

Introduction to Jupyter

Presented by Ben Winjum

IDRE

April 17, 2019

Outline

- What is Jupyter?
- Interactive – notebook basics
- A slightly deeper view of notebooks and the growing use of Jupyter
- Interactive – making a notebook from scratch
 - Including interactive elements
- The research and education ecosystem the Jupyter enables
- Interactive – JupyterLab



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



Julia, Python, R
and now over 100 other languages

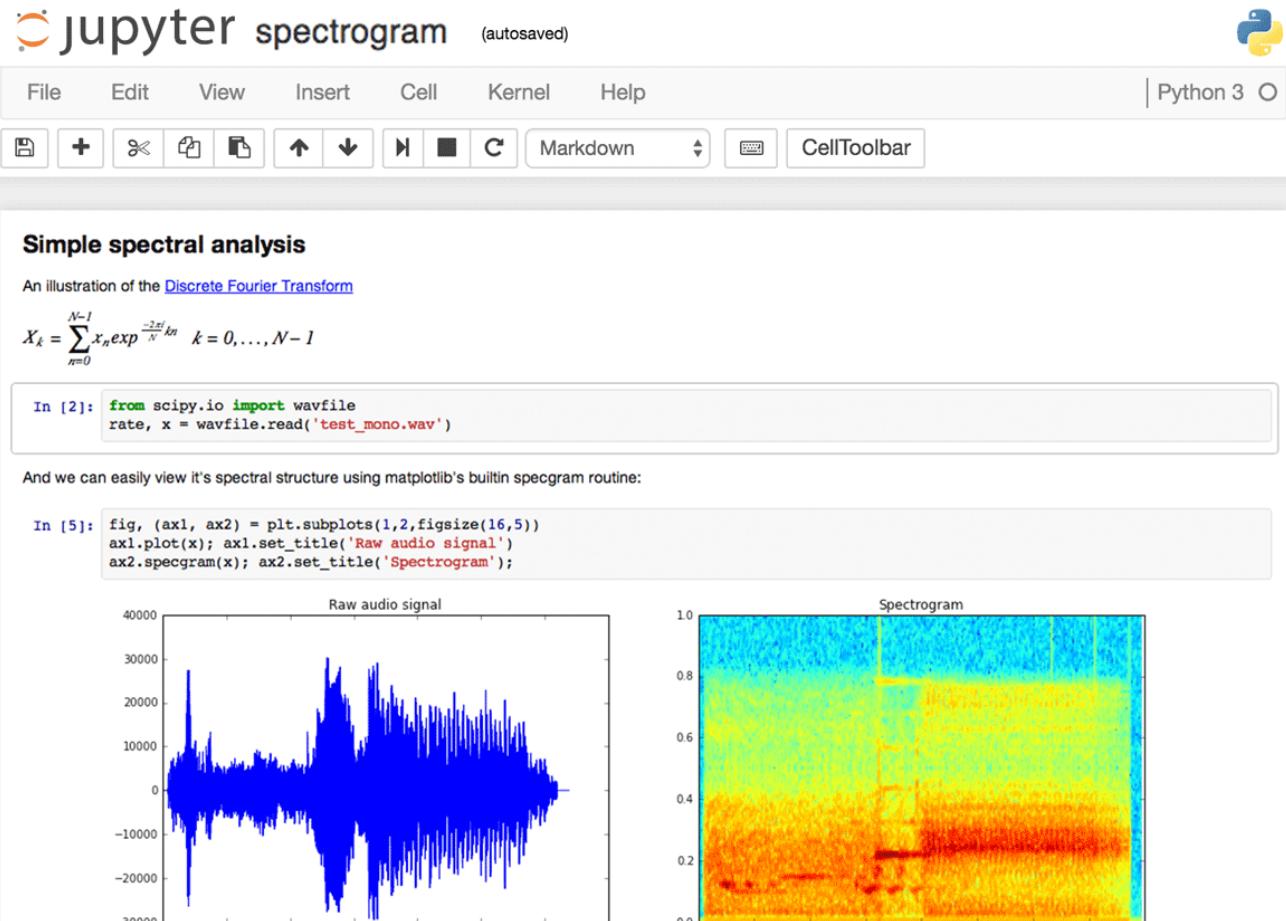


A central component of the Jupyter project
is the Jupyter Notebook

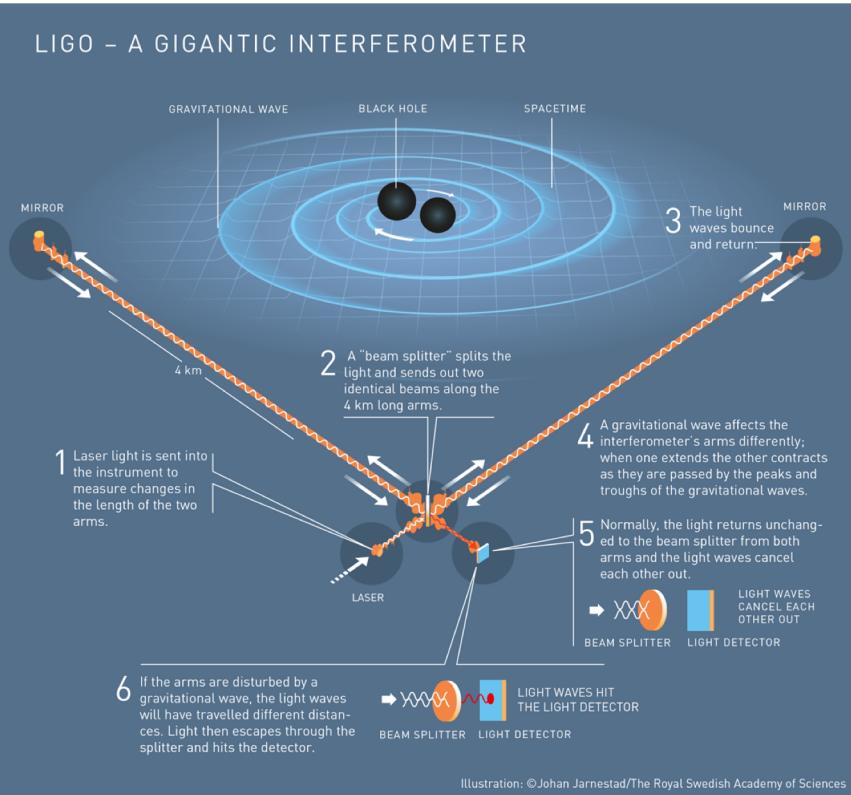
Observations Period			
2. S. 1912	March 12	O **	
3. mon		** O *	
2. Feb:		O *** *	
3. mon		O * *	
3. Febr.		* O *	
4. mon		* O **	
6. mon		** O *	
8. March 13.		* * * O	
10. mon		* * * O *	
11.		* * O *	
12. H. & wyp:		* O *	
13. mon		* * O *	
14. Marz.		* * * O *	

The Jupyter Notebook
an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

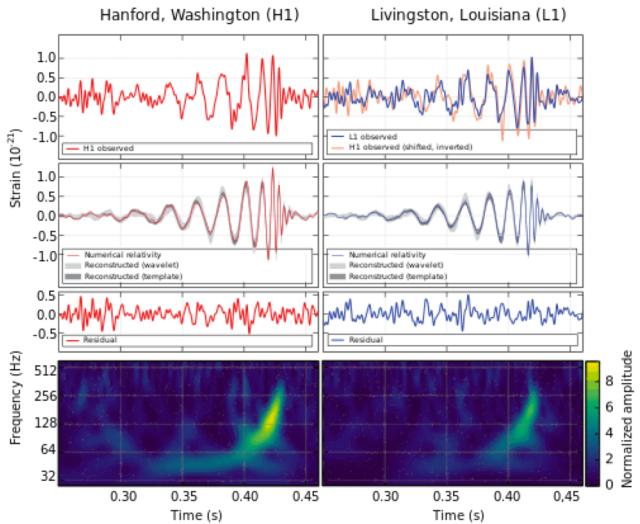
Uses include:
data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.



Jupyter Exploration of 2017 Physics Nobel Work



LIGO & Open Science



jupyter index (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 2 0

Run M Run Cell Kernel Widgets

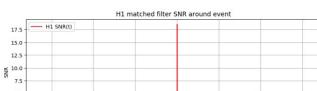
```
plt.ion()
plt.grid('on')
plt.xlabel('Time since (0,dt)'.format(timemax))
plt.ylabel('Whitened strain (units of noise stddev)')
plt.title('Residual whitened data after subtracting template around event')
plt.savefig(eventname+"-det"+_matchname+".plottype")

#-- Display PSD and template
# must multiply by sqrt(dt) to plot template fft on top of ASD:
plt.figure(figsize=(10,6))
template = np.abs(template['template'].fft)*np.sqrt(dt*datafreq)
d_eff = template/(np.abs(template['template']))**0.5
plt.loglog(datafreq, template, 'k', label='template(f)*sqrt(dt)')
plt.loglog(freq, np.sqrt(data_psd), pcolor, label='ASD')
plt.xlim(20, fs/2)
plt.ylim(-24, -10)
plt.grid()
plt.xlabel('frequency (Hz)')
plt.ylabel('strain noise ASD (strain/rtHz), template h(f)*rt(f)')
plt.title('ASD and template around event')
plt.savefig(eventname+"-det"+_matchfreq+".plottype")
```

For detector H1, maximum at 1126259462.4395 with SNR = 18.6, D_{eff} = 814.44, horizon = 1.889.6 Mpc

For detector L1, maximum at 1126259462.4324 with SNR = 13.2, D_{eff} = 999.74, horizon = 1.650.6 Mpc

H1 matched filter SRN around event



This plot shows the matched filter signal-to-noise ratio (SRN) over time. The x-axis represents time from 1126259462.4395, ranging from -10 to 10 seconds. The y-axis represents SRN, ranging from 0.0 to 15.0. A sharp peak is visible at approximately 0 seconds, reaching a value of about 18.6.

H1 SRN(t)

Time since 1126259462.4395

H1 whitened data around event



This plot shows the whitened data around the event. The x-axis represents time from 1126259462.4395, ranging from -0.150 to 0.050 seconds. The y-axis represents whitened strain in units of noise std dev, ranging from -10.0 to 10.0. The data shows a significant transient signal centered around 0 seconds.

H1 whitened h(t)

Template(t)

Time since 1126259462.4395

H1 Residual whitened PSD after subtracting template around event



This plot shows the residual whitened Power Spectral Density (PSD) after subtracting the template. The x-axis represents frequency from 0.020 to 0.030 Hz. The y-axis represents PSD in units of noise std dev, ranging from -10.0 to 10.0. The plot shows a noisy baseline with several sharp peaks, notably around 0.025 Hz.

H1 whitened h(t)

Template(t)

Time since 0.020 to 0.030 Hz

<http://physics.aps.org/featured-article-pdf/10.1103/PhysRevLett.116.061102>
<https://www.gw-openscience.org/tutorials/>

Jupyter notebooks engage learners

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 the piece goes 'C' with 'F'?" or "I wonder which hand was the first to see these chords in this order," or "I'd like to know what kind of harmonic progression (or Indian raga) a piece-based pitch structure (or the form of minutes) 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?

Example: 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("lalalala").show()

Print the twelve-tone matrix for a tone row (in this case: the opening of Schoenberg's Fourth String Quartet):
print(music21.sets.RhythmMatrix([2,1,3,10,5,3,4,6,6,7,6,11]))
```

or show all the `and`-`Music21` classes are already available as objects, you can type:
`print(music21.all)` (www.music21.org/doc/html/musiconall.html)

Convert a file from Humdrum's `.kern` format to MusicXML for editing in Finale or Sibelius:
`converter.parse('/Users/cuthbert/Desktop/composition.krn').write('musicxml')`

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

Out[10]:

bwv7.7.mx1

Movement Name: bwv7.7.mx1

```
def closedPosition(self):
    """Return a new Chord object with
    the same notes, but in closed position.
    >>> chord1 = Chord(["C#4", "G5"])
    >>> chord2 = chord1.closedPosition()
    >>> print(chord2.ally)
    <chord> e' g'4
```

```
newChord = copy.deepcopy(self)
tempPitchList = newChord.pitches
chordList = []
for pitch in tempPitchList:
    while thisPitch > tempChordNotes:
        thisPitch -= 12
    chordList.append(thisPitch)
newChord.pitches = chordList
```

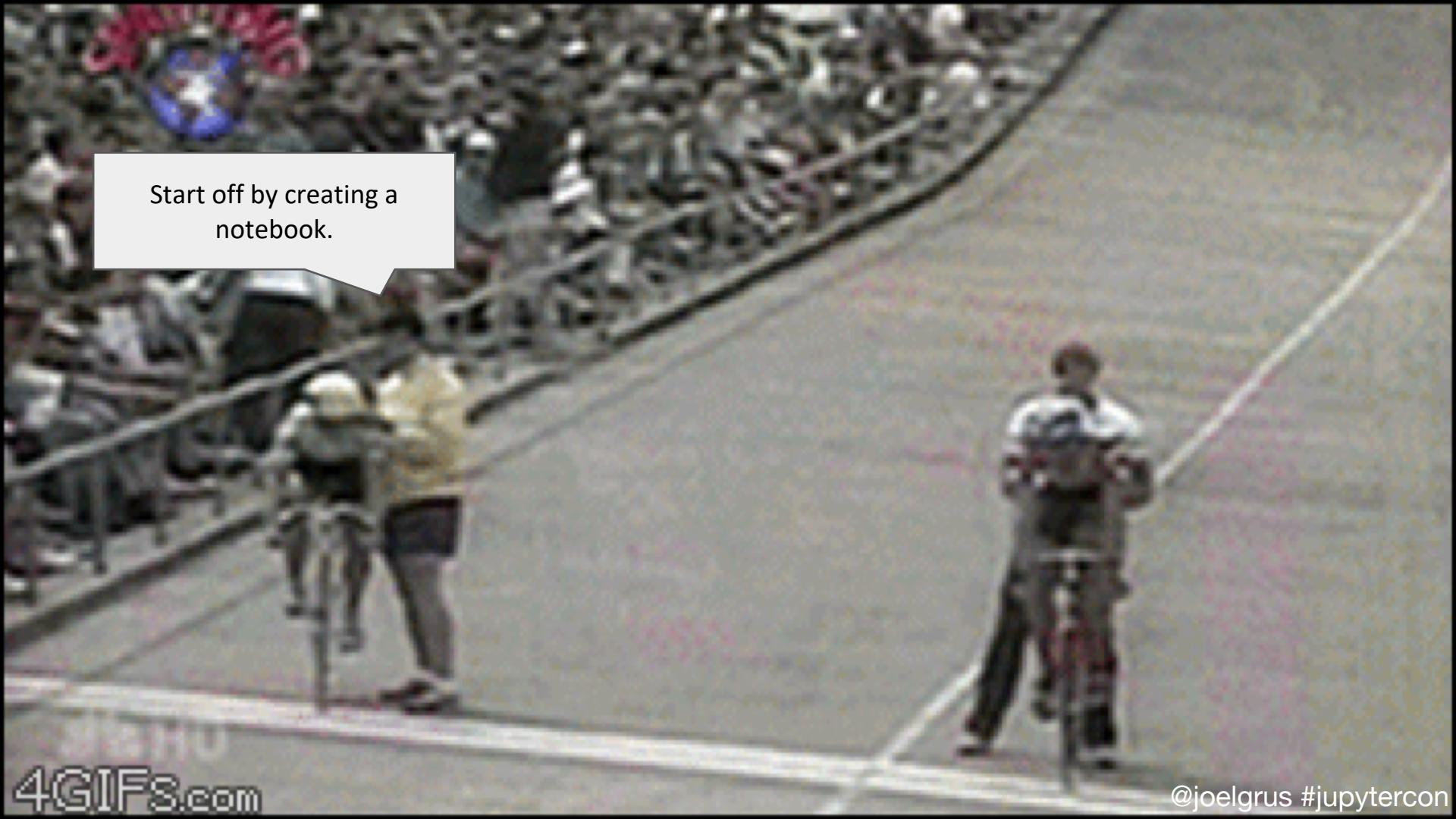


Michael Scott Cuthbert (cuthbert [at] mit.edu) is Associate Professor of Music and Homer A. Burnell Career Development Professor at M.I.T.



From C. Willing, JupyterCon 2017

So How can you build amazing notebooks?



Start off by creating a
notebook.



Before we can move mountains, it helps to learn how our tools work

Notebook basics

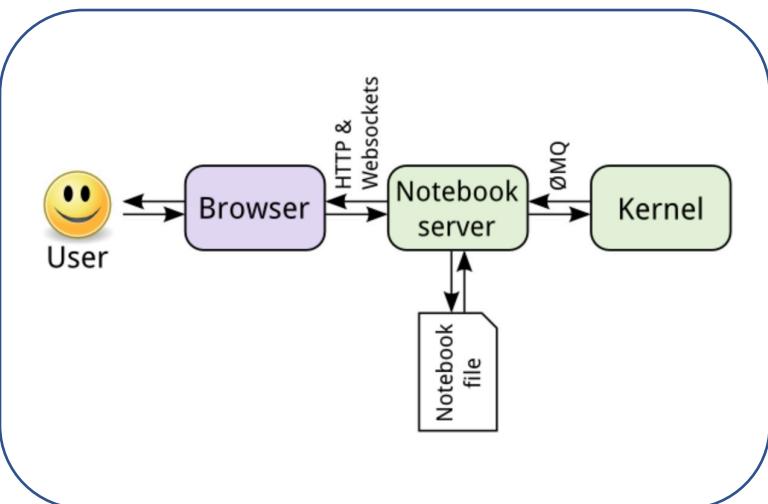
go to <https://jupyter.idre.ucla.edu>

```
$ cd work  
$ git clone git://github.com/benjum/idre-intro-to-jupyter.git
```

The Jupyter Notebook

Jupyter is comprised of several components

Front-end:



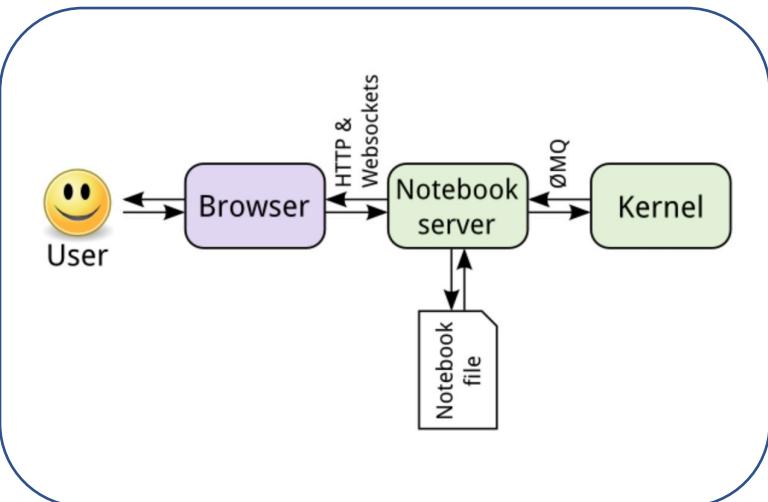
1. **Web Application:** Browser-based tool for interactive development of notebook documents
2. **Notebook Document:** A representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects. These documents are internally JSON files and are saved with the .ipynb extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues.

The Jupyter Notebook

Jupyter is comprised of several components

Back-end:

1. **Kernel:** A separate process responsible for running user code. Jupyter is capable of interfacing with many programming languages.
2. **Notebook Server:** Communicates with kernel and routes the programming language to the web browser.



Different ways to view notebooks

- Lab notebook
 - Single author... and meant primarily for single viewer
 - Can be split into parts before they get too long
 - Can be split into different topics
 - Not meant to be anything other than place for experimentation and development
- Deliverable report
 - Single- or team-authored
 - Meant to share
 - Fully polished
 - Store the final analysis and outputs
- Interactive playground
 - Carefully crafted educational narratives to invite interaction
 - Interleaving regular notebook cells with test cells

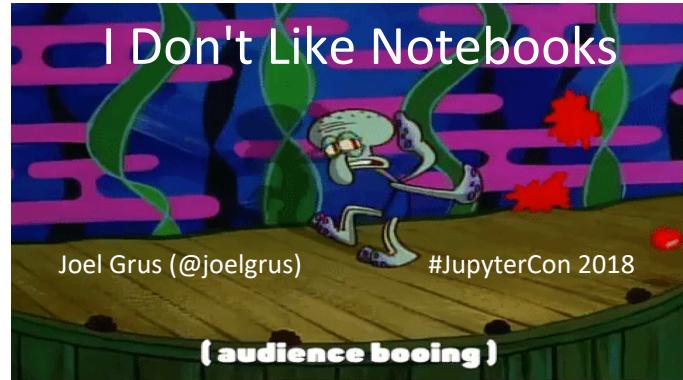
Not all uses of Jupyter are for clearly documented research and education

A. Rule, A. Tabard, J. Hollan. *Exploration and Explanation in Computational Notebooks*. ACM CHI Conference on Human Factors in Computing Systems, Apr 2018, Montréal, Canada.

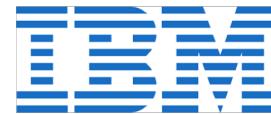
- In a scan of > 1 million notebooks on GitHub, 1/4 had no explanatory text
- In analyzing 200 academic notebooks, < 40% discussed reasoning or explained results
- In interviews with 15 academic data analysts, most considered computational notebooks personal, exploratory, and messy

“Why not to use Jupyter”

- Hidden state and out-of-order execution
- Notebooks are difficult for beginners
- Notebooks encourage bad habits
- Notebooks discourage modularity and testing
- Jupyter’s autocomplete, linting, and way of looking up the help are awkward
- Notebooks encourage bad processes
- Notebooks hinder reproducible + extensible science
- Notebooks make it hard to copy and paste into Slack/Github issues
- Errors will always halt execution
- Notebooks make it easy to teach poorly
- Notebooks make it hard to teach well



Nevertheless, Jupyter tools are being used by some major companies, and they are finding it in their best interest to contribute to open source tools related to Jupyter



So... is Jupyter a good environment for developing code?

So... is Jupyter a good environment for developing code?



YES!

But... like any beautiful instrument, it helps to develop technique

... now let's practice making one

Atlantic Monthly
April 5, 2018



Genomic analysis of elongated skulls and extensive female-biased immigration in early Medieval Bavaria

Krishna R. Veeramah^a, Andreas Rott^{b,1}, Melanie Groß^{c,1}, Lucy van Dorp^d, Saioa López^e, Karola Kirsanow^f, Christian Sell^c, Jens Blöcher^c, Daniel Wegmann^{f,g}, Vivian Link^{f,g}, Zuzana Hofmanová^{f,g}, Joris Peters^{b,h}, Bernd Trautmann^b, Anja Gairhosⁱ, Jochen Haberstroh^j, Bernd Päffgen^k, Garrett Hellenthal^d, Brigitte Haas-Gebhardⁱ, Michaela Harbeck^{b,2,3}, and Joachim Burger^{c,2,3}

^aDepartment of Ecology and Evolution, Stony Brook University, Stony Brook, NY 11794-5245; ^bState Collection for Anthropology and Palaeoanatomy, Bavarian Natural History Collections, 80333 Munich, Germany; ^cPalaeogenetics Group, Institute of Organismic and Molecular Evolution, Johannes Gutenberg University Mainz, 55099 Mainz, Germany; ^dUCL Genetics Institute, Department of Genetics, Evolution and Environment, University College London, WC1E 6BT London, United Kingdom; ^eCancer Institute, University College London, WC1E 6DD London, United Kingdom; ^fDepartment of Biology, University of Fribourg, 1700 Fribourg, Switzerland; ^gSwiss Institute of Bioinformatics, 1700 Fribourg, Switzerland; ^hArchaeoBioCenter and Institute for Palaeoanatomy, Domestication Research and the History of Veterinary Medicine, Ludwig Maximilian University, 80539 Munich, Germany; ⁱBavarian State Archaeological Collection, 80538 Munich, Germany; ^jBavarian State Department of Monuments and Sites, 80539 Munich, Germany; and ^kInstitute of Prehistoric and Prehistoric Archaeology, Ludwig Maximilian University, 80799 Munich, Germany

Manuscript submitted January 18, 2017; accepted November 21, 2017

The Scientific Paper Is Obsolete

Here's what's next.

Modern European genetic structure demonstrates strong correlations with geography, while genetic analysis of prehistoric humans has indicated at least two major waves of immigration from outside the continent during periods of cultural change. However, population-level genome data that could shed light on the demographic processes occurring during the intervening periods have been absent. Therefore, we generated genomic data from 41 individuals dating mostly to the late 5th/early 6th century

to form in the 5th century AD, and that it emanated from a combination of the romanized local population of the border province of the former Roman Empire and immigrants from north of the Danube (2). While the Baiuvarii are less well known than some other contemporary groups, an interesting archaeological feature in Bavaria from this period is the presence of skeletons with artificially deformed or elongated skulls (Fig. 1*A*).

Artificial cranial deformation (ACD), which is only possible

POPULATION
BIOLOGY

Novel uses of Jupyter: Digital Publication (takes a long time for adoption to actually occur)

The "Paper" of the Future

-  Alyssa Goodman (Harvard University)
-  Josh Peek (Space Telescope Science Institute)
-  Alberto Accomazzi (Harvard-Smithsonian Center for Astrophysics (CFA))
-  CB Chris Beaumont (Harvard-Smithsonian Center for Astrophysics (CFA))
-  CB Christine L. Borgman (UCLA - University of California, Los Angeles)
-  Hope How-Huan Chen (Harvard University)
-  Merce Crosas (Harvard University)
-  CE Christopher Erdmann (North Carolina State University)
-  August Muench
-  Alberto Pepe (Authorea Team)
-  CW Curtis Wong (Microsoft)

show less

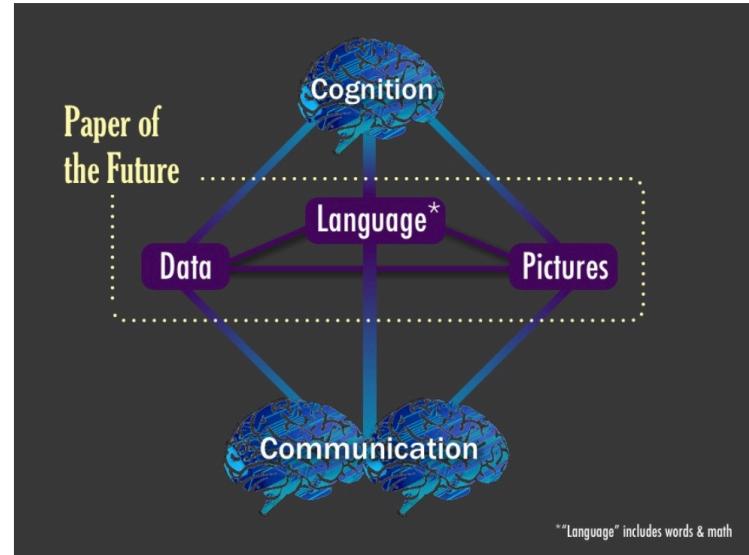


Fig. 1

The Paper of the Future should include seamless linkages amongst **data**, **pictures**, and **language**, where "language" includes both words and math. When an individual attempts to understand each of these kinds of information, different cognitive functions are utilized: communication is inefficient if the channel is restricted primarily to language, without easy interconnection to data and pictures.

R. Connelly and V. Gayle

An investigation of Social Class Inequalities in General Cognitive Ability in Two British Birth Cohorts

“An innovative aspect of this work is that we go beyond providing the usual supplementary material and make the complete workflow openly available. We take the trailblazing step of rendering the complete workflow accessible and reproducible within a Jupyter notebook Jupyter notebooks have been used in Nobel Prize winning high-profile big science applications but are rarely used in Sociology.”

<https://github.com/RoxanneConnelly/Social-Class-Inequalities-in-General-Cognitive-Ability-in-Two-British-Birth-Cohorts>

Jupyter and Education



<https://data.berkeley.edu/news/coursefuture>

Educational Examples

- Interactive notes
 - [12 Steps to Bioinformatics](#), Lorena Barba
- Online textbooks
 - [An Introduction to Applied Bioinformatics](#), Greg Caporaso
- Notebook projects (using research software for education)
 - [JupyterPIC](#) for UCLA EE/Phys M185

Jupyter and HPC



<https://ccit.clemson.edu/research/jupyter/>

Connecting Notebooks with HPC Clusters

- University HPC Clusters
 - Like UCLA's Hoffman2
 - <https://www.hoffman2.idre.ucla.edu/access/jupyter-notebook/>
- National Supercomputing Centers
 - Like DOE's NERSC (National Energy Research Scientific Computing Center)
 - <http://www.nersc.gov/users/data-analytics/data-analytics-2/jupyter-and-rstudio/>

How to share

- Use the notebook menu
- jupyter nbconvert (converts notebooks to html, pdf, etc)
- Binder
- nbviewer.jupyter.org



Turn a GitHub repo into a collection of
interactive notebooks

Have 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

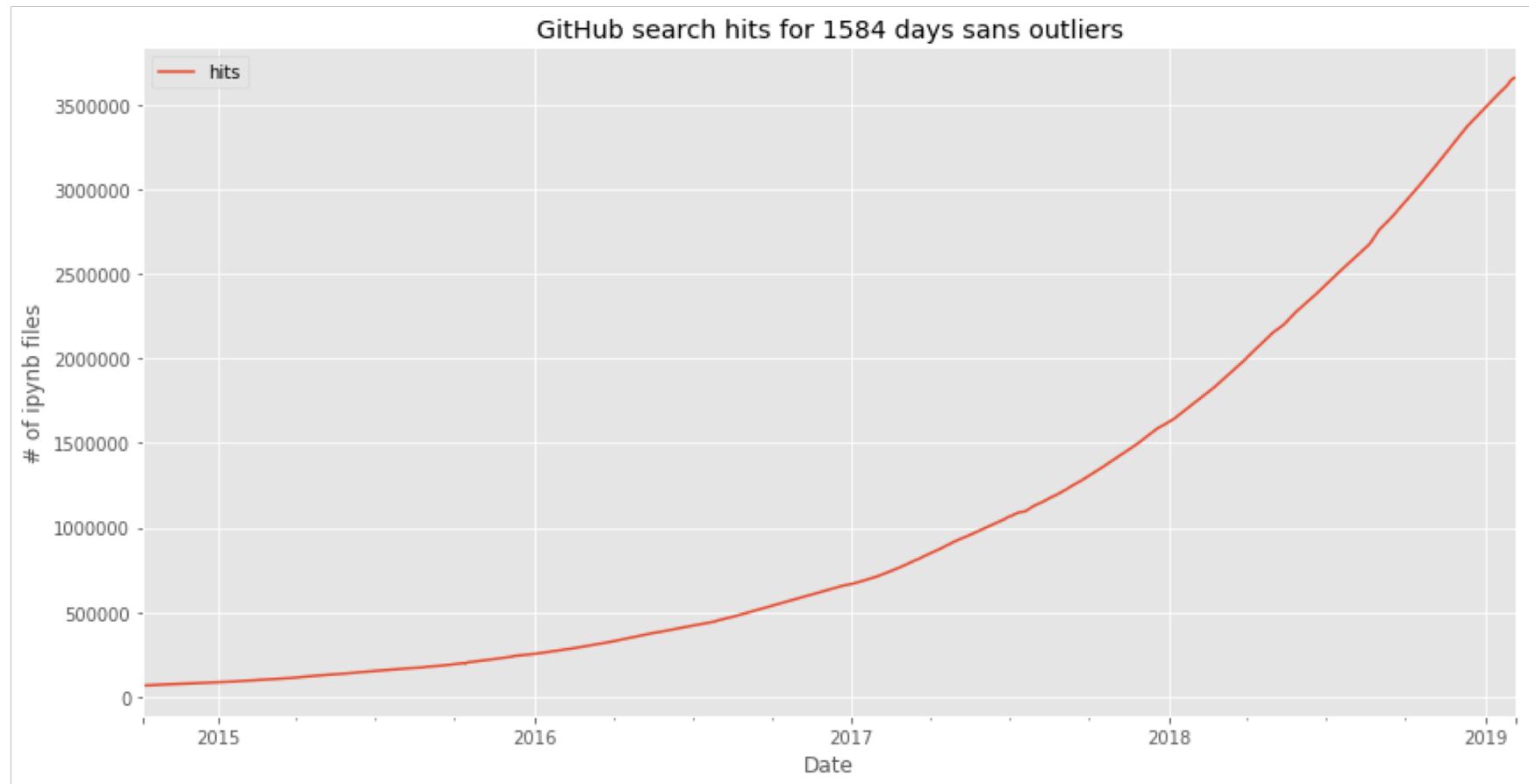
GitHub repository name or URL
 GitHub ▾

Git branch, tag, or commit
 Path to a notebook file (optional) File ▾

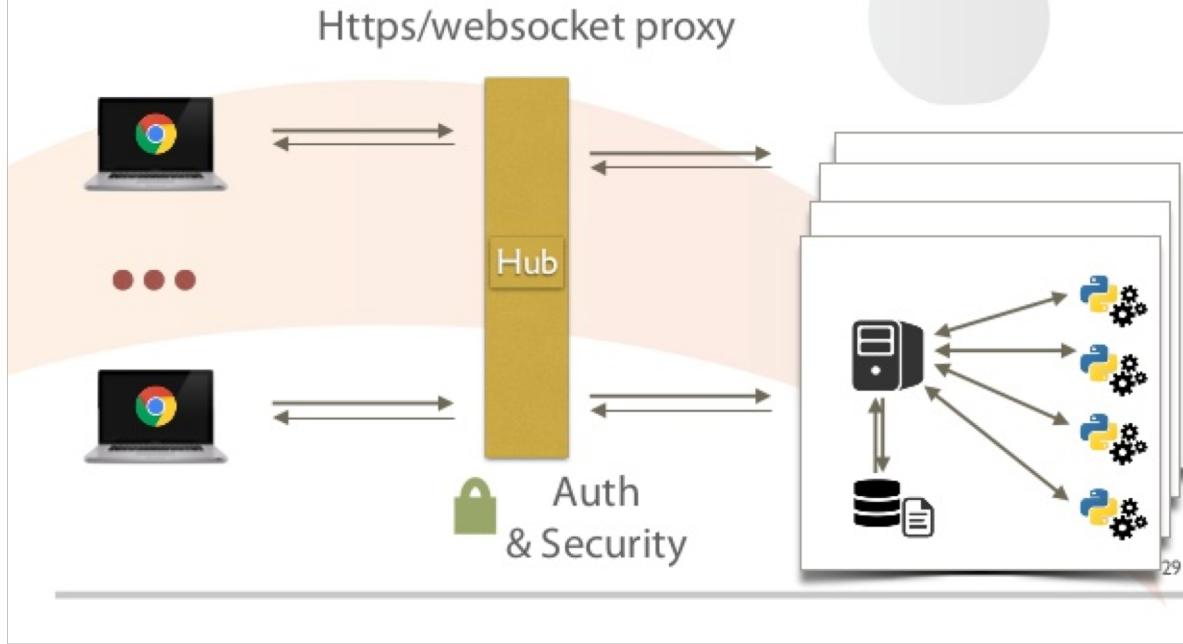
Copy the URL below and share your Binder with others:

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

Estimate of public Jupyter notebooks on GitHub



jupyterHub



Next Generation: JupyterLab

The screenshot shows the JupyterLab interface. On the left is a sidebar with tabs for Files, Running, Commands, Cell Tools, and Tabs. The Files tab is active, showing a list of notebooks: Data.ipynb (an hour ago), Fasta.ipynb (a day ago), Julia.ipynb (a day ago), Lorenz.ipynb (seconds ago), R.ipynb (a day ago), iris.csv (a day ago), lightning.json (9 days ago), and lorenz.py (3 minutes ago). The Running tab shows no running processes.

The main area has several tabs: Lorenz.ipynb (active), Terminal 1, Console 1, Data.ipynb, README.md. The Lorenz.ipynb tab contains text about the Lorenz system and its differential equations:

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

Below this, it says: "Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors."

In [4]:

```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

The Output View tab shows three sliders for parameters: sigma (10.00), beta (2.67), and rho (28.00). Below the sliders is a 2D plot of the Lorenz attractor, which is a complex, fractal-like shape formed by the trajectories of the differential equations.

The code editor tab shows the contents of lorenz.py:

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim(5, 55)

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random(N, 3)
```

Notebooks: in JupyterLab

Final item:
please complete the brief
post-course survey

<https://bit.ly/2VaYkVT>