

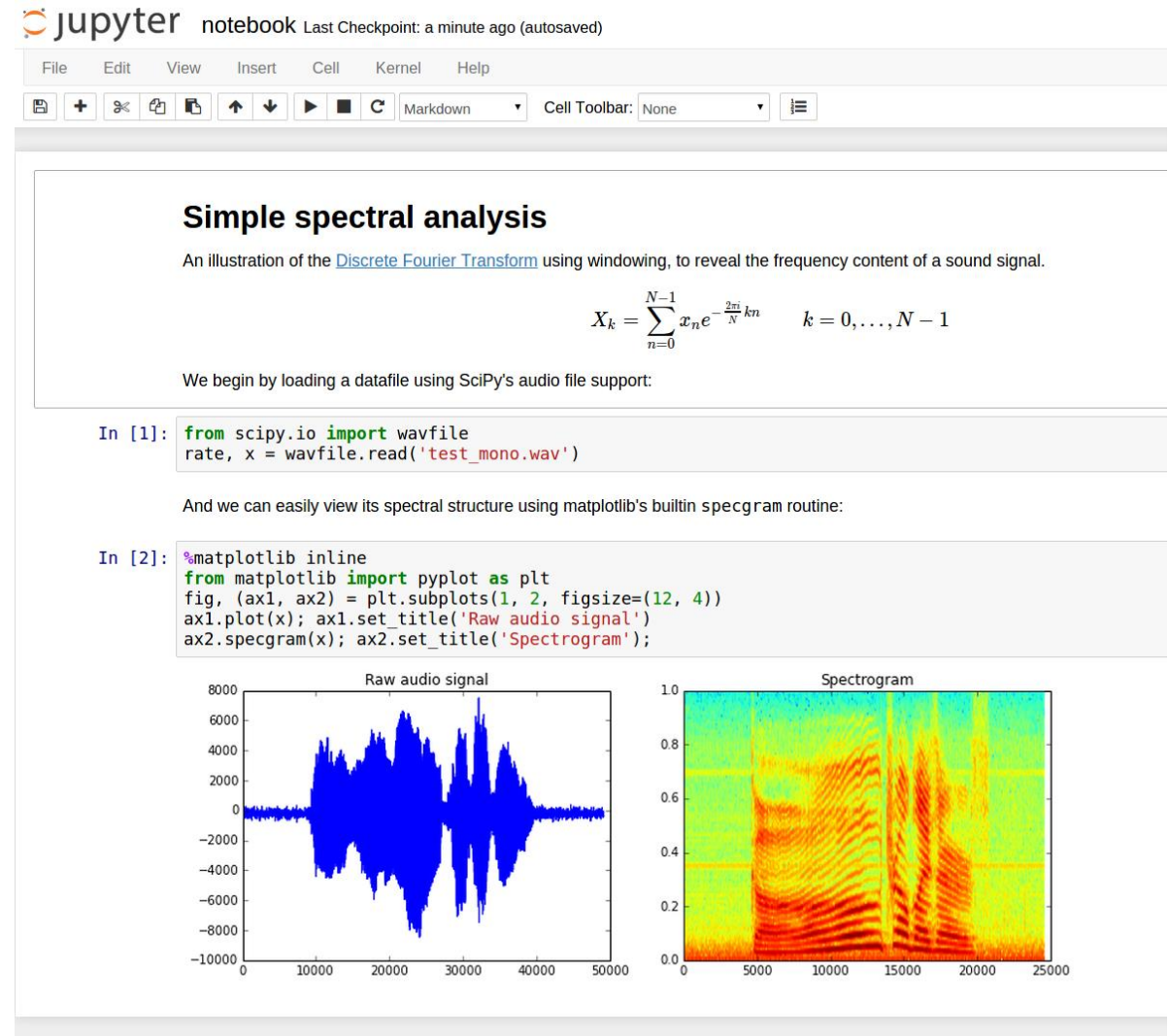


THE JUPYTER NOTEBOOK

Joe Futrelle, October 2018

What is the Jupyter notebook?

- Open-source browser-based tool
 - ▣ Developing code
 - ▣ Visualizing results
 - ▣ Documenting code
 - ▣ Sharing code on the web
- Inspired by Mathematica, MATLAB, and RStudio
- Formerly IPython notebook
 - ▣ But supports many languages



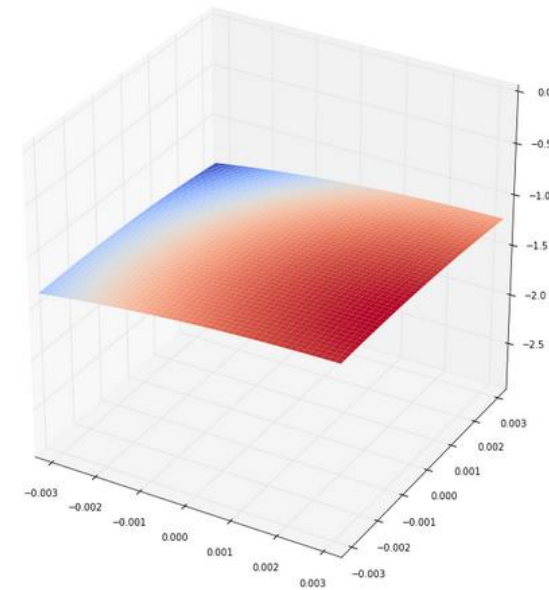
Why should you care?

- Free (no license fee)
- Prototype rapidly with integrated interactive visualization
- Document and share code
 - ▣ Similar to R vignettes
- Allow others to easily run and modify your code
- Lots and lots of integrated tools and libraries
 - ▣ Rapidly growing list

```
In [77]: import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(12,12))

# 'ax' is a 3D-aware axis instance because of the projection='3d' keyword argument to add_subplot
plot
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(I, J, -D, rstride=1, cstride=1, linewidth=0, cmap=cm.coolwarm)
ax.set_zlim(-np.max(D)*2, 0)

Out[77]: (-2.9124530179267176, 0)
```



Now discretize distance map D in the distance dimension as weights on a set of equally-spaced altitude components $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n$ where

$$\alpha_i = \Delta i$$

defining D_Δ as $\frac{D}{\Delta}$, the weight w_i is given by

$$w_i = \begin{cases} 1 - |D_\Delta - i| & i-1 < D_\Delta \leq i+1 \\ 0 & \text{otherwise} \end{cases}$$

which has the property for any $k > \max(D_\Delta)$ (i.e., over the entire altitude range of the distance map)

$$\sum_{i=0}^k w_i = 1$$

```
In [78]: # Now discretize the map in the altitude dimension as elementwise weights
# on a set of discrete altitude components such that the sum of the weights is 1.

# using evenly spaced altitude components
delta = 0.10 # 10cm

# generate a sequence of altitude components
```

Notebook model: cells, markdown, and magic

Notebook is divided into cells

Code is in code cells

Visualizations are inline

jupyter notebook Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Markdown Cell Toolbar: None

Simple spectral analysis

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

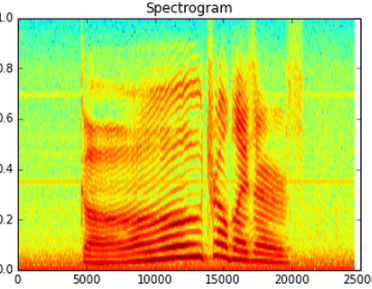
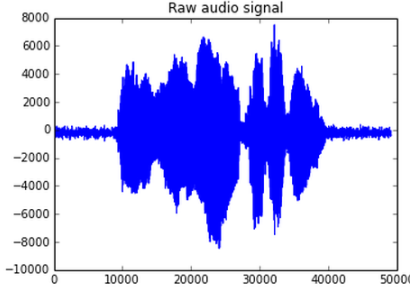
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

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 spectrogram 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.spectrogram(x); ax2.set_title('Spectrogram');
```



The figure displays two side-by-side plots. The left plot, titled 'Raw audio signal', shows a blue waveform of the audio signal over time, with the x-axis ranging from 0 to 50,000 and the y-axis from -10,000 to 8,000. The right plot, titled 'Spectrogram', shows the frequency content of the signal as a heatmap, with the x-axis representing time (0 to 25,000) and the y-axis representing frequency (0.0 to 1.0). The spectrogram uses a color scale from blue (low intensity) to red (high intensity) to show the energy distribution across different frequencies over time.

HTML documentation is in markdown cells

Magics extend notebook functionality

Notebook model: code cells

Code cell

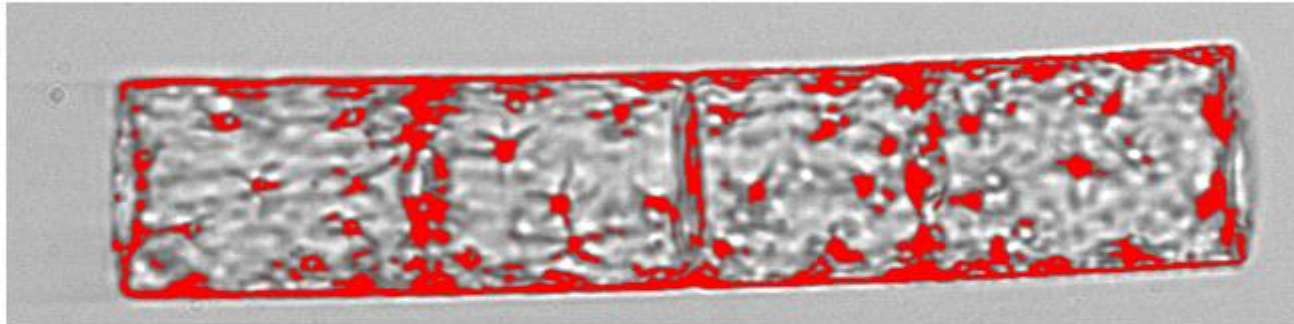
```
In [21]: from scipy.cluster.vq import kmeans2

# step 4: estimate dark areas using kmeans
samples = roi.reshape((roi.size, 1))
(means, _) = kmeans2(samples, k=2)
thresh = np.mean(means)
dark = roi < (thresh * 0.65)

show_masked(rgb, dark)
```

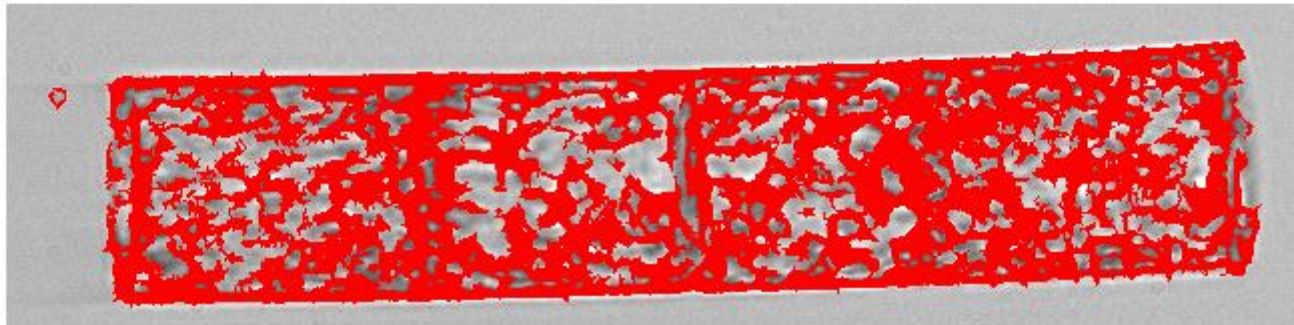
Output of
code cell

Out[21]:



```
In [22]: dark_blob = blob + dark
show_masked(rgb, dark_blob)
```

Out[22]:



Cells can be run
individually (e.g.,
repeatedly) and in
any order

Cells share all
variables

Notebook model: markdown cells

Simple alternative
to HTML

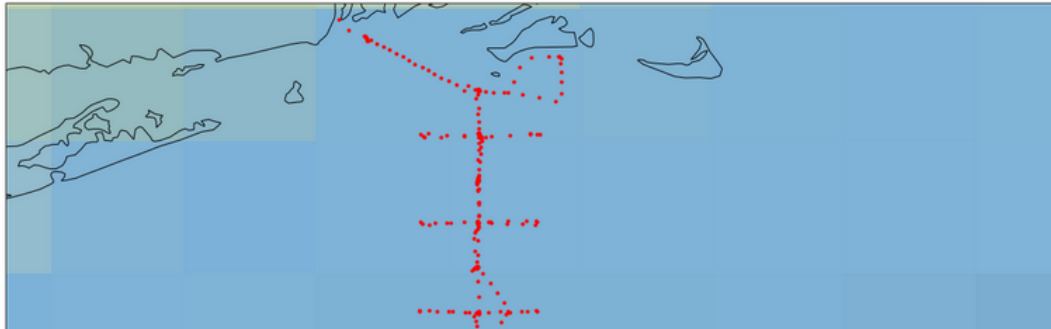
Generates HTML

```
### Plot cruise track using cartopy(https://scitools.org.uk/cartopy/docs/latest/)  
  
* Using 10m coastlines  
* Using stock_img (https://scitools.org.uk/cartopy/docs/latest/matplotlib/geoaxes.html?highlight=stock_img#cartopy.mpl.geoaxes.GeoAxes.stock_img) until I can find a better base map
```

Plot cruise track using [cartopy](https://scitools.org.uk/cartopy/docs/latest/)

- Using 10m coastlines
- Using `stock_img` until I can find a better base map

```
In [8]: %matplotlib inline  
import matplotlib.pyplot as plt  
  
import cartopy.crs as ccrs  
  
fig = plt.figure(figsize=(20, 10))  
ax = fig.add_subplot(1, 1, 1, projection=ccrs.PlateCarree())  
  
ax.coastlines(resolution='10m')  
r = 2  
ax.set_extent([wh_lon-r, wh_lon+r, wh_lat-r, wh_lat], ccrs.PlateCarree())  
  
ax.scatter(lons, lats, color='red', s=9, transform=ccrs.Geodetic())  
  
ax.stock_img()  
None
```



Notebook model: fancier markdown example

Inline equations in
markdown cell

```
Since  $L_s = u \overrightarrow{L_i}$  we only need solve for  $u$ 


$$u = \frac{-\alpha (am - lb)}{-i (bn - mc) + j (an - lc) + f (am - lb)}$$


So distance to substrate for a given  $\begin{bmatrix} i & j \end{bmatrix}$  and  $\alpha$  is  $||\overrightarrow{uL_i}||$  or


$$\left\| \begin{bmatrix} \frac{-\alpha (am - lb)}{-i (bn - mc) + j (an - lc) + f (am - lb)} \\ i \\ j \\ -f \end{bmatrix} \right\|$$

```

Since $L_s = u \overrightarrow{L_i}$ we only need solve for u

$$u = \frac{-\alpha(am - lb)}{-i(bn - mc) + j(an - lc) + f(am - lb)}$$

Rendering

So distance to substrate for a given $\begin{bmatrix} i & j \end{bmatrix}$ and α is $||\overrightarrow{uL_i}||$ or

$$\left\| \begin{bmatrix} \frac{-\alpha(am - lb)}{-i(bn - mc) + j(an - lc) + f(am - lb)} \\ i \\ j \\ -f \end{bmatrix} \right\|$$

Notebook model: magic (“%” and “%%”)

In [1]: `%load_ext Cython`

In [2]: `def python_fib(n):
 a, b = 1, 1
 for _ in range(n):
 a, b = a + b, a
 return a`

Line magic

`%timeit python_fib(75)`

19.1 μ s \pm 534 ns per loop (mean \pm std. dev. of 7 runs, 10000 loops each)

Cell magic

In [3]: `%%cython`

```
def cython_fib(int n):  
    cdef long a, b  
    cdef int i  
    a, b = 1, 1  
    for i in range(n):  
        a, b = a + b, a  
    return a
```

In [4]: `%timeit cython_fib(75)`

340 ns \pm 5.47 ns per loop (mean \pm std. dev. of 7 runs, 1000000 loops each)

Live demos: wish me luck!



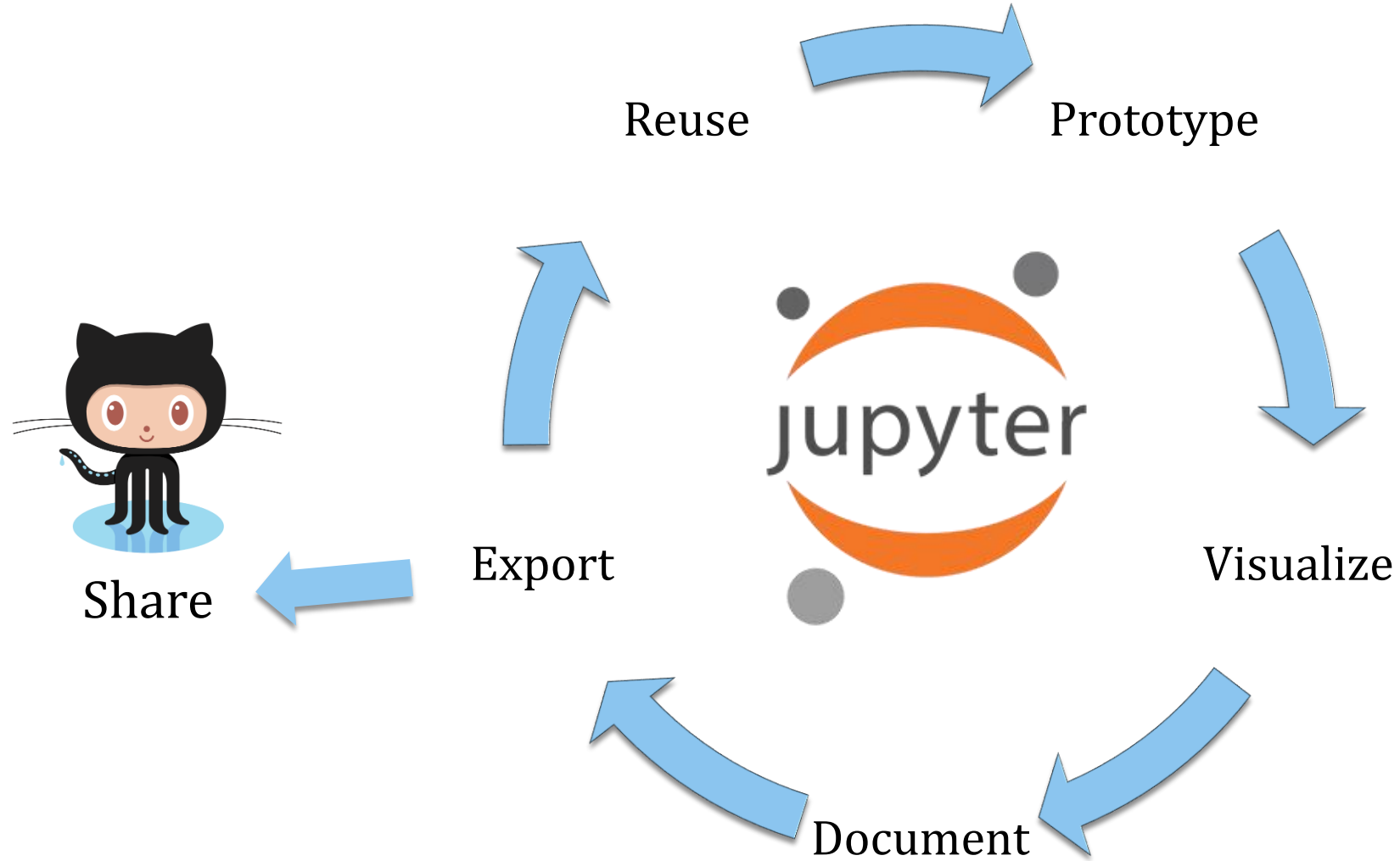
Sharing notebooks

- Upload to GitHub and GitHub will render them
 - ▣ e.g., <https://gist.github.com/joefutrelle/9898646>
- Use nbviewer.jupyter.org for similar functionality
 - ▣ e.g., <https://nbviewer.jupyter.org/gist/joefutrelle/9898646>
- Embed in blogs and discussion forums
- Share .ipynb files
 - ▣ Simple JSON text format
 - ▣ Includes output such as plots, etc.
- Hosting services for notebooks: not out there yet
 - ▣ But you can run your own multi-user JupyterHub
 - ▣ Integration with HPC through Pangeo

Caveats: things that Jupyter isn't good at (yet)

- No built-in way to manage dependencies
 - ▣ But integrated external tools like Anaconda help a lot
- Hard to version control notebooks
 - ▣ .ipynb files are JSON, not ordinary plain text files with lines of code
 - ▣ Snapshotting works
- No built-in automated testing
 - ▣ But ordinary defensive coding works
- Code editing lacks many features found in IDEs (yet)
- Notebook model has some issues
 - ▣ Out of order code execution can be confusing

My Jupyter workflow



Bottom line

- Jupyter is an emerging and popular open-source project in the data science community
- It's very good for exploratory data analysis and development of interactive visualizations
- It's good for prototyping new capabilities or libraries
- It's great for sharing code on the web
- New features are being added all the time
- It's fun!