IP[y]:
IPython

The Architecture of Jupyter

Interactive by design

Fernando Pérez
(@fperez_org & fperez@lbl.gov)

LBL & UC Berkeley

Mandatory COI disclosure

“The purpose of computing is insight, not numbers”

–Hamming’62

The Lifecycle of a Scientific Idea (schematically)

1. Individual exploratory work
2. Collaborative development
3. Parallel production runs (HPC, cloud, ...)
4. Publication & communication (reproducibly!)
5. Education
6. Goto 1

The Lifecycle of a Scientific Idea (schematically)

1. Individual exploratory work
2. Collaborative development
3. Parallel production runs (HPC, cloud, ...)
4. Publication & communication (reproducibly!)
5. Education
6. Goto 1

We treat this as a single, coherent problem

IPython: CU Boulder, 2001

or how to best procrastinate on a Physics dissertation

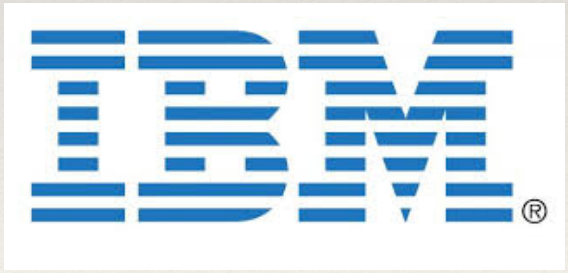
/bin/bash

```
In [13]: run ~/scratch/error
reps: 5
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/home/fperez/scratch/error.py in <module>()
     70 if __name__ == '__main__':
     71     #explode()
     72
---> 72     main()
     73     g2='another global'

/home/fperez/scratch/error.py in main()
     60     array_num = zeros(size,'d')
     61     for i in xrange(reps):
---> 62         RampNum(array_num, size, 0.0, 1.0)
     63     RNtime = time.clock()-t0
     64     print 'RampNum time:', RNtime

/home/fperez/scratch/error.py in RampNum(result, size, start, end)
     43     tmp = zeros(size+1)
     44     step = (end-start)/(size-1-tmp)
---> 45     result[:] = arange(size)*step + start
     46
     47 def main():

ValueError: shape mismatch: objects cannot be broadcast to a single shape

In [14]:
```

Real credit goes to whole team

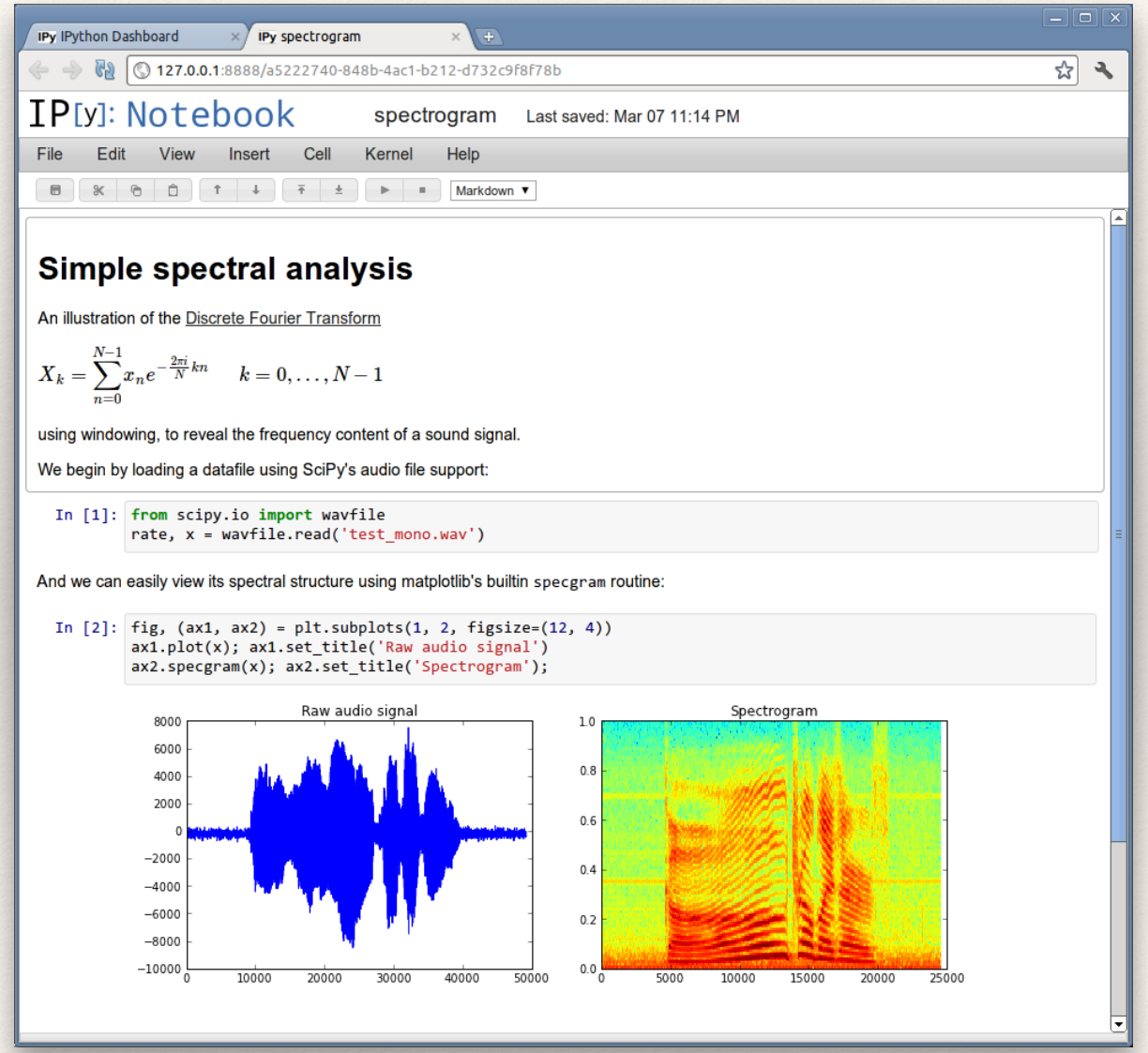
Plus ~ 500 more Open source contributors!

Funding and partnerships



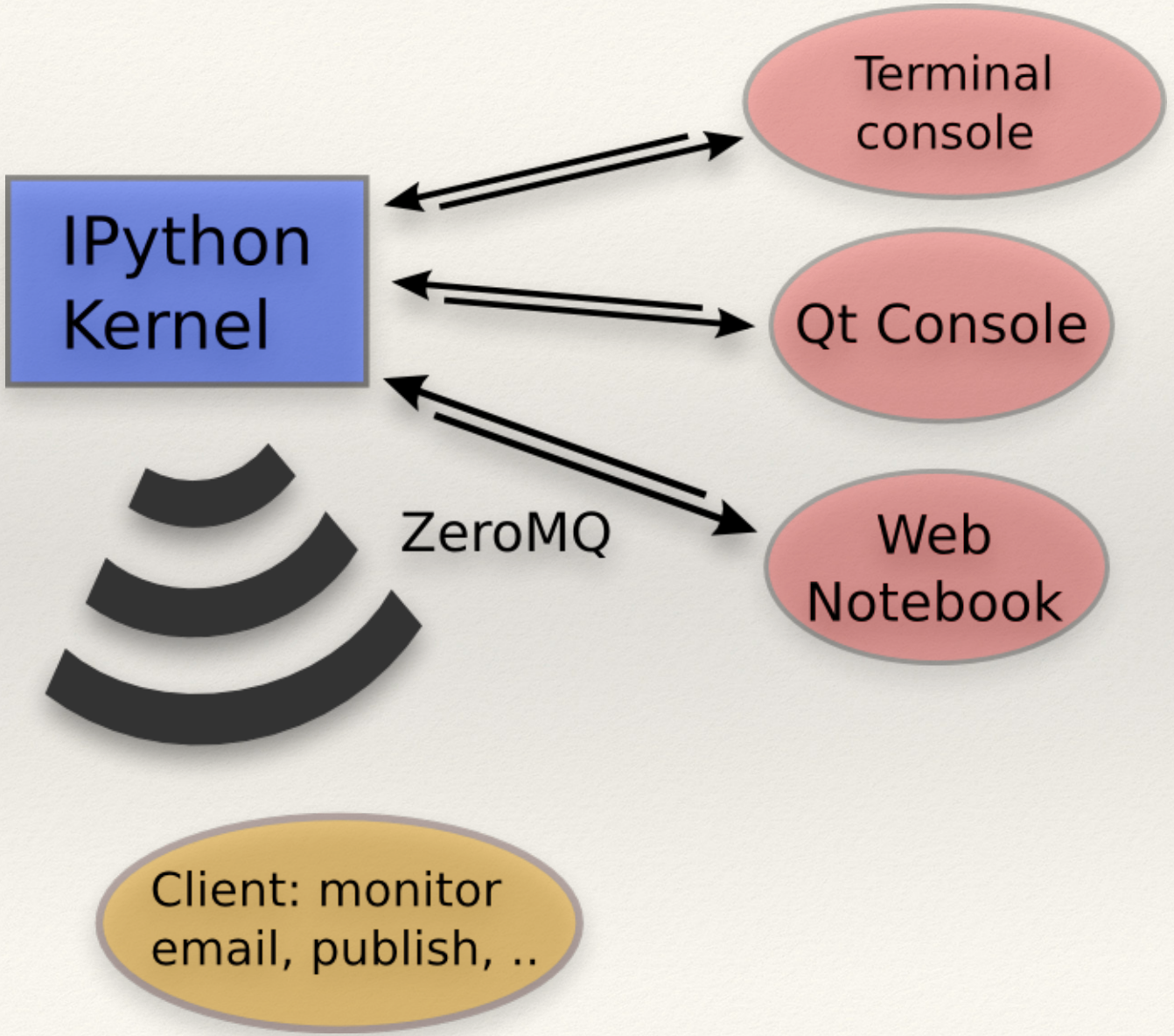
2011: The IPython Notebook

- Rich web client
- Text & math
- Code
- Results
- Share, reproduce.



From IPython to Project Jupyter

A simple and generic architecture



IPython

- Interactive Python shell at the terminal
- Kernel for this protocol in Python
- Tools for Interactive Parallel computing
- Network protocol for interactive computing
- Clients for protocol
  - Console
  - Qt Console
  - Notebook
- Notebook file format & tools (nbconvert...)
- Nbviewer

IPython ... Jupyter

- Interactive Python shell at the terminal
- Kernel for this protocol in Python
- Tools for Interactive Parallel computing

- Network protocol for interactive computing
- Clients for protocol
  - Console
  - Qt Console
  - Notebook
- Notebook file format & tools (nbconvert...)
- Nbviewer

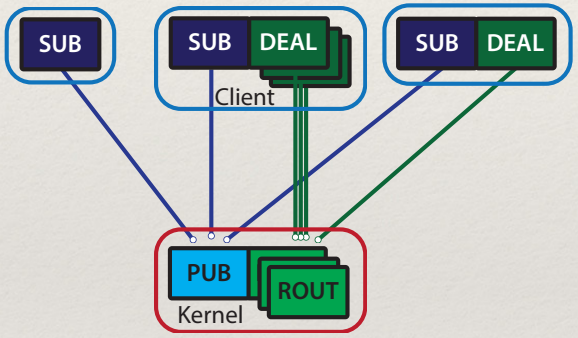
Language Agnostic

Foundational ideas: the web

- HTTP and HTML:
  - A protocol to transfer content.
  - A format to represent it.

Foundational ideas: Jupyter

Rich REPL Protocol



ØMQ + JSON

Document Format

We have already computed $P(X|A)$ above. On the other hand, $P(X|\sim A)$ is subjective: our code can pass tests but still have a bug in it, though the probability there is a bug present is reduced. Note this is dependent on the number of tests performed, the degree of complication in the tests, etc. Let's be conservative and assign $P(X|\sim A) = 0.5$. Then

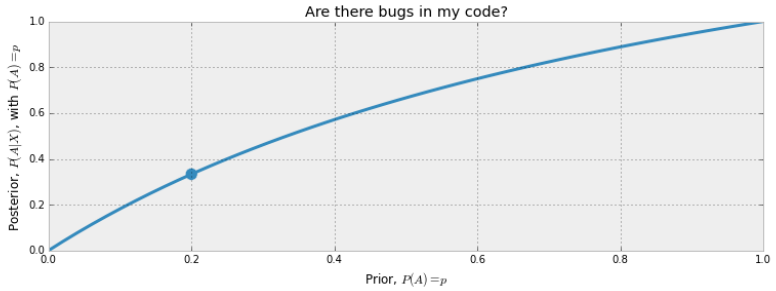
$$P(A|X) = \frac{1 \cdot p}{1 \cdot p + 0.5(1-p)}$$

$$= \frac{2p}{1+p}$$

This is the posterior probability. What does it look like as a function of our prior, $p \in [0, 1]$?

```
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, 2 * p / (1 + p), color="#348ABD", lw=3)
# plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=["#A60628"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=140, c="#348ABD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, with $P(A) = p$")
plt.title("Are there bugs in my code?")
```

```
<matplotlib.text.Text at 0x1051de650>
```



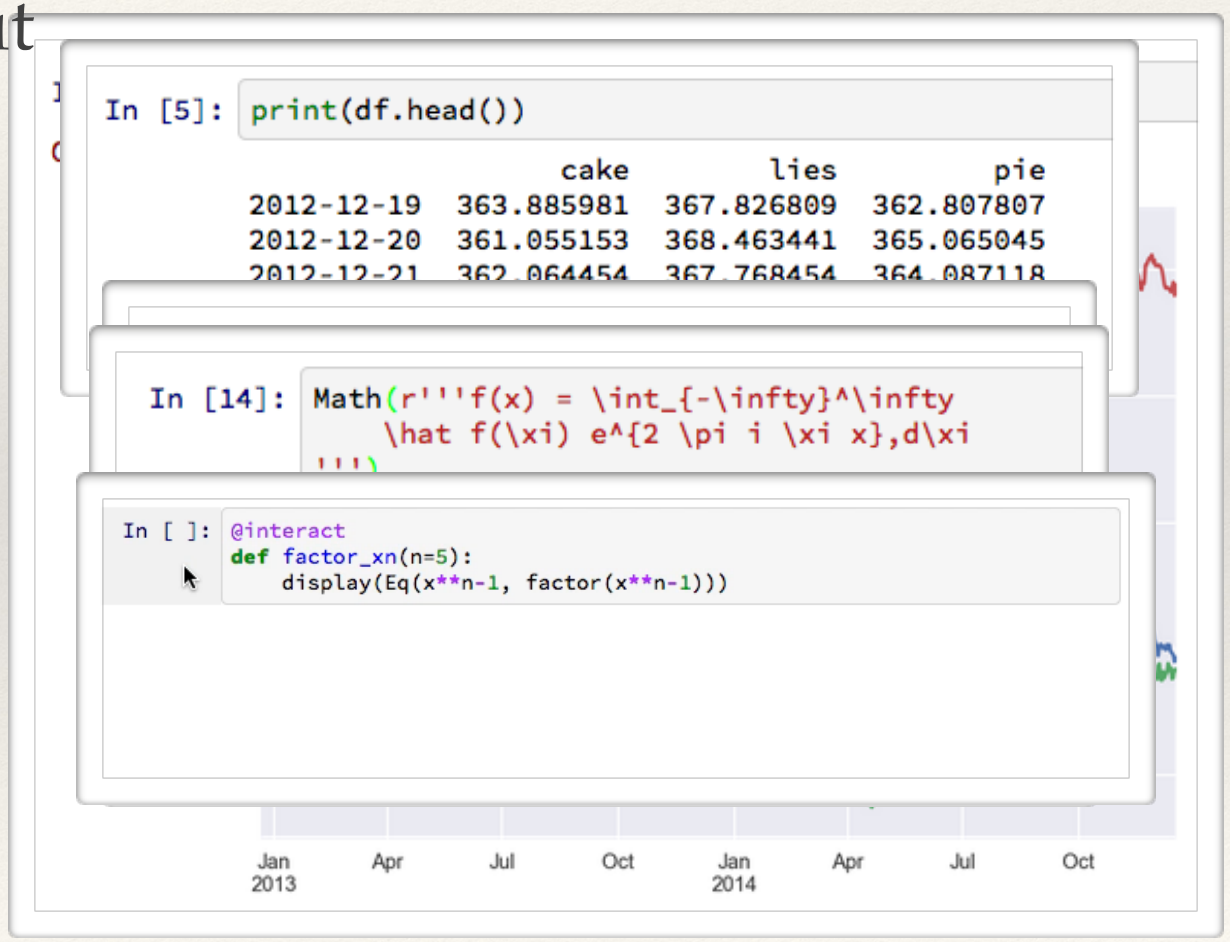
https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

Jupyter Protocol

supercharge the P in REP*L

any mime-type output

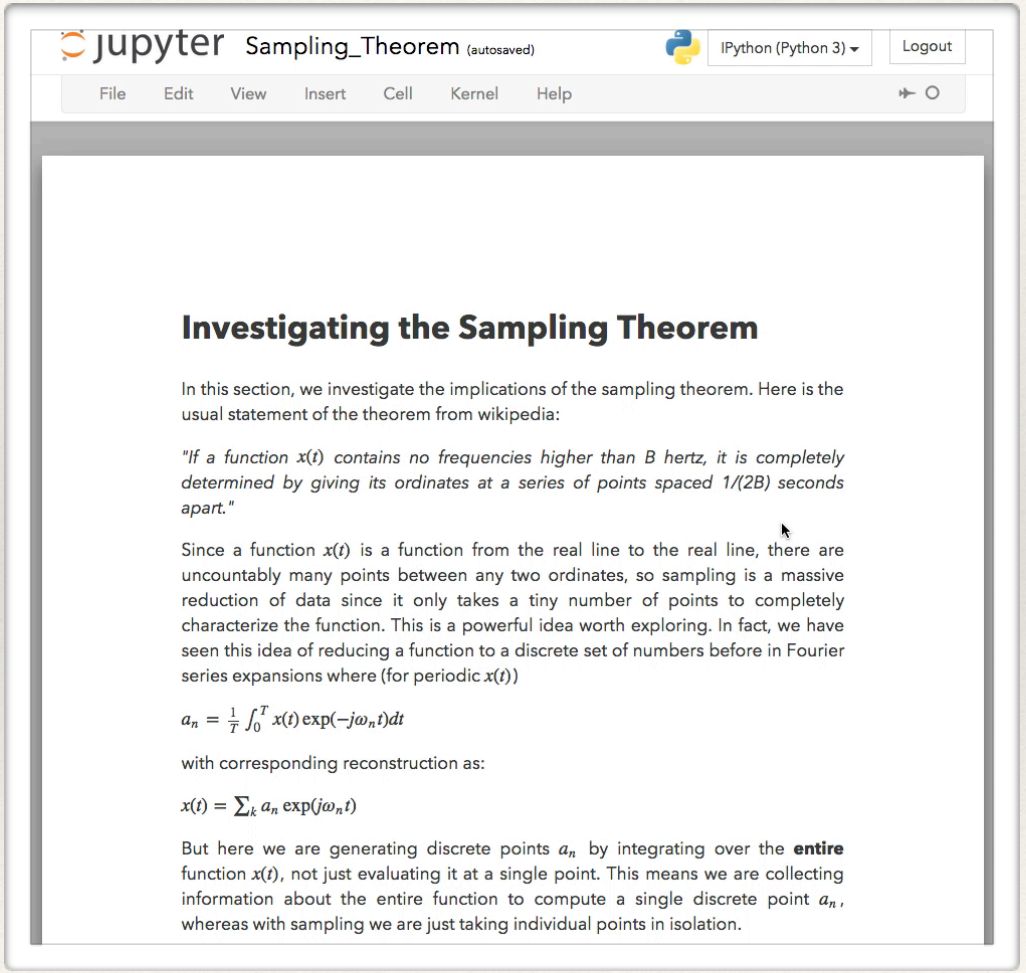
- text
- svg, png, jpeg
- latex, pdf
- html, javascript
- interactive widgets



Jupyter Protocol
is language agnostic

Jupyter Notebooks

- Plain text (JSON)
- Publicly documented schema
- Machine readable, easy to understand (metadata everywhere)



- Notebook = sequence of cells
- text cell = markdown + latex)
- code cell = input + output
- Transformable (nbconvert)

https://github.com/unpingco/Python-for-Signal-Processing

The Jupyter Notebook Ecosystem

nbviewer: seamless notebook sharing

- Zero-install reading of notebooks
- Just share a URL
- nbviewer.jupyter.org

Reproducible Research (2012, w/Rob Knight): Paper, Notebooks and Virtual Machine

www.nature.com/ismej/journal/v7/n3/full/ismej2012123a.html

The ISME Journal
Multidisciplinary Journal of Microbial Ecology

Login Cart

Search go Advanced search

Journal home > Archive > Commentaries > Full text

Journal home
Advance online publication
About AOP
Current issue
Archive
Focuses

Commentary

The ISME Journal (2013) **7,** 461–464; doi:10.1038/ismej.2012.123; published online 25 October 2012

Collaborative cloud-enabled tools allow rapid, reproducible biological insights

Open

FULL TEXT
Previous | Next
Table of contents
Download PDF
Send to a friend
View interactive PDF in

agan-Kelley[1,12], William Anton Walters[2,12], onald[3,6,12], Justin Riley[4], Brian E Granger[5], nzalez[6], Rob Knight[7,8], Fernando Perez[9] and J poraso[10,11]

Group in Applied Science and Technology, University of t Berkeley, Berkeley, CA, USA
nt of Molecular, Cellular and Developmental Biology, f Colorado at Boulder, Boulder, CO, USA
s Institute, University of Colorado at Boulder, Boulder, CO,
ducational Innovation and Technology, Massachusetts Technology, Cambridge, MA, USA
epartment, California Polytechnic State University, San Luis USA
nt of Computer Science, University of Colorado at Boulder,), USA

This notebook is intended to calculate the positions of primers in an alignment, using functions from PrimerProspector.

Import the needed functions, and define the primer sequences

```
In [8]: # Code modified from PrimerProspector library slice_aligned_region.py (development version
        )

        # Imports and definitions
        from string import lower, upper
        from operator import itemgetter

        from cogent import LoadSeqs, DNA
        from cogent.core.alphabet import AlphabetError
        from cogent.align.align import make_dna_scoring_dict, local_pairwise
        from cogent.parse.fasta import MinimalFastaParser
        from cogent.core.moltype import IUPAC_DNA_ambiguities

        DNA_CODES = ['A', 'C', 'T', 'G', 'R', 'Y', 'M', 'K',
         'W', 'S', 'B', 'D', 'H', 'V', 'N']


        # Note that these are all written 5'->3', the reverse primers are reverse complemented for
         the local alignment

        # If one wanted to test different primers, they would be defined here.

        # 27f/338r = V2 (also includes V1, but generally just referred to as V2)
        # 349f/534r = V3
        # 515f/806r = V4
        # 967f/1046r = V6
        # 1391f/1492r = V9

        primer_seqs = {
                    '27f':'AGAGTTTGATCMTGGCTCAG',
                    '338r':DNA.rc('GCTGCCTCCCGTAGGAGT'),
                    '349f':'GYGCASCAGKCGMGAAW',
                    '534r':DNA.rc('ATTACCGCGGCTGCTGG'),
                    '515f':'GTGCCAGCMGCCGCGGTAA',
                    '806r':DNA.rc('GGACTACVSGGGTATCTAAT'),
                    '967f':'CAACGCGAAGAACCTTACC',
                    '1048r':DNA.rc('CGRCRGCCATGYACCWC'),
                    '1391f':'TGYACACACCGCCCGTC',
                    '1492r':DNA.rc('GGCTACCTTGTTACGACTT'),
                    '1391r':'TGYACACACCGCCCGTC' # Need this rather than forward primer to get prope
        r 3' position of reverse version
                    }

        reference_aligned_file = '/home/ubuntu/qiime_software/gg_otus-4feb2011-release/rep_set/gg_
        76_otus_4feb2011_aligned.fasta'
```

Instructions and supporting data for the QIIME/IPython/StarCluster demo at the *2012 NIH Cloud Computing the Microbiome* workshop and our corresponding paper in the ISME Journal.

The analysis made use of the IPython Notebook, QIIME, StarCluster, PyCogent, and PrimerProspector. All of these tools are pre-installed in the `ami-9f69c1f6` public Amazon EC2 instance, which was used in this study.

Supporting Files

The IPython notebooks supporting this study can be viewed here and are available here in PDF format:

- NIH Cloud Demo (Complete)
- NIH Cloud Demo (Fast)
- Timing*
- Variable Region Position Boundaries
- Pearson v Robinson-Foulds Distances
- V3 and V4 Regions Only

* Note that the Timing notebook is for reference as related to the paper only - it will not be directly reproducible on re-runs of the above notebooks as it relies on the semi-manual creation of the `tasks.log` file. The `tasks.log` file used to generate the original timing data is available for download here.

The Greengenes reference OTU collection used in this study is available for download here.

The IPython notebook files (`.ipynb`) are available for download here.

The tree metadata mapping file used in generating the coloring categories in the 3D PCoA plot is available here.

The paper for this analysis, "Collaborative cloud-enabled tools allow rapid, reproducible biological insights", is available here.

Reproducing the analysis

Four `m2.4xlarge` instances were booted using StarCluster to create a 32 core cluster with approximately 280GB of RAM (70GB per 8 core instance). This was used for the full analysis (a more complete analysis then was done during the workshop, where the workshop analysis was optimized to run quickly). To support the large quantity of data that is generated during the analysis, you should create an EBS volume which will be attached to the running instance. A 20 GB volume will be sufficient. The volume used for running these notebooks is available as `snap-75eb8005`.

To reproduce the analyses presented in this paper you should install StarCluster locally, and configure it according to the instructions on the StarCluster website. You can then add the following to your `~/.starcluster/config` file:

```
[plugin ipcluster]
setup_class = starcluster.plugins.ipcluster.IPCluster
enable_notebook = true
# If you leave notebook_passwd out, a random password
# will be generated instead.
notebook_passwd = YOUR-PASSWORD

[cluster qiime-ipython]
node_image_id = ami-9f69c1f6
cluster_user = ubuntu
keyname = YOUR-KEY
cluster_size = 4
node_instance_type = m2.4xlarge
plugins = ipcluster
volumes = qiime-ipython-data

[volume qiime-ipython-data]
VOLUME_ID = YOUR-VOLUME-ID
MOUNT_PATH = /home/ubuntu/data
```

http://www.nature.com/ismej/journal/v7/n3/full/ismej2012123a.html

http://qiime.org/home_static/nih-cloud-apr2012

Today: mybinder.org

binder

Turn a GitHub repo into a collection of interactive notebooks powered by Jupyter and Kubernetes.

github.com/freeman-lab

1 Tell us your GitHub repo

user/project OR github url

2 Configure dependencies

> none for basic Python projects

requirements.txt for pip Python projects

environment.yml for conda Python projects

Dockerfile for custom builds

3 Attach services

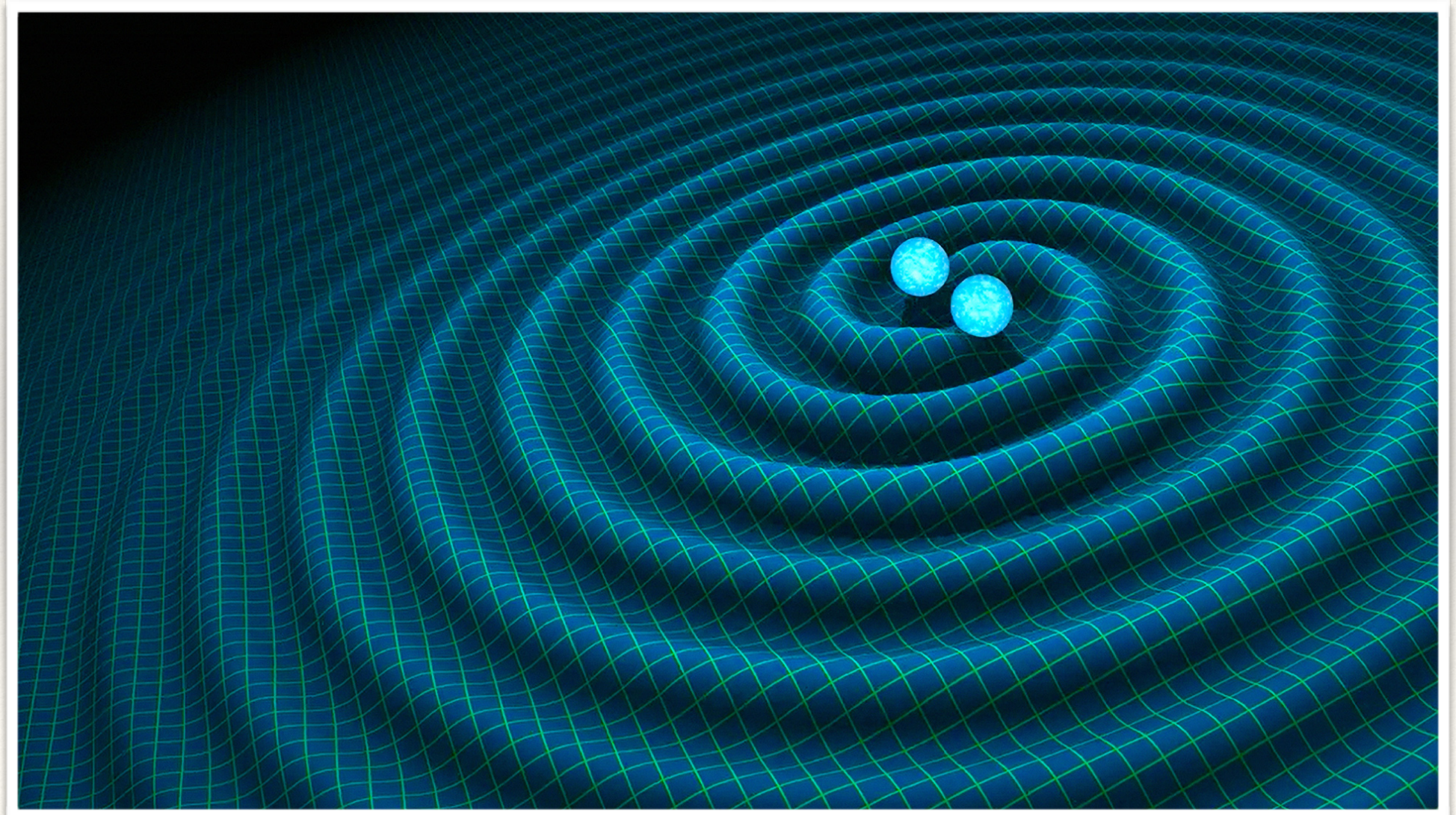
Postgres

Spark

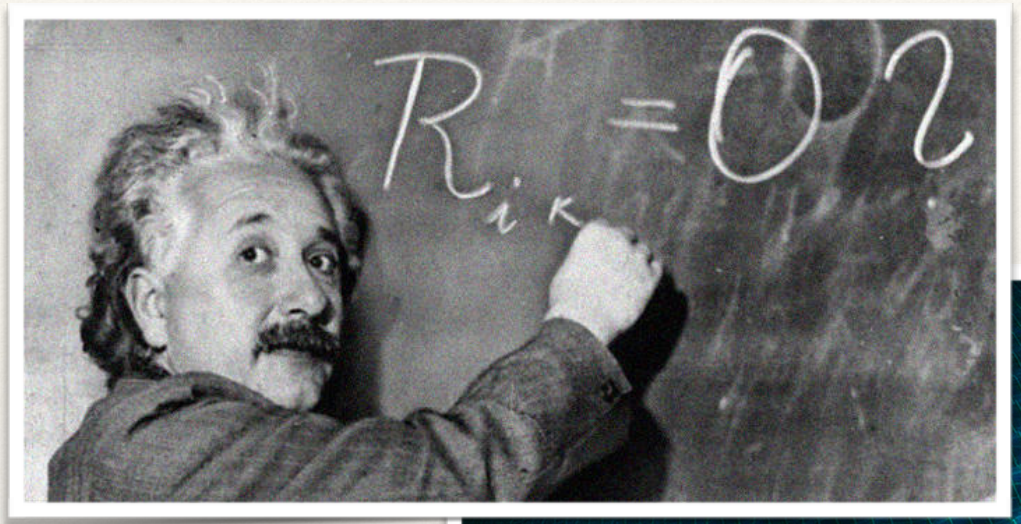
make my binder

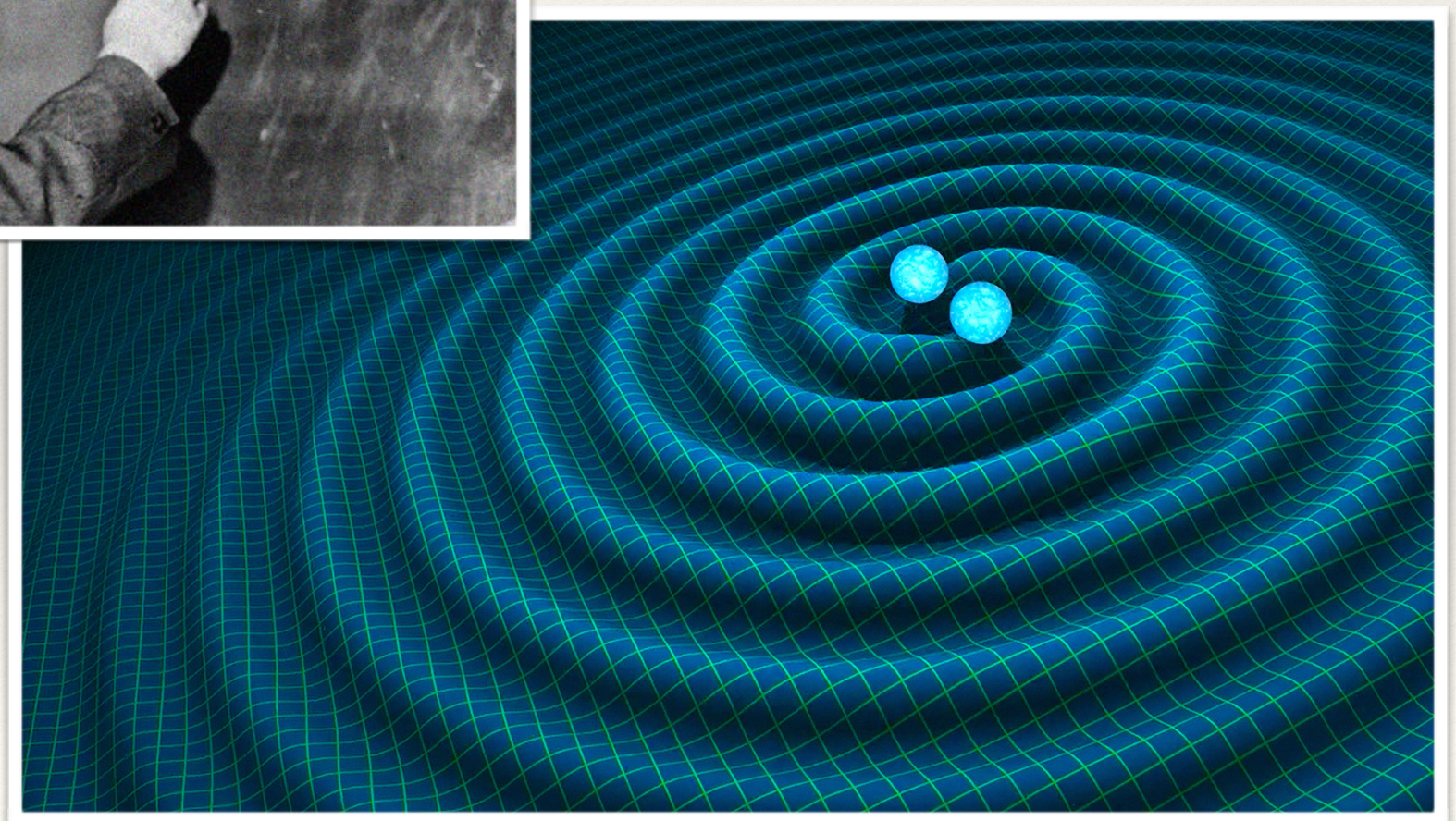
This just in!

A long time ago in a galaxy far, far away…



A long time ago in a galaxy far, far away…





Two identical detectors: Hanford, WA and Livingston, LA

LIGO: a feat of science & engineering

Detection problem:

- ~ 1/1000 proton over 4 km.
- Sensitivity ~ 1e-21
- Milky Way: 1e+21m across!

September 14, 2015

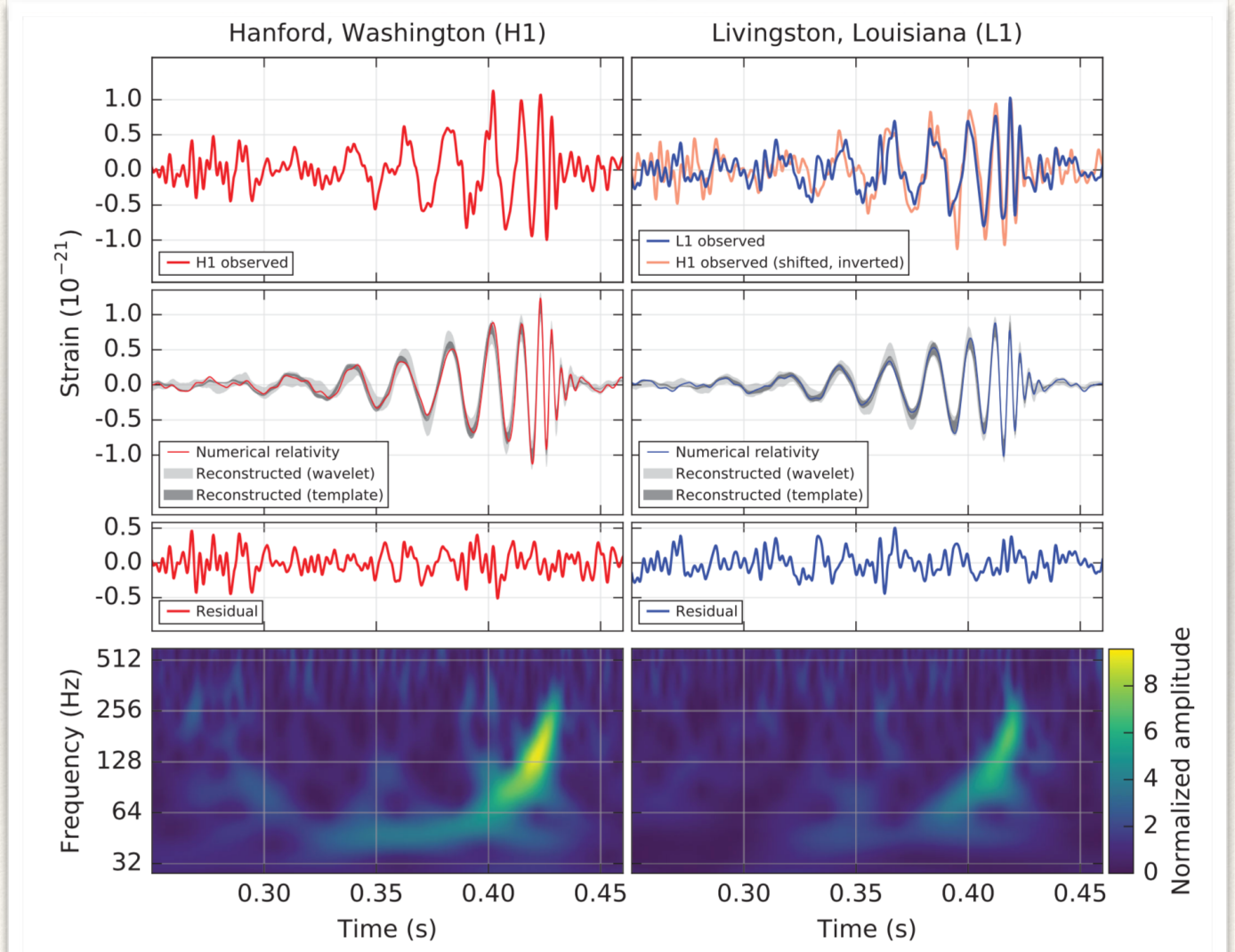
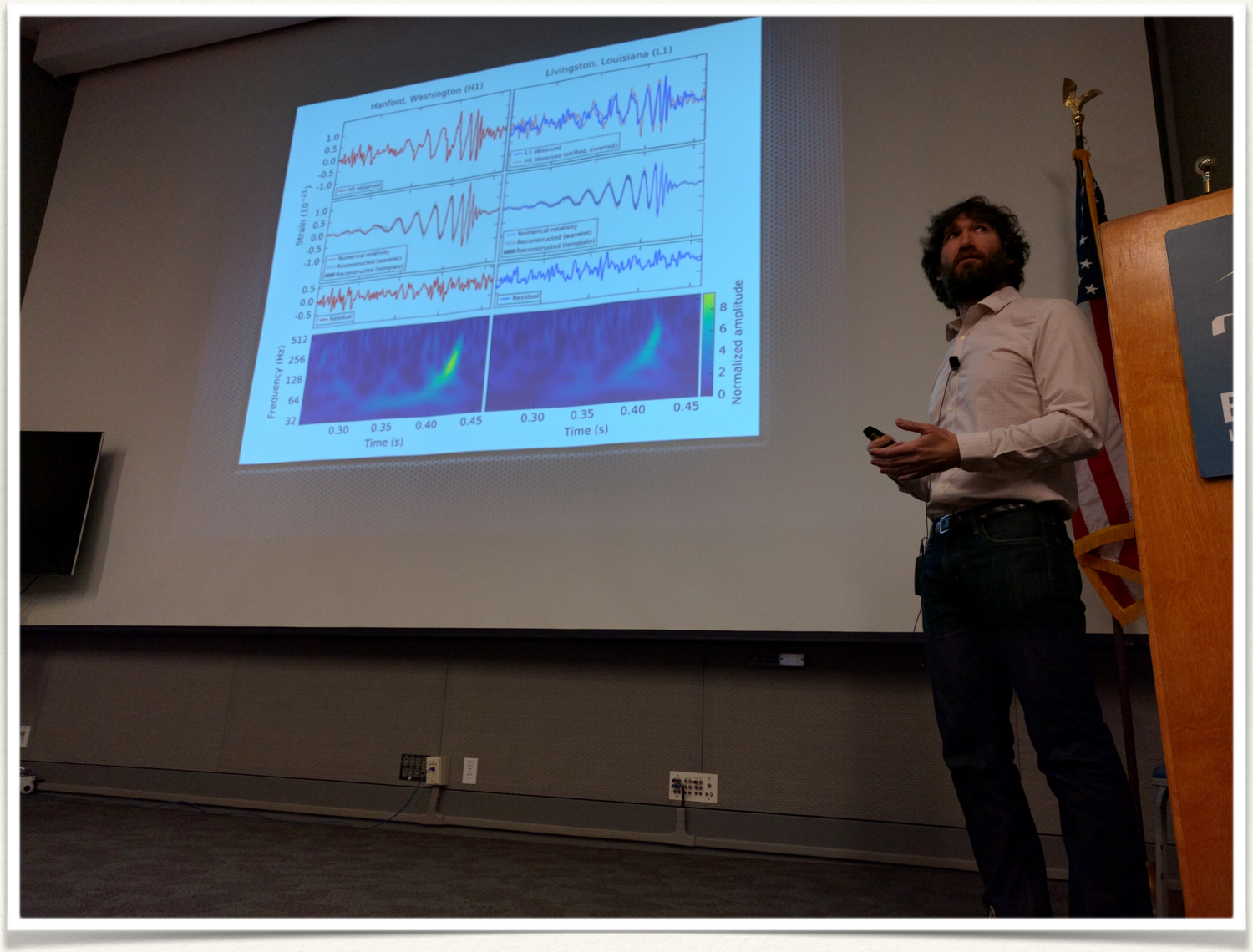


FIG. 1. The gravitational-wave event GW150914 observed by the LIGO Hanford (H1, left column panels) and Livingston (L1, right column panels) detectors. Times are shown relative to September 14, 2015 at 09:50:45 UTC. For visualization, all time series are filtered with a 35–350 Hz bandpass filter to suppress large fluctuations outside the detectors' most sensitive frequency band, and band-reject

February 19, 2016: Dan Holz @ LBL



Gravitational waves detected on Jupyter!

jupyter GW150914_tutorial Last Checkpoint: 02/13/2016 (unsaved changes)

File Edit View Insert Cell Kernel Help IPython (Python 3)

The filtered data peak at around 1.e-21, 1000 times smaller than the scale in the first plot. The "DC" offset between H1 and L1 data visible in the first plot is no longer visible here; the bandpassing cuts off frequency components below around 40 Hz.

Now, as with whitening, the signal is visible as an oscillation sweeping from low to high frequency from -0.10 seconds to 0, then damping down into the random noise. Again, it looks roughly the same in both detectors, after shifting and flipping the L1 data with respect to H1. It's exactly the kind of signal we expect from the inspiral, merger and ringdown of two massive black holes.

And as with whitening, the NR waveform looks, by eye, to be a good match to the data in both detectors; the signal is consistent with the waveform predicted from General Relativity.

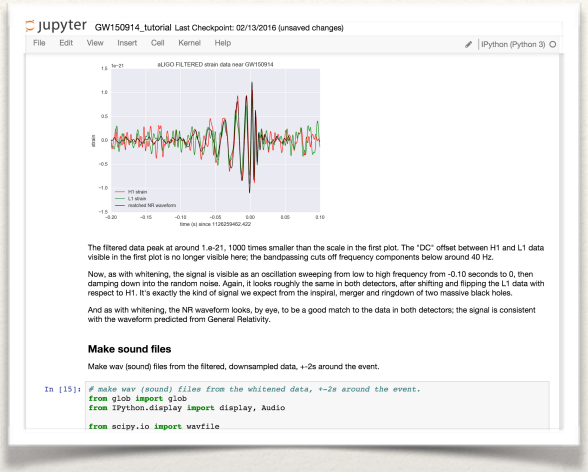
Make sound files

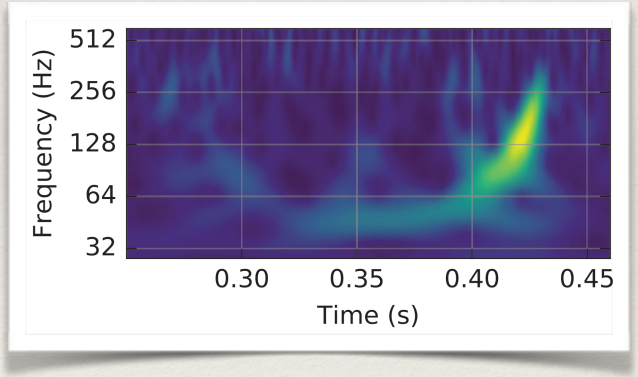
Make wav (sound) files from the filtered, downsampled data, +-2s around the event.

In [15]:
```
# make wav (sound) files from the whitened data, +-2s around the event.
from glob import glob
from IPython.display import display, Audio

from scipy.io import wavfile
```

From LIGO Open Science Center, binder-ified: github.com/minrk/ligo-binder

The song of the universe





Make sound files

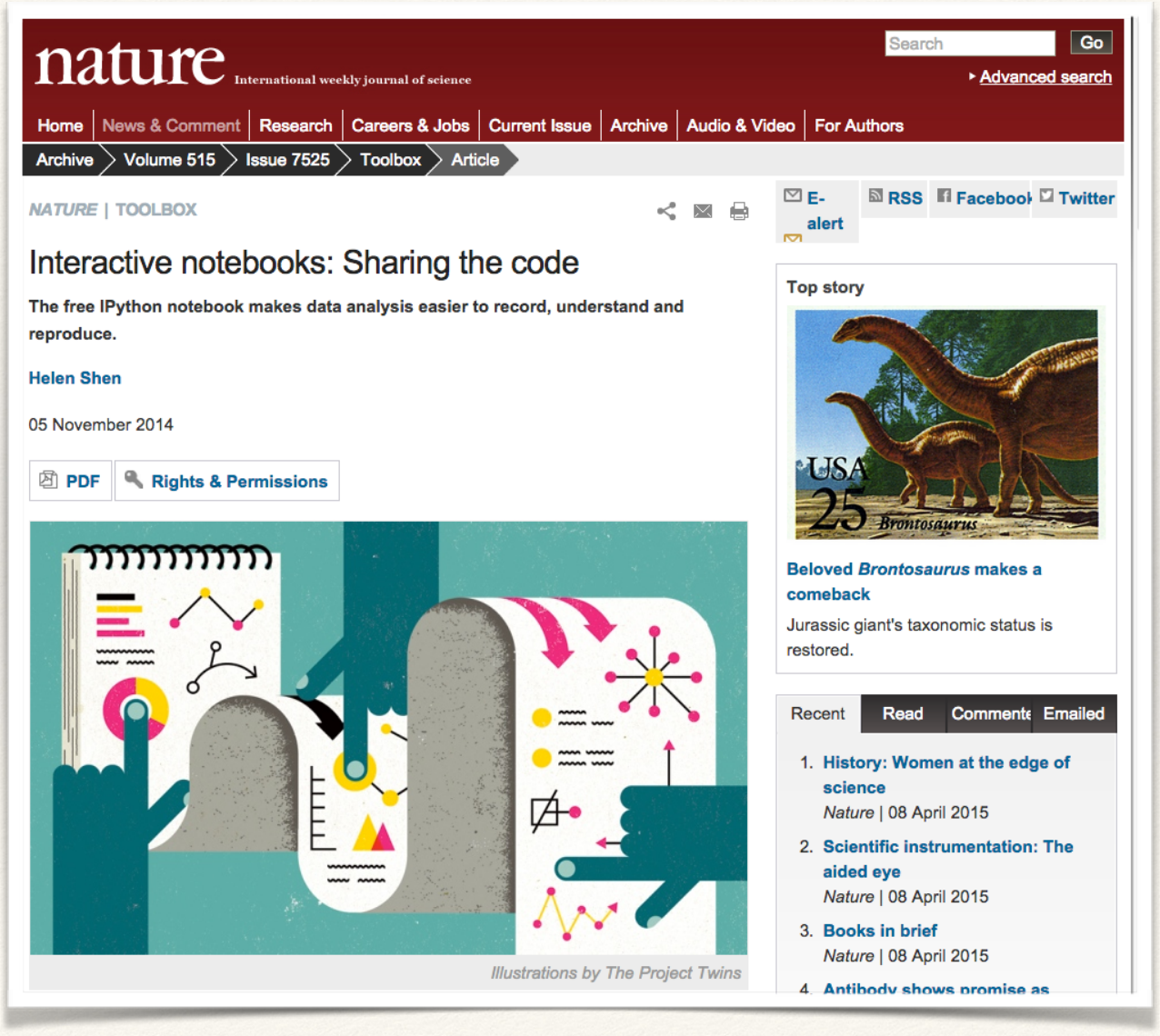
Make wav (sound) files from the filtered, downsampled data, +-2s around the event.

```
# make wav (sound) files from the whitened data, +-2s around the event.
from glob import glob
from IPython.display import display, Audio

from scipy.io import wavfile

# function to keep the data within integer limits, and write to wavfile:
def write_wavfile(filename,fs,data):
    d = np.int16(data/np.max(np.abs(data)) * 32767 * 0.9)
    wavfile.write(filename,int(fs), d)

tevent = 1126259462.422         # Mon Sep 14 09:50:45 GMT 2015
deltat = 2.                     # seconds around the event

# index into the strain time series for this time interval:
indxt = np.where((time >= tevent-deltat) & (time < tevent+deltat))

# write the files:
write_wavfile("GW150914_H1_whitenbp.wav",int(fs), strain_H1_whitenbp[indxt])
write_wavfile("GW150914_L1_whitenbp.wav",int(fs), strain_L1_whitenbp[indxt])
write_wavfile("GW150914_NR_whitenbp.wav",int(fs), NR_H1_whitenbp)

for wav in glob('*whitenbp.wav'):
    display(wav)
    display(Audio(filename=wav))
```

```
'GW150914_H1_whitenbp.wav'
```

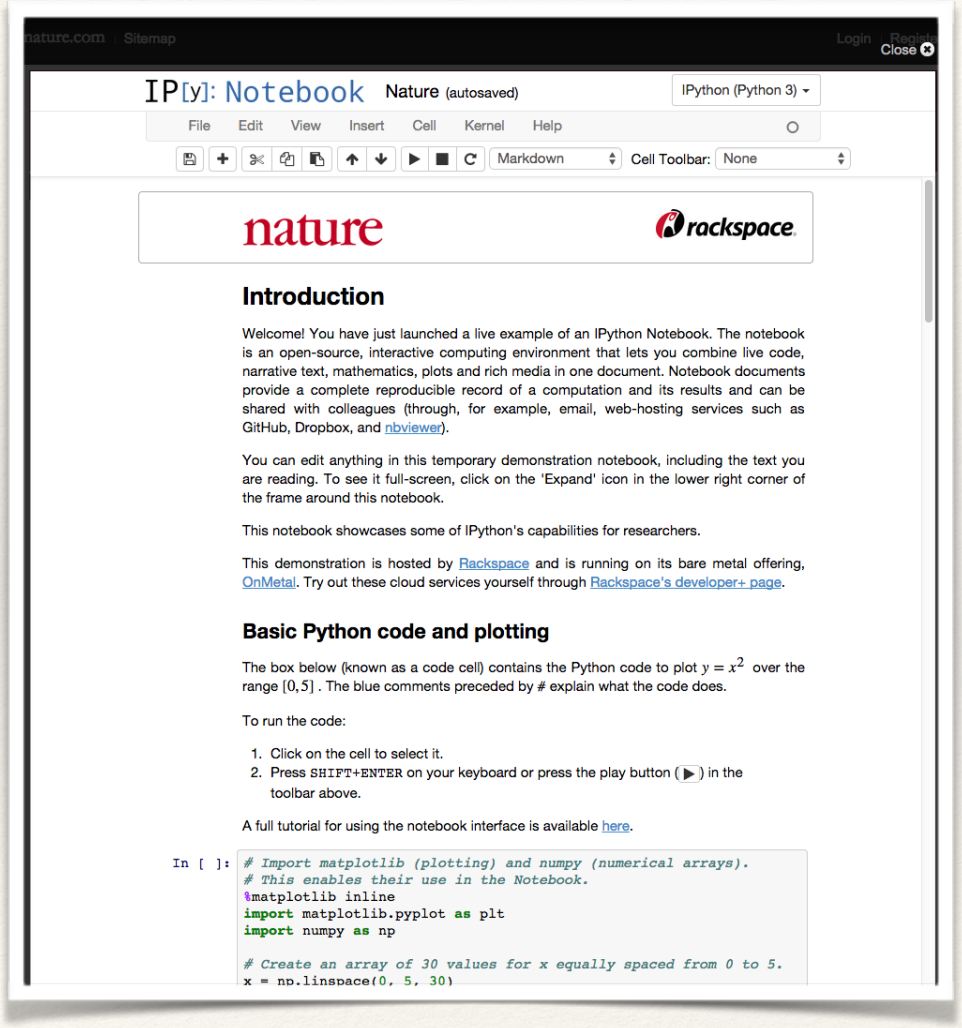
Using the IPython.display.Audio object

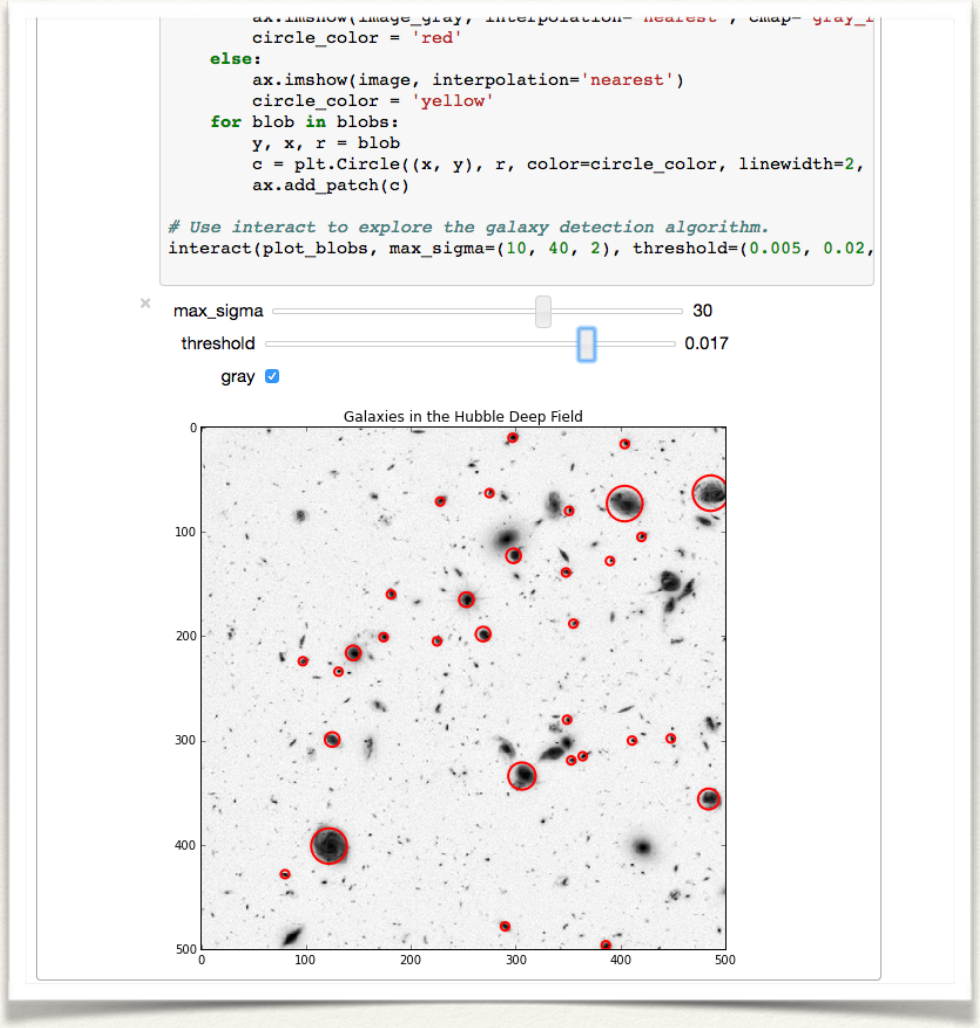
Changing the scientific culture



http://www.nature.com/news/interactive-notebooks-sharing-the-code-1.16261

Executable papers: the future?



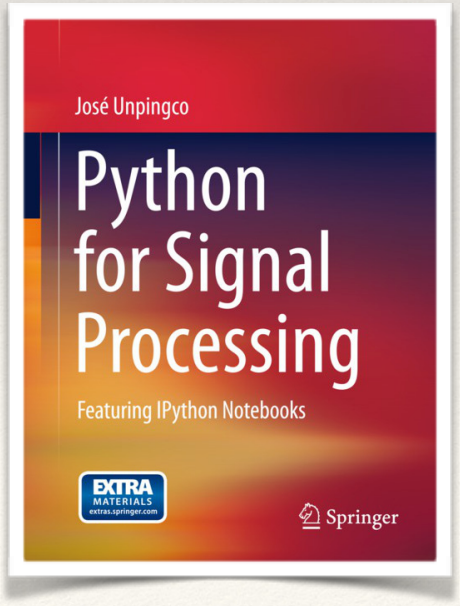


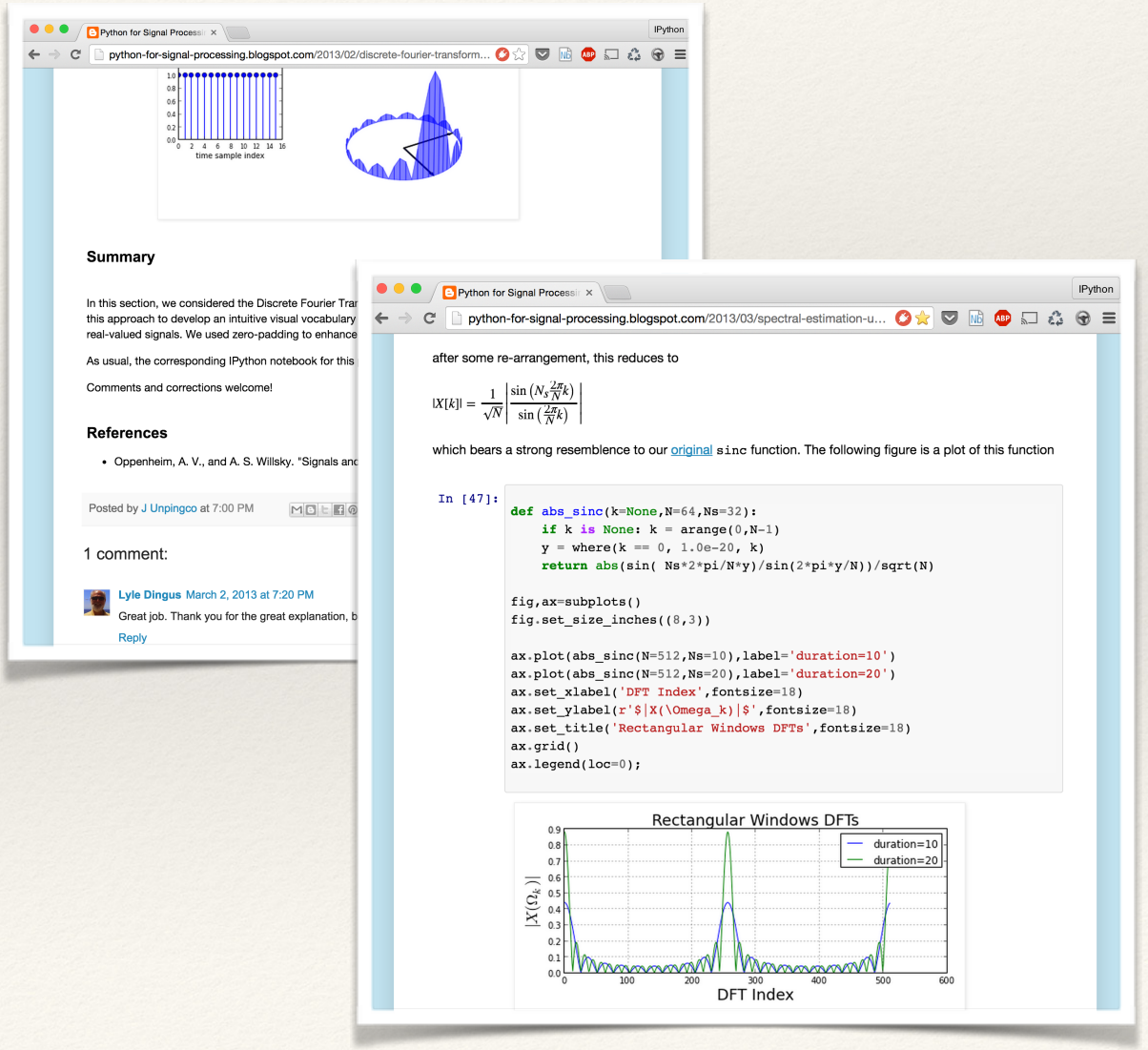
http://www.nature.com/news/ipython-interactive-demo-7.21492?article=1.16261

Executable books

Python for Signal Processing, by José Unpingco

- Springer hardcover book
- Chapters: IPython Notebooks
- Posted as a blog entry
- All available as a Github repo





JupyterHub: multiuser support

Jupyter for Organizations

JupyterHub is a multiuser version of the notebook designed for centralized deployments in companies, university classrooms and research labs.

Pluggable authentication

Manage users and authentication with PAM, OAuth or integrate with your own directory service system. Collaborate with others through the Linux permission model.

Centralized deployment

Deploy the Jupyter Notebook to all users in your organization on centralized servers on- or off-site.

Container friendly

Use Docker containers to scale your deployment and isolate user processes using a growing ecosystem of prebuilt Docker containers.

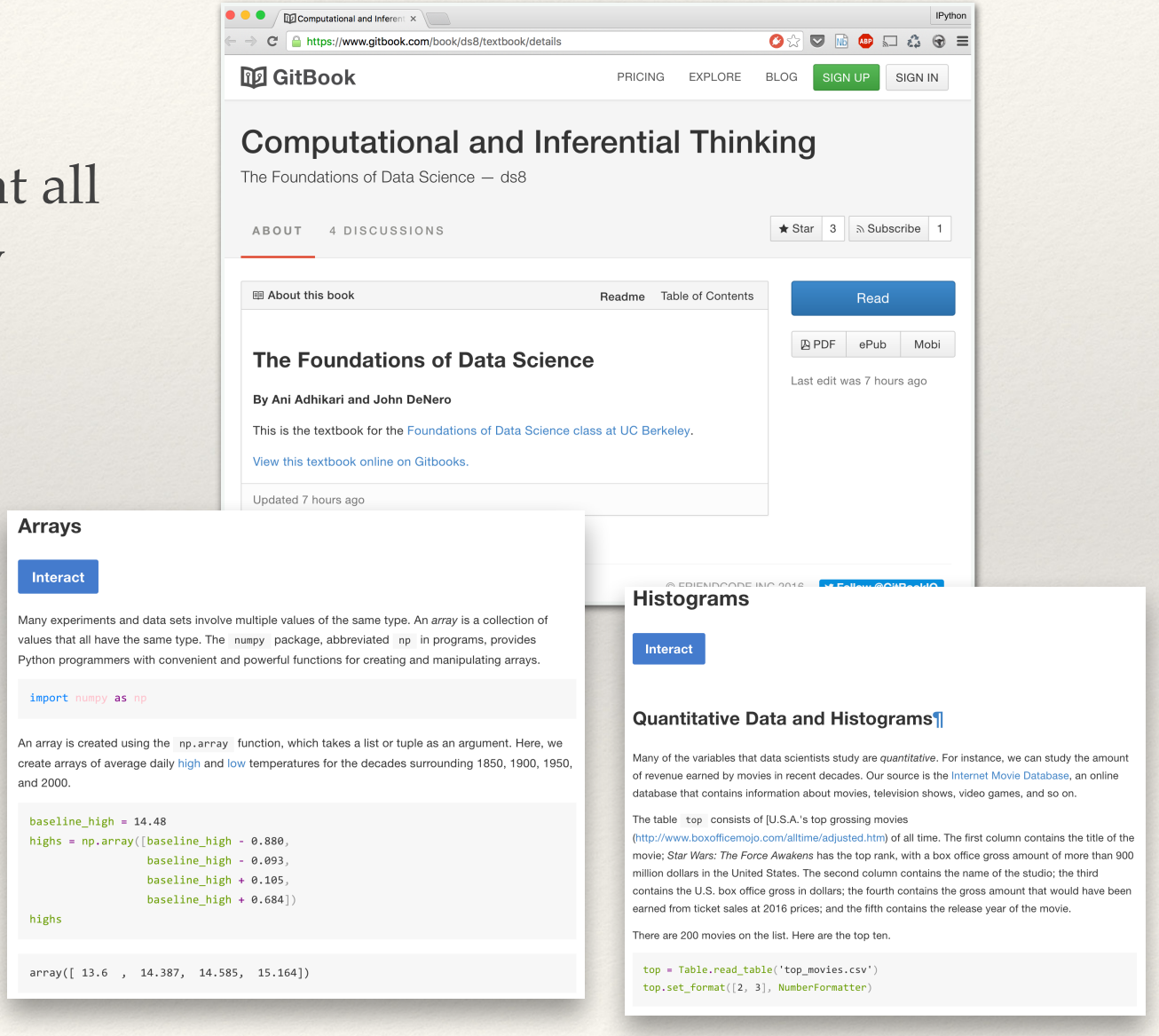
Code meets data

Deploy the Notebook next to your data to provide unified software management and data access within your organization.

Berkeley's *Foundations of Data Science*

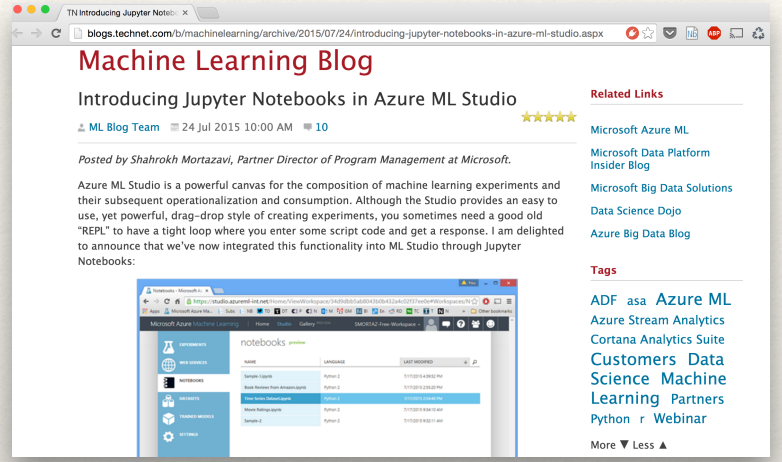
- New curriculum aimed at all freshmen at UC Berkeley
- Interactive textbook is Jupyter Notebooks
- Course deployment is JupyterHub

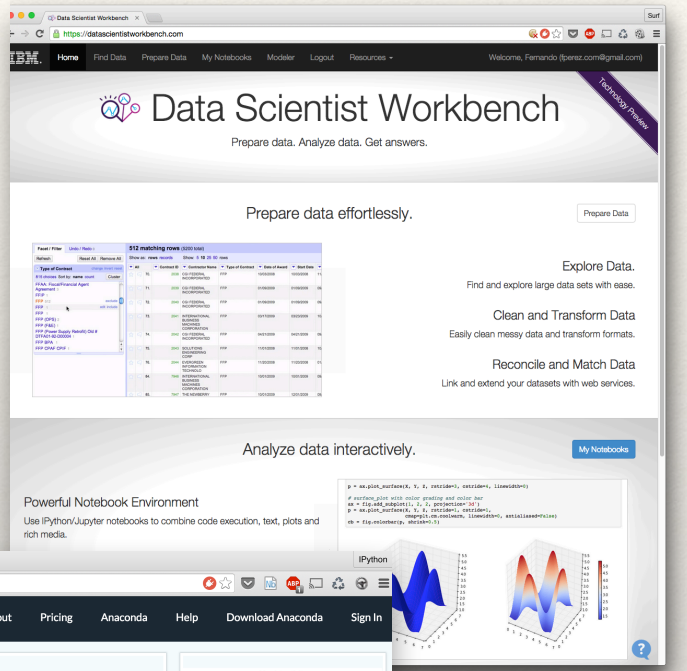
http://data8.org

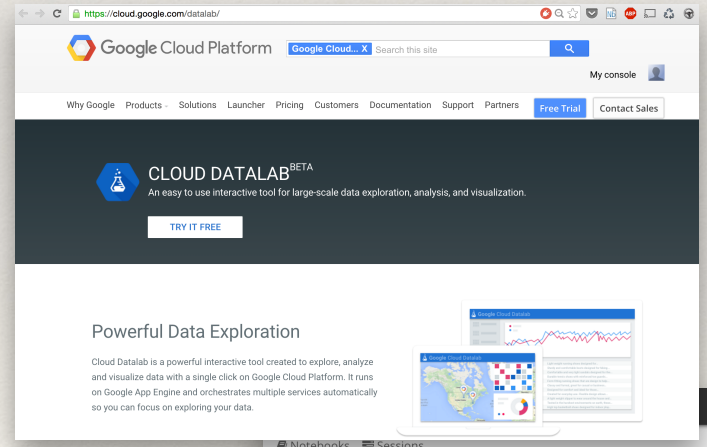


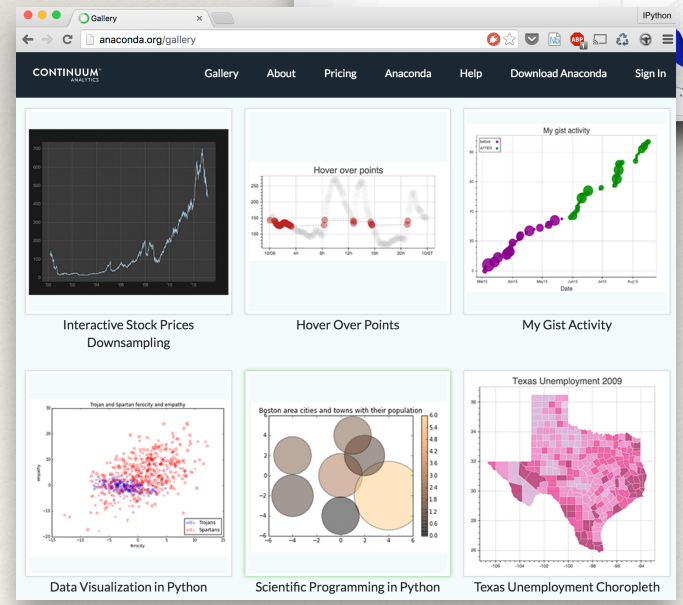
These foundations have become infrastructure

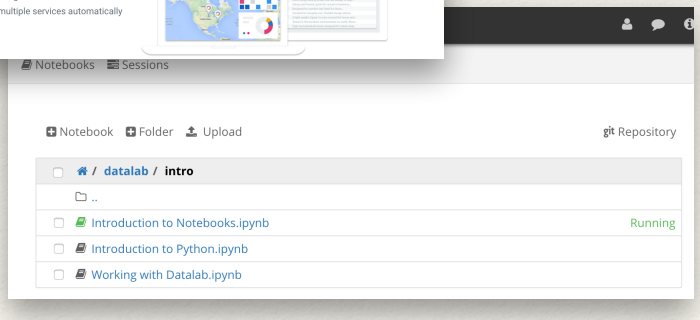
Microsoft, IBM, Google, Continuum…



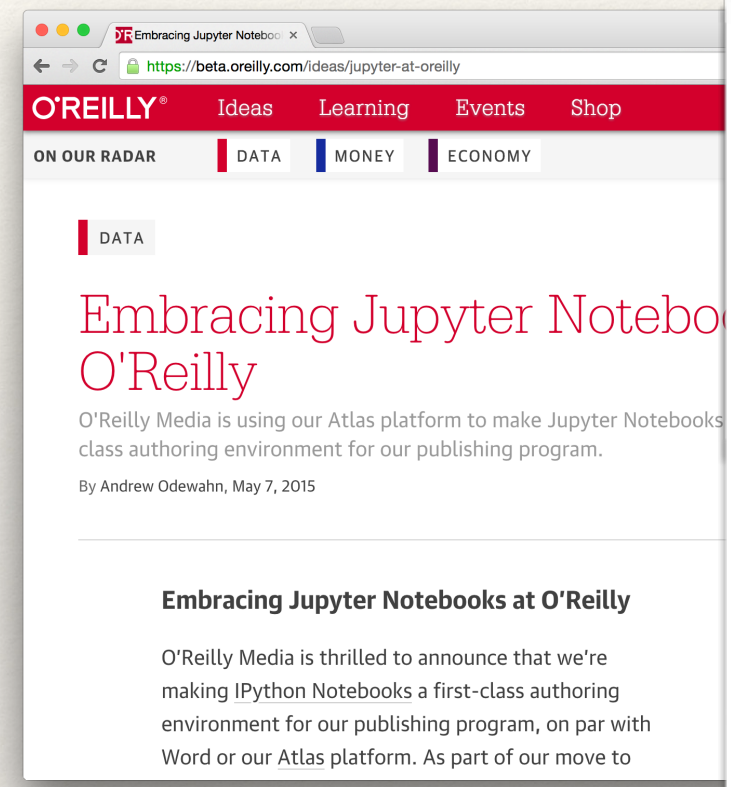


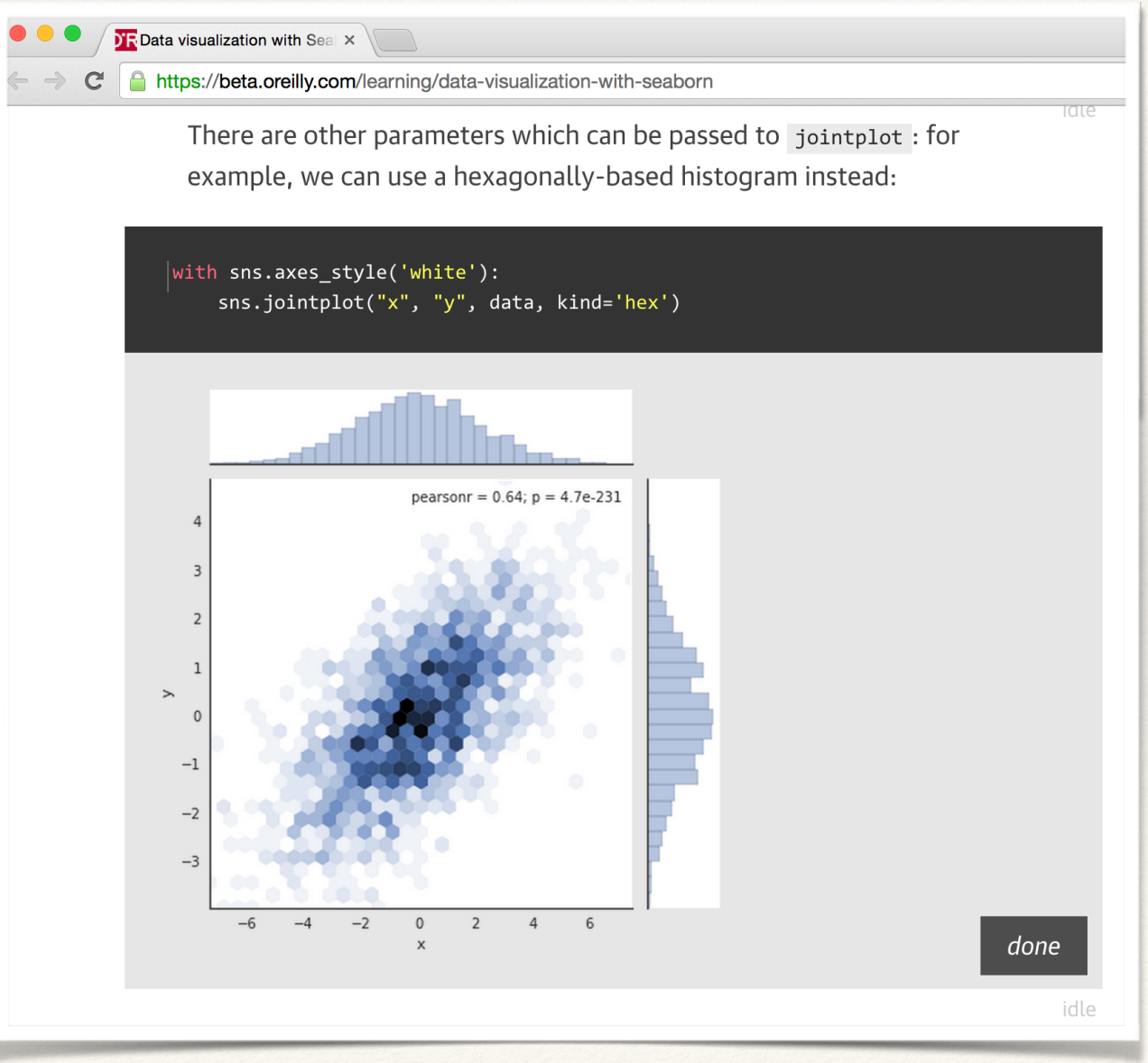




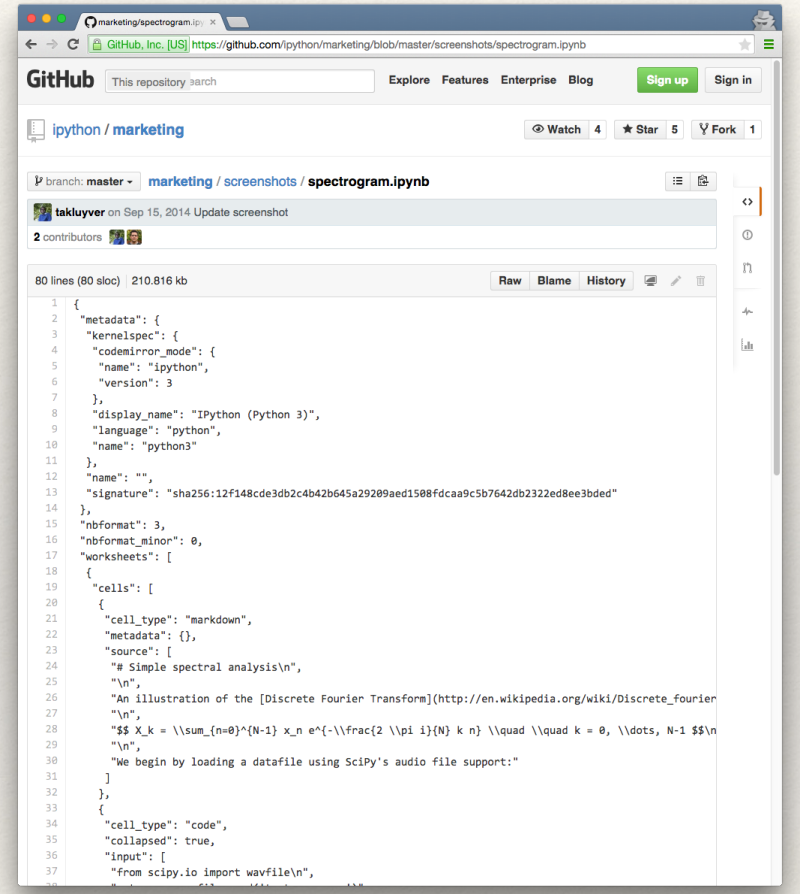


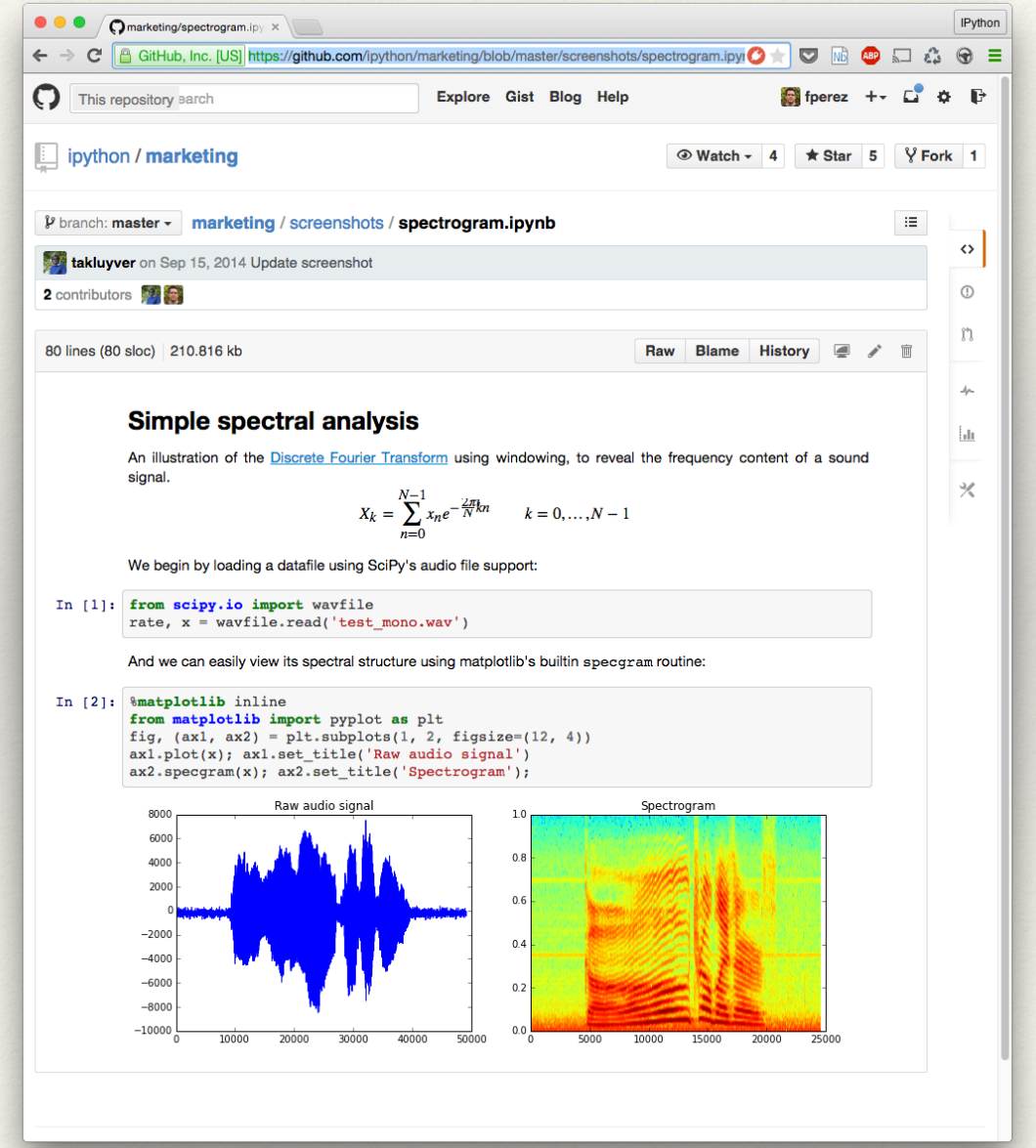
O'Reilly Thebe: kernels as a service





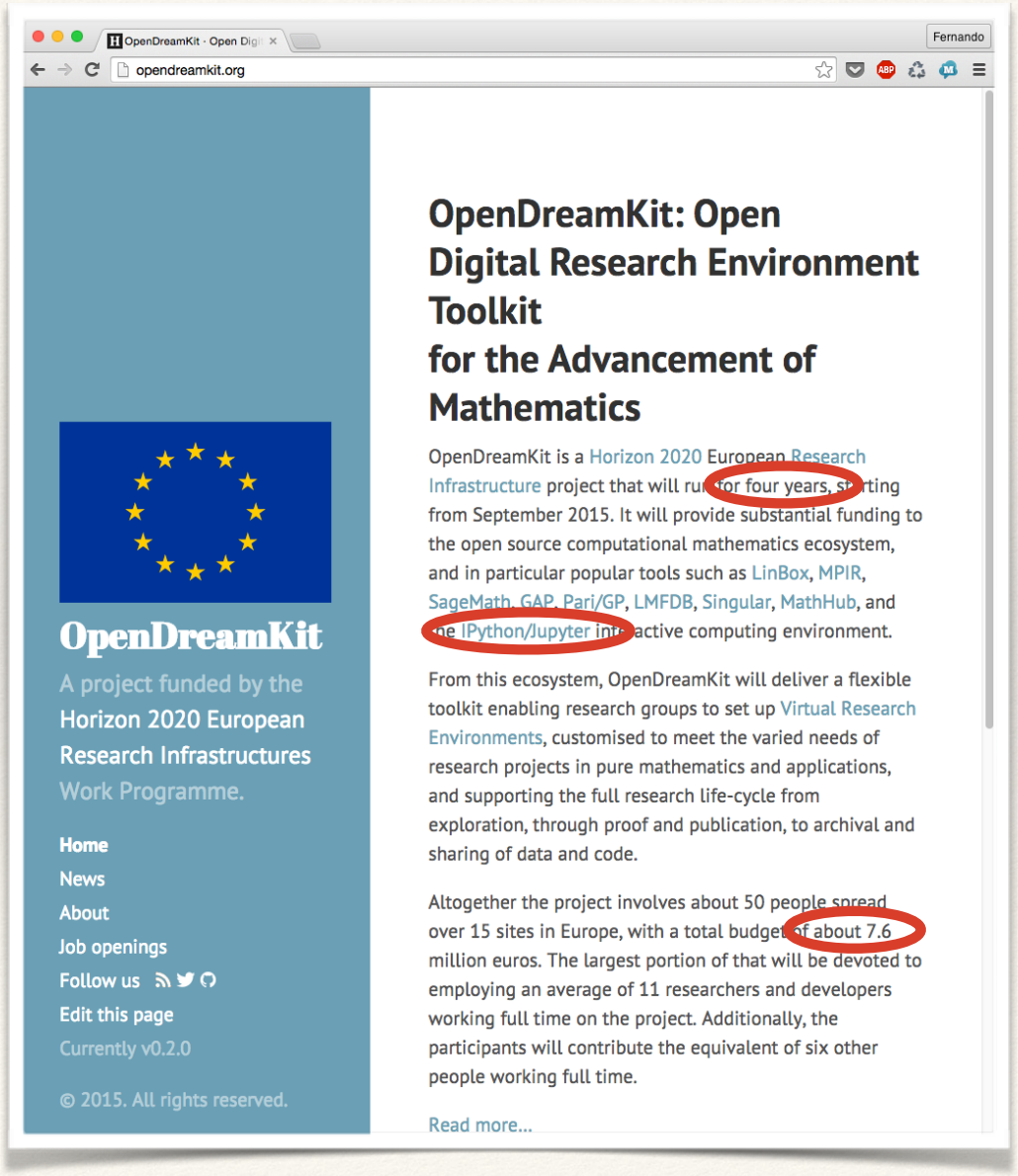
Over 500,000 notebooks on GitHub





After May 2015:

EU team for Jupyter (UK & Norway)



Thank You!

@fperez_org fperez@lbl.gov

@ProjectJupyter @IPythonDev

Try it out at

try.jupyter.org