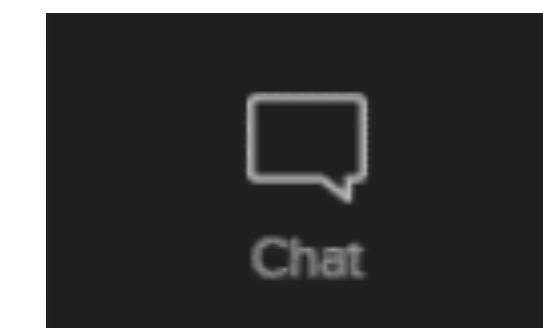
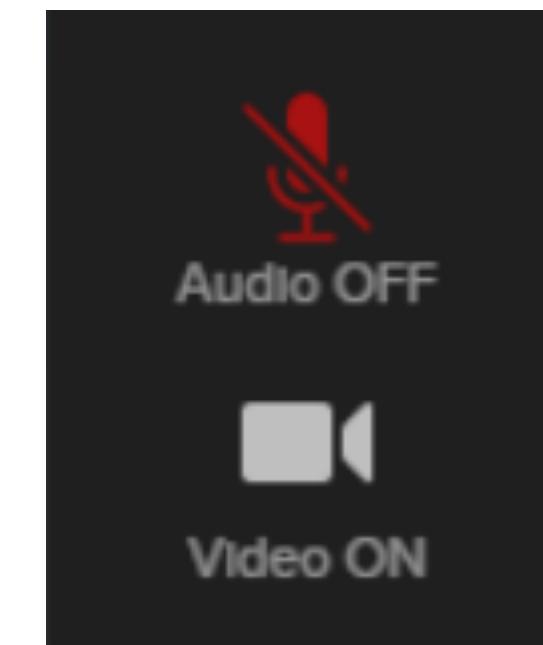


Welcome to the SGCI Webinar!

- We will be starting shortly.
- Your audio has been muted, and you are encouraged to **turn off your video** during the presentation.
- Controls for these are near the bottom of the right-side control panel for BlueJeans.
- You may submit questions at any time using **Chat**, and the moderator will share them with the presenter when appropriate.
- This presentation will be **recorded** and slides will be posted.



SGCI's week-long **Science Gateways Bootcamp** in October teaches strategies for successful gateway development & sustainability.

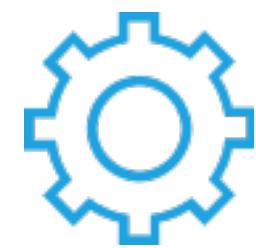
Apply by Friday 7/28: <https://sciencegateways.org/bootcamp>

A few brief words about the Science Gateways Community Institute (SGCI)

Our goal: To facilitate community *sharing of experiences, technologies, and practices* at little or no cost to community members through *NSF-funded, online and in-person resources and services*



Incubator: Learn best practices from our consultants or Bootcamp.



Extended Developer Support: Get direct, custom development help.



Scientific Software Collaborative: Find gateways or software components (or promote your own).



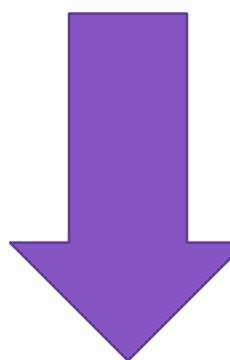
Community Engagement & Exchange: Engage with and learn from the gateways community.



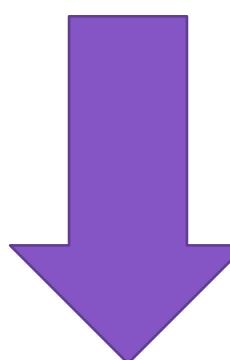
Workforce Development: Build your professional career as a student or young professional.

A quick favor at the end of this webinar...

NSF gives money to SGCI.



SGCI gives you a free webinar.



*Could you give SGCI
30 seconds of feedback?*



Jupyter

A Gateway for Scientific Collaboration and Education

The Project Jupyter Team

Carol Willing, Cal Poly
Brian Granger, Cal Poly
Fernando Perez, LBNL/Berkeley
Min Ragan-Kelley, Simula

The Larger Jupyter Team

@ProjectJupyter on Twitter

July 20, 2017



Proud
member
of the
Jupyter
community



Carol Willing

- Steering Council, [Project Jupyter](#)
- Core Developer, [Project Jupyter](#)
- Software Engineer, [Cal Poly SLO](#)
- Director, [Python Software Foundation](#)
- Core Developer, [CPython](#)
- Geek in Residence, [Fab Lab San Diego](#)



@willingcarol

Agenda

- Gateways
- Jupyter Notebook
- JupyterHub
- JupyterLab
- Next steps



2013 - Engage users

Convert from kern format to MusicXML

```
In [12]: c = converter.parse('/Users/carol/Downloads/duet/edokomuri.krn')
c.show()
```

Out[12]:

music21: a toolkit for computer-aided musicology

What is music21?

Music21 is a set of tools for helping scholars and other active listeners answer questions about music quickly and simply. If you've ever asked yourself a question like, "I wonder how often Bach does that" or "I wish I knew which band was the first to use these chords in this order," or "I'll bet we'd know more about Renaissance counterpoint (or Indian ragas or post-tonal pitch structures or the form of raijuks) if I could write a program to automatically write more of them," then music21 can help you with your work.

How simple is music21 to use?

Extremely. After starting Python and typing "from music21 import *" you can do all of these things with only a single line of music21 code:

```
-----  
Display a short melody in musical notation:  
converter.parse("tinyminiaton: 3/4 c1 d8 E q16 A g E")().show()
```

```
Print the twelve-tone matrix for a tone row (In this case the opening of Schoenberg's Fourth String Quartet):  
print(tonetools.rowToMatrix([2,1,6,10,5,3,4,9,8,7,6,11]))
```

```
or since all the 2nd-Viennese school rows are already available as objects, you can type:  
print(tonetools.getATonality('RowByNumber(1RowSchoenbergOp37)').matrix())
```

```
Convert a file from Humdrum's <+> data format to MusicXML, for editing in Finale or Sibelius:  
converter.parse('/Users/carthbert/docs/composition.krn').write('musicxml')
```

```
def closedPosition(self):  
    """  
    Returns a new Chord object with  
    the same pitches as self, but with  
    the lowest note as the root.  
    """  
  
>>> chord1 = Chord(["C#4", "G5",  
>>> chord2 = chord1.closedPosition()  
>>> print(chord2.tidy.value)  
<idis' e' g'>4  
    """  
  
    newChord = copy.deepcopy(self)  
    tempChordNotes = newChord.pitches  
    chordBassPS = self.bass().ps  
    for thisPitch in tempChordNotes:  
        while thisPitch.ps > chordBassPS:  
            thisPitch.octave = thisPitch.octave - 1  
    newChord.pitches = tempChordNotes
```

- Get Started with music21
- Browse the [music21 documentation](#)
- Download music21 from Google Code
- Get our latest news and updates at the [music21 blog](#)
- Read the [Frequently Asked Questions](#)
- Sign up for the [music21list](#) mailing list through Google Groups.

```
In [10]: sBach = corpus.parse('bach/bwv7.7')
sBach.show()
```

Out[10]:

bwv7.7.mxl

The figure shows a Mac OS X application window titled "tmpOph_K4.txt". The window contains two main sections: musical notation at the top and score representation code at the bottom.

Musical Notation: At the top, there are three staves of music. The top staff is in treble clef, the middle staff is in treble clef, and the bottom staff is in bass clef. All staves are in common time (indicated by 'C') and major key (indicated by a sharp sign). The notation includes various note values (eighth notes, sixteenth notes) and rests.

Score Representation Code: Below the notation is a text-based representation of the score. It starts with "Movement Name: bwv7.7.mxl" followed by several measures of music represented as sequences of note heads and rests. The code uses standard musical notation conventions like 'C' for common time and 'F#' for major key.

Michael Scott Cuthbert (cuthbert [at] mit.edu) is Associate Professor of Music and Homer A. Burnell Career Development Professor at M.I.T.

2014 - Break down barriers to entry

Intro to Python San Diego Python

- Start with a proven curriculum

<http://pyvideo.org/pycon-us-2013/a-hands-on-introduction-to-python-for-beginning-p.html>

- Hands on to engage students
- Takeaway notebooks reduce student stress

<https://github.com/pythonsd/intro-to-python>



2014 - Build a community

Andrea Zonca: notes about Python, high performance computing, data analysis

Posts:

How to create pull requests on Github	[FRI 30 JUNE 2017]
Deploy Jupyterhub on a Supercomputer with SSH Authentication	[TUE 16 MAY 2017]
Configure Globus on your local machine for GridFTP with XSEDE authentication	[WED 19 APRIL 2017]
Sample deployment of Jupyterhub in HPC on SDSC Comet	[SUN 26 FEBRUARY 2017]
Customize your Python environment in Jupyterhub	[FRI 24 FEBRUARY 2017]
Automated deployment of Jupyterhub with Ansible	[FRI 03 FEBRUARY 2017]
How to publish your research software to Github	[WED 01 FEBRUARY 2017]
Run Ubuntu in HPC with Singularity	[FRI 13 JANUARY 2017]
Jupyterhub Docker Spawner with GPU support	[WED 12 OCTOBER 2016]
Jupyterhub deployment on multiple nodes with Docker Swarm	[TUE 24 MAY 2016]
Quick Jupyterhub deployment for workshops with pre-built image	[THU 28 APRIL 2016]
Deploy Jupyterhub on a Virtual Machine for a Workshop	[SAT 16 APRIL 2016]
Use your own Python installation (kernel) in Jupyterhub	[MON 05 OCTOBER 2015]
IPython/Jupyter notebook setup on NERSC Edison	[THU 24 SEPTEMBER 2015]
IPython/Jupyter notebook setup on SDSC Comet	[THU 17 SEPTEMBER 2015]
Run Jupyterhub on a Supercomputer	[THU 02 APRIL 2015]
Accelerate groupby operation on pixels with Numba	[TUE 24 MARCH 2015]
Software Carpentry setup for Chromebook	[TUE 10 FEBRUARY 2015]
Zero based indexing	[WED 22 OCTOBER 2014]
Write unit tests as cells of IPython notebooks	[TUE 30 SEPTEMBER 2014]

WORDs Workflows for Data Science Center of Excellence

Team

software carpentry

Past Workshops

Scripps Research Institute: Nov 15-16, 2012
Salk Institute: Jun 09-10, 2013
Scripps Institution of Oceanography, UC San Diego: May 29-30, 2014
UC San Diego: May 17-18, 2016
UC San Diego: Jul 18-22, 2016
Scripps Institution of Oceanography, UCSD: Sep 19-20, 2016
Biomedical Library Building Classroom 4: May 23-24, 2017

Credits: <http://zonca.org/> <https://words.sdsc.edu/team> <https://software-carpentry.org>

Serve users and foster collaboration

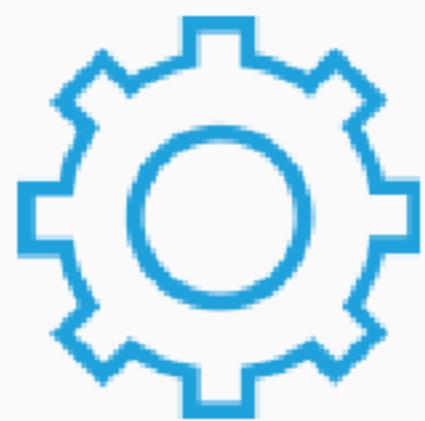


2017 and
beyond



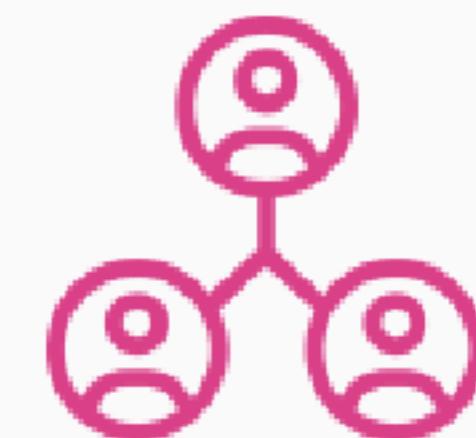
Incubator

Learn best practices
from our consultants
or Bootcamp



Extended Developer Support

Get direct, custom
development help



Scientific Software Collaborative

Find gateways or
software components
(or promote your own)



Community Engagement & Exchange

Engage with and learn
from the gateways
community



Workforce Development

Build your professional
career as a student or
young professional



Credit: <http://sciencegateways.org/>



“Project Jupyter serves not only the academic and scientific communities but also a much broader constituency of data scientists in research, education, industry and journalism...”

- *Fernando Pérez*

UC Berkeley

“

...we see uses of our tools that range from **high school education** in programming to the nation's **supercomputing** facilities and the **leaders of the tech industry**.

- *Fernando Pérez*

UC Berkeley

“

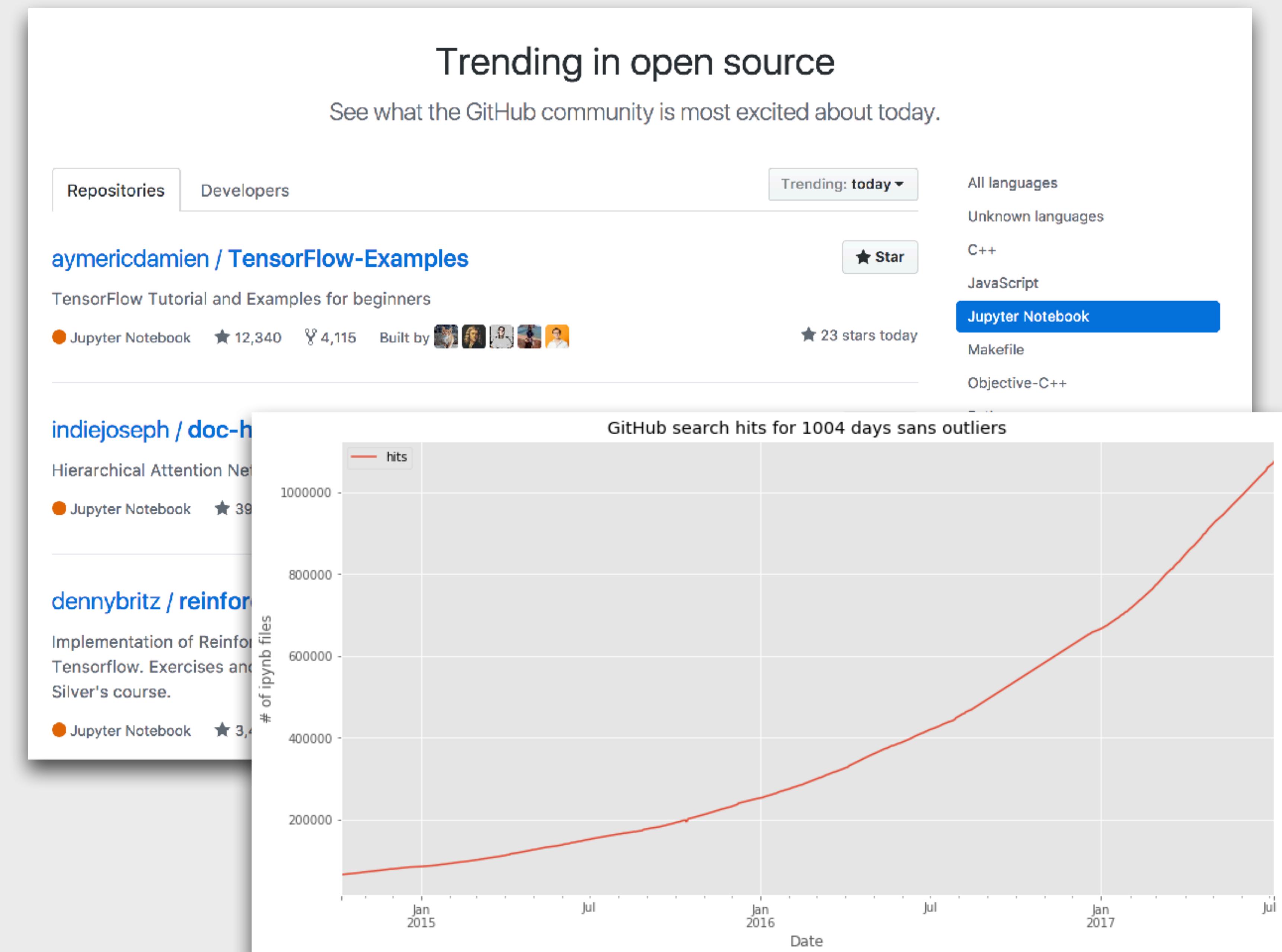
More than **a million people** are
currently using Jupyter for everything
from...

*-Prof. Brian Granger
Cal Poly*

>6M Users



Over 1M Notebooks on GitHub



“...analyzing massive gene sequencing
datasets to processing **images** from
the Hubble Space Telescope and
developing **models** of financial
markets.

*-Prof. Brian Granger
Cal Poly*

Scaling globally



Credit: <http://pythonineducation.org/>

“We are excited by the potential of Project Jupyter to **reach even wider audiences** and to contribute to **increased cross-disciplinary collaboration** in the sciences.

-Betsy Fader

Helmsley Charitable Trust

“

Jupyter Notebook... will enable **data exploration, visualization, and analysis** in a way that **encourages sound science** and **speeds progress**.

-Chris Mentzel

The Gordon and Betty Moore Foundation

Agenda

- Gateways
- Jupyter Notebook
- JupyterHub
- JupyterLab
- Next steps

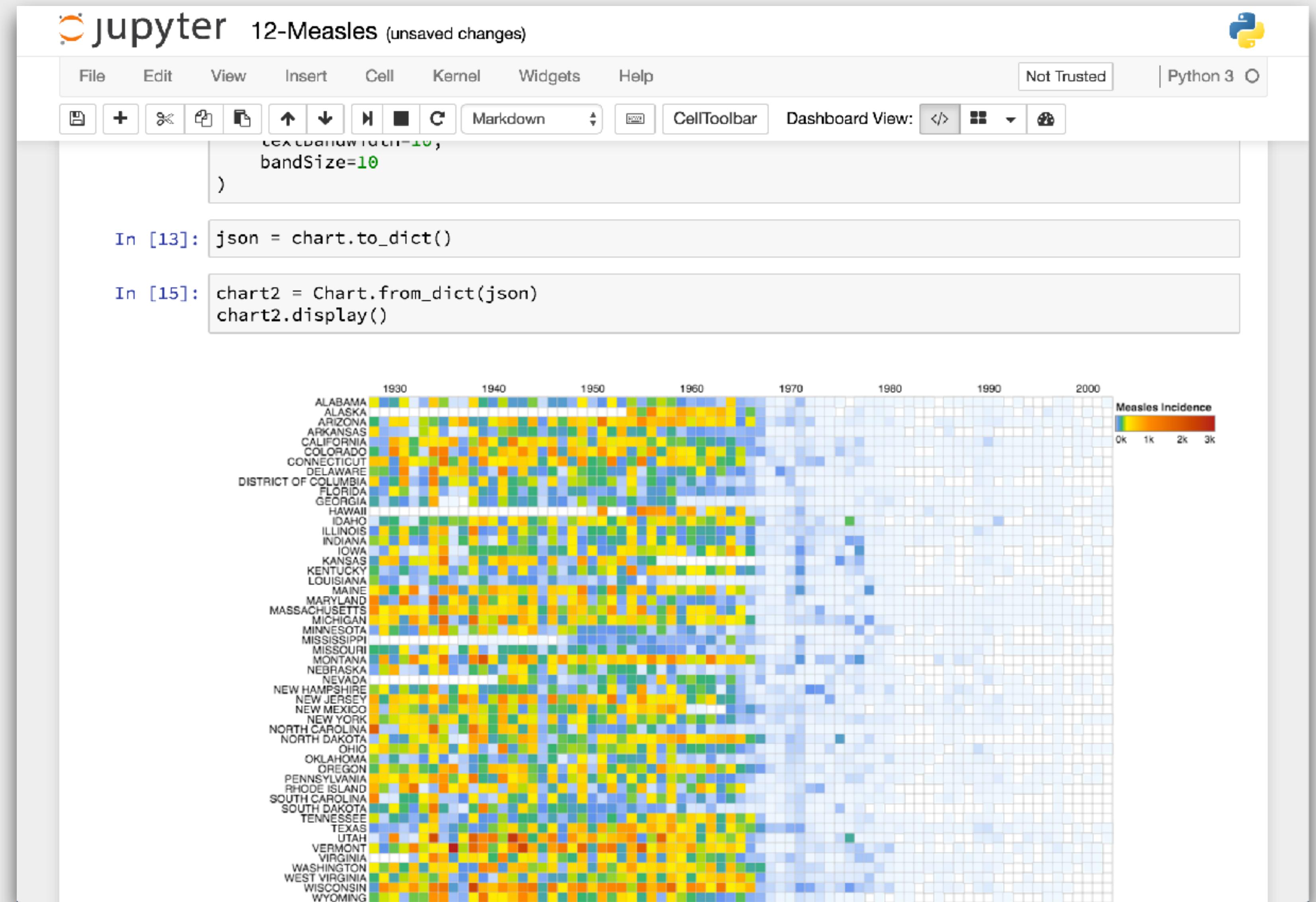
Jupyter Notebook



Interactive, Exploratory, Reproducible

- **Interactive**, browser-based computing environment
- **Exploratory** data science, ML, visualization, analysis, stats
- **Reproducible** document format:
 - Code
 - Narrative text (markdown)
 - Equations (LaTeX)
 - Images, visualizations
- Over 50 programming languages
- Everything open-source (BSD license)

Jupyter Notebook



A Jupyter Notebook document with a visualization of measles data.

Interactive Documentation

Engaging User Content

Rapid “what if” scenarios



ipywidgets

- **Docs** <https://ipywidgets.readthedocs.io>
- **Website** <http://jupyter.org/widgets.html>
- **Blog 6.0 release** <https://blog.jupyter.org/2017/03/01/ipywidgets-6-release/>
- **cookiecutter** to simplify creating new widgets

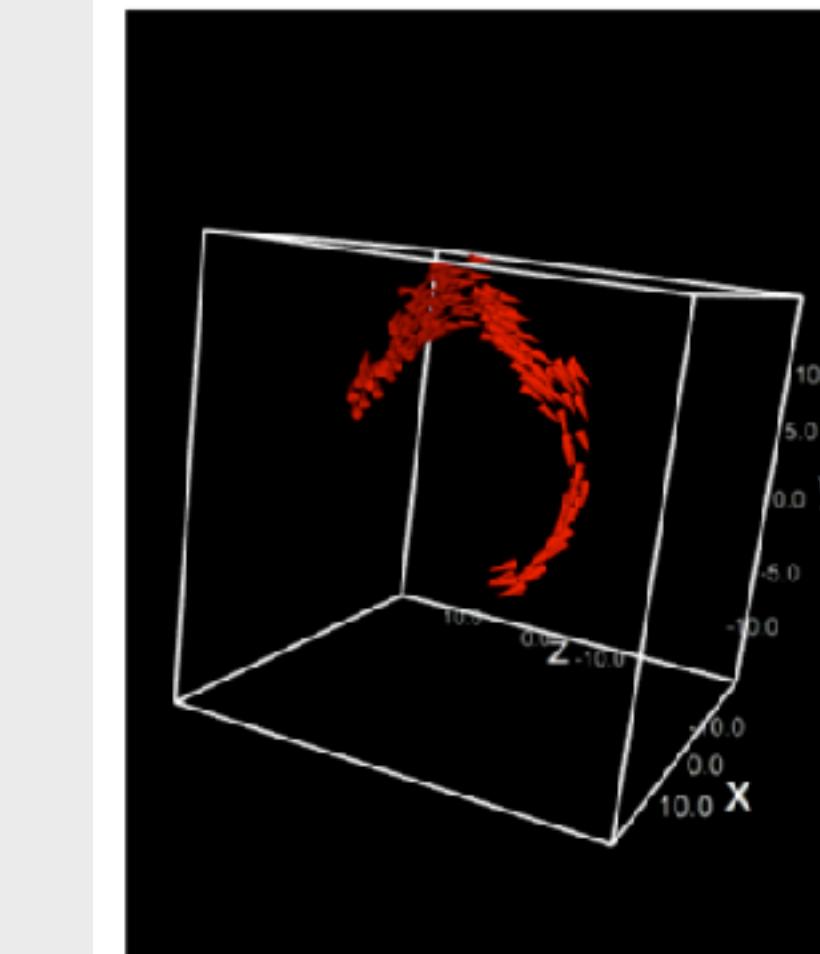
ipyvolume

3d plotting for Python in the Jupyter notebook based on IPython widgets using WebGL.

Example

```
import ipyvolume.pylab as p3
import numpy as np

fig = p3.figure()
q = p3.quiver(*stream.data[:,0:50,:200], color="red", size=7)
p3.style.use("dark") # looks better
p3.animate_glyphs(q, interval=200)
p3.show()
```



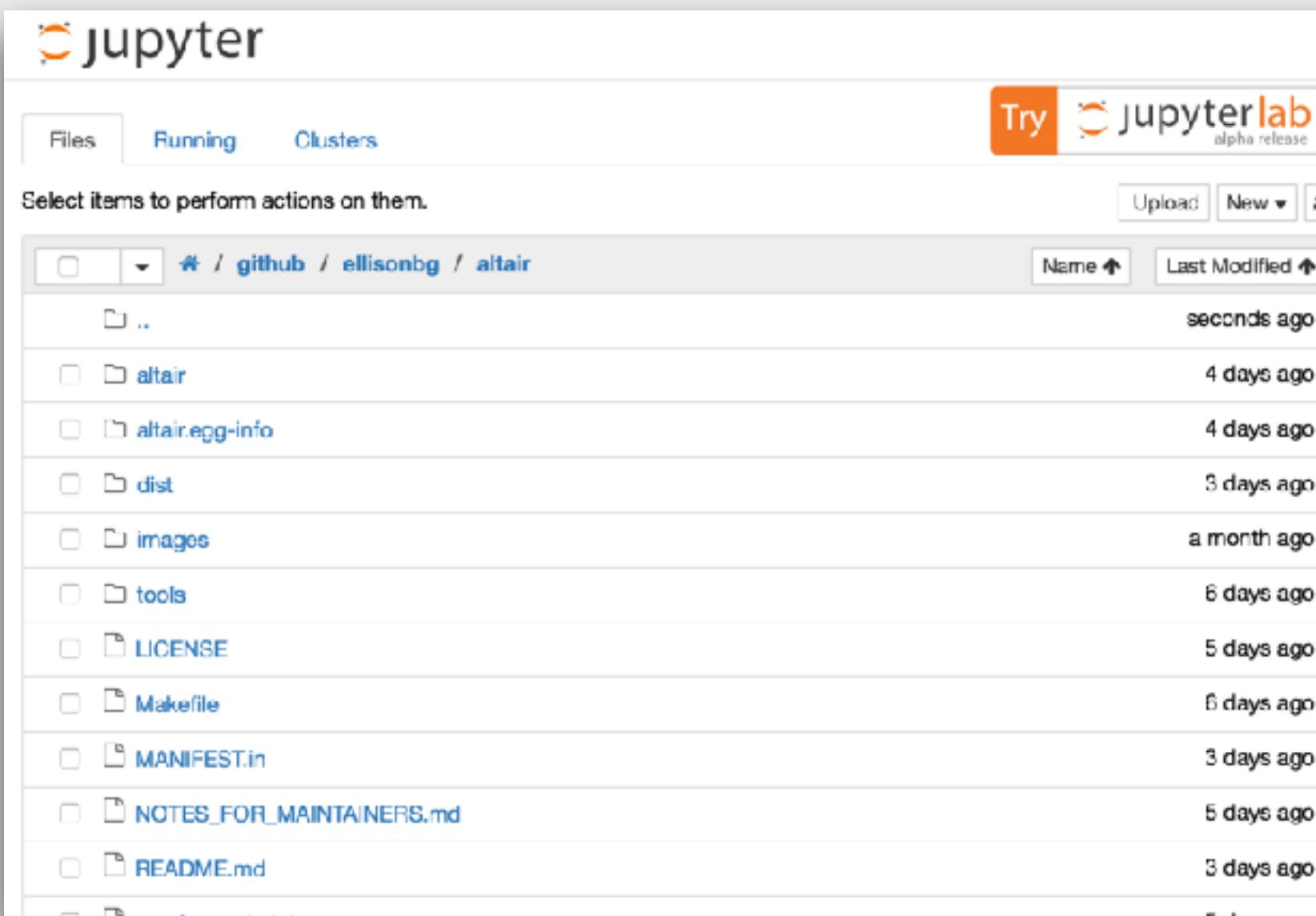
Pushing
the
boundaries



ipyvolume

- **3D interactivity** in notebooks
- **Innovation** by Maarten Breddels and team
- **Documentation** engages and demonstrates
- **Try and enjoy** at ipyvolume.readthedocs.io

Classic Jupyter: More Than Just Notebooks



This screenshot shows a Jupyter notebook cell with the title 'setup.py'. The code is written in Python and defines two functions: 'read' and 'version'. The 'read' function takes a path and returns the contents of the file. The 'version' function reads the package version from a python file like __init__.py. It uses regular expressions to search for the __version__ variable. The code is color-coded for syntax highlighting.

```
import io
import os
import re

try:
    from setuptools import setup
except ImportError:
    from distutils.core import setup

def read(path, encoding='utf-8'):
    path = os.path.join(os.path.dirname(__file__), path)
    with io.open(path, encoding=encoding) as fp:
        return fp.read()

def version(path):
    """Obtain the package version from a python file e.g. pkg/__init__.py
    See <https://packaging.python.org/en/latest/single_source_version.html>.
    """
    version_file = read(path)
    version_match = re.search(r'''^__version__ = ['"]([^"]*)['"]''',
                             version_file, re.M)
    if version_match:
        return version_match[1]
    else:
        raise RuntimeError('No version string found in %r' % path)
```

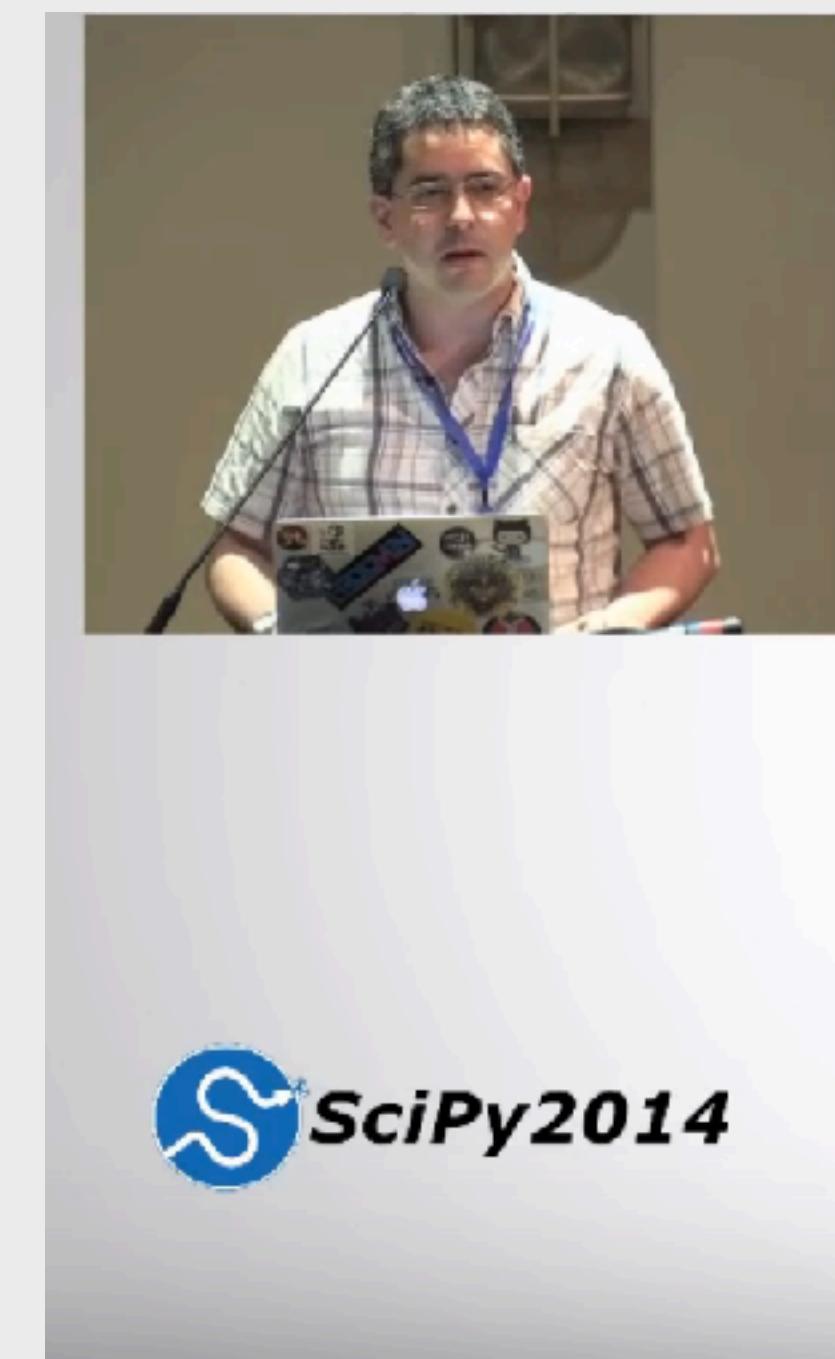
This screenshot shows a terminal window with a black background and white text. The user is in the directory '/github/ellisonbg/altair'. They run the command 'ls' which lists several files and directories: LICENSE, MANIFEST.in, Makefile, NOTES_FOR_MAINTAINERS.md, README.md, altair, and a subdirectory 'altair/notebooks/'. Inside 'notebooks/' are several Jupyter notebooks: 01-Index.ipynb, 02-Introduction.ipynb, 03-ScatterCharts.ipynb, 04-BarCharts.ipynb, 05-LineCharts.ipynb, 06-AreaCharts.ipynb, and 07-LayeredCharts.ipynb. There are also other files like altair.egg-info, dist, images, requirements.txt, and setup.py.



Agenda

- Gateways
- Jupyter Notebook
- JupyterHub
- JupyterLab
- Next steps

Jupyter for Science and Data Science



Galileo's *Sidereal Messenger*: 1610

OBSERVATIONS OF THE STARS

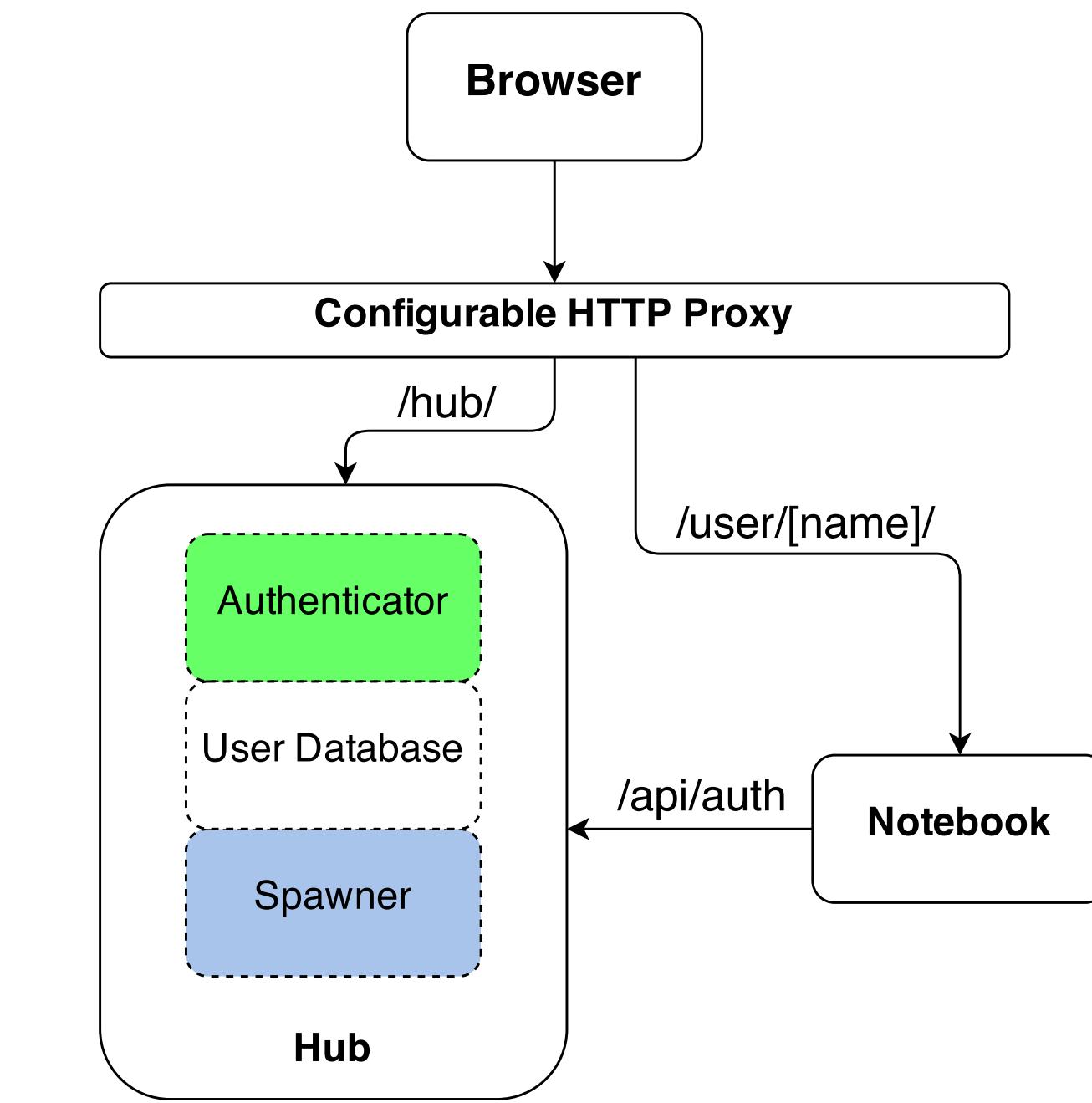
Ori. * * O * Occ.

was greater than the star furthest to the west; but both were very conspicuous and bright; the distance of each one from Jupiter was two minutes. A third star, certainly not in view before, began to appear at the third hour; it nearly touched Jupiter on the east side, and was exceedingly small. They were all arranged in the same straight line, along the ecliptic.

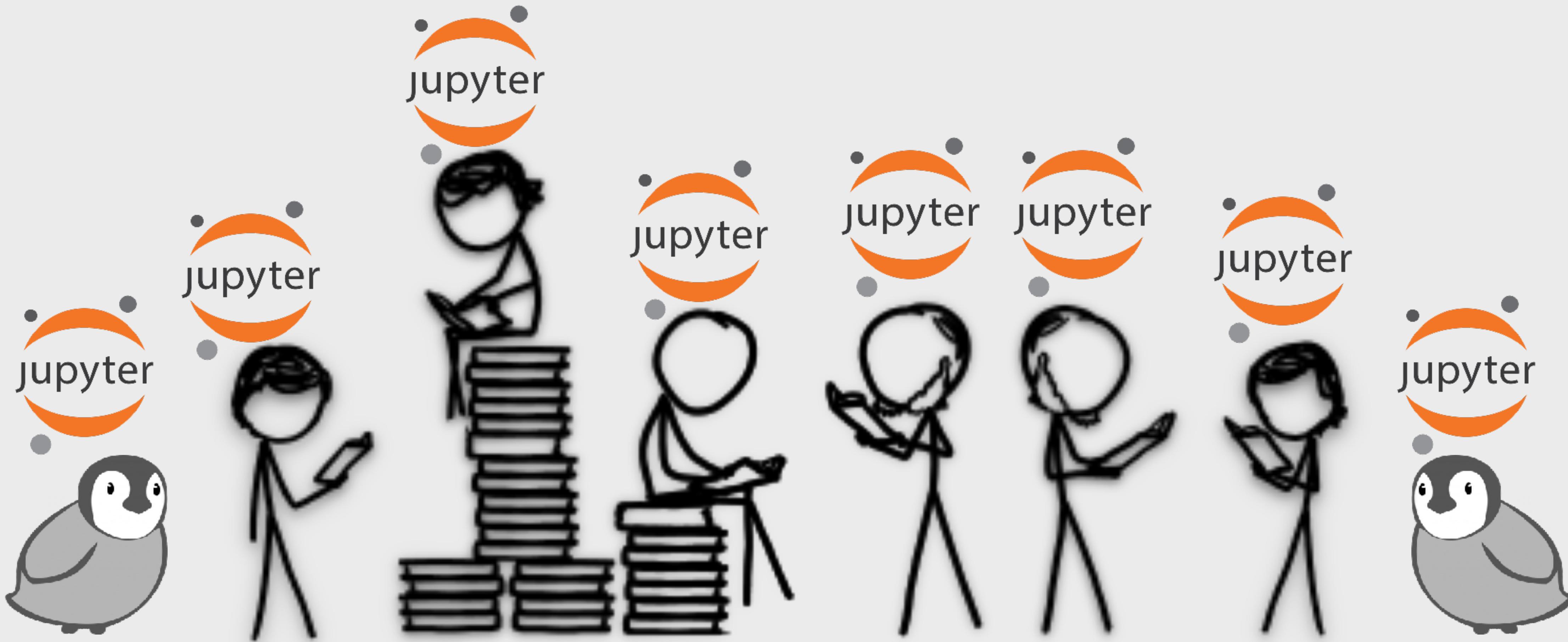
Jan. 13. For the first time four stars were in view in the following position with regard to Jupiter. There were three to the west, and one to the east; they made a nearly straight line,

Ori. * * O * Occ.

but the middle star of the straight line toward was at a distance of 2' only between Jupiter and themselves, west of Jupiter, in size, and though small in comparison with the fixed stars of the same magnitude.



JupyterHub

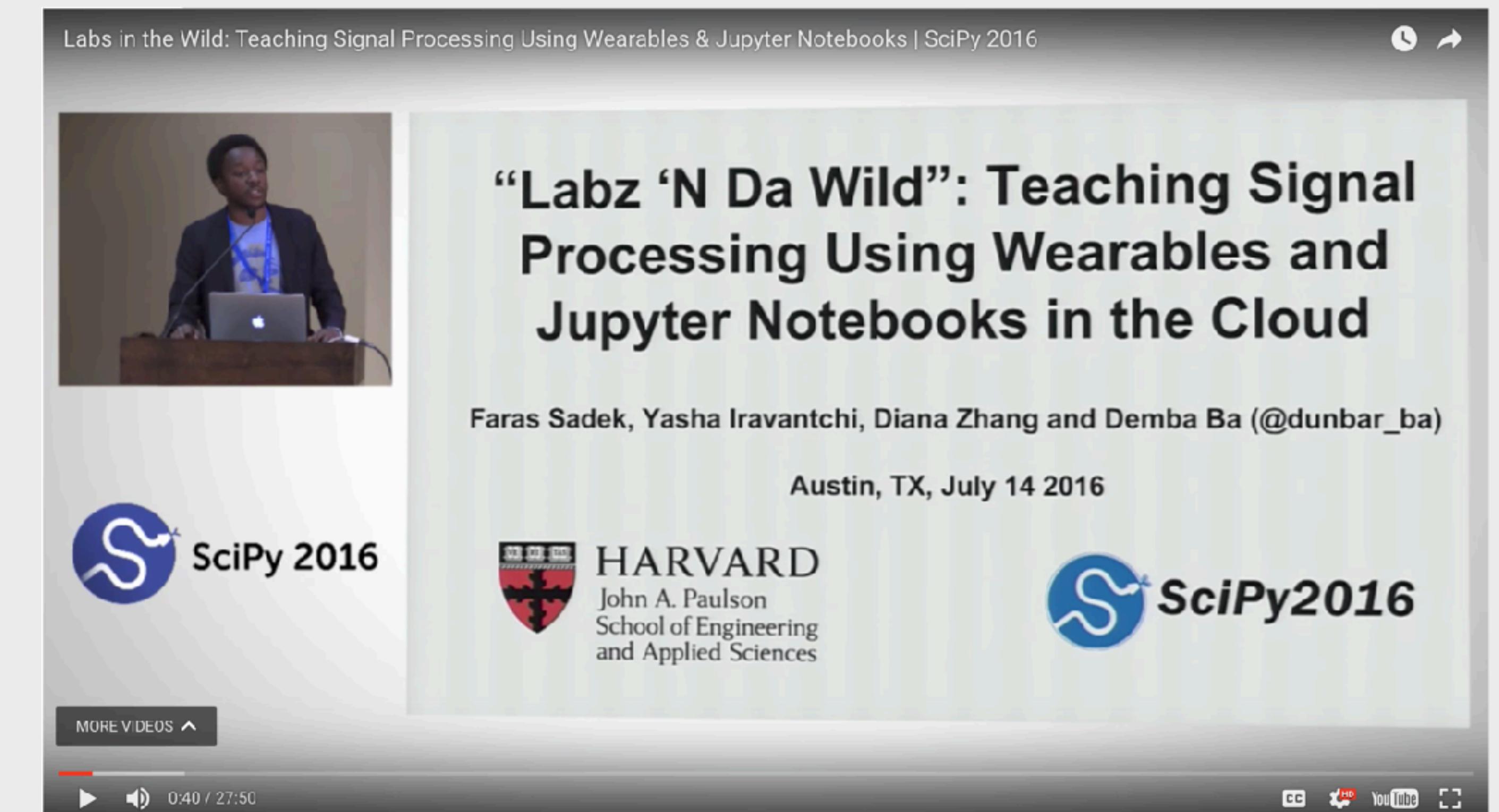


Exploration and prototyping

Teaching Signal Processing using Wearables and Jupyter Notebooks

Dr. Demba Ba

- Exploration and experimentation
<http://pyvideo.org/scipy-2016/labs-in-the-wild-teaching-signal-processing-using-wearables-jupyter-notebooks-scipy-2016.html>
- Physical media with wearables and electronics
- Real world, self-directed projects



Visualize and communicate

Python for Geosciences

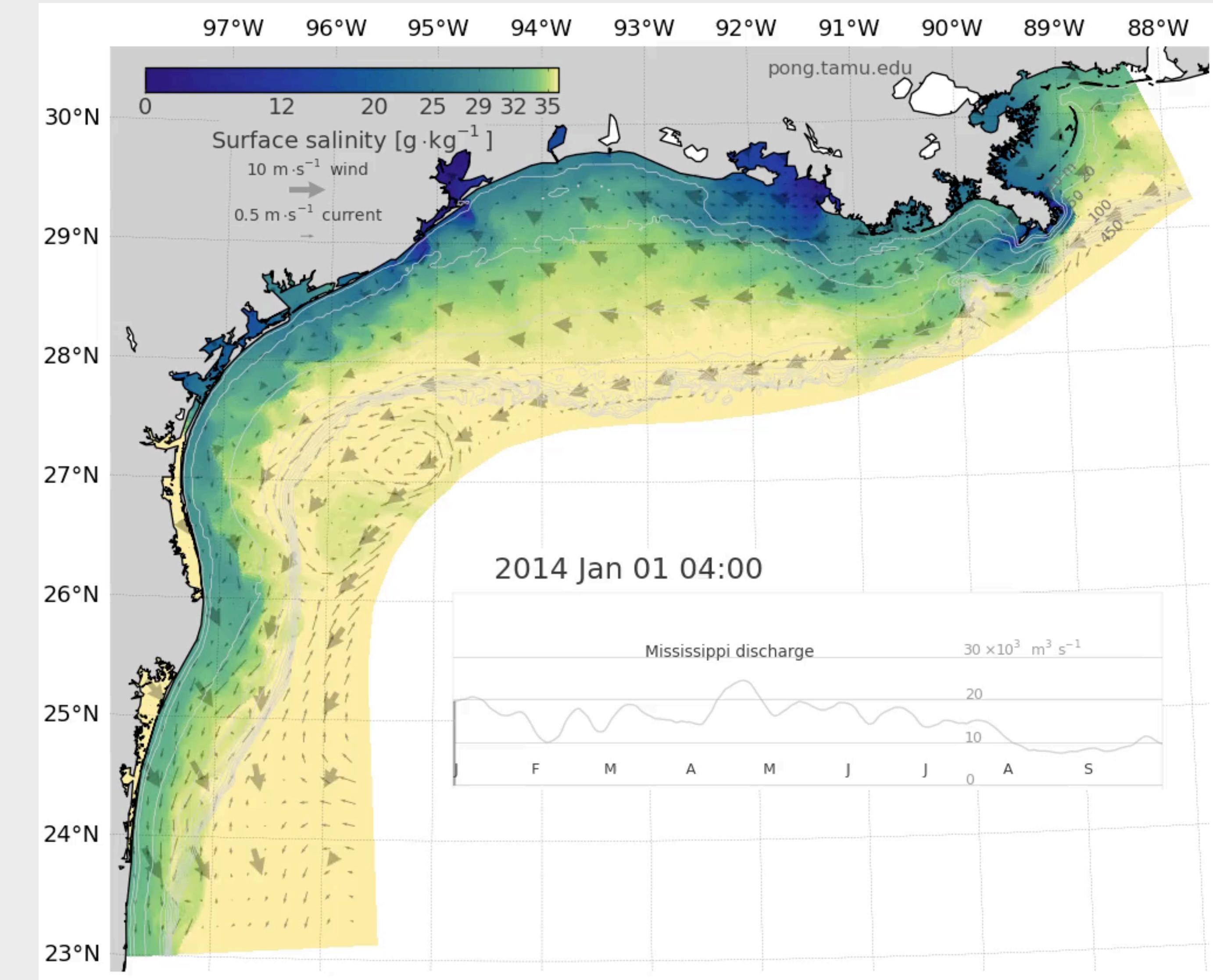
Dr. Kristen Thyng

- Feedback and communication with students using nbgrader

<http://kristenthyngh.com/blog/2016/09/07/jupyterhub+nbgrader/>

- Progression to complex examples and tasks

<https://github.com/kthyng/python4geosciences>

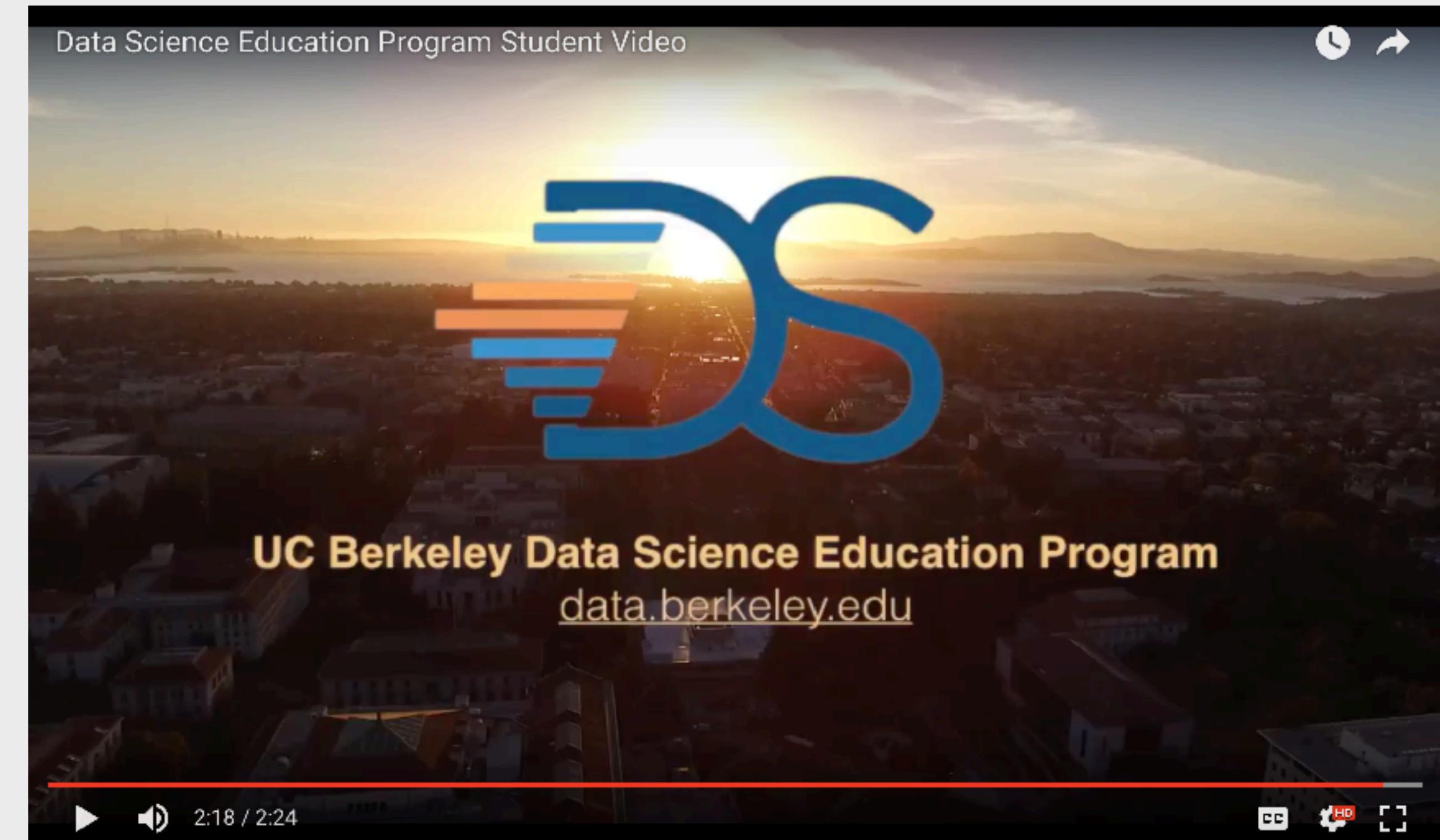


Scale learning with research tools

Berkeley Data Science Data8 UC Berkeley

- Campus wide curriculum
- Cross-discipline
- Zero to JupyterHub with Kubernetes

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



<http://denero.org/data-8-in-spring-2017.html>

<https://github.com/data-8/jupyterhub-k8s>

<http://data8.org/>

<http://data.berkeley.edu/>

<http://data.berkeley.edu/about/videos>

Live
Code on
Binder



LIGO Binder

The screenshot shows the LIGO Binder interface. At the top is the binder logo, followed by the text "Turn a GitHub repo into a collection of interactive notebooks". Below this, a subtext explains: "Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere." A large central box contains fields for "GitHub repo or URL" (set to <https://github.com/minrk/ligo-binder.git>), "Git branch, tag, or commit" (set to "master"), and "Path to a notebook file (Optional)" (set to "index.ipynb"). A "Launch" button is to the right of these fields. Below the form is a progress bar with segments labeled "Waiting", "Already built!", and "Launching". At the bottom is a "Build logs" section with a "show" link.

<https://losc.ligo.org/tutorials/>
<https://beta.mybinder.org/v2/gh/minrk/ligo-binder/master?filepath=index.ipynb>

Enabling Reproducible Science



LIGO Open Science Center

LIGO is operated by California Institute of Technology and Massachusetts Institute of Technology and supported by the U.S. National Science Foundation.

Welcome to the LIGO Open Science Center

[About LIGO](#)
[Get Started with LIGO data](#)
[Join the E-mail list for updates](#)
For general information on LIGO, please visit ligo.org
If you have LSC credentials, you may go to the [development site](#)

More discoveries from LIGO!
Data Releases from two events and a candidate event

released 2016 June 15:
[Event of December 26, GW151226: Chirp mass 9](#)

released 2016 June 15:
[Candidate event of October 12, LVT151012: Chirp mass 15](#)

released 2016 Feb 11:
[Event of September 14, GW150914: Chirp mass 30](#)

The [LIGO Laboratory's Data Management Plan](#) describes the scope and timing of LIGO data releases.

Jupyter notebook
See the new tutorial on signal processing with LIGO data, as a Jupyter (iPython) notebook.
[Tutorial on Binary Black Hole Signals in LIGO Open Data](#)

<https://losc.ligo.org/about/>



JupyterHub

Where we
are



0.7 - 12/2016

- introduce **Services**
- Anything that can talk to the Hub's API that's not a **User**
- **Managed Service:** A process started by the Hub
- **External Service:** Anything not started by the Hub
(may or may not be a process)

Where
we are



0.7 Services can...

- run a web service at /services/:service-name
- authenticate requests with the **Hub** via **HubAuth**
- talk to the **Hub API** with their **API token(s)**

Where
we are



0.7 Services are for...

- interacting with the **Hub**
- nbgrader formgrader
- culling idle servers
- sharing files
- shared notebook server(s)
- nbviewer

Where we are going

JupyterHub

0.8



Where we are going



0.8

- abstract **Proxy** API
- define spec and Python API for Hub's proxy needs
- Better support nginx, kubernetes proxies
- Requires moving activity tracking to single-user servers (done in notebook 5.0)

Where we are going



0.8

- multiple **servers** per user
 - useful when single **Hub** exposes a variety of computational resources (clusters)
 - servers can have different configurations (different Spawners?)
 - need to keep common single server-per-user case well supported, to avoid overcomplicating things
 - contributions started by Christian Barra

Where we are going



OAuth

- JupyterHub as **OAuth provider**
 - removes need for complicated cookie management by the Hub
 - Will be needed as number of endpoints for which users are authorized grows (shared servers for collaboration)

Where
we are
going

HubShare

- **Service** for sharing
- unit of sharing: directory
- push/pull model
- simple REST spec (possibly WebDAV)
- share with individuals, groups
- target use case: nbgrader assignments

<https://github.com/jupyterhub/hubshare>



Agenda

- Gateways
- Jupyter Notebook
- JupyterHub
- JupyterLab
- Next steps

Building Blocks



File Browser

Notebooks

Terminal

Text Editor

Kernels

Output

Introducing JupyterLab: The Evolution of the Jupyter Notebook (almost beta)

The JupyterLab Team

Chris Colbert, Continuum
Steven Silvester, Continuum
Afshin Darian, Continuum
Jason Grout, Bloomberg
Brian Granger, Cal Poly
Grant Nestor, Cal Poly
Cameron Oelsen, Cal Poly
Fernando Perez, LBNL/Berkeley
Ian Rose, Berkeley
Cal Poly Interns
The Larger Jupyter Team

@jupyterlab on GitHub
@ProjectJupyter on Twitter



A Whirlwind Tour of JupyterLab

The image is a composite of two parts. On the left, a man with glasses and a black t-shirt is speaking into a microphone at a podium during a presentation at the SciPy 2017 conference. On the right, a screenshot of the JupyterLab interface is displayed. The interface features a top navigation bar with File, Notebook, Editor, Terminal, Console, and Help. Below this is a sidebar with tabs for Files, Running, Commands, and Tabs. The Files tab shows a list of files and notebooks, including Fasta.ipynb, heat_equation.ipynb, notebook.ipynb, Untitled.ipynb, all.fasta, big.csv, burrito.csv, iris.csv, markdown_python.md, Messier106_NGC421..., Museums_in_DC.geojson, package.json, smaller.csv, test.py, and vega.vl.json. The 'package.json' file is currently selected, highlighted with a blue background. To the right of the sidebar is a 'Launcher' panel containing sections for Notebook, Console, and Other. The Notebook section has icons for Python 3, Julia 0.5.0, jupyterlab-demo, and R. The Console section also has icons for Python 3, Julia 0.5.0, jupyterlab-demo, and R. The Other section includes icons for Terminal and Text Editor.

SciPy 2017



Learn more about JupyterLab

PyData Seattle: https://www.youtube.com/watch?v=tHZT_mpNRcY

Demo from SciPy 2017: <https://youtu.be/X8zPuBu22Y4?t=44m50s>

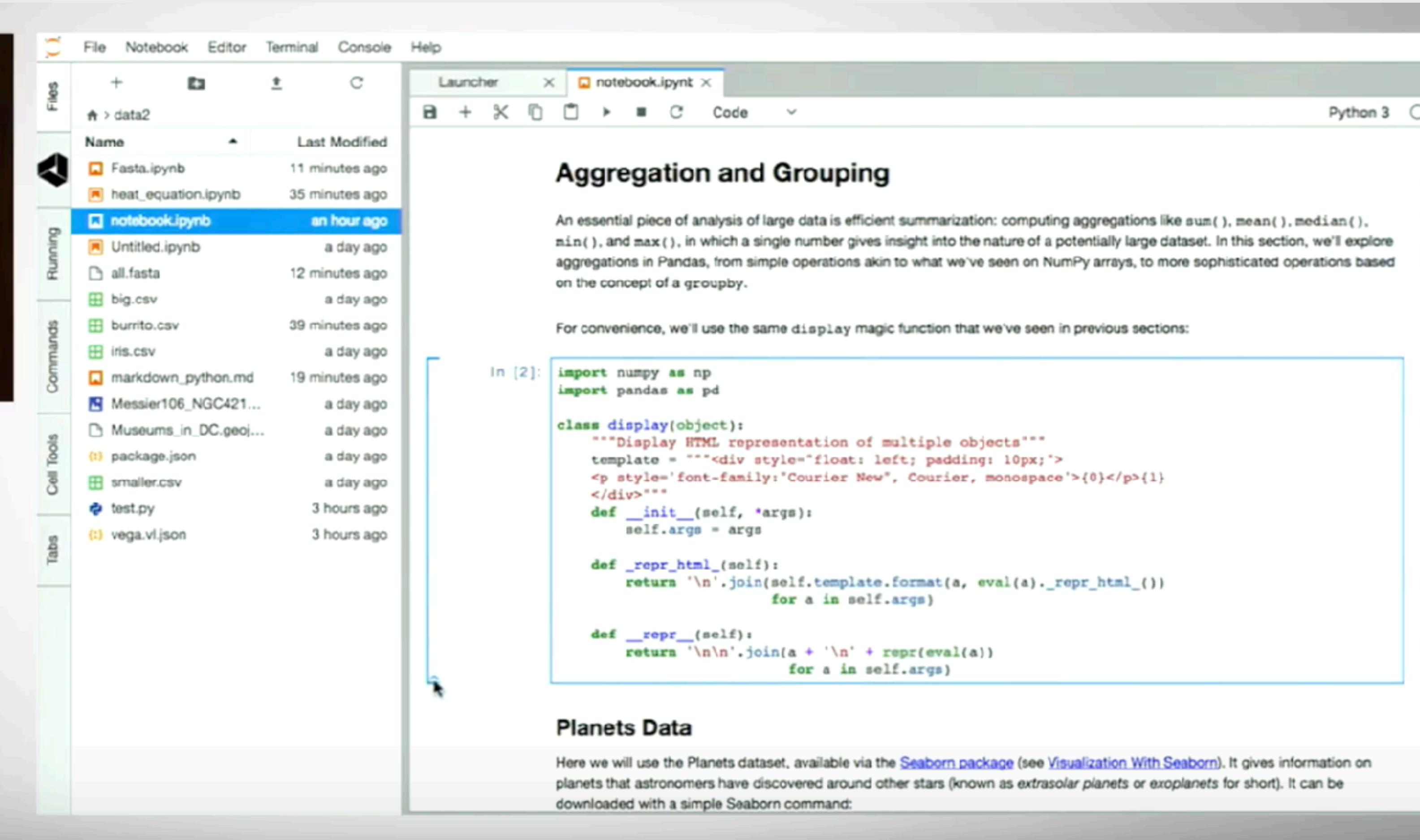
<http://pyvideo.org/pydata-dc-2016/jupyterlab-building-blocks-for-interactive-computing.html>



New implementation of the notebook



SciPy 2017



The image shows a Jupyter Notebook interface. On the left, there's a video feed of a man speaking at a podium. On the right, the notebook interface is displayed. The top navigation bar includes File, Notebook, Editor, Terminal, Console, and Help. The left sidebar has tabs for Files, Running, Commands, Cell Tools, and Tabs. The 'Files' tab shows a list of files: Fasta.ipynb, heat_equation.ipynb, notebook.ipynb (selected), Untitled.ipynb, all.fasta, big.csv, burrito.csv, iris.csv, markdown_python.md, Messier106_NGC421..., Museums_in_DC.geo..., package.json, smaller.csv, test.py, and vega.vl.json. The 'Running' tab shows a list of running notebooks. The main area contains two code cells. The first cell, In [2], contains Python code for a custom display class:

```
import numpy as np
import pandas as pd

class display(object):
    """Display HTML representation of multiple objects"""
    template = """<div style="float: left; padding: 10px; >
    <p style='font-family: 'Courier New', Courier, monospace'>{0}</p>{1}
    </div>"""
    def __init__(self, *args):
        self.args = args

    def _repr_html_(self):
        return '\n'.join(self.template.format(a, eval(a)._repr_html_())
                        for a in self.args)

    def __repr__(self):
        return '\n\n'.join(a + '\n' + repr(eval(a))
                          for a in self.args)
```

The second cell, In [3], contains the text "Planets Data". Below it, a note says: "Here we will use the Planets dataset, available via the [Seaborn package](#) (see [Visualization With Seaborn](#)). It gives information on planets that astronomers have discovered around other stars (known as *extrasolar planets* or *exoplanets* for short). It can be downloaded with a simple Seaborn command:"

Collapsible cells and draggable cells

The image is a composite of two screens. On the left, a video feed shows a man with glasses and a black t-shirt speaking into a microphone. Below the video is the SciPy 2017 logo, which is a blue circle with a white 'S' and the text 'SciPy 2017'. On the right, a Jupyter Notebook interface is displayed. The top navigation bar includes File, Notebook, Editor, Terminal, Console, and Help. The 'Files' tab shows a list of files: Fasta.ipynb, heat_equation.ipynb, notebook.ipynb (selected), Untitled.ipynb, all.fasta, big.csv, burrito.csv, iris.csv, markdown_python.md, Messier106_NGC421..., Museums_in_DC.geojson, package.json, smaller.csv, test.py, and vega.vl.json. The main content area has a section titled 'Aggregation and Grouping' with text about data summarization and Pandas operations. It also contains a code cell showing the import of Seaborn and loading of the 'planets' dataset, followed by its shape and head. The data head output is:

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.82	2007



Collaboration between tools

SciPy 2017

For convenience, we'll use the same display magic function that we've seen in previous sections:

Planets Data

Here we will use the Planets dataset, available via the [Seaborn package](#) (see [Visualization With Seaborn](#)). It gives information on planets that astronomers have discovered around other stars (known as *extrasolar planets* or *exoplanets* for short). It can be downloaded with a simple Seaborn command:

```
In [9]: import seaborn as sns  
planets = sns.load_dataset('planets')  
planets.shape
```

```
Out[9]: (1035, 6)
```

```
In [10]: planets.head()
```

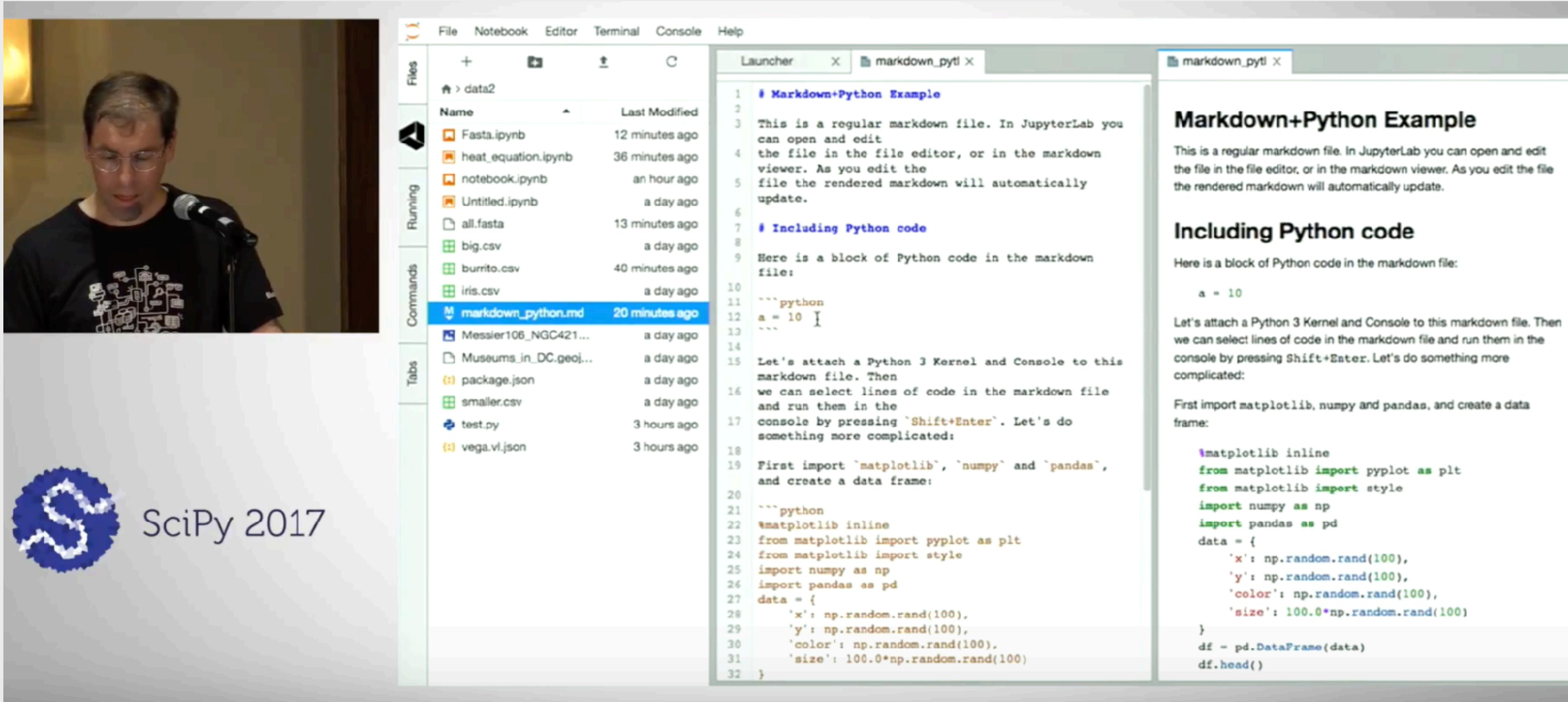
```
Out[10]:
```

	method	number	orbital_period	mass	distance
0	Radial Velocity	1	269.300	7.10	77.40
1	Radial Velocity	1	874.774	2.21	56.95
2	Radial Velocity	1	763.000	2.60	19.84
3	Radial Velocity	1	326.030	19.40	110.62
4	Radial Velocity	1	516.220	10.50	119.47

```
In [ ]: p1
```

A log in the console of commands executed
Explore data in console without messing up your notebook

Editors



The slide is titled "Editors" and includes the SciPy 2017 logo. The main content is a screenshot of the JupyterLab interface. On the left, there's a file browser showing various files like Fasta.ipynb, heat_equation.ipynb, notebook.ipynb, Untitled.ipynb, all.fasta, big.csv, burrito.csv, and iris.csv. A file named "markdown_python.md" is selected. The central area shows a code editor with the following content:

```
1 # Markdown+Python Example
2
3 This is a regular markdown file. In JupyterLab you
4 can open and edit
5 the file in the file editor, or in the markdown
6 viewer. As you edit the
7 file the rendered markdown will automatically
8 update.
9
10 # Including Python code
11
12 Here is a block of Python code in the markdown
13 file:
14
15 ````python
16 a = 10
17 ````

18 Let's attach a Python 3 Kernel and Console to this
19 markdown file. Then
20 we can select lines of code in the markdown file
21 and run them in the
22 console by pressing `Shift+Enter`. Let's do
23 something more complicated:
24
25 First import `matplotlib`, `numpy` and `pandas`,
26 and create a data frame:
27
28 ````python
29 %matplotlib inline
30 from matplotlib import pyplot as plt
31 from matplotlib import style
32 import numpy as np
33 import pandas as pd
34 data = {
35     'x': np.random.rand(100),
36     'y': np.random.rand(100),
37     'color': np.random.rand(100),
38     'size': 100.0*np.random.rand(100)
39 }
40 df = pd.DataFrame(data)
41 df.head()
```

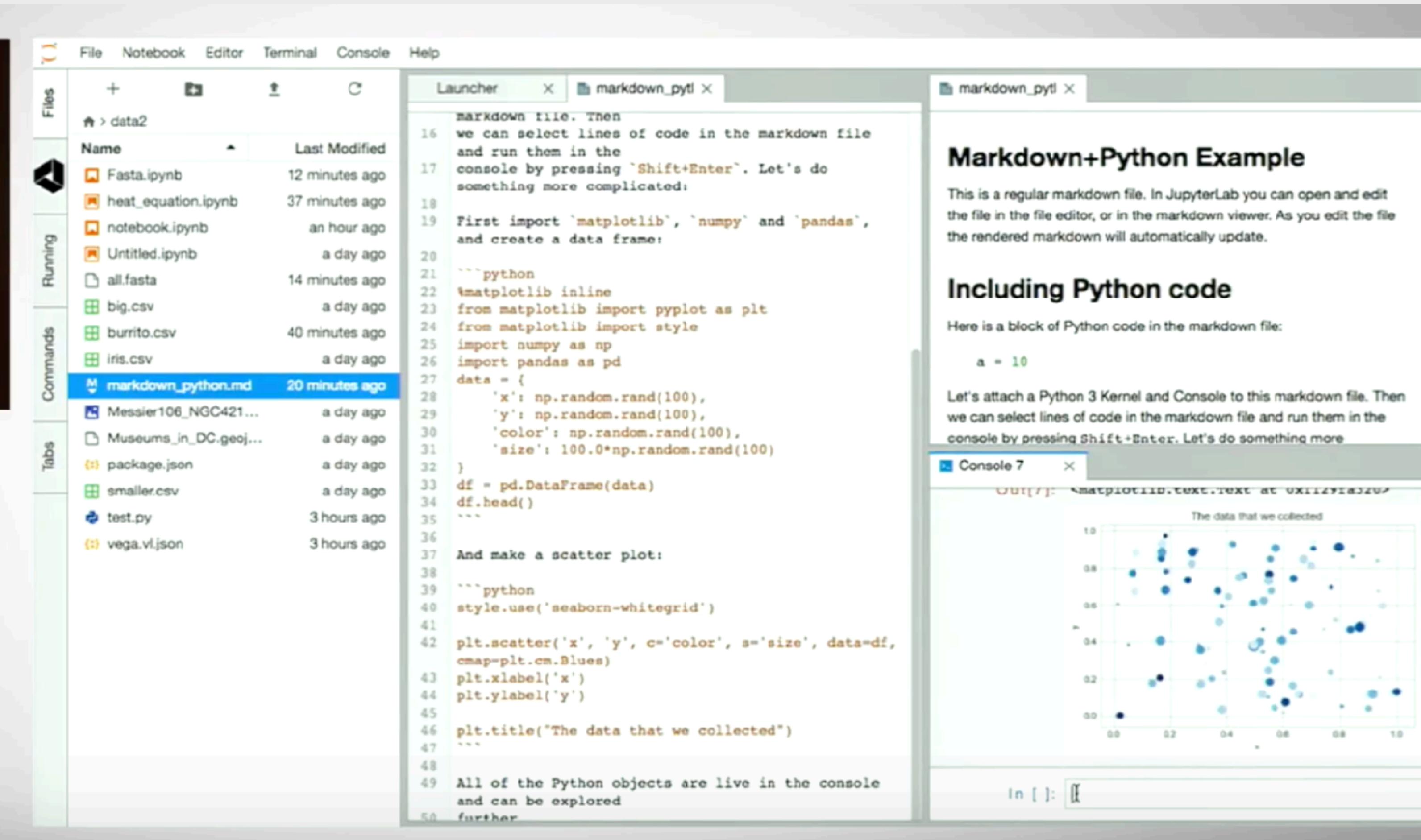
The right side of the interface shows a preview of the rendered Markdown content, which includes the rendered code blocks and the resulting output.

Many different editors; preview markdown

Editor connected to a console



SciPy 2017



The screenshot shows the JupyterLab interface. On the left, there's a file browser with a list of files including 'data2', 'Fasta.ipynb', 'heat_equation.ipynb', 'notebook.ipynb', 'Untitled.ipynb', 'all.fasta', 'big.csv', 'burrito.csv', 'iris.csv', 'markdown_python.md' (which is selected), 'Messier106_NGC421...', 'Museums_in_DC.geojson', 'package.json', 'smaller.csv', 'test.py', and 'vega.vl.json'. The 'Running' tab shows no active kernels. The 'Commands' tab has a few items listed. The main area contains a 'Launcher' tab and a 'markdown_pytl' tab. The 'markdown_pytl' tab displays a Markdown document with Python code examples. The code includes imports for matplotlib, numpy, and pandas, and creates a scatter plot. A 'Console 7' tab is open, showing the output of the plotted data. The text in the Markdown file says:

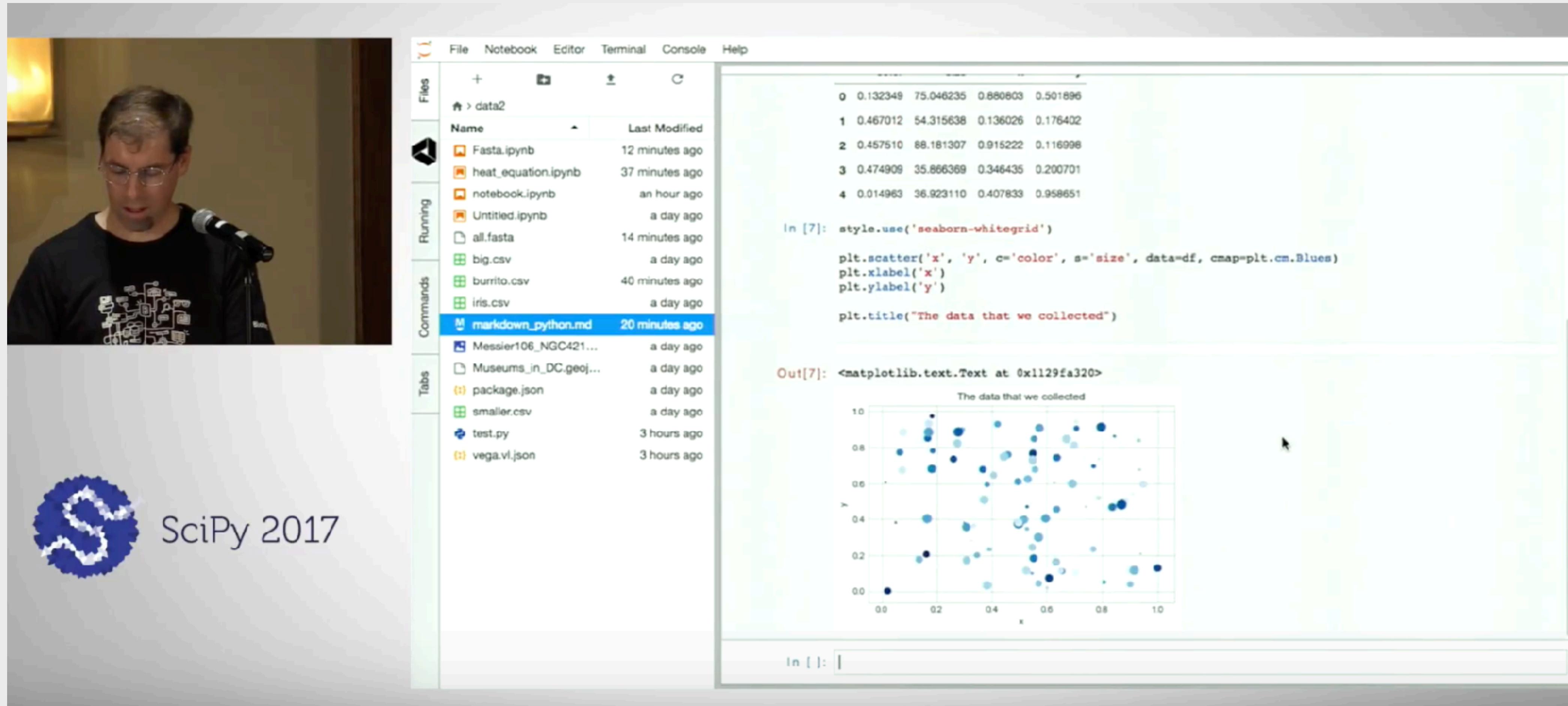
```
16 markdown file. Then
17 we can select lines of code in the markdown file
18 and run them in the
19 console by pressing `Shift+Enter`. Let's do
20 something more complicated:
21
22 First import 'matplotlib', 'numpy' and 'pandas',
23 and create a data frame:
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
```

The output in the 'Console 7' tab shows a scatter plot with the title "The data that we collected". The text at the bottom of the Markdown file says:

```
All of the Python objects are live in the console
and can be explored
further
```

Connect to console and Shift-Enter to run code snippet

Single document mode



Shift-Command-Enter to enter single document mode. Similar to classic notebook.

Extensible

SciPy 2017

Launch X markd X Messir X Museu X all.fasta X

File Notebook Editor Terminal Console Help

Files

Name Last Modified

- Fasta.ipynb 13 minutes ago
- heat_equation.ipynb 37 minutes ago
- notebook.ipynb an hour ago
- Untitled.ipynb a day ago
- all.fasta** 14 minutes ago
- big.csv a day ago
- burrito.csv 41 minutes ago
- iris.csv a day ago
- markdown_python.md 21 minutes ago
- Messier106_NGC421... a day ago
- Museums_in_DC.geojson a day ago
- package.json a day ago
- smaller.csv a day ago
- test.py 3 hours ago
- vega.vl.json 3 hours ago

Running

Commands

Tabs

Console 7

```
a = 10
```

Let's attach a Python 3 Kernel and Console to this markdown file. Then we can select lines of code in the markdown file and run them in the console by pressing Shift+Enter. Let's do something more.

```
In [7]: style.use('seaborn-whitegrid')  
plt.scatter('x', 'y', c='color', s='size'  
plt.xlabel('x')  
plt.ylabel('y')
```

```
In [ ]:
```

“In one night and a couple of dozen lines of code we wrote a Fasta viewer.”

Becomes a notebook extension

The image is a composite of two parts. On the left, there is a presentation slide with a photo of a man speaking into a microphone, the text "SciPy 2017", and the SciPy logo. On the right, there is a screenshot of the JupyterLab interface. The interface includes a file browser on the left showing files like "Fasta.ipynb", "heat_equation.ipynb", and "notebook.ipynb". The main area contains two code cells: In [6] and In [7]. In [6] contains Python code for creating a Fasta viewer class. In [7] contains code for reading an Fasta file and passing it to the Fasta class. Below the code cells, there is a heatmap visualization of DNA sequence data. To the right of the code cells, there is a "Markdown+Python Example" section with text and a "Console 7" tab showing a list of numbers.

Markdown+Python Example

This is a regular markdown file. In JupyterLab you can open and edit the file in the file editor, or in the markdown viewer. As you edit the file the rendered markdown will automatically update.

Including Python code

Here is a block of Python code in the markdown file:

```
a = 10
```

Let's attach a Python 3 Kernel and Console to this markdown file. Then we can select lines of code in the markdown file and run them in the console by pressing Shift+Enter. Let's do something more.

Console 7

```
0 0.132349 75.046235 0.880803 0.501896  
1 0.467012 54.315638 0.136026 0.176402  
2 0.457510 88.181307 0.915222 0.116998  
3 0.474909 35.866369 0.346435 0.200701  
4 0.014963 36.923110 0.407833 0.958651
```

```
In [7]: style.use('seaborn-whitegrid')  
plt.scatter('x', 'y', c='color', s='size'  
plt.xlabel('x')  
plt.ylabel('y')
```

```
In [ ]:
```



With the same code, the Fasta viewer becomes an extension usable in the notebook.

Datasets, grids, and scale

The image shows a JupyterLab interface running on a screen. On the left, there's a video feed of a man with glasses speaking into a microphone. Below the video is the SciPy 2017 logo. The main area is a JupyterLab dashboard. On the left side of the dashboard, there's a sidebar with sections for 'Files', 'Running', 'Commands', and 'Tabs'. The 'Files' section shows a list of files including 'big.csv' (selected), 'burrito.csv', 'iris.csv', 'messier_python.md', 'Museums_in_DC.geojson', 'package.json', 'smaller.csv', 'test.py', and 'vega.vl.json'. The 'Running' section shows a list of kernel instances. The 'Commands' section shows a list of recent commands. The 'Tabs' section shows several open tabs: 'Launch X', 'markd X', 'Messier X', 'burrito X', 'big.csv X', and 'markdown_python X'. The 'big.csv' tab is active, displaying a large dataset with columns: date, routeCode, serviceVehi, tripid, vehicleId, patternId, and tripDirection. The data consists of 1.2 million rows of timestamped vehicle trip information. To the right of the file browser is a code editor window titled 'markdown_python X' containing the text 'Markdown+Python Example'. It describes how to edit a markdown file and run Python code in the console. Below the code editor is a 'Console 7' window showing the output of a Python script that uses 'seaborn-whitegrid' style and creates a scatter plot. The script includes code to import 'style', 'plt', and 'scatter', and to set 'x' and 'y' labels.

1.2M rows 200Mb csv file. Excel can't open.

A few seconds to load and then "smooth as butter" when scrolling.
Rumor has it that Chris Colbert has a trillion row by column demo too.



Agenda

- Gateways
- Jupyter Notebook
- JupyterHub
- JupyterLab
- Next steps



Call to action

- Join **Jupyter** mailing lists
- Participate in a **sprint**
- Give a **talk** or **write** a post
- Offer a **workshop**
- **Contribute** to a favorite project
- **Share** your trials and successes



Resources

<https://github.com/willingc/2017-science-gateways/blob/master/resources/resources.md>



jupyter.org

jupyter google
groups and
Gitter

Trending
notebooks
on GitHub

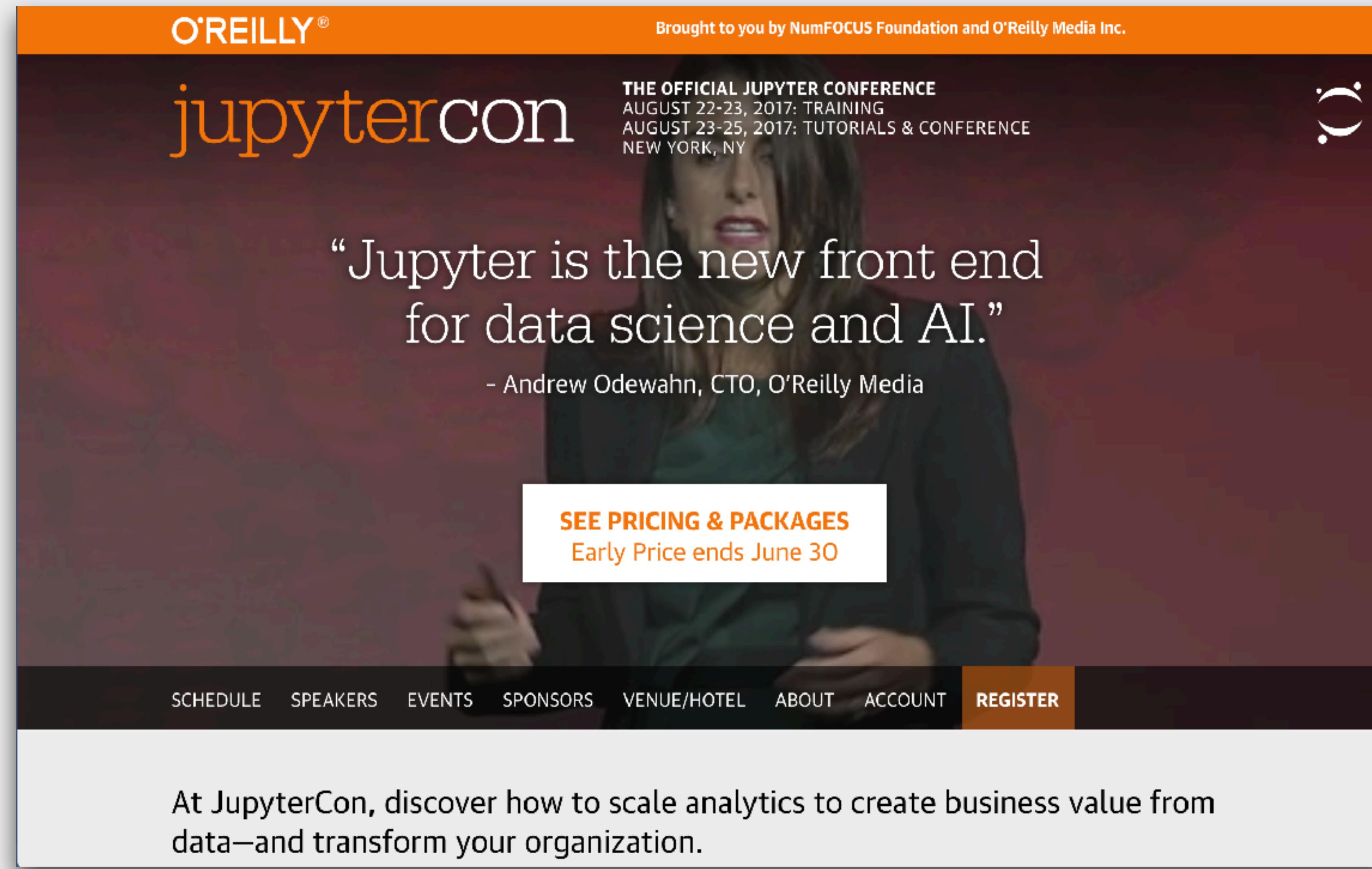
pyvideo.org

try.jupyter.org

nbviewer

JupyterCon, August 2017, NYC

<https://conferences.oreilly.com/jupyter/jup-ny>



The image shows the homepage of the JupyterCon conference website. At the top, the O'Reilly logo is on the left, and the text "Brought to you by NumFOCUS Foundation and O'Reilly Media Inc." is on the right. The main title "jupytercon" is prominently displayed in large orange letters. To the right of the title, event details are listed: "THE OFFICIAL JUPYTER CONFERENCE AUGUST 22-23, 2017: TRAINING AUGUST 23-25, 2017: TUTORIALS & CONFERENCE NEW YORK, NY". A quote from Andrew Odewahn follows: "Jupyter is the new front end for data science and AI." - Andrew Odewahn, CTO, O'Reilly Media. Below the quote is a button labeled "SEE PRICING & PACKAGES Early Price ends June 30". At the bottom of the page, a navigation bar includes links for SCHEDULE, SPEAKERS, EVENTS, SPONSORS, VENUE/HOTEL, ABOUT, ACCOUNT, and a highlighted "REGISTER" button. A footer message encourages attendees to "discover how to scale analytics to create business value from data—and transform your organization."

O'REILLY®

Brought to you by NumFOCUS Foundation and O'Reilly Media Inc.

jupytercon

THE OFFICIAL JUPYTER CONFERENCE
AUGUST 22-23, 2017: TRAINING
AUGUST 23-25, 2017: TUTORIALS & CONFERENCE
NEW YORK, NY

“Jupyter is the new front end
for data science and AI.”

- Andrew Odewahn, CTO, O'Reilly Media

SEE PRICING & PACKAGES
Early Price ends June 30

SCHEDULE SPEAKERS EVENTS SPONSORS VENUE/HOTEL ABOUT ACCOUNT REGISTER

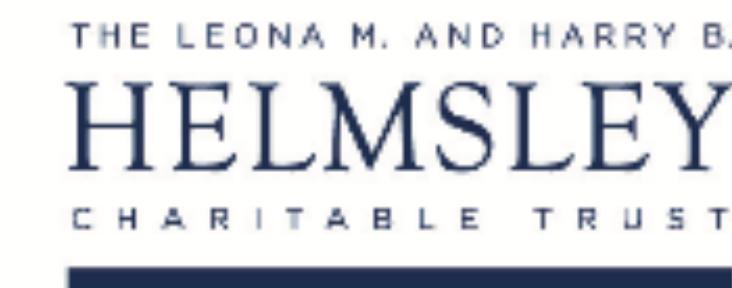
At JupyterCon, discover how to scale analytics to create business value from data—and transform your organization.

WE'RE OPEN FOR YOU.



Attributions and recognition

A huge thank you to the Project Jupyter team and community.
Your hard work and passion makes this all possible.



- Kristen Thng
- San Diego Python
- Demba Ba
- Jeremy Freeman, Binder
- Michael Cuthbert, music21
- LIGO
- Andrea Zonca, SDSC, Ilkay Altinas, Software Carpentry
- Photo credits on individual slides



Thank you



Thank you

try.jupyter.org

www.jupyter.org

ipython.org

numfocus.org

GitHub: willingc
@willingcarol



Thank you for participating!

- Please share your feedback through our 30-second evaluation:
<http://sciencegateways.org/webinareval>
- Join us next month (August 9)

Interactive Best Practices: Job Management & Scheduling

Presented by Miron Livny and Todd Tannenbaum (Condor Project), Mark Miller (CIPRES), Sudhakar Pamidighantam (SEAggrid), and others

- Upcoming opportunities for students/educators: <http://sciencegateways.org/engage/student-focused>