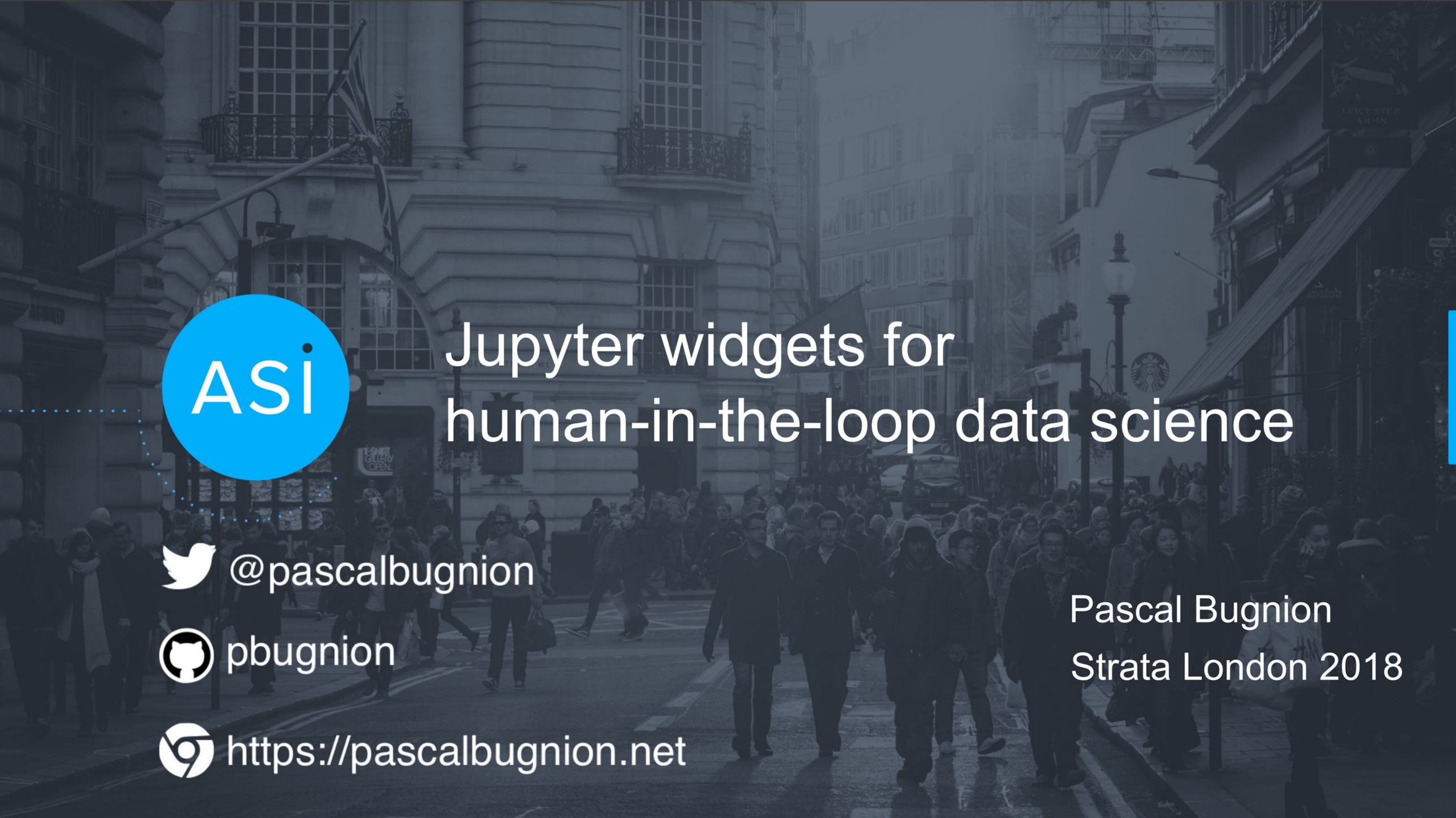


Jupyter widgets for human-in-the-loop data science

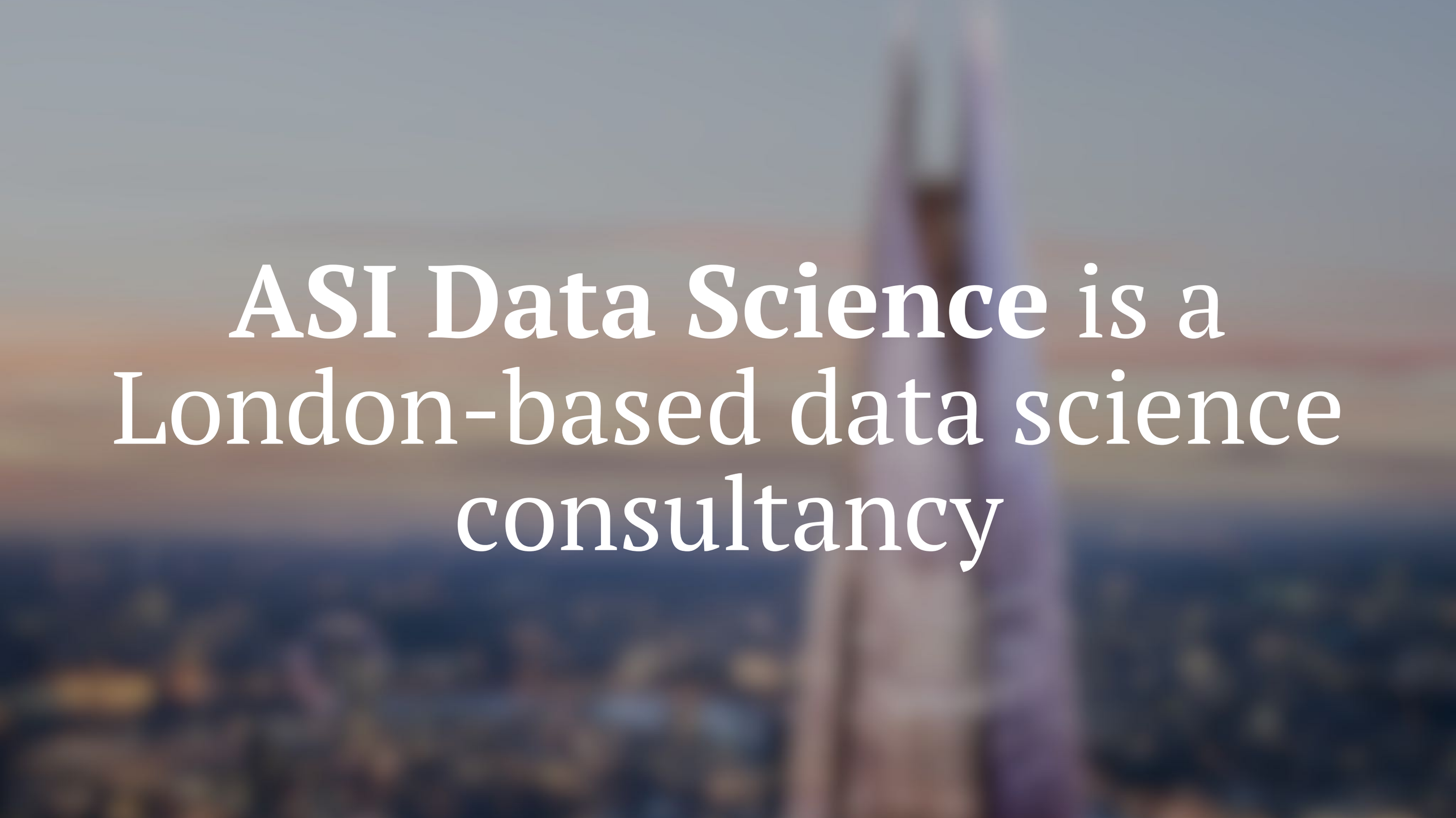
@pascalbugnion

pbugnion

https://pascalbugnion.net

Pascal Bugnion

Strata London 2018



ASI Data Science is a London-based data science consultancy

About me

— Committer for **Jupyter widgets**

— Main author of **jupyter-gmaps**, a library for visualizing geographical data in Jupyter notebooks

— Author of **Scala for data science** (Packt Publishing)

Jupyter widgets for human-in-the-loop data science


Developing machine
learning software is hard

Developing machine learning software is hard

— almost always stochastic

— often black box

Developing machine learning software is hard

— robustness
— overfitting
— overreliance on certain features or groups of features

Traditional software development workflows are inadequate

Human intuition

It should be easy for humans to explore the model.

Human intuition

Computers think in terms of bytes and instructions, and humans think in terms of concepts and images.

Human intuition

We need a framework to rapidly create UIs that allow the human to think at a higher level of abstraction.

The UI should not be a black box.

Jupyter widgets

jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

Simple spectral analysis

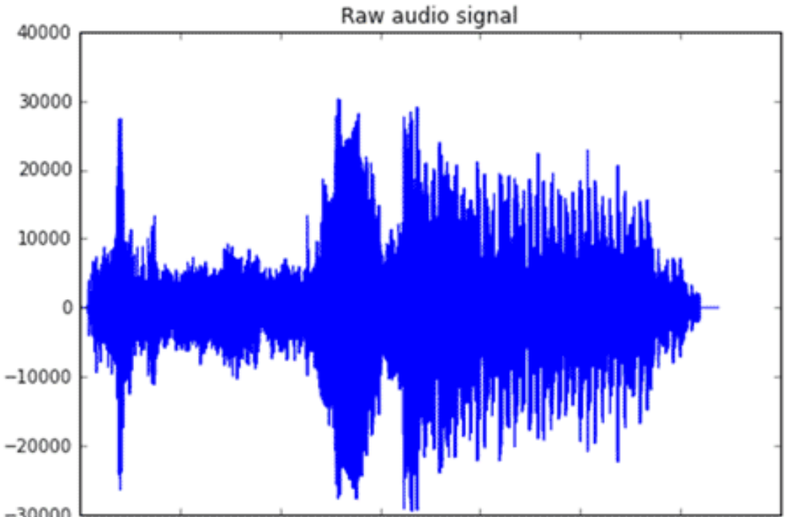
An illustration of the Discrete Fourier Transform

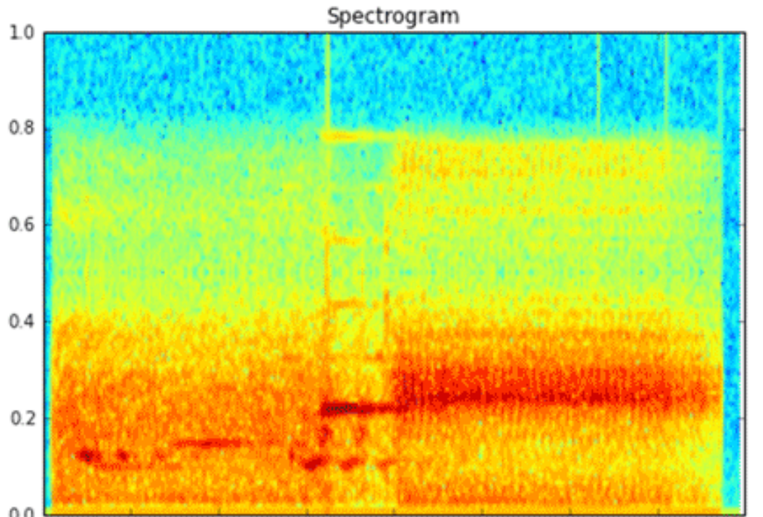
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

In [2]:
```
from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

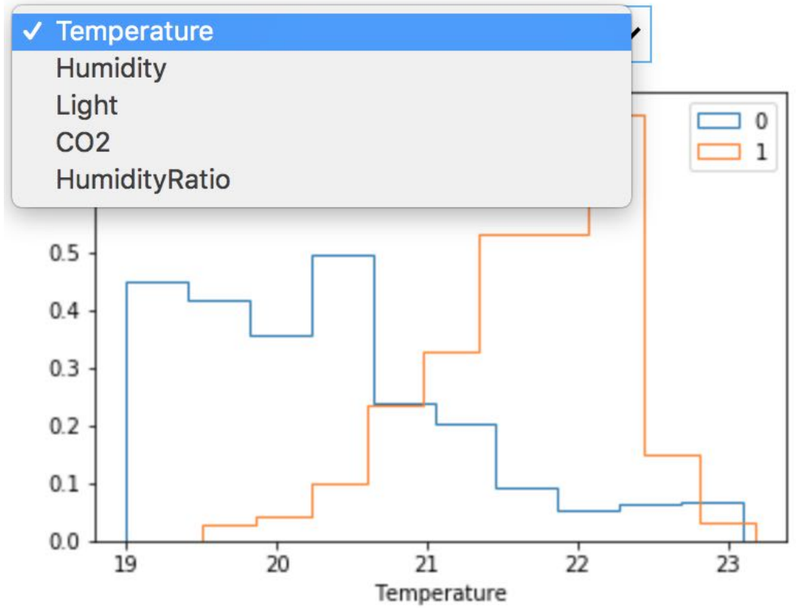
And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

In [5]:
```
fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```





```
columns = [column for column in df.columns if column not in {'date', 'Occupancy'}]
column_selector = widgets.Dropdown(options=columns)
out = widgets.interactive_output(
    lambda training_column_name: plot_classification(df, 'Occupancy', training_column_name),
    {'training_column_name': column_selector}
)
widgets.VBox([column_selector, out])
```



Jupyter widgets allow building user interfaces entirely in Python, directly in Jupyter notebooks.

Build user interfaces in Python directly in Jupyter notebooks

Examples

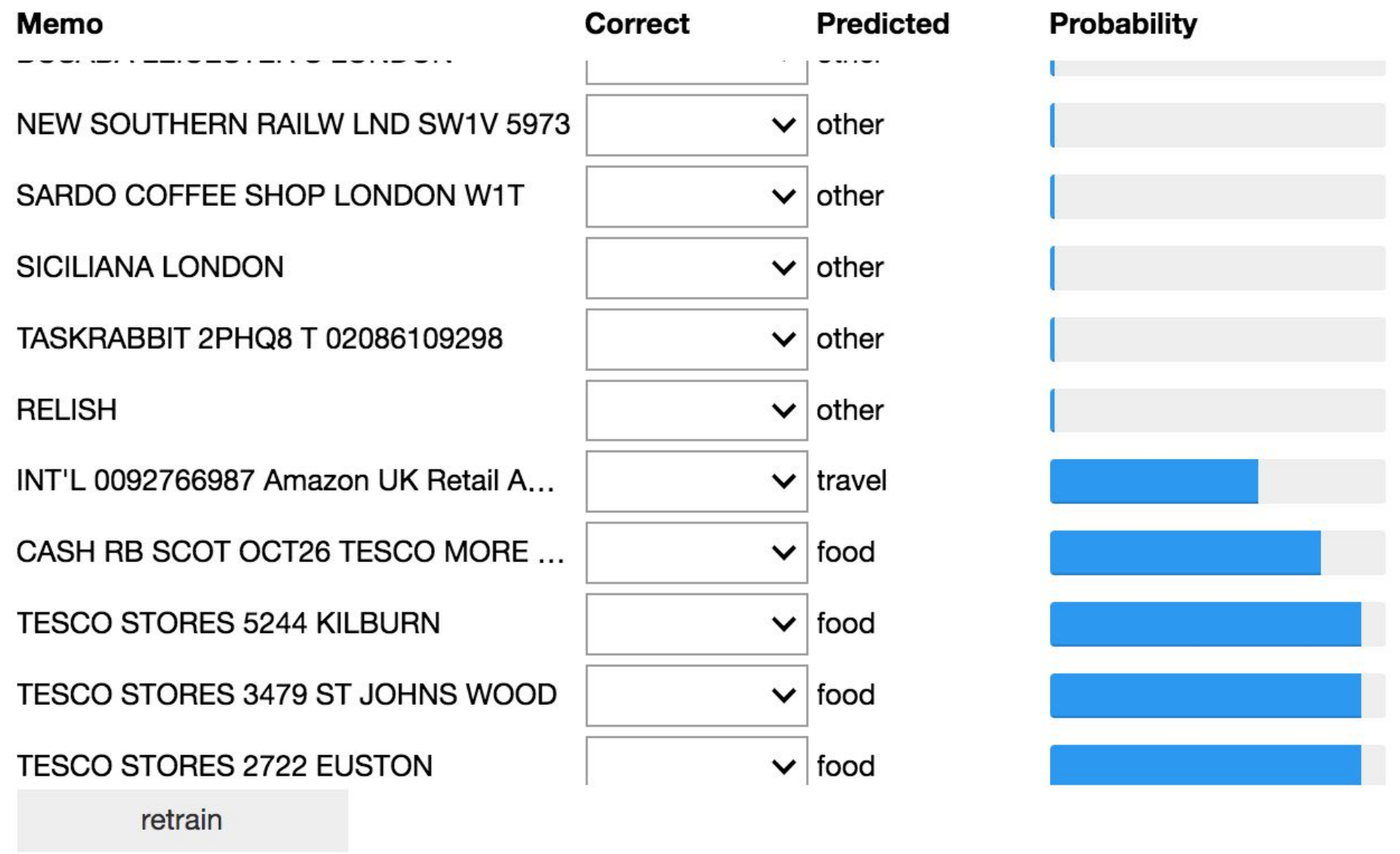
Jupyter widgets

- Jupyter widgets are written entirely in Python
- They are written in the environment the data scientist is currently working in
- Widgets have access to the entire state of the notebook

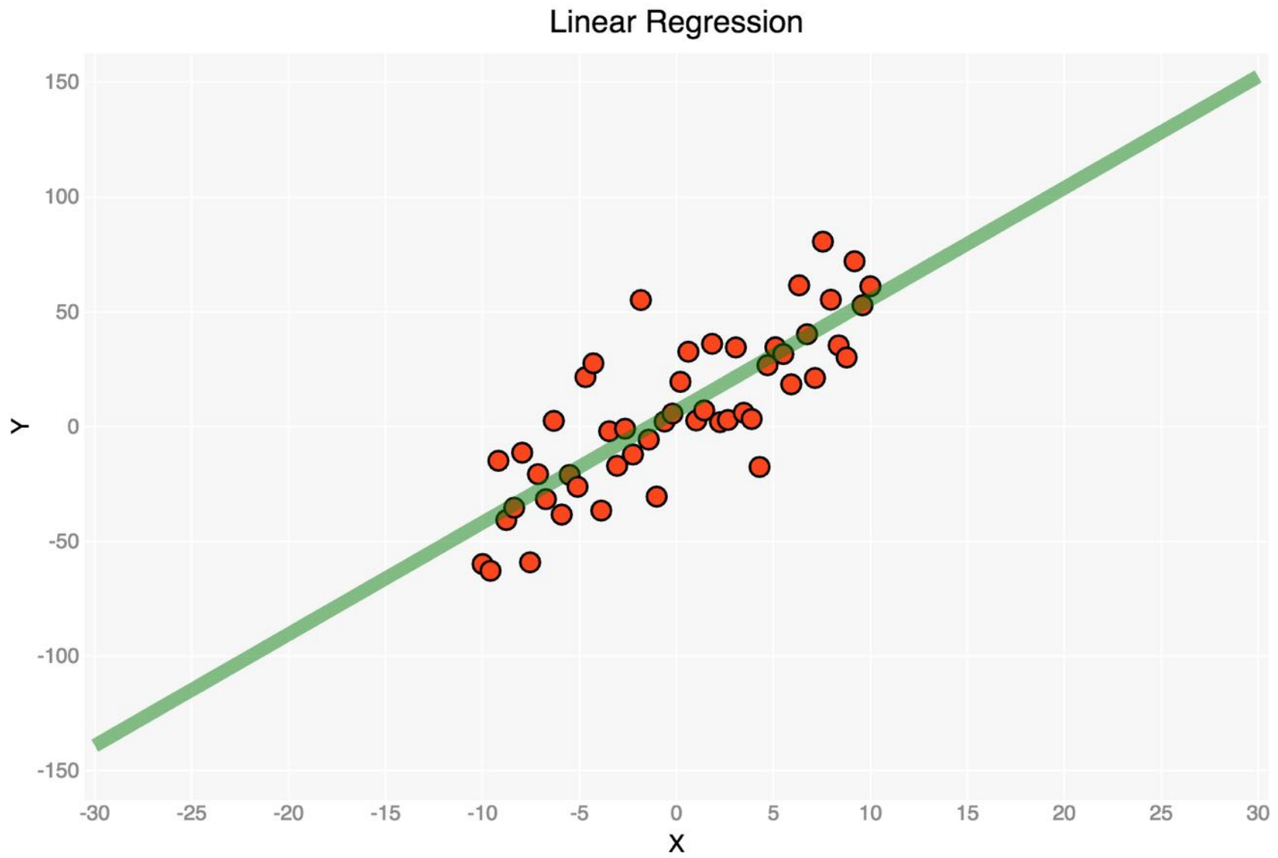
Ecosystem

Core: ipywidgets

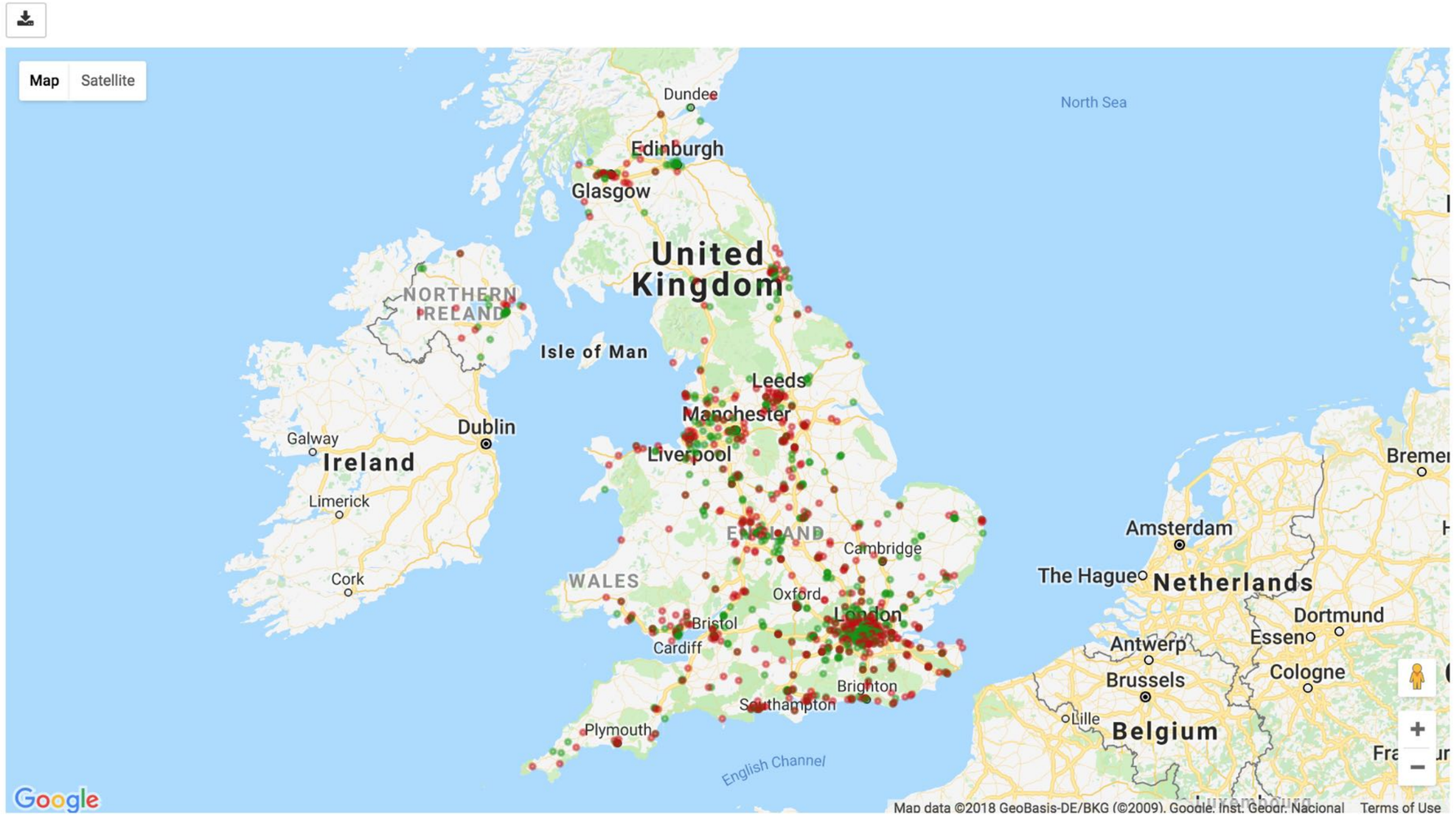
```
labeller = Labeller(transactions, CATEGORIES)
labeller.render()
```



bqplot

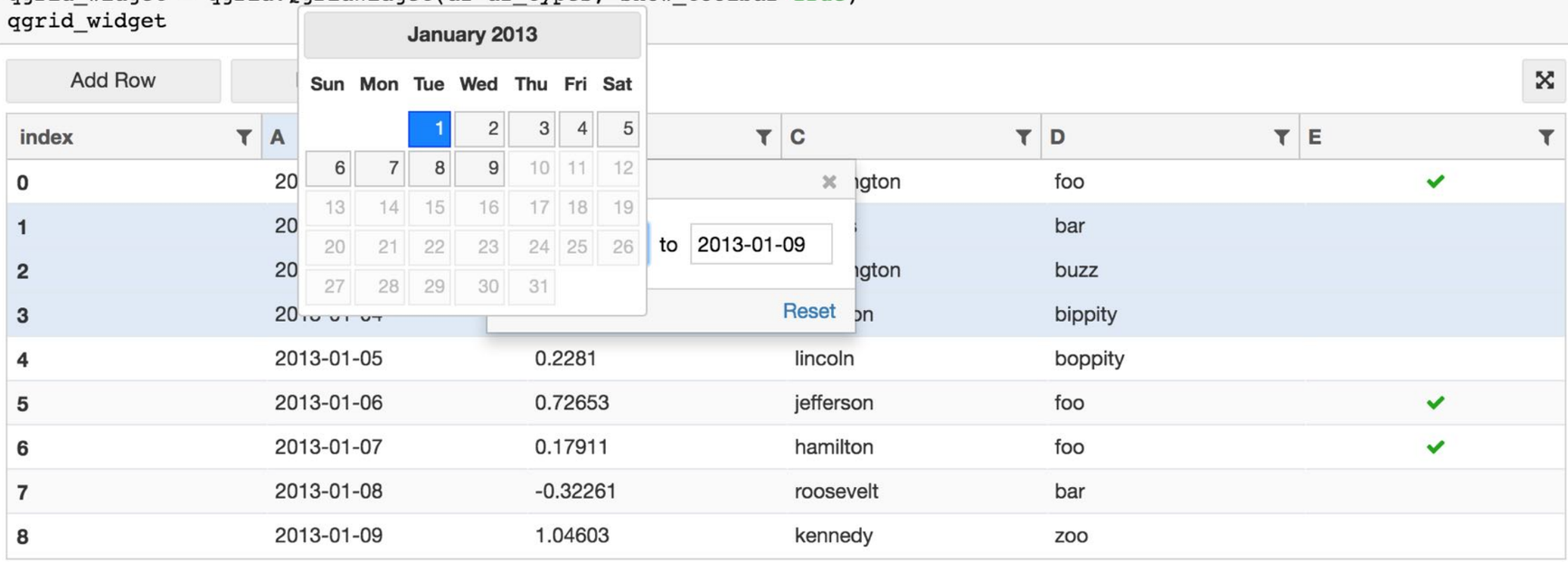

Linear Regression
150
100
50
0
-50
-100
-150
Y
-30
-25
-20
-15
-10
-5
0
5
10
15
20
25
30
X
Reset
Regression Line: $6.81 + 4.86x$

ipyleaflet and gmaps



qgrid

```
import numpy as np
import pandas as pd
import qgrid
randn = np.random.randn
df_types = pd.DataFrame({
    'A' : pd.Series(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
                '2013-01-05', '2013-01-06', '2013-01-07', '2013-01-08', '2013-01-09'],index=list(range(9)),dtype='dateti
    'B' : pd.Series(randn(9),index=list(range(9)),dtype='float32'),
    'C' : pd.Categorical(["washington", "adams", "washington", "madison", "lincoln","jefferson", "hamilton", "roosevelt
    'D' : ["foo", "bar", "buzz", "bippity","boppity", "foo", "foo", "bar", "zoo"] })
df_types['E'] = df_types['D'] == 'foo'
qgrid_widget = qgrid.QgridWidget(df=df_types, show_toolbar=True)
qgrid_widget
```



index	A		C	D	E
0	20		ngton	foo	✔
1	20			bar	
2	20		ngton	buzz	
3	20		on	bippity	
4	2013-01-05	0.2281	lincoln	boppity	
5	2013-01-06	0.72653	jefferson	foo	✔
6	2013-01-07	0.17911	hamilton	foo	✔
7	2013-01-08	-0.32261	roosevelt	bar	
8	2013-01-09	1.04603	kennedy	zoo	

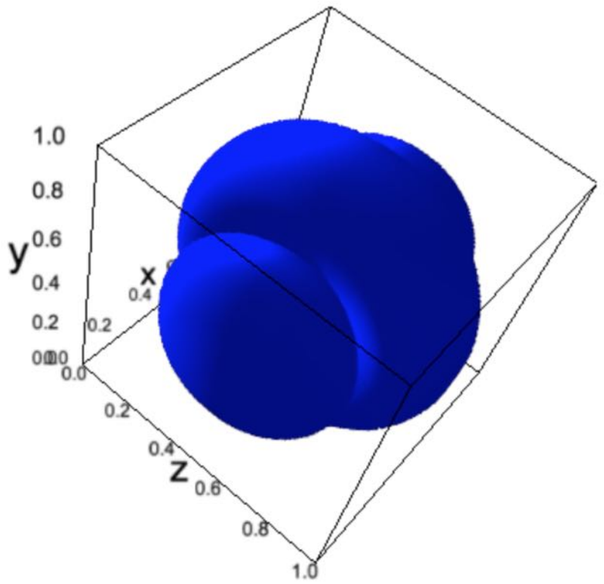
ipyvolume

```
In [1]: import ipyvolume

In [2]: ipyvolume.examples.ball(rmax=3, rmin=2.5, shape=64, lighting=True)
```

levels: 0.17

opacities: 0.00

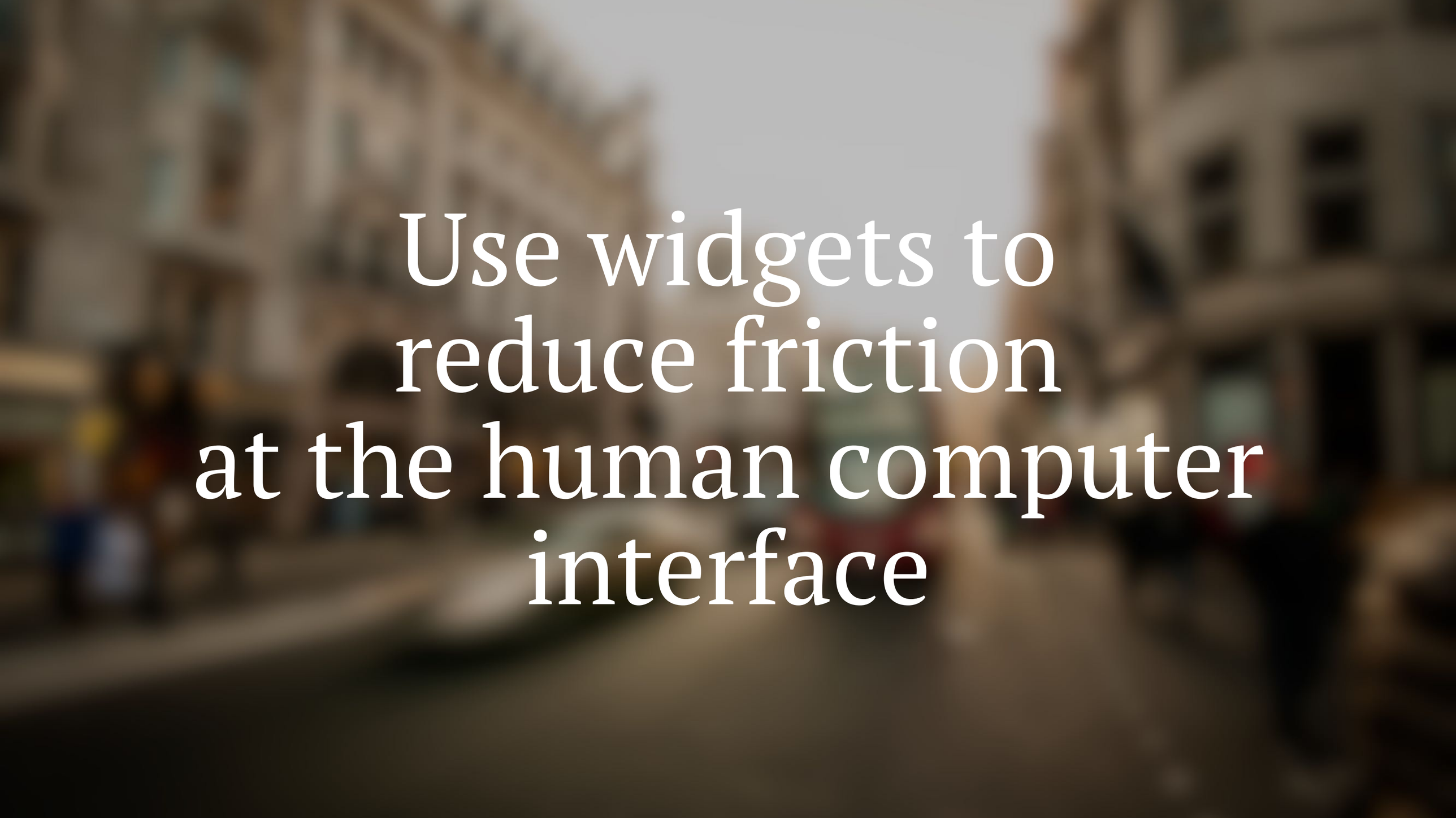


Learning about widgets

- mlviz: visualising machine learning algorithms with Jupyter widgets and bqplot.
- Jupyter widgets tutorial
- Coding a simple widget from scratch: video and code
- Jupyter widgets documentation

Libraries used in this talk

- ipywidgets
- bqplot
- gmaps
- lens
- superintendent


Use widgets to
reduce friction
at the human computer
interface

Acknowledgements

— Jupyter widgets developers: Jason Grout, Sylvain Corlay, Maarten Breddels, Matt Craig, Vidar Tonaas Fauske

— ASI Data Science

— Chakri Cherukuri (ChakriCherukuri)

— Victor Zabalza (zblz) and Scott Stevenson (srstevenson)

— Jan Freyberg (janfreyberg)

— SherlockML