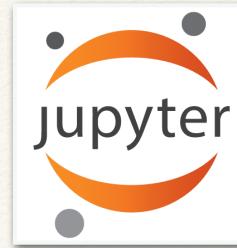


IP[y]:
IPython

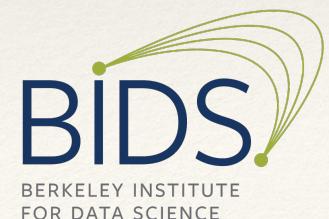


The Architecture of Jupyter

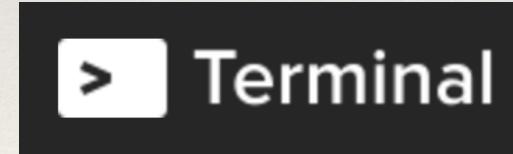
Interactive by design

Fernando Pérez
([@fperez_org](https://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

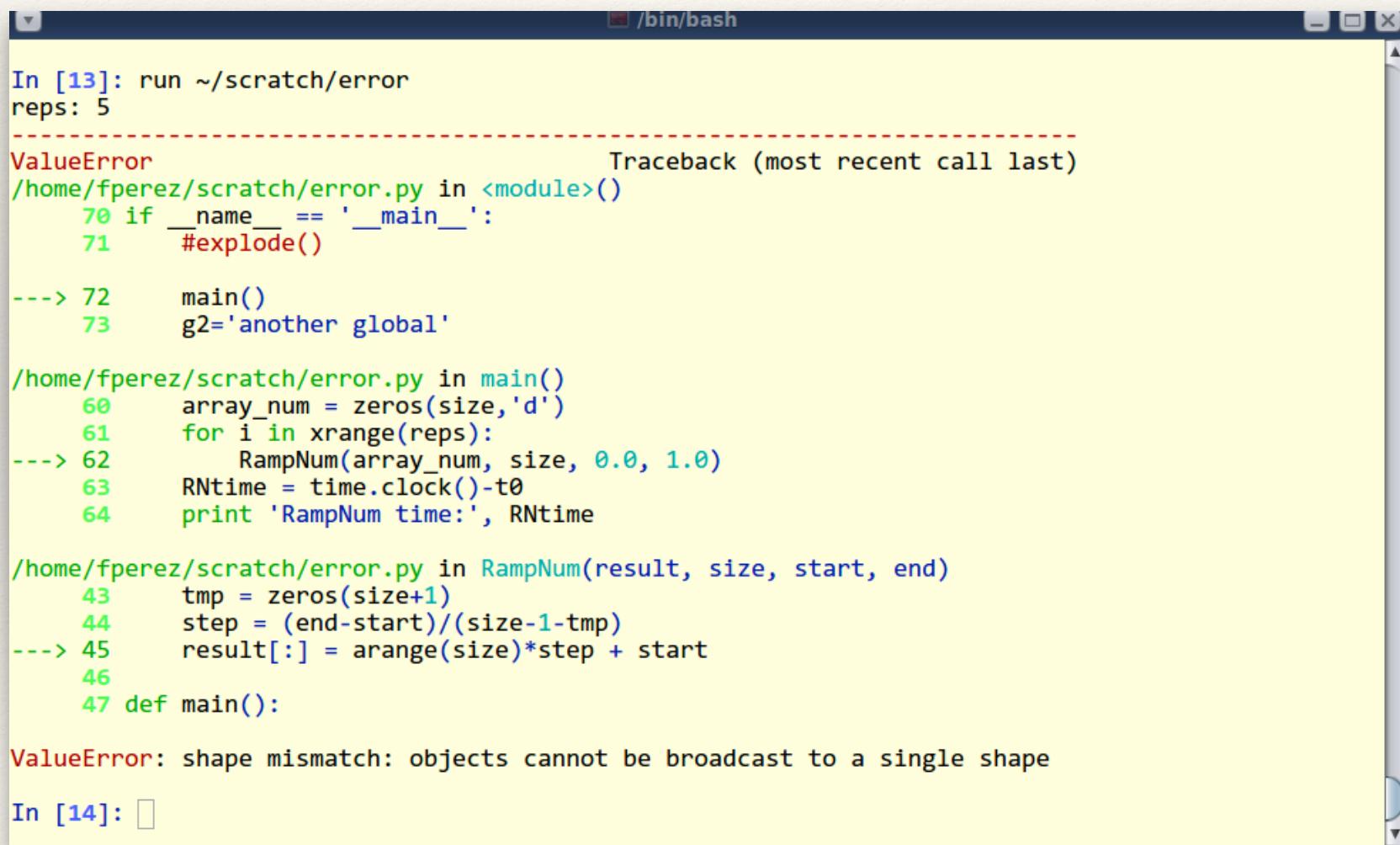
The Lifecycle of a Scientific Idea (schematically)

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

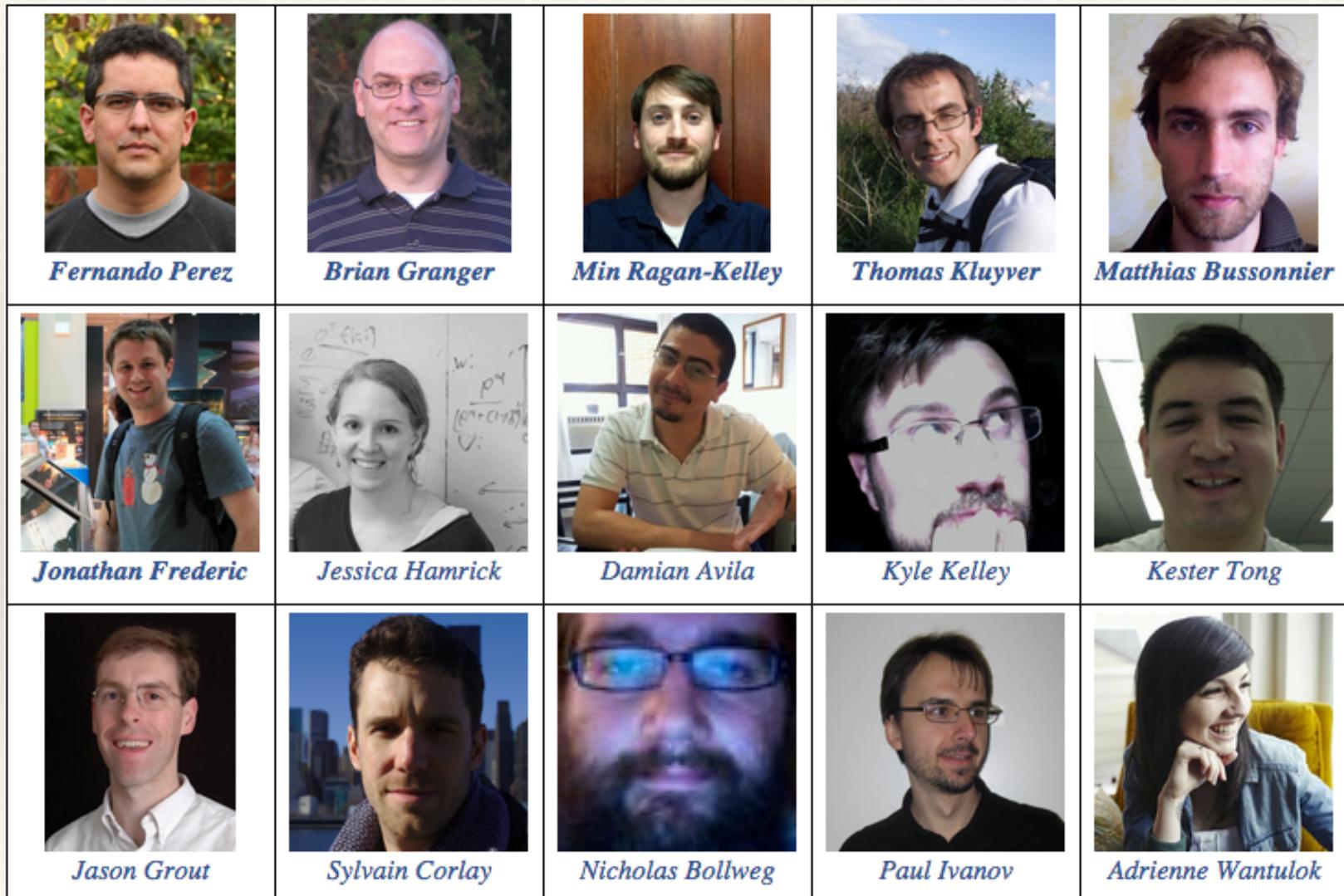


In [13]: run ~/scratch/error
reps: 5

```
-----  
ValueError                                Traceback (most recent call last)  
/home/fperez/scratch/error.py in <module>()  
    70     if __name__ == '__main__':  
    71         #explode()  
---> 72     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!

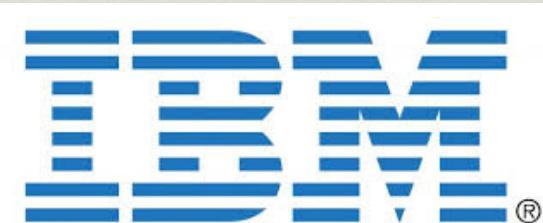
Funding and partnerships



ALFRED P. SLOAN
FOUNDATION

GORDON AND BETTY
MOORE
FOUNDATION

CONTINUUM
ANALYTICS



ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

Google

THE LEONA M. AND HARRY B.
HELMESLEY
CHARITABLE TRUST

SIMONS FOUNDATION

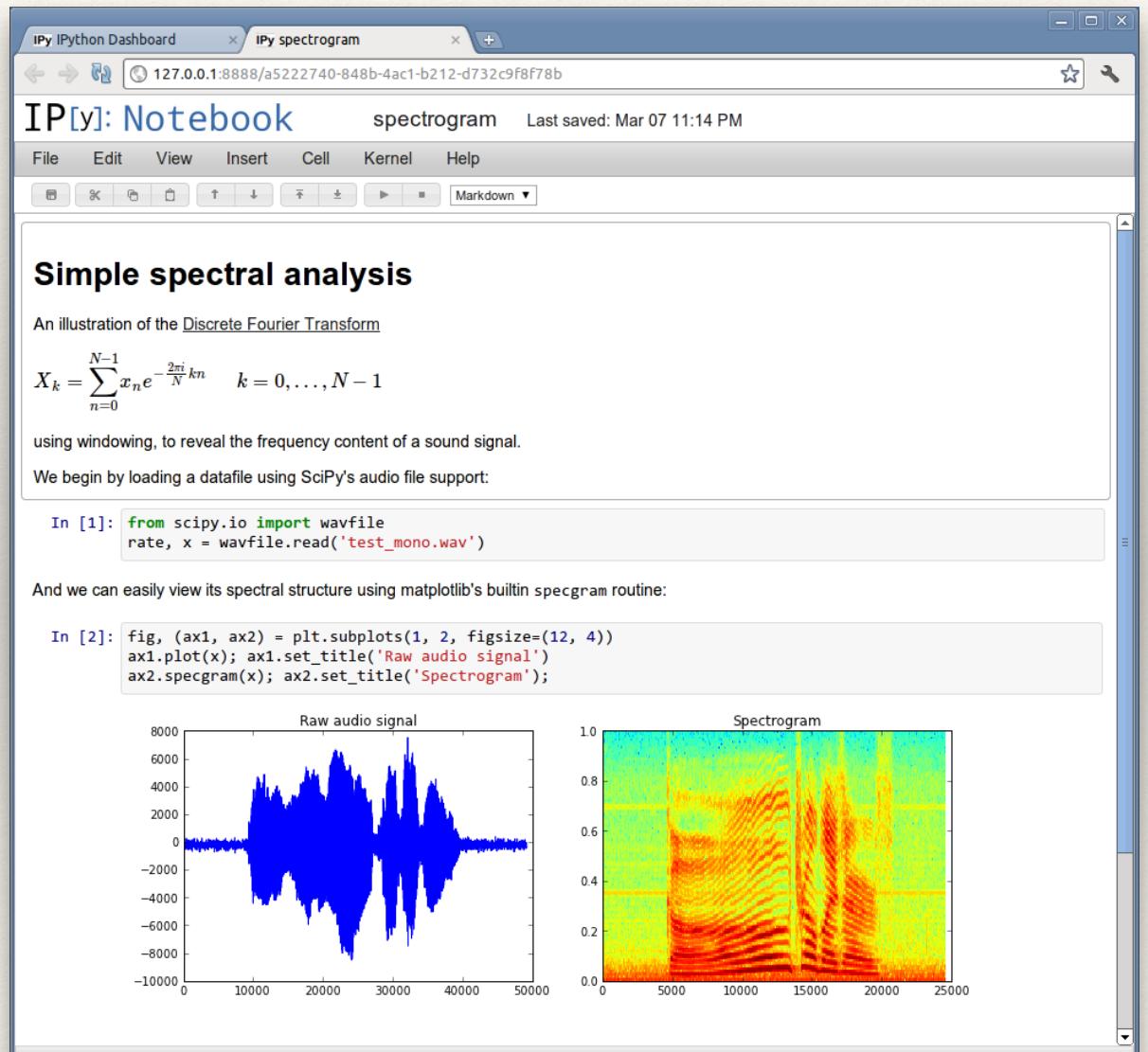
POWERED BY
rackspace®
the open cloud company

 Microsoft

Bloomberg

2011: The IPython Notebook

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

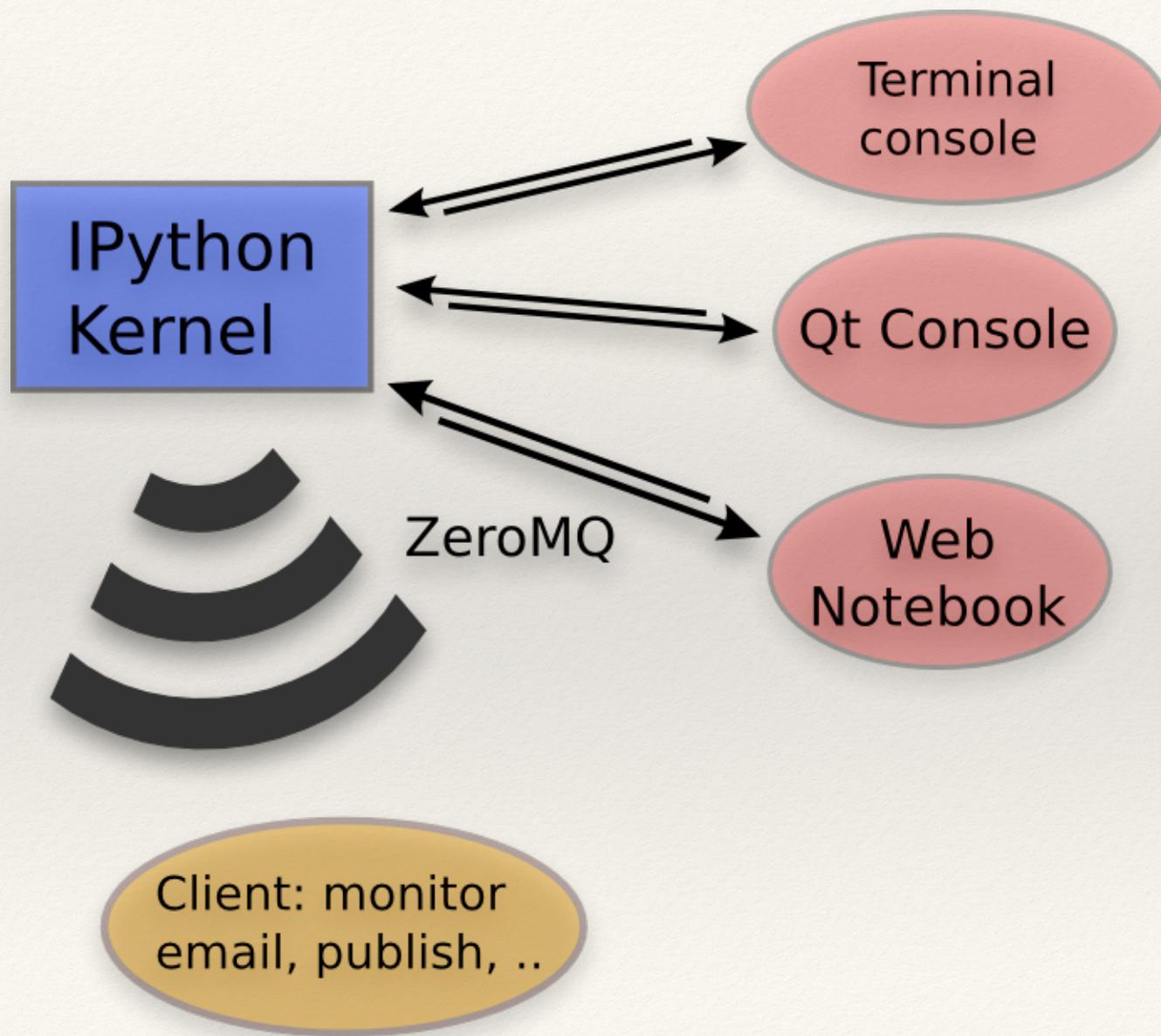


From IPython to Project Jupyter

IP[y]:
IPython



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



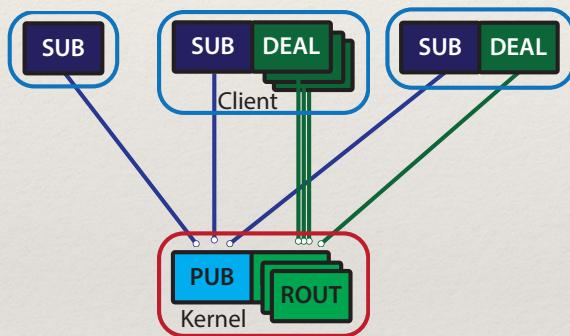
Language Agnostic

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

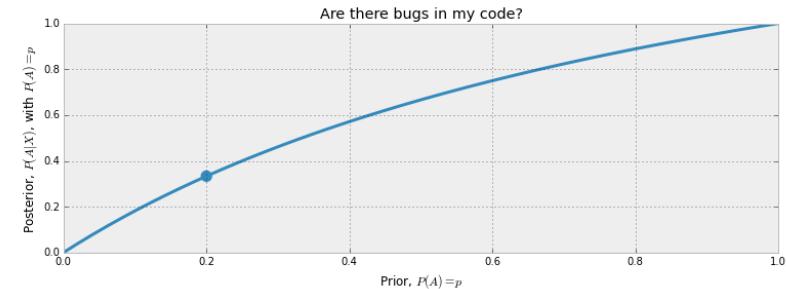
We have already computed $P(X|A)$ above. On the other hand, $P(X| \sim A)$ is subjective: our code can pass tests but still have a bug in it, though the probability there is a bug present is reduced. Note this is dependent on the number of tests performed, the degree of complication in the tests, etc.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

$$\begin{aligned} P(A|X) &= \frac{1 \cdot p}{1 \cdot p + 0.5(1 - p)} \\ &= \frac{2p}{1 + p} \end{aligned}$$

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>



Jupyter Protocol

supercharge the P in REP*L

any mime-type output

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

The screenshot shows a Jupyter Notebook interface with three visible code cells:

- In [5]:**

```
print(df.head())
```

Output:

	cake	lies	pie
2012-12-19	363.885981	367.826809	362.807807
2012-12-20	361.055153	368.463441	365.065045
2012-12-21	362.064454	367.768454	364.087118
- In [14]:**

```
Math(r'''f(x) = \int_{-\infty}^{\infty}\hat{f}(\xi) e^{2\pi i \xi x}, d\xi
```
- In []:**

```
@interact
def factor_xn(n=5):
    display(Eq(x**n-1, factor(x**n-1)))
```

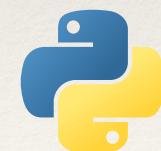
At the bottom of the interface, there is a timeline showing months from Jan 2013 to Oct 2014.

Jupyter Protocol is language agnostic

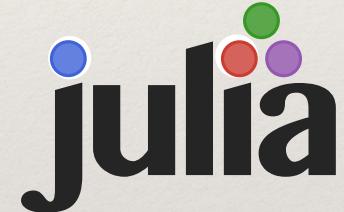
 Scala



 Spark

 python™



 julia

 ERLANG





Jupyter Notebooks

- Notebook (JS on) sequence of cells
- Publicly documented
- ~~text cell~~ = markdown + latex)
- Machine readable,
- ~~easy to understand~~ input and output
- Transformable
- ~~metadata are~~ everywhere

The screenshot shows a Jupyter Notebook interface with the title "Sampling_Theorem" in the header. The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The IPython (Python 3) logo is in the top right, along with Logout. The main content area contains a section titled "Investigating the Sampling Theorem". It includes a text block about the theorem, mathematical formulas for coefficients a_n and reconstruction $x(t)$, and a note about generating discrete points a_n by integrating over the entire function $x(t)$.

Investigating the Sampling Theorem

In this section, we investigate the implications of the sampling theorem. Here is the usual statement of the theorem from wikipedia:

"If a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart."

Since a function $x(t)$ is a function from the real line to the real line, there are uncountably many points between any two ordinates, so sampling is a massive reduction of data since it only takes a tiny number of points to completely characterize the function. This is a powerful idea worth exploring. In fact, we have seen this idea of reducing a function to a discrete set of numbers before in Fourier series expansions where (for periodic $x(t)$)

$$a_n = \frac{1}{T} \int_0^T x(t) \exp(-j\omega_n t) dt$$

with corresponding reconstruction as:

$$x(t) = \sum_k a_n \exp(j\omega_n t)$$

But here we are generating discrete points a_n by integrating over the **entire** function $x(t)$, not just evaluating it at a single point. This means we are collecting information about the entire function to compute a single discrete point a_n , whereas with sampling we are just taking individual points in isolation.

The Jupyter Notebook Ecosystem

nbviewer: seamless notebook sharing

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

nbviewer FAQ IPython Jupyter

nbviewer

A simple way to share Jupyter Notebooks

URL | GitHub username | GitHub username/repo | Gist ID

Programming Languages

IPython IRuby IJulia

An IJulia Preview

Basic Julia interaction

Python for Signal Processing Books Probabilistic Programming

Data Visualization with Lightning Misc Interactive plots with Plotly

Mining the Social Web 2nd Edition

DATA MINING FACEBOOK, TWITTER, LINKEDIN, GOOGLE+, GITHUB, AND MORE

PROBABILISTIC PROGRAMMING & BAYESIAN METHODS FOR HACKERS

Lightning Data Visualization

Bokeh

Plotly

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

The screenshot shows a web browser displaying a journal article from The ISME Journal. The article title is "Collaborative cloud-enabled tools allow rapid, reproducible biological insights". The right side of the screen shows a Jupyter Notebook cell containing Python code for primer analysis.

Journal Article Content:

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 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': 'AGAGTTTGATCTGGCTTCAG',
    '338r': 'DNA.rc("GCTGCCCTCCCGTAGGAGT"),
    '349f': 'GYCACASCAGKGCGGAAN',
    '534r': 'DNA.rc("ATTACCCGGGCTGTCTGG"),
    '515r': 'DNA.rc("GTCCGGGCGGCGGCGG"),
    '806r': 'DNA.rc("GGACTTAWSGGGGATCTAAAT"),
    '967f': 'GGGGAAAGACCTTAC',
    '1046r': 'GGRGCGGCGACGACGCW',
    '1391f': 'TGYACACACCCCGCCOT',
    '1492r': 'DNA.rc("GGCTACCTCTGTACGACT"),
    '1391r': 'TGYACACACCCGGCGTC' # Need this rather than forward primer to get proper
r 3' position of reverse version
}

reference_aligned_file = '/home/ubuntu/qiime_software/gg_otsu-4feb2011-release/rep_set/gg_
76_otsu_4feb2011_aligned.fasta'
```

Right Panel (Jupyter Notebook):

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

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-9fe9c1f6 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 than 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/configfile`:

```
[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-9fe9c1f6
cluster_size = 1
keyname = YOUR-KEY
cluster_size = 4
node_instance_type = m2.4xlarge
plugins = ipcluster
volumes = glime-ipython-data

(volume glime-ipython-data)
VOLUME_ID = /dev/vdb
MOUNT_PATH = /home/ubuntu/data
```

Today: mybinder.org



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



[github.com / freeman-lab](https://github.com/freeman-lab)

1

Tell us your GitHub repo

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

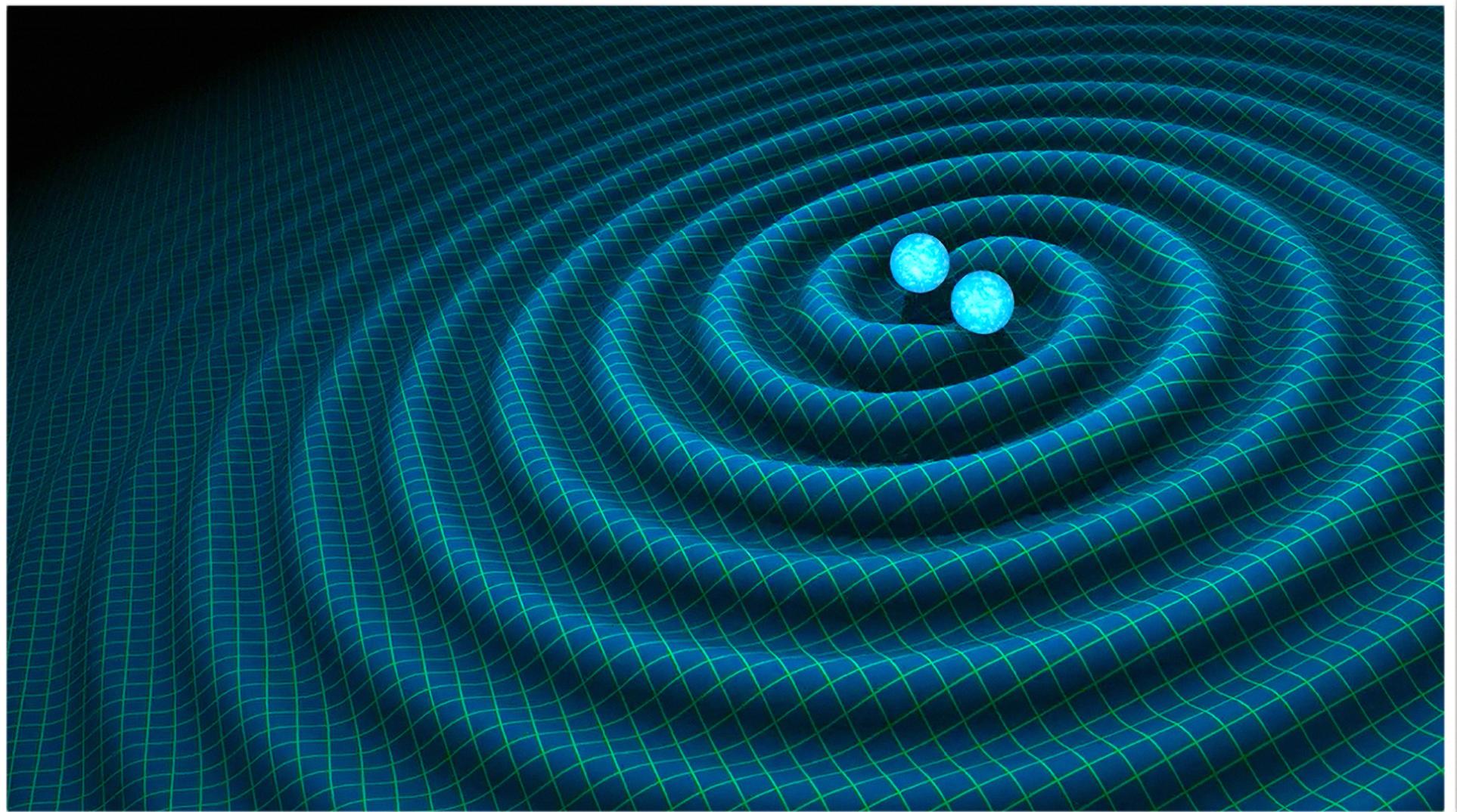
Attach services

Postgres
 Spark

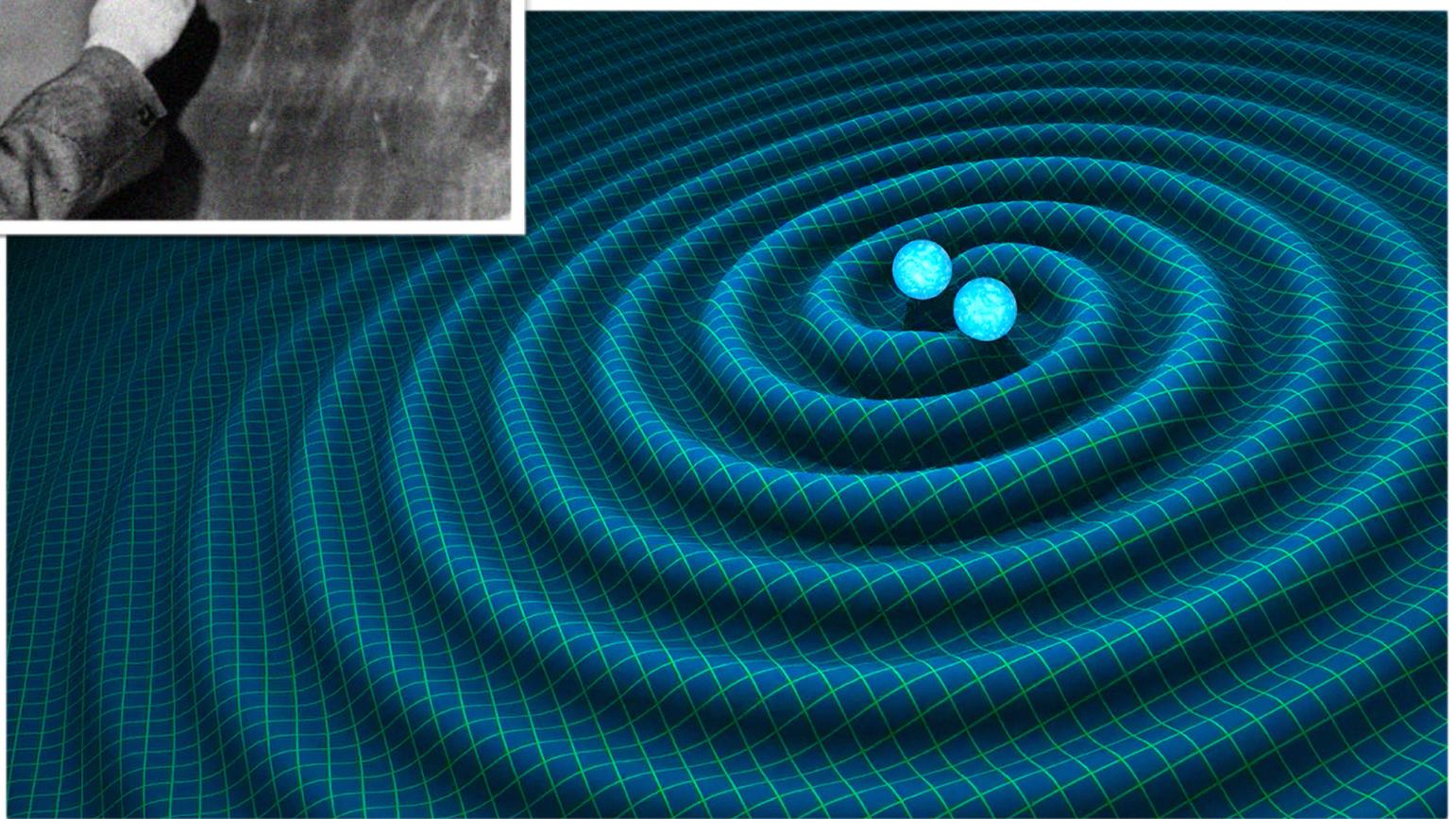
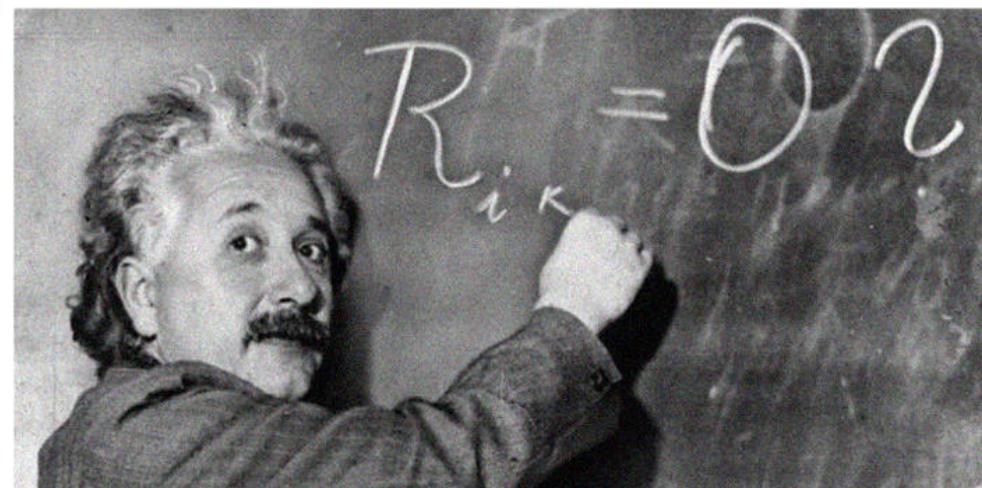
make my binder

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 ~ $1\text{e}-21$
- Milky Way: $1\text{e}+21\text{m}$ across!

September 14, 2015

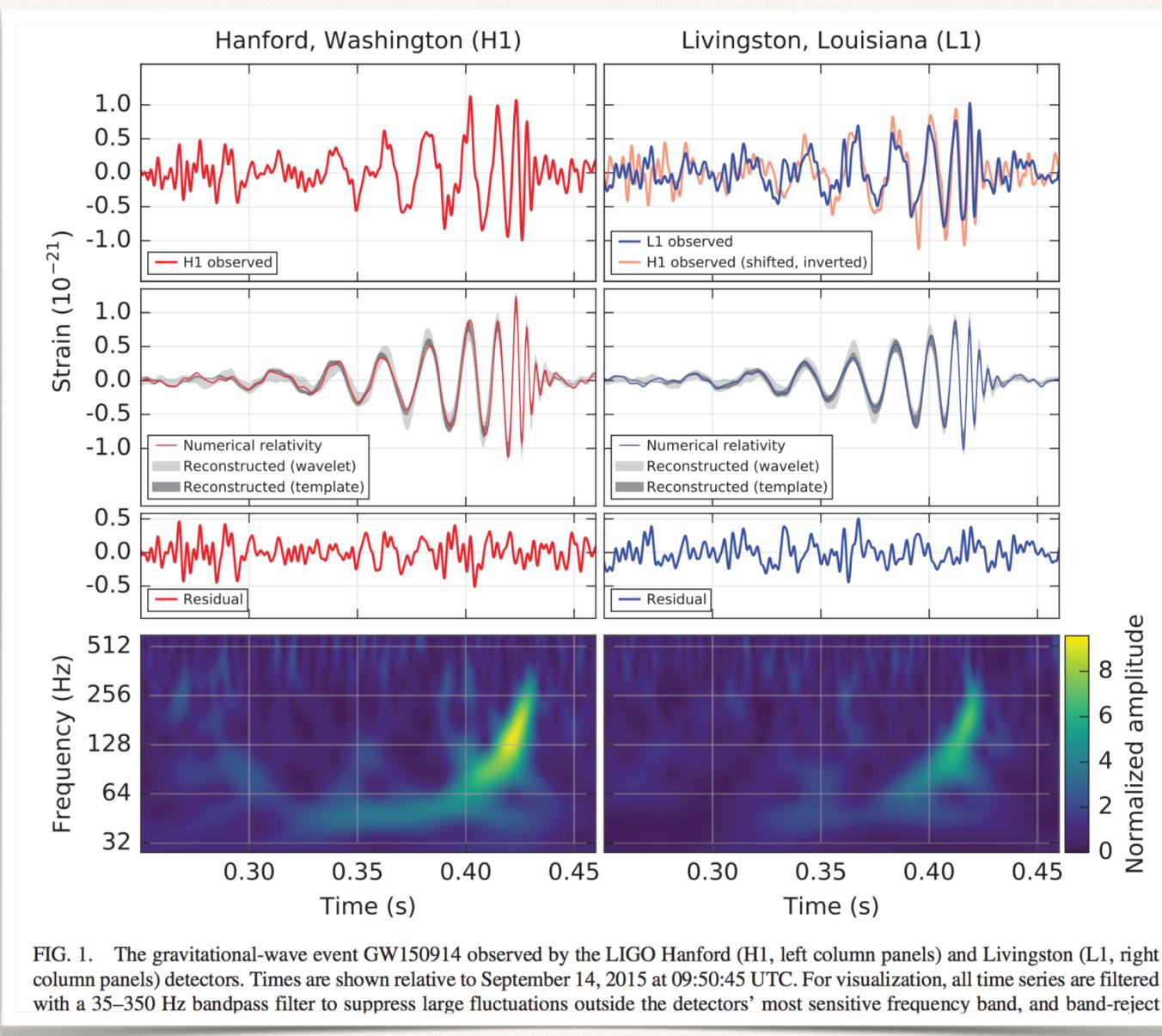
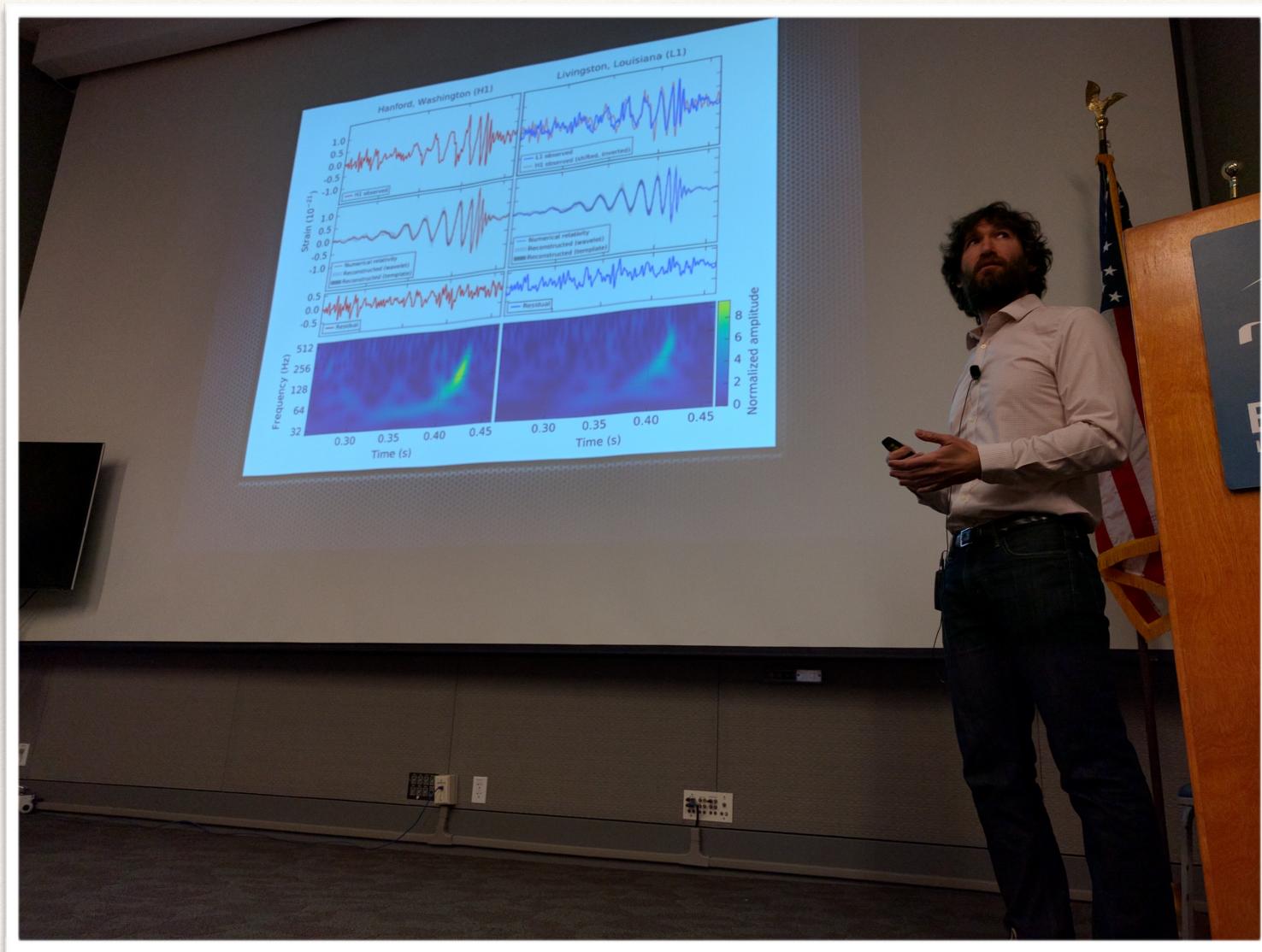
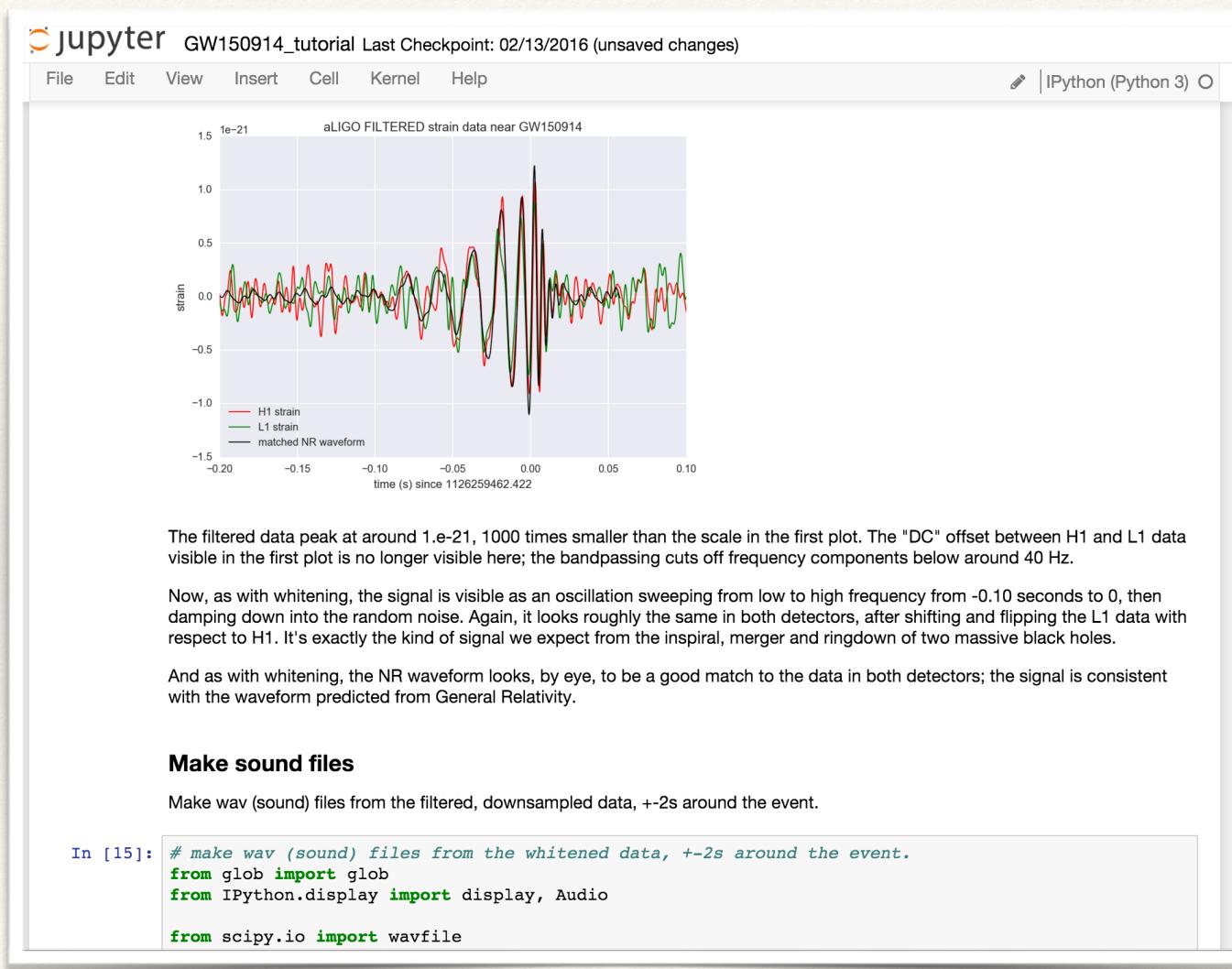


FIG. 1. The gravitational-wave event GW150914 observed by the LIGO Hanford (H1, left column panels) and Livingston (L1, right column panels) detectors. Times are shown relative to September 14, 2015 at 09:50:45 UTC. For visualization, all time series are filtered with a 35–350 Hz bandpass filter to suppress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

February 19, 2016: Dan Holz @ LBL

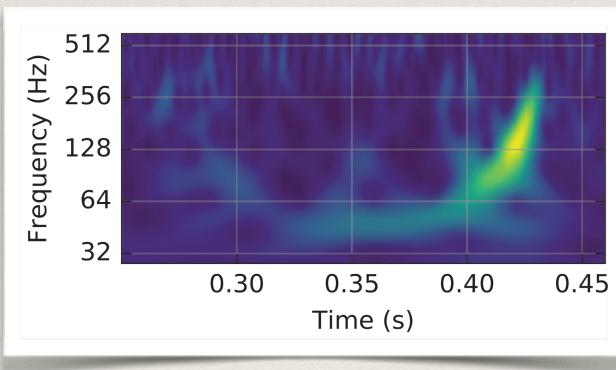
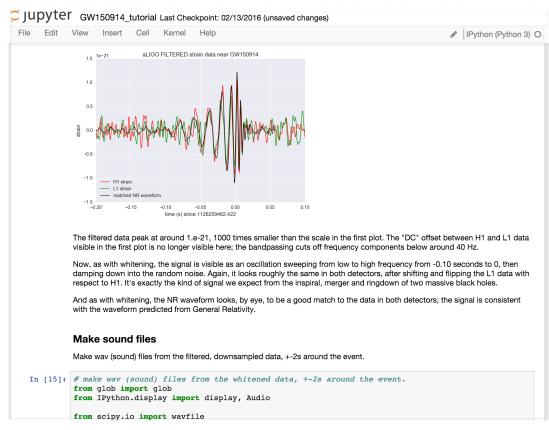


Gravitational waves detected on Jupyter!



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

nature
International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Authors

Archive > Volume 515 > Issue 7525 > Toolbox > Article

NATURE | TOOLBOX

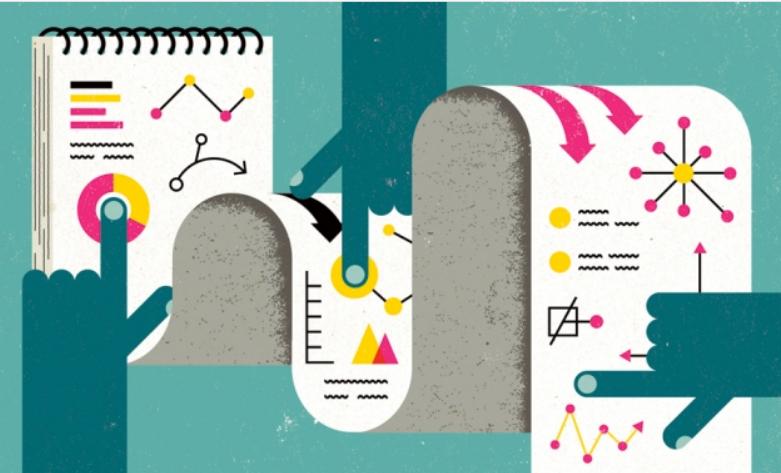
Interactive notebooks: Sharing the code

The free IPython notebook makes data analysis easier to record, understand and reproduce.

Helen Shen

05 November 2014

PDF Rights & Permissions



Illustrations by The Project Twins

Search Go Advanced search

E-alert RSS Facebook Twitter

Top story



USA 25 Brontosaurus

Beloved Brontosaurus makes a comeback

Jurassic giant's taxonomic status is restored.

Recent Read Comments Emailed

1. History: Women at the edge of science
Nature | 08 April 2015
2. Scientific instrumentation: The aided eye
Nature | 08 April 2015
3. Books in brief
Nature | 08 April 2015
4. Antibody shows promise as

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

Executable papers: the future?

nature.com · Sitemap

Login | Register | Close

IP[y]: Notebook Nature (autosaved) | IPython (Python 3)

File Edit View Insert Cell Kernel Help

Cell Toolbar: None

nature | rackspace

Introduction

Welcome! You have just launched a live example of an IPython Notebook. The notebook is an open-source, interactive computing environment that lets you combine live code, narrative text, mathematics, plots and rich media in one document. Notebook documents provide a complete reproducible record of a computation and its results and can be shared with colleagues (through, for example, email, web-hosting services such as GitHub, Dropbox, and [nbviewer](#)).

You can edit anything in this temporary demonstration notebook, including the text you are reading. To see it full-screen, click on the 'Expand' icon in the lower right corner of the frame around this notebook.

This notebook showcases some of IPython's capabilities for researchers.

This demonstration is hosted by [Rackspace](#) and is running on its bare metal offering, [OnMetal](#). Try out these cloud services yourself through [Rackspace's developer+ page](#).

Basic Python code and plotting

The box below (known as a code cell) contains the Python code to plot $y = x^2$ over the range $[0, 5]$. The blue comments preceded by # explain what the code does.

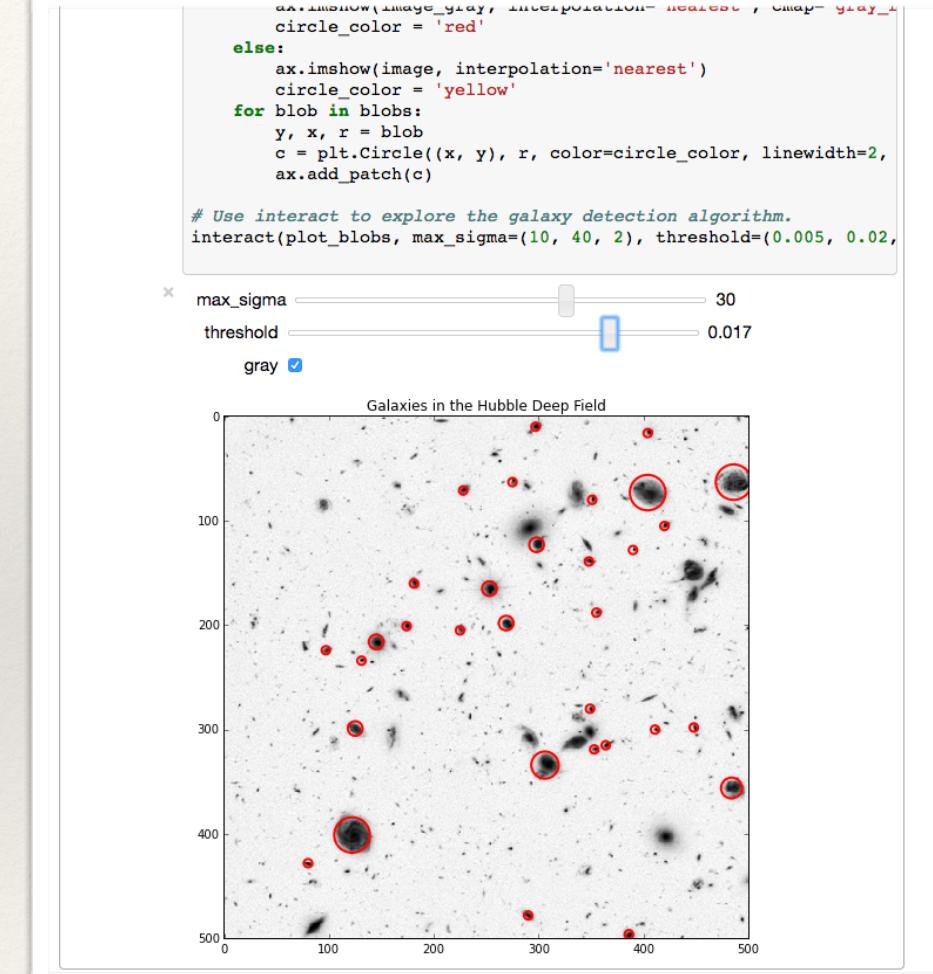
To run the code:

1. Click on the cell to select it.
2. Press SHIFT+ENTER on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available [here](#).

```
In [ ]: # Import matplotlib (plotting) and numpy (numerical arrays).
# This enables their use in the Notebook.
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Create an array of 30 values for x equally spaced from 0 to 5.
x = np.linspace(0, 5, 30)
```

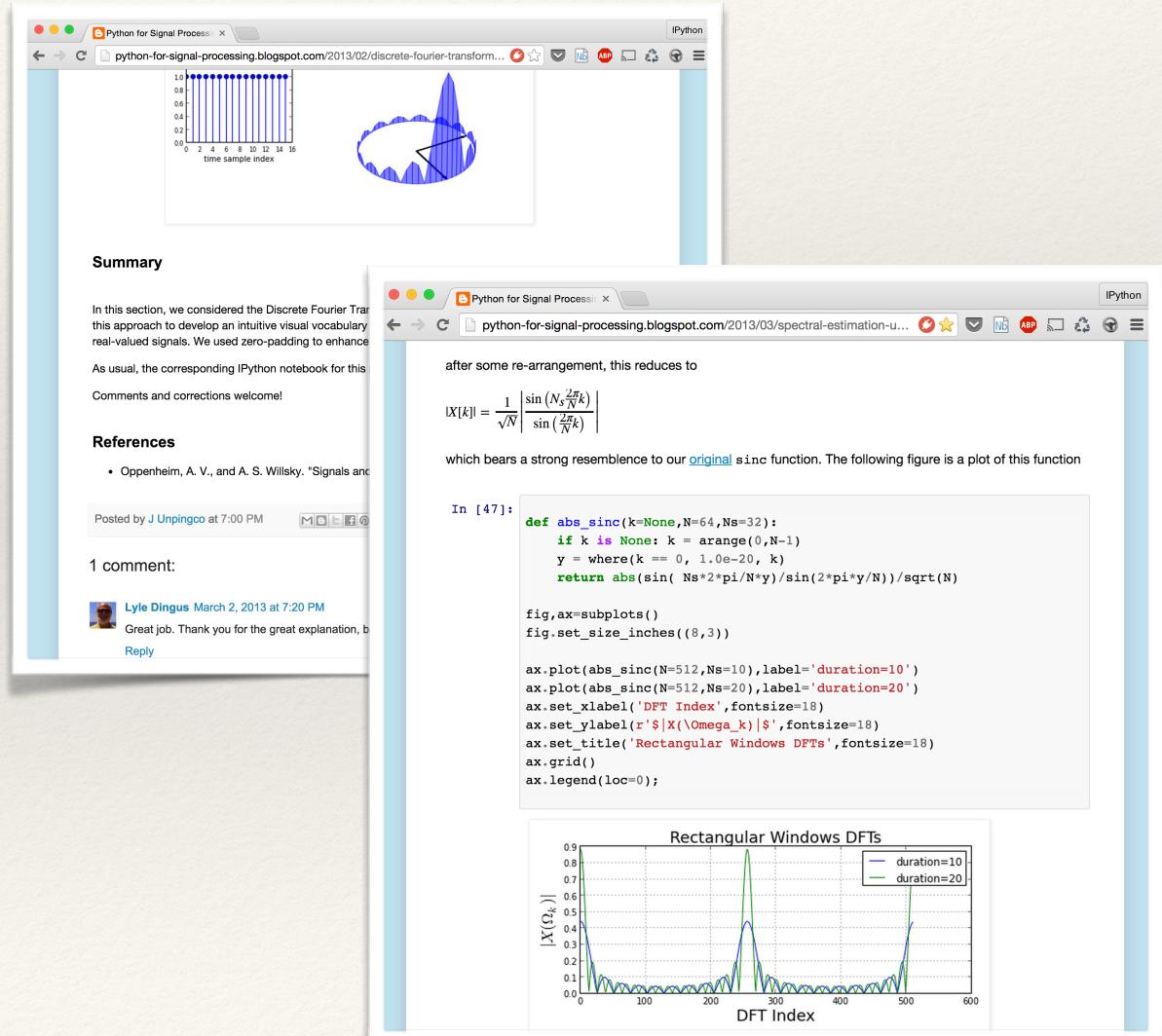
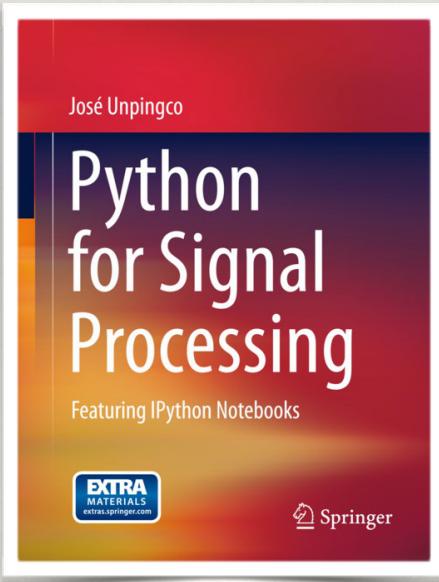


<http://www.nature.com/news/ipython-interactive-demo-7.21492?article=1.16261>

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.



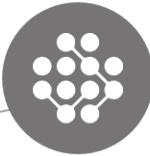
Centralized deployment

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

The screenshot shows the Berkeley Foundations of Data Science course website. At the top, there is a navigation bar with links for PRICING, EXPLORE, BLOG, SIGN UP, and SIGN IN. Below the navigation, the title "Computational and Inferential Thinking" is displayed, along with the subtitle "The Foundations of Data Science — ds8". There are buttons for "Star" (3), "Subscribe" (1), and "Read". Below this, there is a section titled "About this book" with links to "Readme" and "Table of Contents". A large blue "Read" button is prominent. Below the book summary, there are two Jupyter Notebook interactives:

- Arrays**: This interactive shows Python code for creating arrays from lists and manipulating them. It includes a snippet of code:

```
import numpy as np
```

An array is created using the `np.array` function, which takes a list or tuple as an argument. Here, we create arrays of average daily `high` and `low` temperatures for the decades surrounding 1850, 1900, 1950, and 2000.

```
baseline_high = 14.48
highs = np.array([baseline_high - 0.880,
                 baseline_high - 0.093,
                 baseline_high + 0.105,
                 baseline_high + 0.684])
highs
```

```
array([ 13.6 ,  14.387,  14.585,  15.164])
```
- Histograms**: This interactive shows a histogram of movie box office gross data. It includes a snippet of code:

```
top = Table.read_table('top_movies.csv')
top.set_format([2, 3], NumberFormatter)
```

<http://data8.org>

These foundations have become
infrastructure

Microsoft, IBM, Google, Continuum...

This screenshot shows a blog post titled "Introducing Jupyter Notebooks in Azure ML Studio". The post is by Shahrokh Mortazavi, Partner Director of Program Management at Microsoft. It discusses the integration of Jupyter Notebooks into Azure ML Studio, allowing users to use a drag-drop style for creating experiments. The post includes a screenshot of the Azure ML Studio interface showing a list of notebooks.

This screenshot shows the Data Scientist Workbench interface. It features a dashboard with sections for "Prepare data effortlessly", "Explore Data", "Clean and Transform Data", and "Analyze data interactively". The "Analyze data interactively" section displays a Jupyter notebook cell with Python code and a 3D surface plot visualization.

This screenshot shows the Google Cloud Platform Cloud Databab interface. It highlights the "TRY IT FREE" button and provides a brief description of Cloud Databab as an easy-to-use interactive tool for large-scale data exploration, analysis, and visualization.

This screenshot shows the Google Cloud Platform Notebooks interface. It displays a file tree with notebooks like "Introduction to Notebooks.ipynb" and "Introduction to Python.ipynb", both marked as "Running".

This screenshot shows the Continuum Analytics Anaconda gallery. It features several data visualizations: "Interactive Stock Prices Downsampling", "Hover Over Points", "My Gist Activity", "Data Visualization in Python" (Trojan and Spartan fencity and empathy), "Scientific Programming in Python" (Boston area cities and towns with their population), and "Texas Unemployment Choropleth".

O'Reilly Thebe: kernels as a service

The screenshot shows a web browser window with the O'Reilly logo at the top. The main content is a blog post titled "Embracing Jupyter Notebooks" by O'Reilly. The post discusses how O'Reilly Media is using their Atlas platform to make Jupyter Notebooks a first-class authoring environment. It includes a snippet of Python code and a scatter plot generated using Seaborn.

Embracing Jupyter Notebooks
O'Reilly

O'Reilly Media is using our Atlas platform to make Jupyter Notebooks a first-class authoring environment for our publishing program.

By Andrew Odewahn, May 7, 2015

Embracing Jupyter Notebooks at O'Reilly

O'Reilly Media is thrilled to announce that we're making IPython Notebooks a first-class authoring environment for our publishing program, on par with Word or our Atlas platform. As part of our move to

The screenshot shows a web browser window with the O'Reilly logo at the top. The main content is a notebook titled "Data visualization with Seaborn". It displays a hexagonal jointplot with marginal histograms. The code used to generate the plot is shown in a code block:

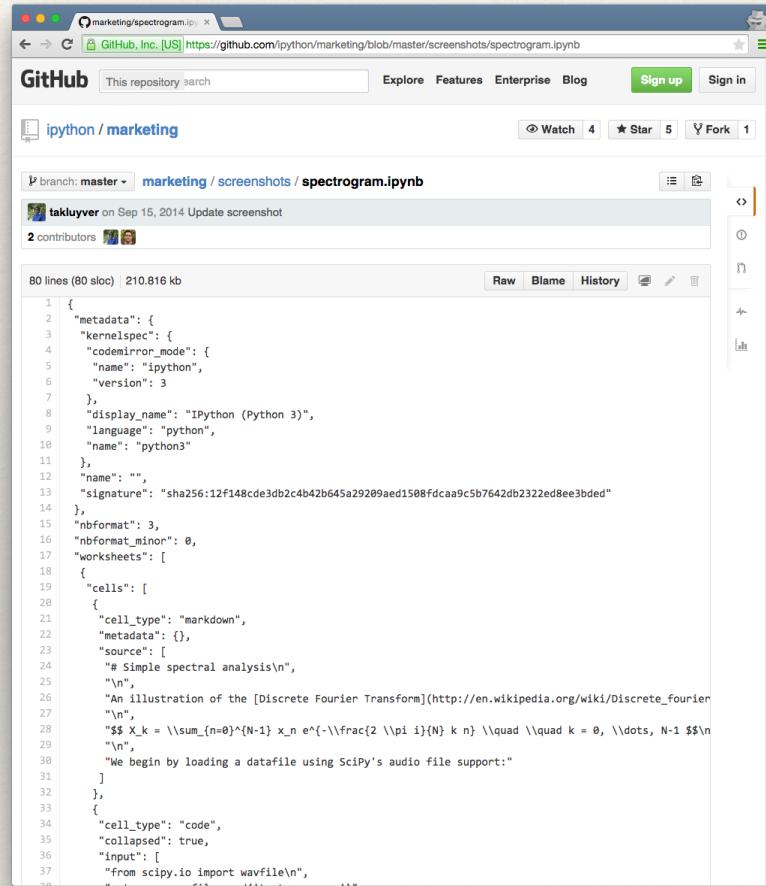
```
with sns.axes_style('white'):
    sns.jointplot("x", "y", data, kind='hex')
```

The plot shows a positive correlation between the variables x and y. The x-axis ranges from -6 to 6, and the y-axis ranges from -3 to 4. A text annotation above the plot states: "There are other parameters which can be passed to `jointplot`: for example, we can use a hexagonally-based histogram instead:"

done

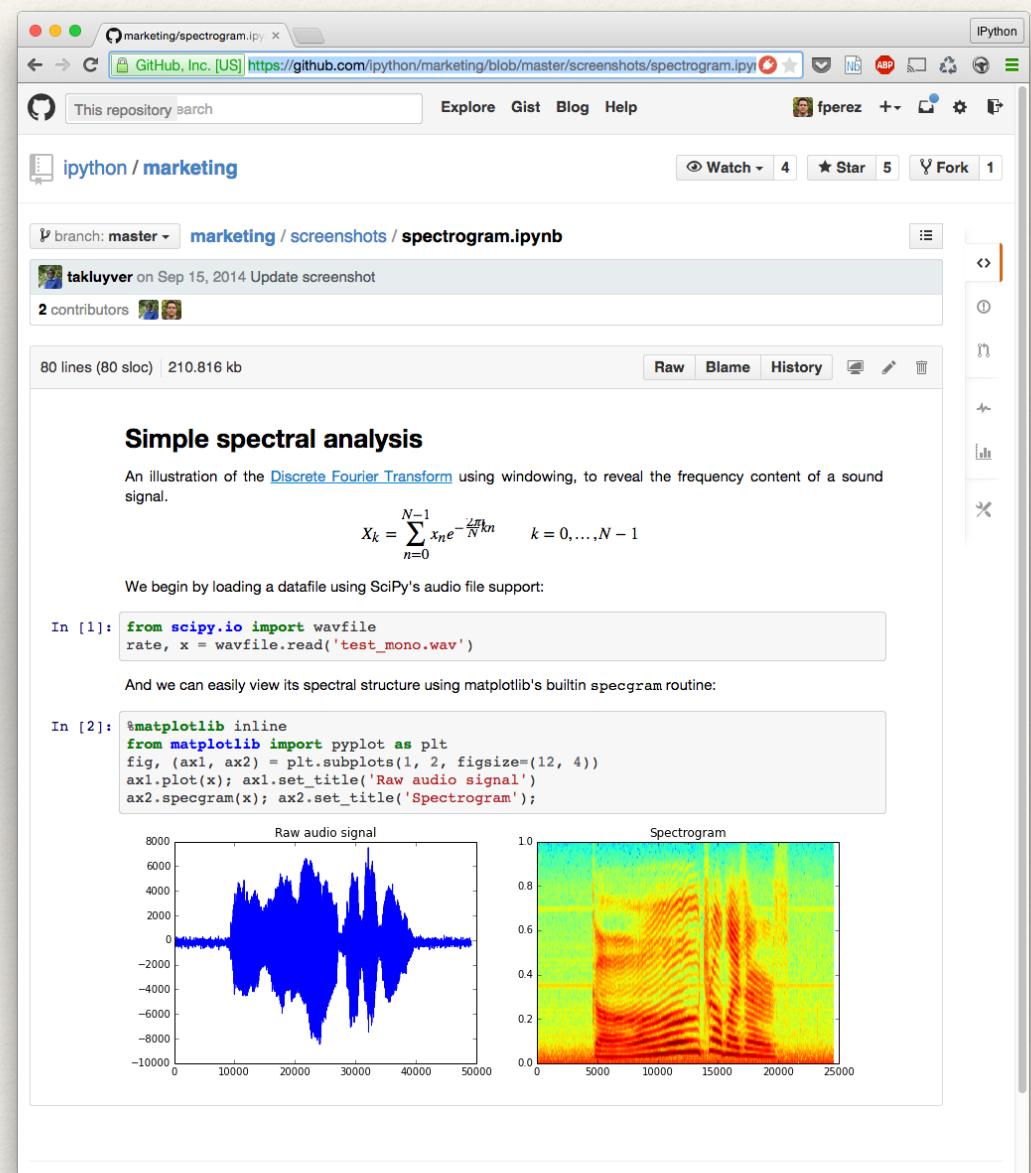
idle

Over 500,000 notebooks on GitHub



A screenshot of a GitHub notebook repository. The URL is <https://github.com/python/marketing/blob/master/screenshots/spectrogram.ipynb>. The notebook contains a single code cell with 80 lines of Python code. The code is related to spectral analysis, specifically using SciPy's audio file support and matplotlib's specgram routine. The GitHub interface shows basic navigation and repository details.

```
1 {  
2     "metadata": {  
3         "kernelspec": {  
4             "codemirror_mode": {  
5                 "name": "ipython",  
6                 "version": 3  
7             },  
8             "display_name": "IPython (Python 3)",  
9             "language": "python",  
10            "name": "python3"  
11        },  
12        "name": "",  
13        "signature": "sha256:12f148cde3db2c4b42b645a29209aed1508fdcaa9c5b7642db2322ed8ee3bded"  
14    },  
15    "nbformat": 3,  
16    "nbformat_minor": 0,  
17    "worksheets": [  
18        {  
19            "cells": [  
20                {  
21                    "cell_type": "markdown",  
22                    "metadata": {},  
23                    "source": [  
24                        "# Simple spectral analysis\n",  
25                        "\n",  
26                        "An illustration of the [Discrete Fourier Transform](http://en.wikipedia.org/wiki/Discrete_fourier\n",  
27                        "\n",  
28                        "$\$ X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1 \$\$\\n  
29                        "\n",  
30                        "We begin by loading a datafile using SciPy's audio file support."  
31                    ]  
32                },  
33                {  
34                    "cell_type": "code",  
35                    "collapsed": true,  
36                    "input": [  
37                        "from scipy.io import wavfile\\n",  
38                        "...\\n",  
39                        "...\\n",  
40                        "...\\n",  
41                        "...\\n",  
42                        "...\\n",  
43                        "...\\n",  
44                        "...\\n",  
45                        "...\\n",  
46                        "...\\n",  
47                        "...\\n",  
48                        "...\\n",  
49                        "...\\n",  
50                        "...\\n",  
51                        "...\\n",  
52                        "...\\n",  
53                        "...\\n",  
54                        "...\\n",  
55                        "...\\n",  
56                        "...\\n",  
57                        "...\\n",  
58                        "...\\n",  
59                        "...\\n",  
60                        "...\\n",  
61                        "...\\n",  
62                        "...\\n",  
63                        "...\\n",  
64                        "...\\n",  
65                        "...\\n",  
66                        "...\\n",  
67                        "...\\n",  
68                        "...\\n",  
69                        "...\\n",  
70                        "...\\n",  
71                        "...\\n",  
72                        "...\\n",  
73                        "...\\n",  
74                        "...\\n",  
75                        "...\\n",  
76                        "...\\n",  
77                        "...\\n",  
78                        "...\\n",  
79                        "...\\n",  
80                        "...\\n",  
81                    ]  
82                }  
83            ]  
84        }  
85    ]  
86}
```



A screenshot of a GitHub notebook repository. The URL is <https://github.com/python/marketing/blob/master/screenshots/spectrogram.ipynb>. The notebook contains a code cell with a mathematical formula for the Discrete Fourier Transform (DFT) and two plots: a raw audio signal waveform and a spectrogram. The GitHub interface shows basic navigation and repository details.

Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#) using windowing, to reveal the frequency content of a sound signal.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N - 1$$

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile  
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline  
from matplotlib import pyplot as plt  
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))  
ax1.plot(x); ax1.set_title('Raw audio signal')  
ax2.specgram(x); ax2.set_title('Spectrogram');
```

Raw audio signal

Spectrogram

After May 2015:

EU team for Jupyter (UK & Norway)

The screenshot shows a web browser window displaying the OpenDreamKit website. The page features a large blue header with the EU flag. Below it, the title "OpenDreamKit" is displayed in white. A sidebar on the left contains links to "Home", "News", "About", "Job openings", "Follow us", "Edit this page", and "Currently v0.2.0". The main content area has a white background. It features a large title: "OpenDreamKit: Open Digital Research Environment Toolkit for the Advancement of Mathematics". Below the title is a paragraph about the project being a Horizon 2020 European Research Infrastructure project. Two specific budget-related phrases are circled in red: "for four years, starting from September 2015" and "of about 7.6 million euros". At the bottom of the main content area is a link "Read more...".

OpenDreamKit

A project funded by the Horizon 2020 European Research Infrastructures Work Programme.

Home

News

About

Job openings

Follow us

Edit this page

Currently v0.2.0

© 2015. All rights reserved.

OpenDreamKit: Open Digital Research Environment Toolkit for the Advancement of Mathematics

OpenDreamKit is a Horizon 2020 European Research Infrastructure project that will run **for four years, starting from September 2015**. It will provide substantial funding to the open source computational mathematics ecosystem, and in particular popular tools such as LinBox, MPIR, SageMath, GAP, Pari/GP, LMFDB, Singular, MathHub, and the **IPython/Jupyter** interactive computing environment.

From this ecosystem, OpenDreamKit will deliver a flexible toolkit enabling research groups to set up **Virtual Research Environments**, customised to meet the varied needs of research projects in pure mathematics and applications, and supporting the full research life-cycle from exploration, through proof and publication, to archival and sharing of data and code.

Altogether the project involves about 50 people spread over 15 sites in Europe, with a total budget **of about 7.6 million euros**. The largest portion of that will be devoted to employing an average of 11 researchers and developers working full time on the project. Additionally, the participants will contribute the equivalent of six other people working full time.

[Read more...](#)

Thank You!

@fperez_org fperez@lbl.gov

@ProjectJupyter @IPythonDev

Try it out at
try.jupyter.org