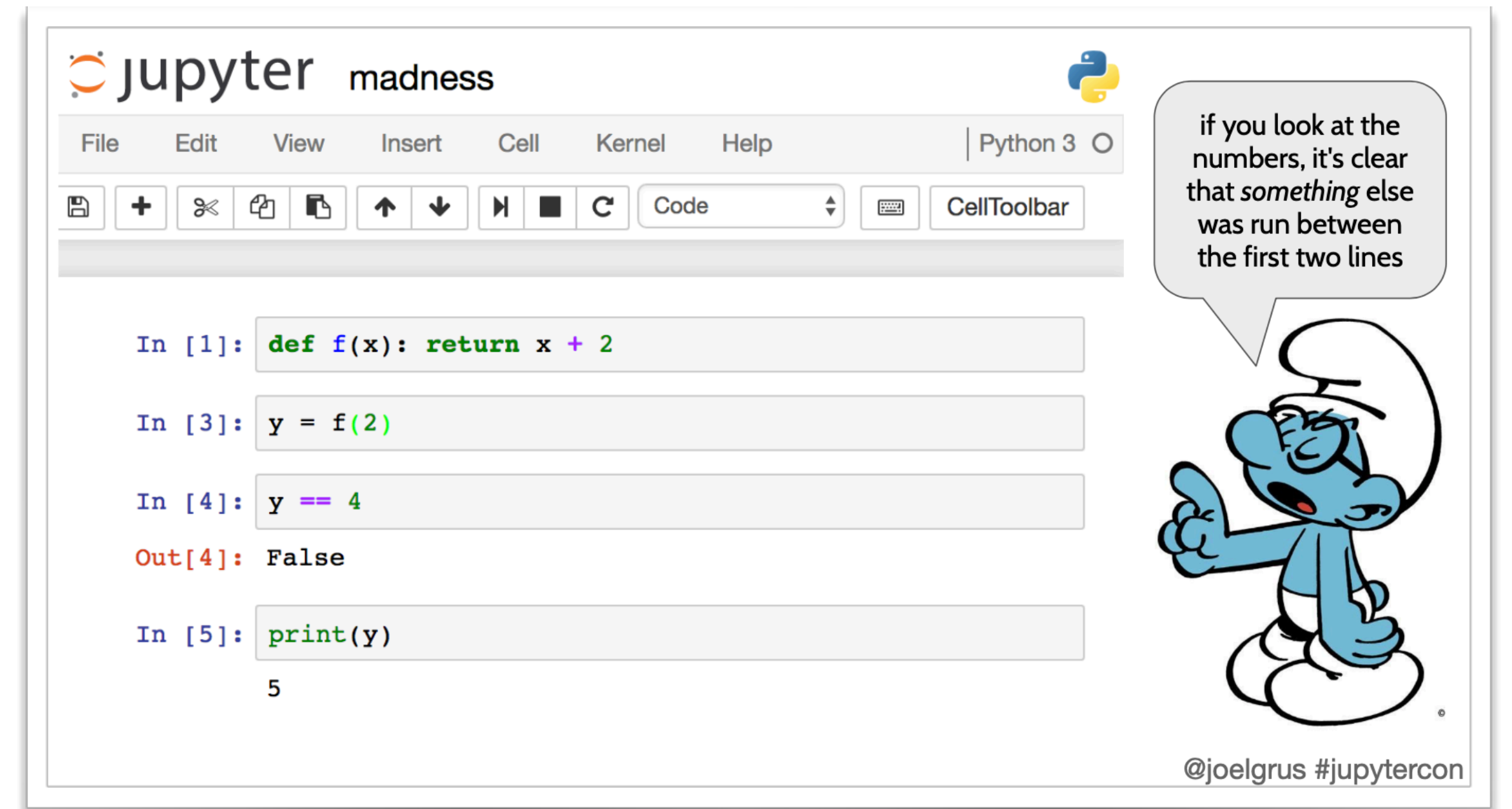
```
Out[4]: False
```

```
In [5]: print(y)
```

5

if you look at the numbers, it's clear those cells weren't even executed in order!

@joelgrus #jupytercon



jupyter madness Python 3

File Edit View Insert Cell Kernel Help

Code CellToolbar

```
In [1]: def f(x): return x + 2
```

```
In [3]: y = f(2)
```

```
In [4]: y == 4
```

```
Out[4]: False
```

```
In [5]: print(y)
```

5

if you look at the numbers, it's clear that *something* else was run between the first two lines

@joelgrus #jupytercon



Jake VanderPlas @jakevdp · 27 Nov 2017

Idea: Jupyter notebooks could have a "reproducibility mode" where:

- 1) Code cells are read-only once executed
- 2) New code cells cannot be inserted above previously executed cells
- 3) No cell can be executed until all previous cells are executed

49 131 599

<https://github.com/jupytercalpoly/reactivepy>

<https://github.com/stitchfix/nodebook>

# Versioning

Jupyter

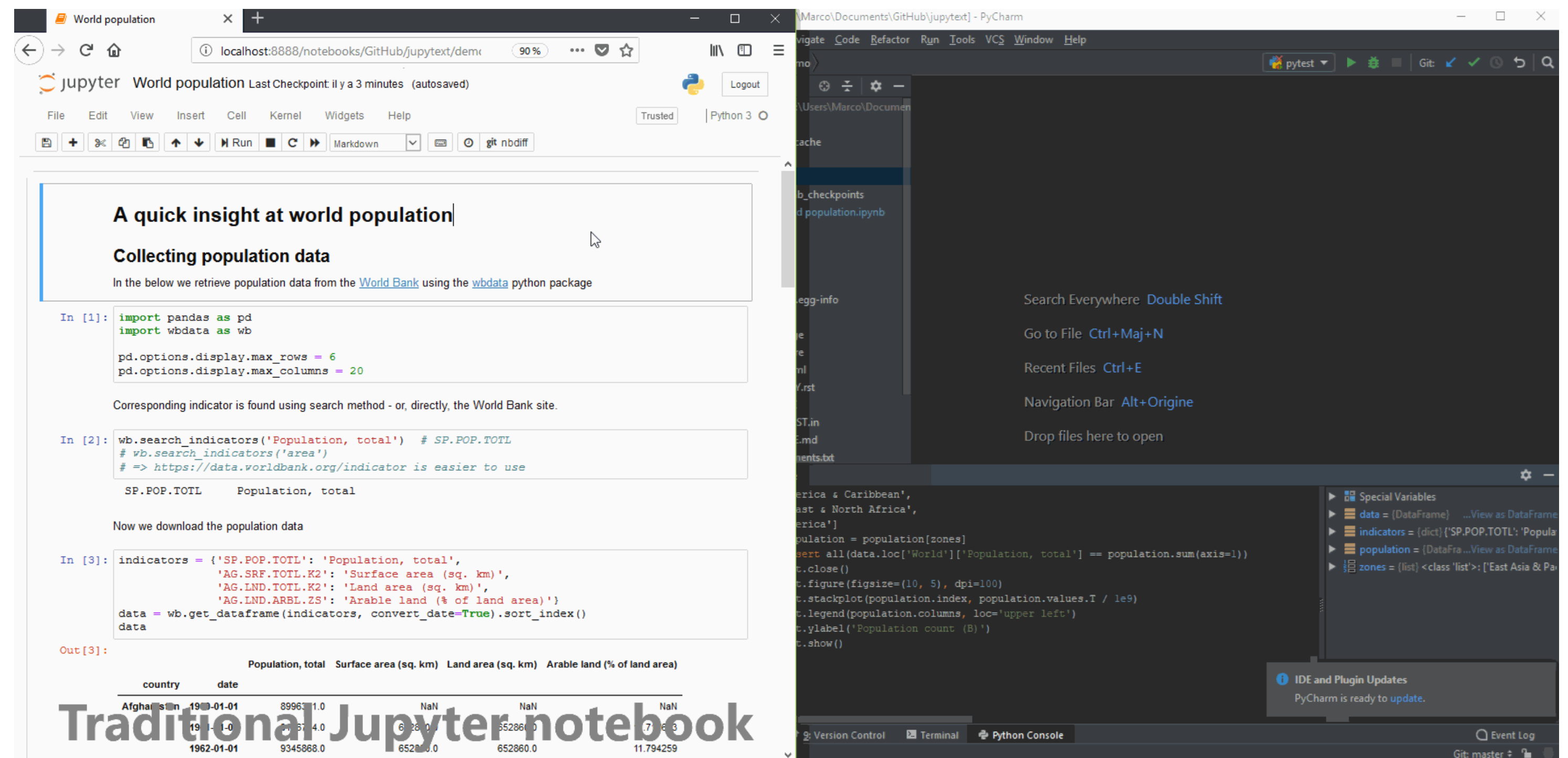
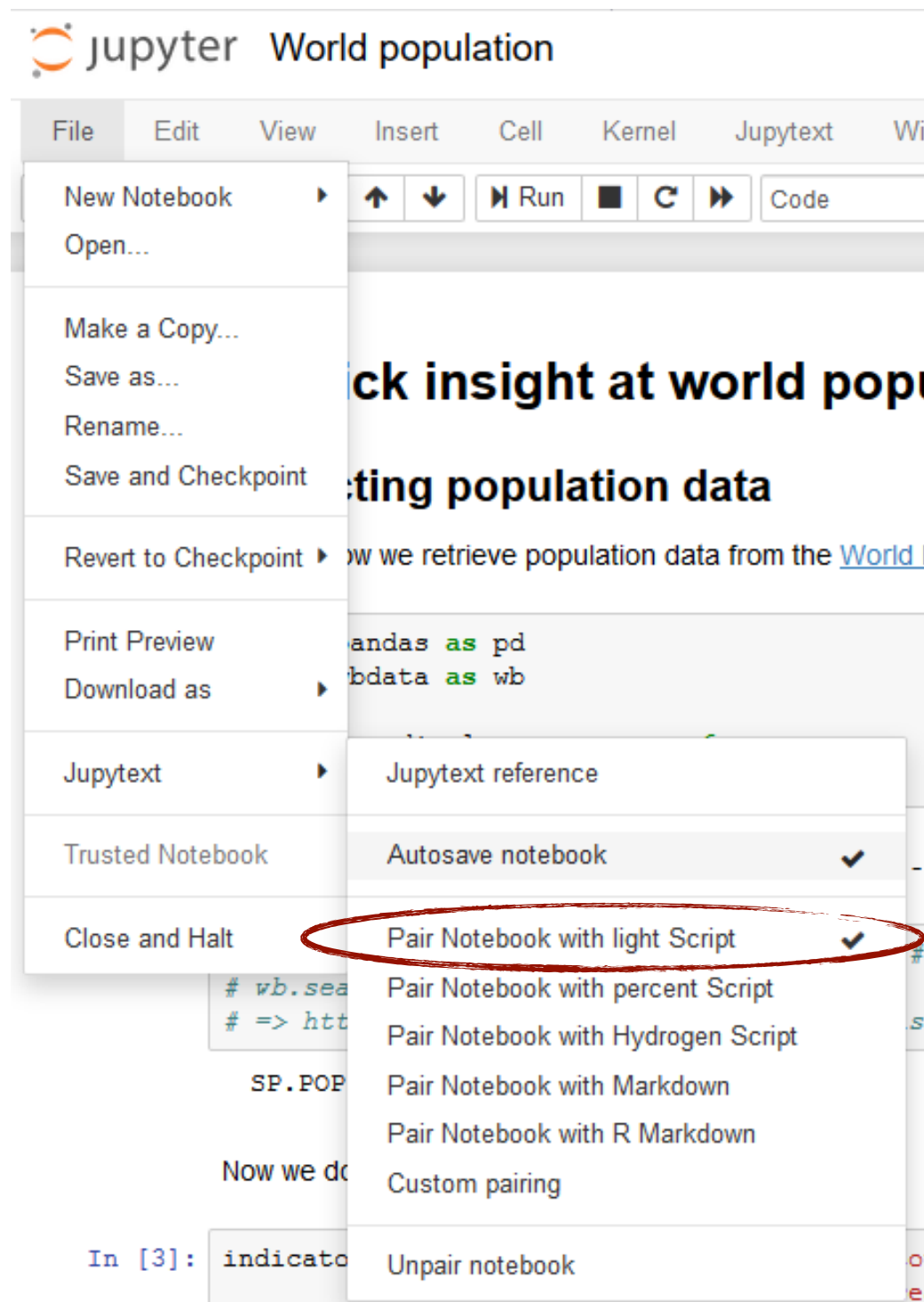
<https://github.com/mwouts/jupyter>

git diff is **#!@ish**

- small changes change metadata
- especially if output is in the notebook

strip the output cells

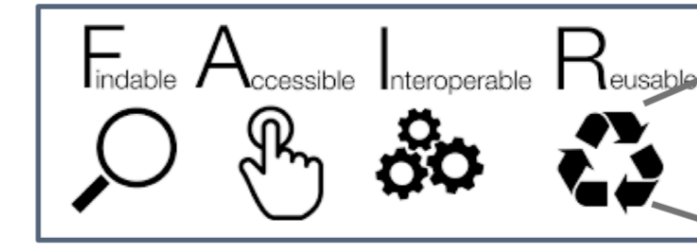
<https://github.com/kynan/nbstripout>



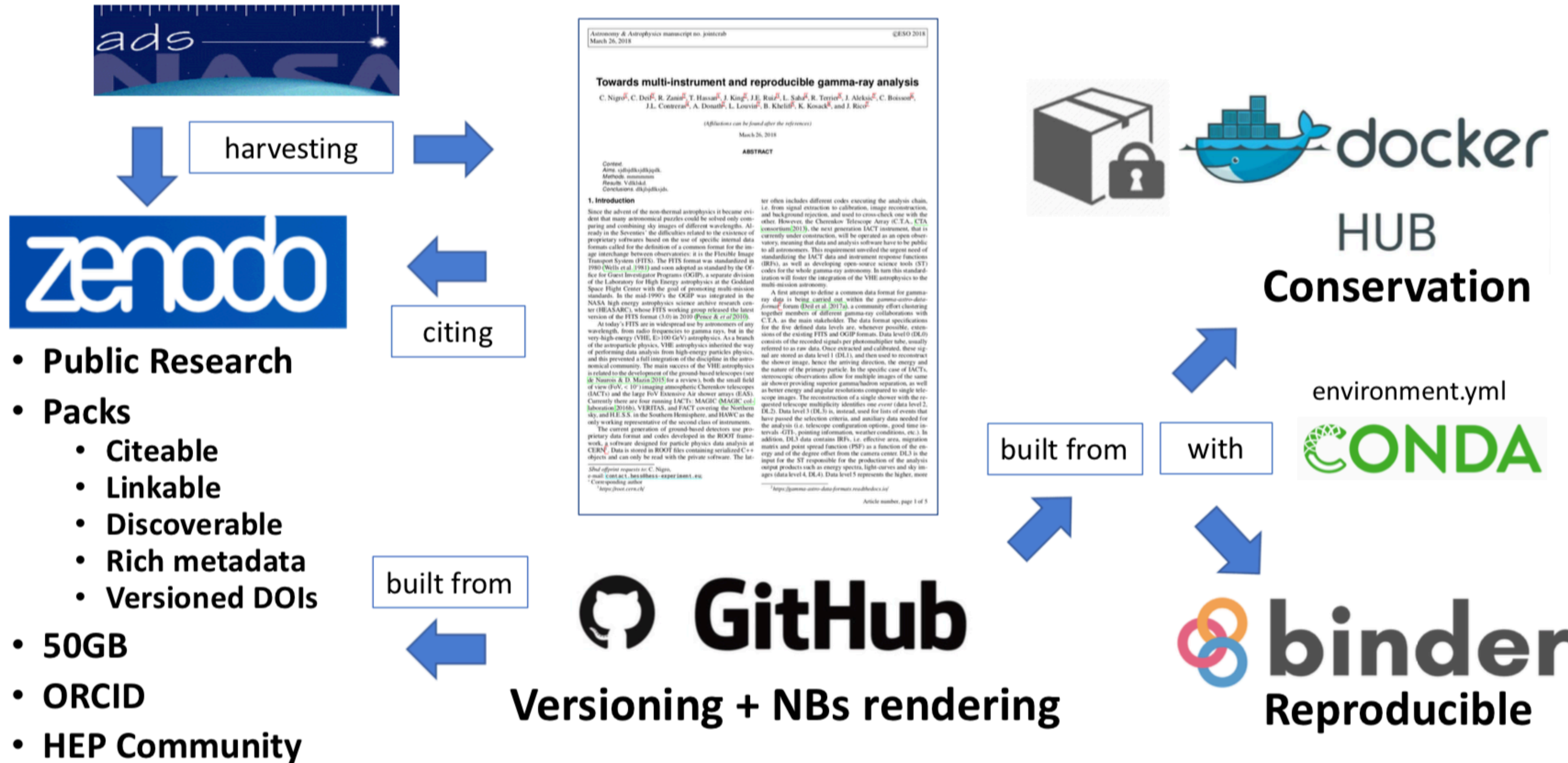
A paired notebook is an .ipynb notebook that is synchronized with a text representation - say a Python script. When the notebook is saved, Jupyter updates both files. The script has a clear diff history, focused on the input cells only

# Publishing reproducible papers

➤ All ESFRI projects (among the others CTA) need to follow the FAIR data principles & infrastructures



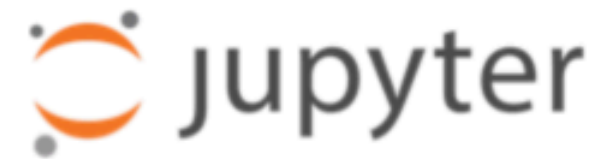
## An inter-linked storage for a reproducible pack





# Building versioned executable tutorials

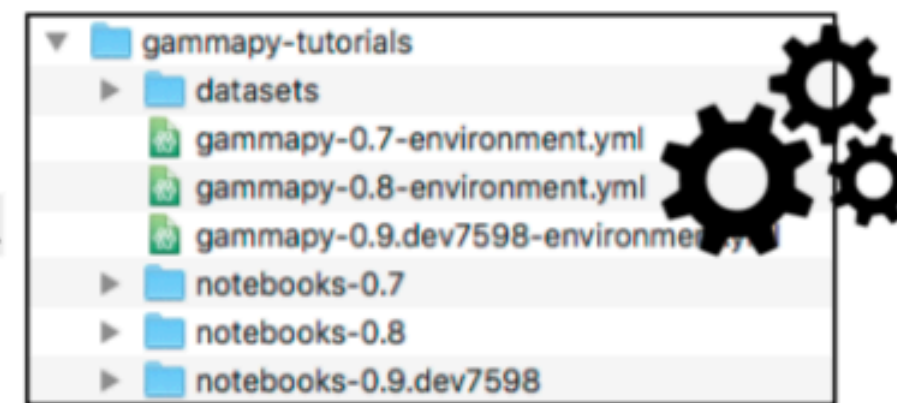
```
gammapy jupyter run
gammapy jupyter test
gammapy jupyter strip
```



Lookup files  
Datasets



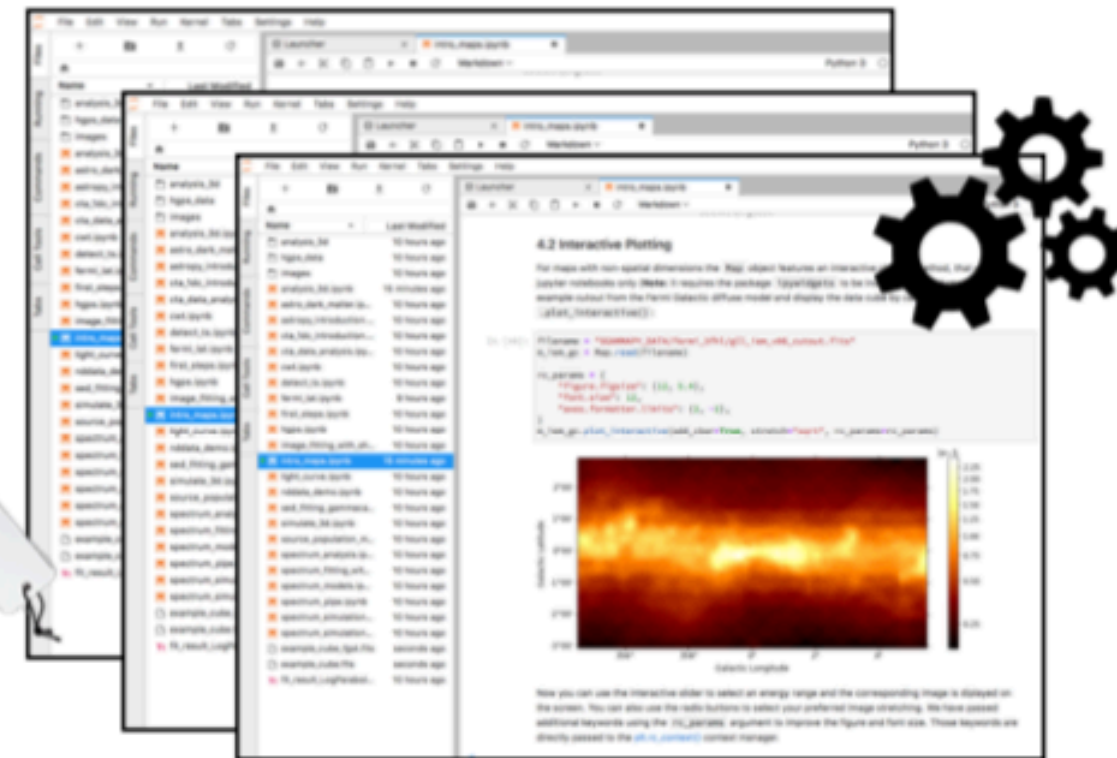
gammapy download



make docs



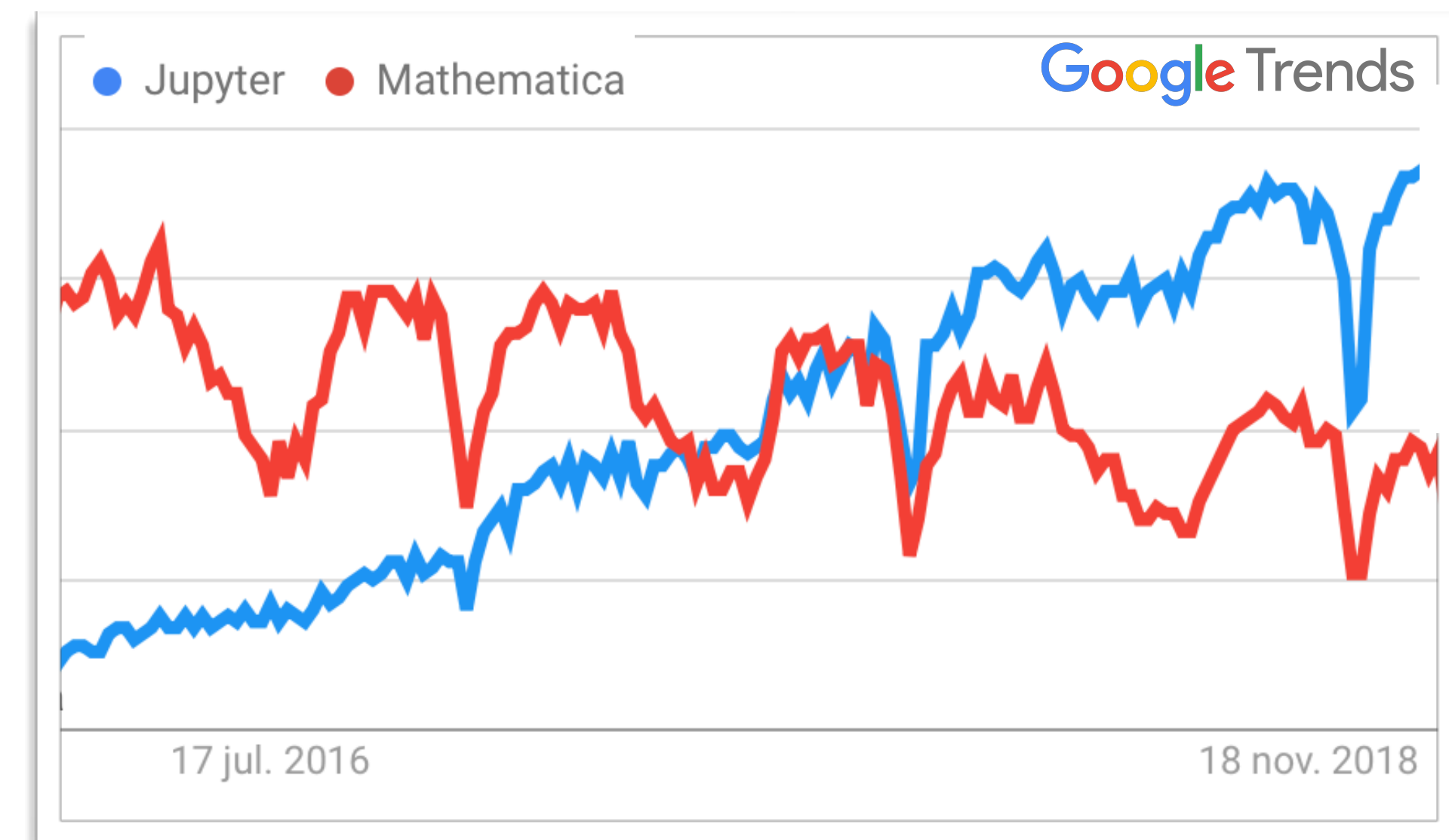
launch binder



- Tutorials integration: tutorials in the form of *Jupyter* notebooks are integrated into the software documentation with *Sphinx* + *nbsphinx* extension.
- Executable on-line: the documentation provides links to *myBinder* platform, where tutorials can be executed in the cloud using a versioned kernel provided by a *Dockerfile*.
- Version-coupling: the base-code, the tutorials and the *Dockerfile* are stored in the same *Github* versioned repository.
- Authoring and review: seamless code review for the tutorials with `gammapy jupyter` using diff comparisons in pull requests is possible, since the notebooks only store markdown and code cells with no outputs.
- Regression tests: tutorials execute in *Travis Continuous Integration* system, checking that their output cells do not throw any errors.
- Reproducibility: deterministic environments are defined for each version of the software in the form of *conda* configuration files, with pinned version numbers for each dependency package.
- Shipping: `gammapy download` command allows to retrieve versioned tutorials, composed of *Jupyter* notebooks, the datasets needed and the *conda* configuration file to build the environment.
- Maintainability: for each versioned environment we define its requirements, which tutorials to provide and where to fetch them with centralized index lookup files.

# A working methodology

- The web **browser** as the working desktop environment
- Capture exploratory and data analysis tasks into **log-like notebooks**
- Multi format display for rich **explanatory notebooks** - code, data, plots, equations, videos, etc..
- **Shareable, re-usable and executable** documentation / recipes
- Complementary executable format of **published books**
- First-class citizens in **GitHub** - easily discovered >3M notebooks
- Used as **executable tutorials** reduce the learning curve
- **Multi-language** support even possible in the same notebook
- Highly **extensible** and customizable in functionalities
- Local execution or **multi-user server-side** deploy
- **Scalable** and **parameterisable**



# A working methodology

- The web **browser** as the working desktop environment
- Capture exploratory and data analysis tasks into **log-like notebooks**
- Multi format display for rich **explanatory notebooks** - code, data, plots, equations, videos, etc..
- **Shareable, re-usable and executable** documentation / recipes
- Complementary executable format of **published books**
- First-class citizens in **GitHub** - easily discovered >3M notebooks
- Used as **executable tutorials** reduce the learning curve
- **Multi-language** support even possible in the same notebook
- Highly **extensible** and customizable in functionalities
- Local execution or **multi-user server-side** deploy
- **Scalable** and **parameterisable**

complex state allowed due to non-linear execution of cells

hard to test

hard to re-use into other formats i.e. copy/paste content

painful exploration of versioning and code-review

hard to make modular poorly factored code

frontend/environment dependencies

productivity flaws as code linting, type checking or tab completion