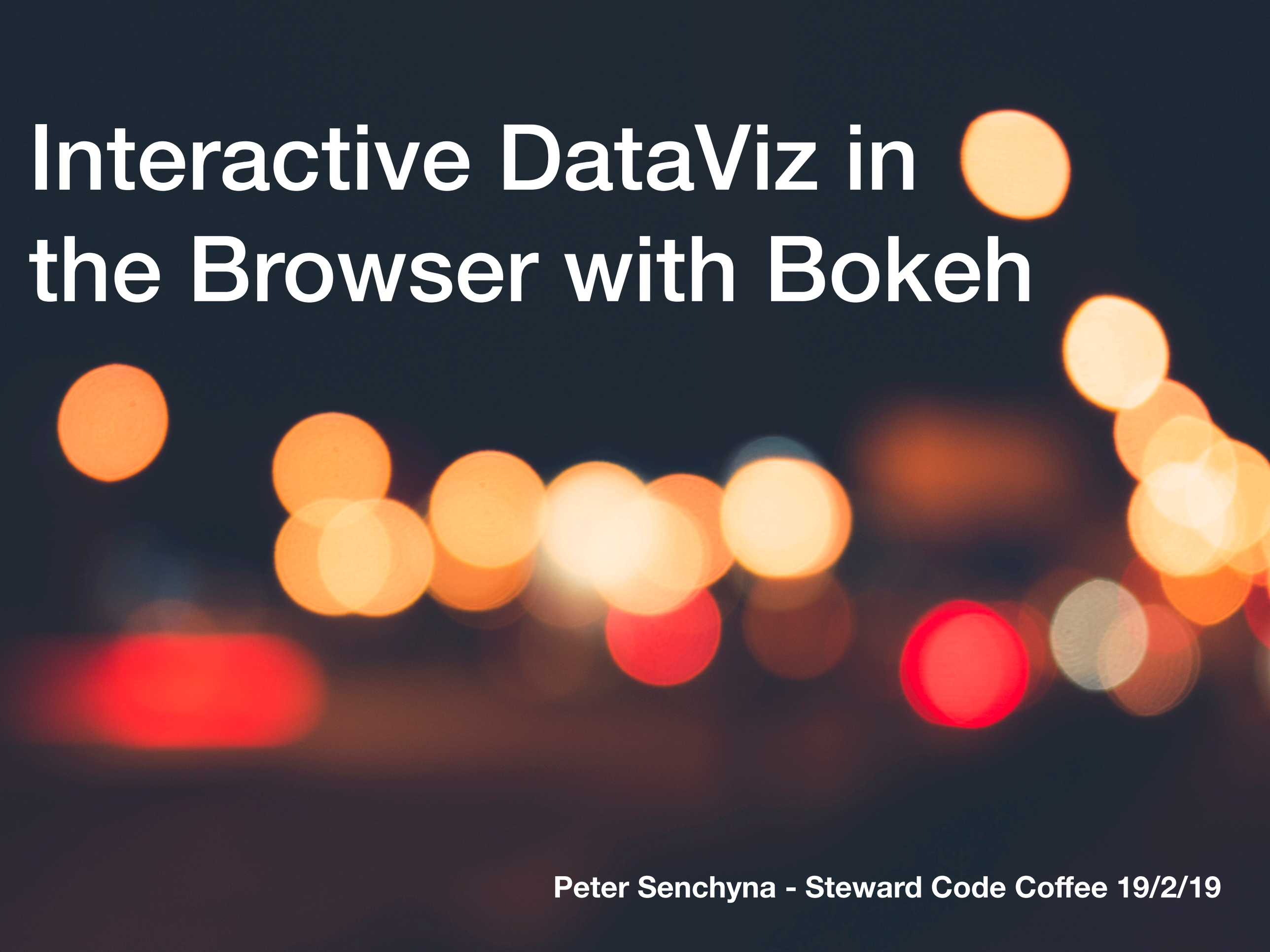
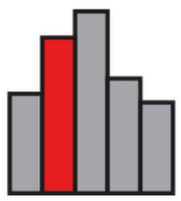
Interactive DataViz in the Browser with Bokeh



Peter Senchyna - Steward Code Coffee 19/2/19

Glue: multi-dimensional linked-data exploration

Glue is a Python library to explore relationships within and between related datasets



Linked Visualizations

With Glue, users can create scatter plots, histograms and images (2D and 3D) of their data. Glue is focused on the brushing and linking paradigm, where selections in any graph propagate to all others.

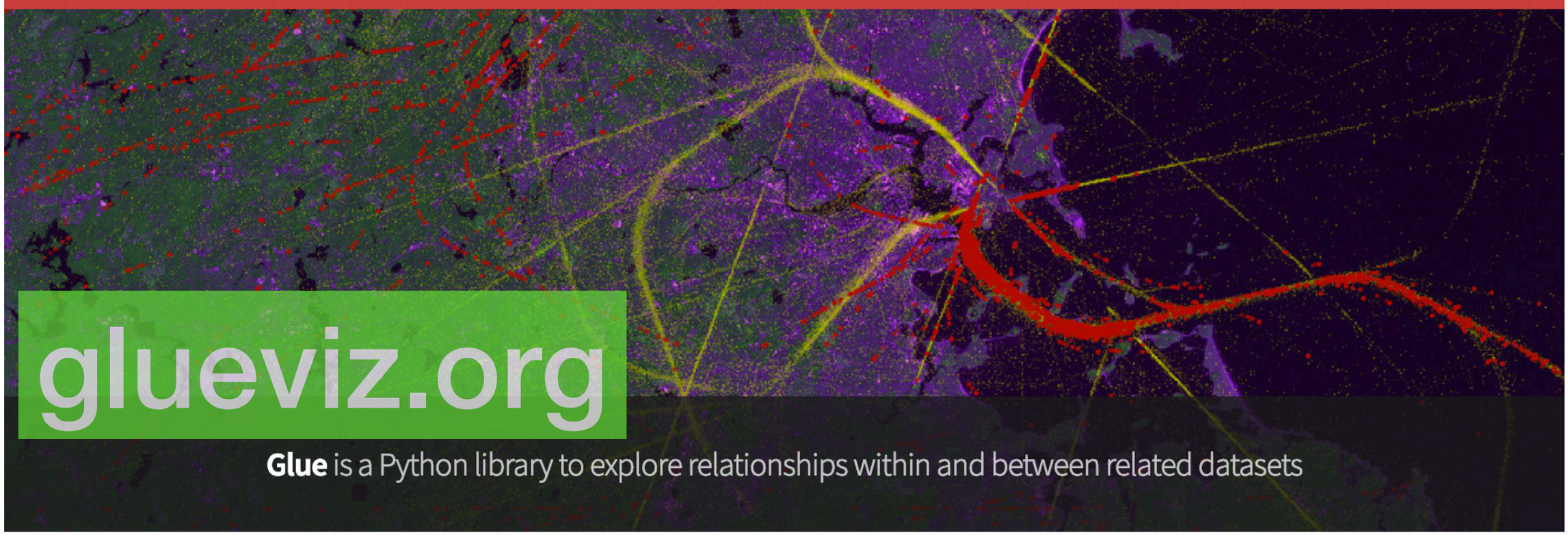
Flexible linking across data

Glue uses the logical links that exist between different data sets to overlay visualizations of different data, and to propagate selections across data sets. These links are specified by the user, and are arbitrarily flexible

Full scripting capability

Glue is written in Python, and built on top of its standard scientific libraries (i.e., Numpy, Matplotlib, Scipy). Users can easily integrate their own python code for data input, cleaning, and analysis.

Glue: multi-dimensional linked-data exploration



Linked Visualizations

With Glue, users can create scatter plots, histograms and images (2D and 3D) of their data. Glue is focused on the brushing and linking paradigm, where selections in any graph propagate to all others.

Flexible linking across data

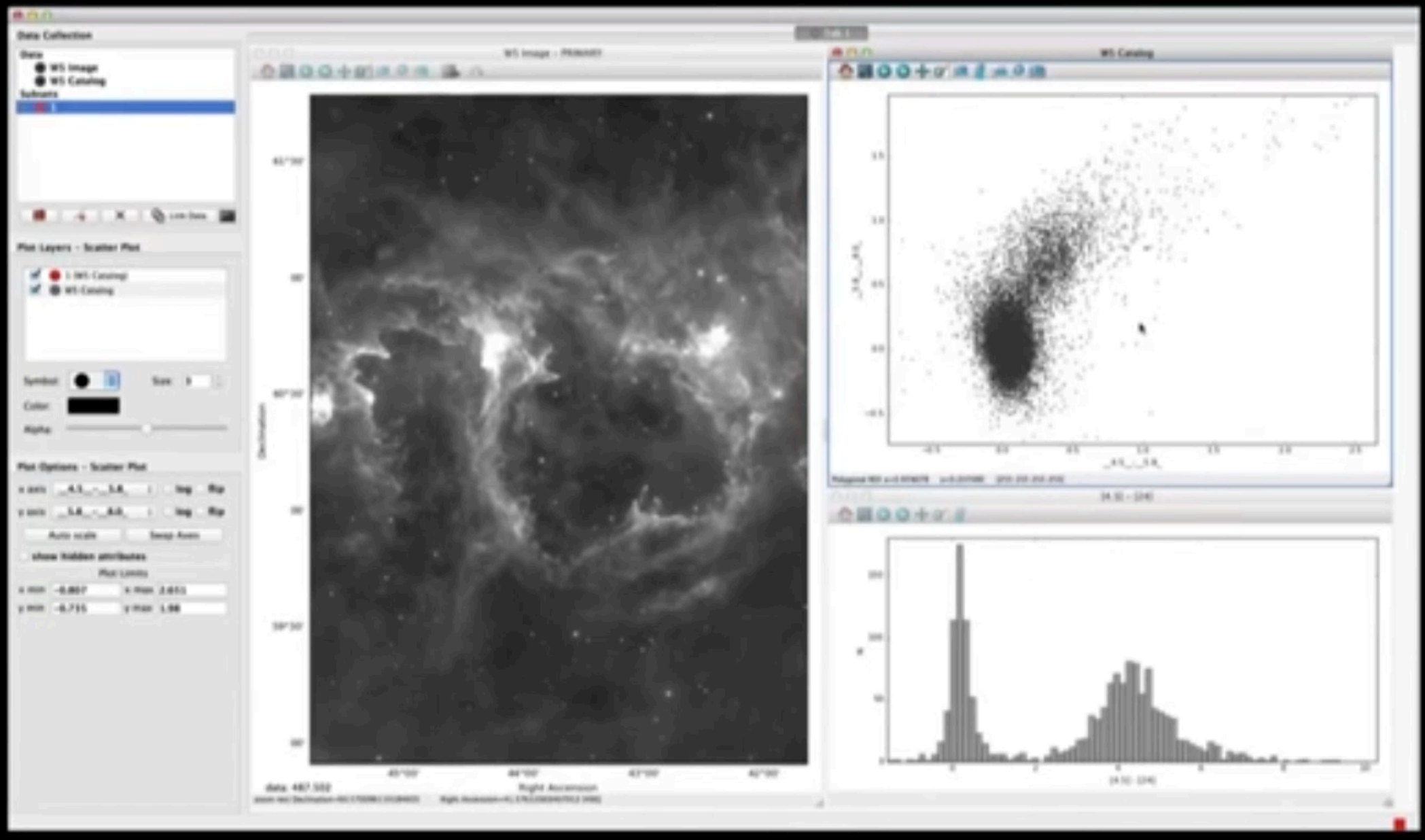
Glue uses the logical links that exist between different data sets to overlay visualizations of different data, and to propagate selections across data sets. These links are specified by the user, and are arbitrarily flexible

Full scripting capability

Glue is written in Python, and built on top of its standard scientific libraries (i.e., Numpy, Matplotlib, Scipy). Users can easily integrate their own python code for data input, cleaning, and analysis.

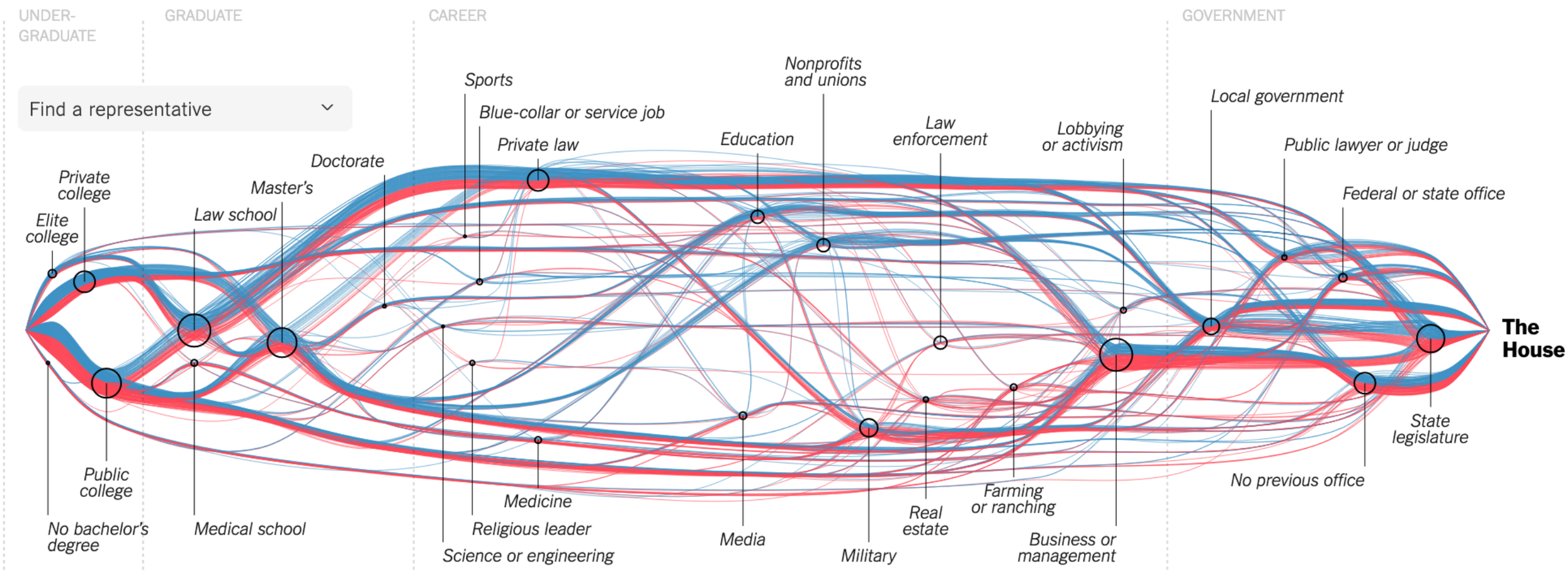
LINKED VIEWS OF HIGH-DIMENSIONAL DATA (IN PYTHON)

GLUE



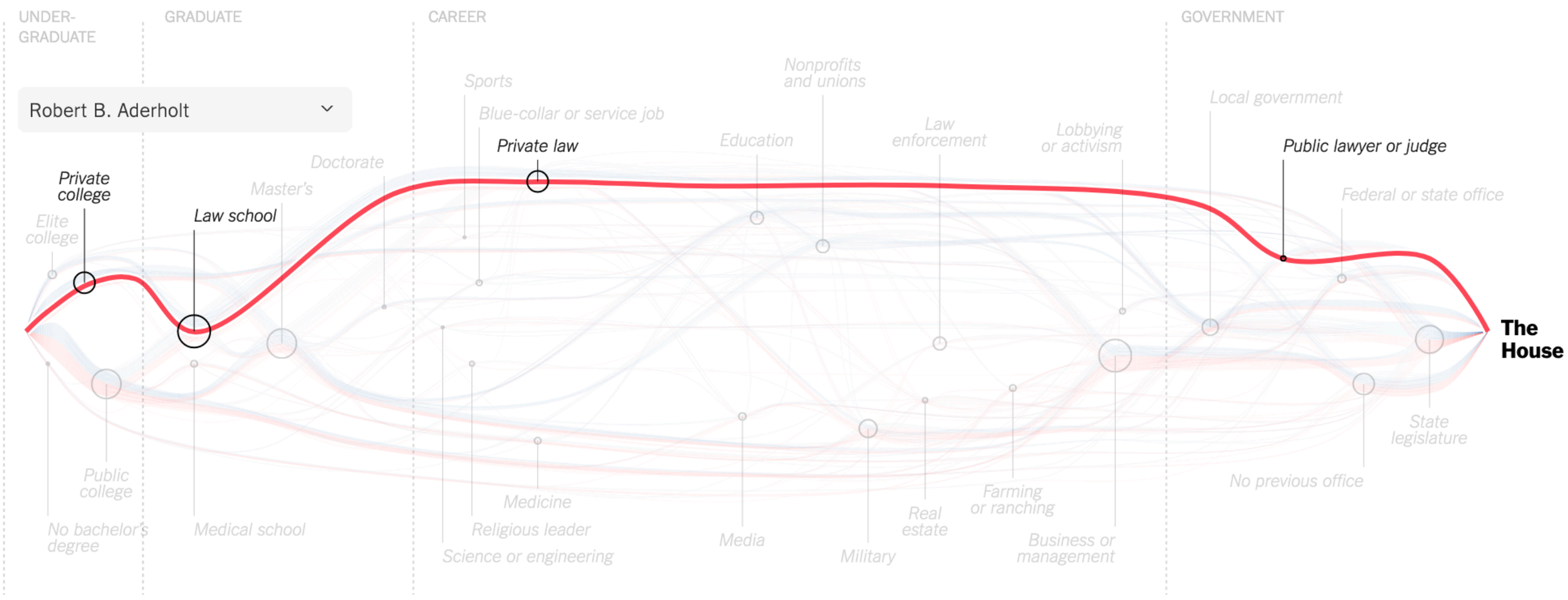
what we're actually talking about today:

data visualization in the browser



what we're actually talking about today:

data visualization in the browser



what we're actually talking about today:

data visualization in the browser

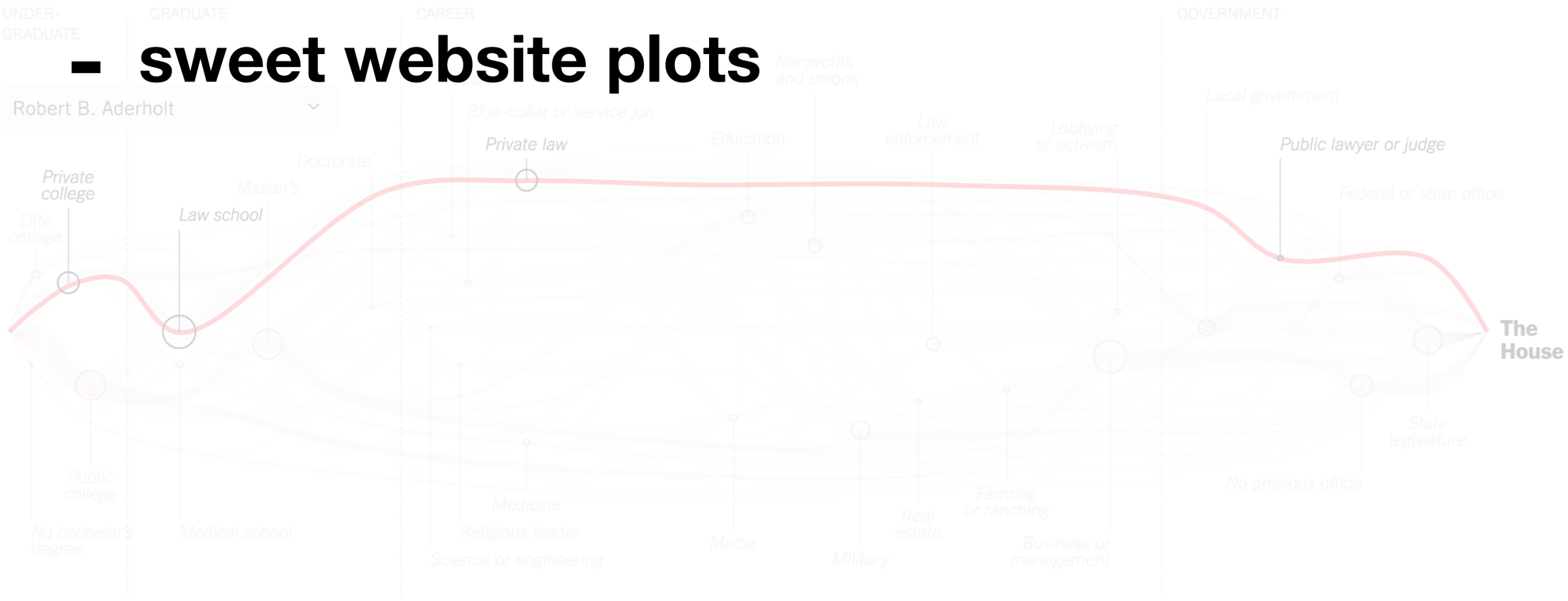
why would you want to do this

what we're actually talking about today:

data visualization in the browser

why would you want to do this

- sweet website plots



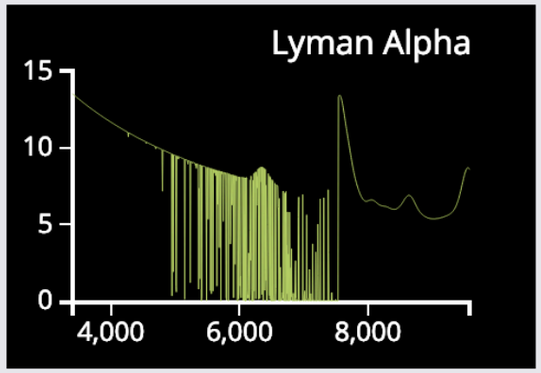
what we're actually talking about today:

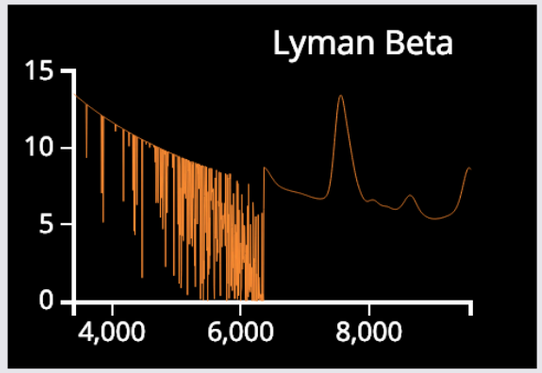
data visualization in the browser

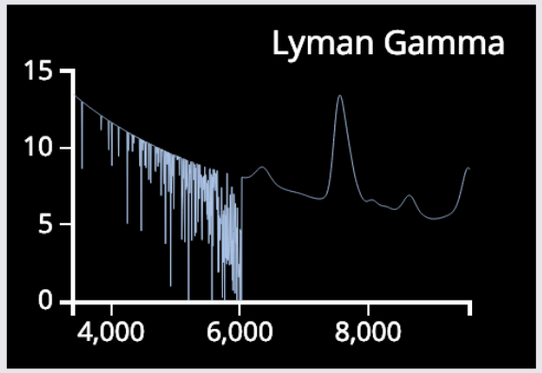
why would you want to do this

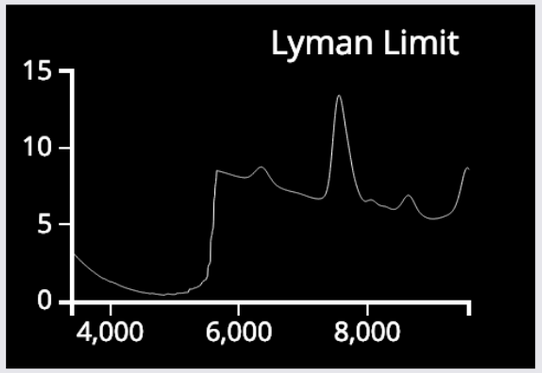
- sweet website plots
- good way to convey complex topics

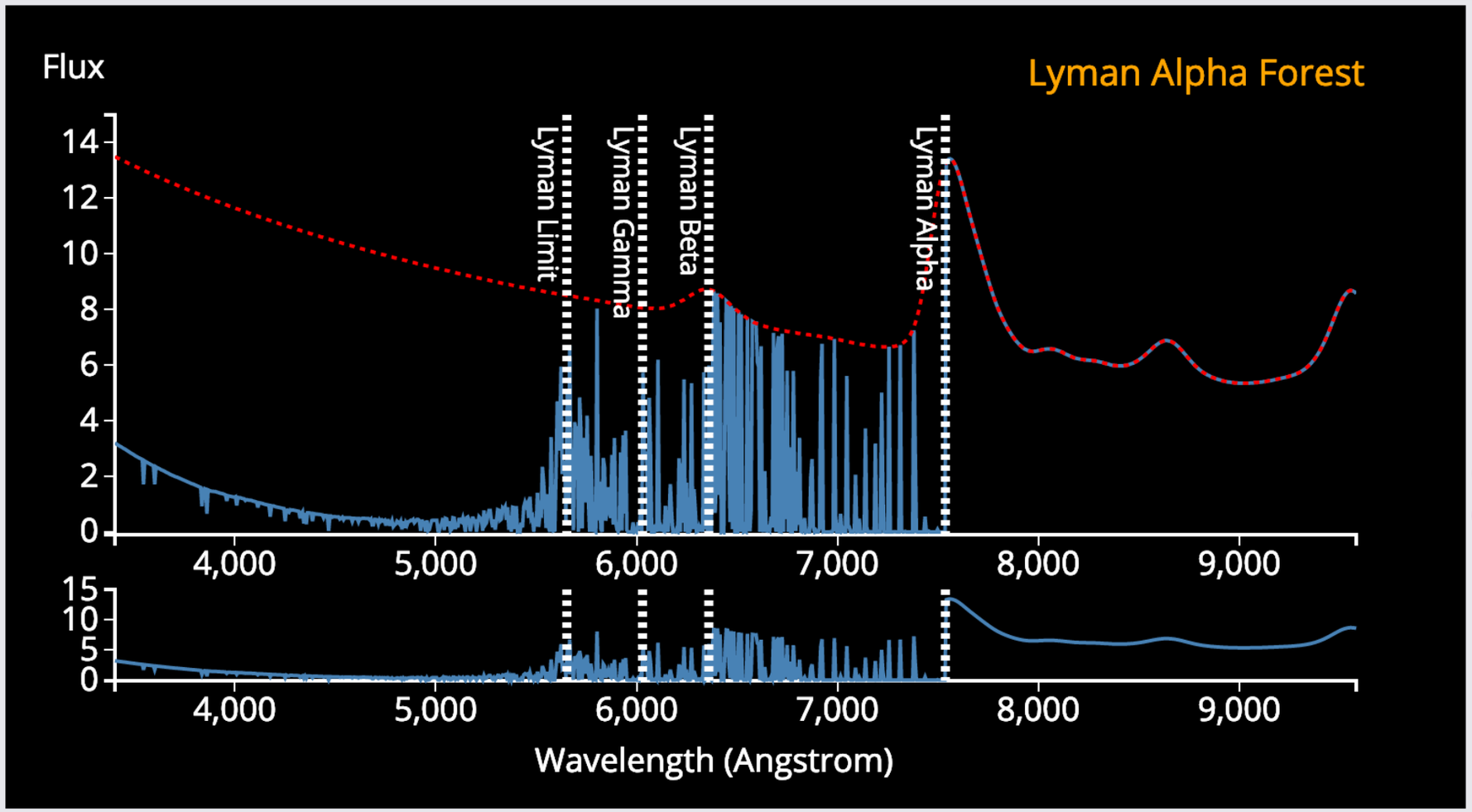
tinyurl.com/lymanalphademo











OFF Remove labels

ON Lyman-alpha
ON Lyman-beta
ON Lyman-gamma
ON Lyman-limit

Continuum subtracted
Quasar mode
ON Quasar Spectrum

Animate

Total column density
7.9e17 atoms/cm²

Redshift index, β
8.1

Column density index, α
2.1

Quasar redshift
5.2

Yuan-Sen Ting (CfA)

D3.js

low level
highly customizable
lots of overhead

bokeh

high level
python based interface
quick and fairly robust

bokeh_line.py

```
from bokeh.plotting import figure, output_file, show

output_file("test.html")
p = figure()
p.line([1,2,3,4,5], [2,3,5,9,8], line_width=2)
show(p)
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Bokeh Plot</title>
      <link rel="stylesheet" href="https://cdn.pydata.org/bokeh/release/bokeh-1.0.2.min.css" type="text/css" />
    <script type="text/javascript" src="https://cdn.pydata.org/bokeh/release/bokeh-1.0.2.min.js"></script>
    <script type="text/javascript">
        Bokeh.set_log_level("info");
    </script>
  </head>
```

<script type="application/json" id="1112">
{"5763bcd5-f135-4b2f-baf9-8e107fccd9a1":{"roots":{"references":[{"attributes":{"line_alpha":0.1,
"line_color":"#1f77b4","line_width":2,"x":{"field":"x"},"y":{"field":"y"}},"id":"1037","type":"Line"},
{"attributes":{"callback":null,"data":{"x":[1,2,3,4,5],"y":[6,7,2,4,5]},"selected":{"id":"1047",
"type":"Selection"},"selection_policy":{"id":"1048","type":"UnionRenderers"}},"id":"1035",
"type":"ColumnDataSource"},{"attributes":{"source":{"id":"1035","type":"ColumnDataSource"}},"id":"1039",
"type":"CDSView"},{"attributes":{},"id":"1020","type":"PanTool"},{"attributes":{"formatter":{"id":"1042",
"type":"BasicTickFormatter"},"plot":{"id":"1001","subtype":"Figure","type":"Plot"},"ticker":{"id":"1011",
"type":"BasicTicker"}},"id":"1010","type":"LinearAxis"},{"attributes":{},"id":"1021","type":"WheelZoomTool"},
{"attributes":{},"id":"1011","type":"BasicTicker"},{"attributes":{"below":[{"id":"1010","type":"LinearAxis"}],
"left":[{"id":"1015","type":"LinearAxis"}],"renderers":[{"id":"1010","type":"LinearAxis"},{"id":"1014",
"type":"Grid"},{"id":"1015","type":"LinearAxis"},{"id":"1019","type":"Grid"},{"id":"1028",
"type":"BoxAnnotation"},{"id":"1038","type":"GlyphRenderer"}],"title":{"id":"1041","type":"Title"},"toolbar":
{"id":"1026","type":"Toolbar"},"x_range":{"id":"1002","type":"DataRange1d"},"x_scale":{"id":"1006",
"type":"LinearScale"},"y_range":{"id":"1004","type":"DataRange1d"},"y_scale":{"id":"1008","type":"LinearScale"}}
,"id":"1001","subtype":"Figure","type":"Plot"},{"attributes":{},"id":"1042","type":"BasicTickFormatter"},
{"attributes":{"overlay":{"id":"1028","type":"BoxAnnotation"}},"id":"1022","type":"BoxZoomTool"},{"attributes":
{"plot":{"id":"1001","subtype":"Figure","type":"Plot"},"ticker":{"id":"1011","type":"BasicTicker"}},"id":"1014",
"type":"Grid"},{"attributes":{"plot":null,"text":""},"id":"1041","type":"Title"},{"attributes":{},"id":"1023",
"type":"SaveTool"},{"attributes":{"formatter":{"id":"1044","type":"BasicTickFormatter"},"plot":{"id":"1001",
"subtype":"Figure","type":"Plot"},"ticker":{"id":"1016","type":"BasicTicker"}},"id":"1015","type":"LinearAxis"},
{"attributes":{},"id":"1024","type":"ResetTool"},{"attributes":{},"id":"1044","type":"BasicTickFormatter"},

bokeh_linked.py

```
from bokeh.plotting import figure
from bokeh.io import output_notebook, output_file, show
from bokeh.layouts import gridplot
from bokeh.models import ColumnDataSource
from bokeh.plotting import figure

# output_notebook()
# uncomment above line to automatically output in-line in jupyter notebook
output_file('brushing.html')

x = list(range(-20, 21))
y0 = [abs(xx) for xx in x]
y1 = [xx**2 for xx in x]

# create a column data source for the plots to share
source = ColumnDataSource(data=dict(x=x, y0=y0, y1=y1))

tools = "box_select,lasso_select,help"

# create a new plot and add a renderer
left = figure(tools=tools, plot_width=300, plot_height=300, title=None)
left.circle('x', 'y0', source=source)

# create another new plot and add a renderer
right = figure(tools=tools, plot_width=300, plot_height=300, title=None)
right.circle('x', 'y1', source=source)

p = gridplot([[left, right]])

show(p)
```

lots of useful basic plotting abstractions:

- **scatter plots**
- **polygonal glyphs**
- **hex tilings**
- **images**

bokeh.mpl.to_bokeh(fig=mpl_fig_instance) **attempts to convert matplotlib figs to bokeh!**

but what if we want something more complicated?

bokeh_slider.py

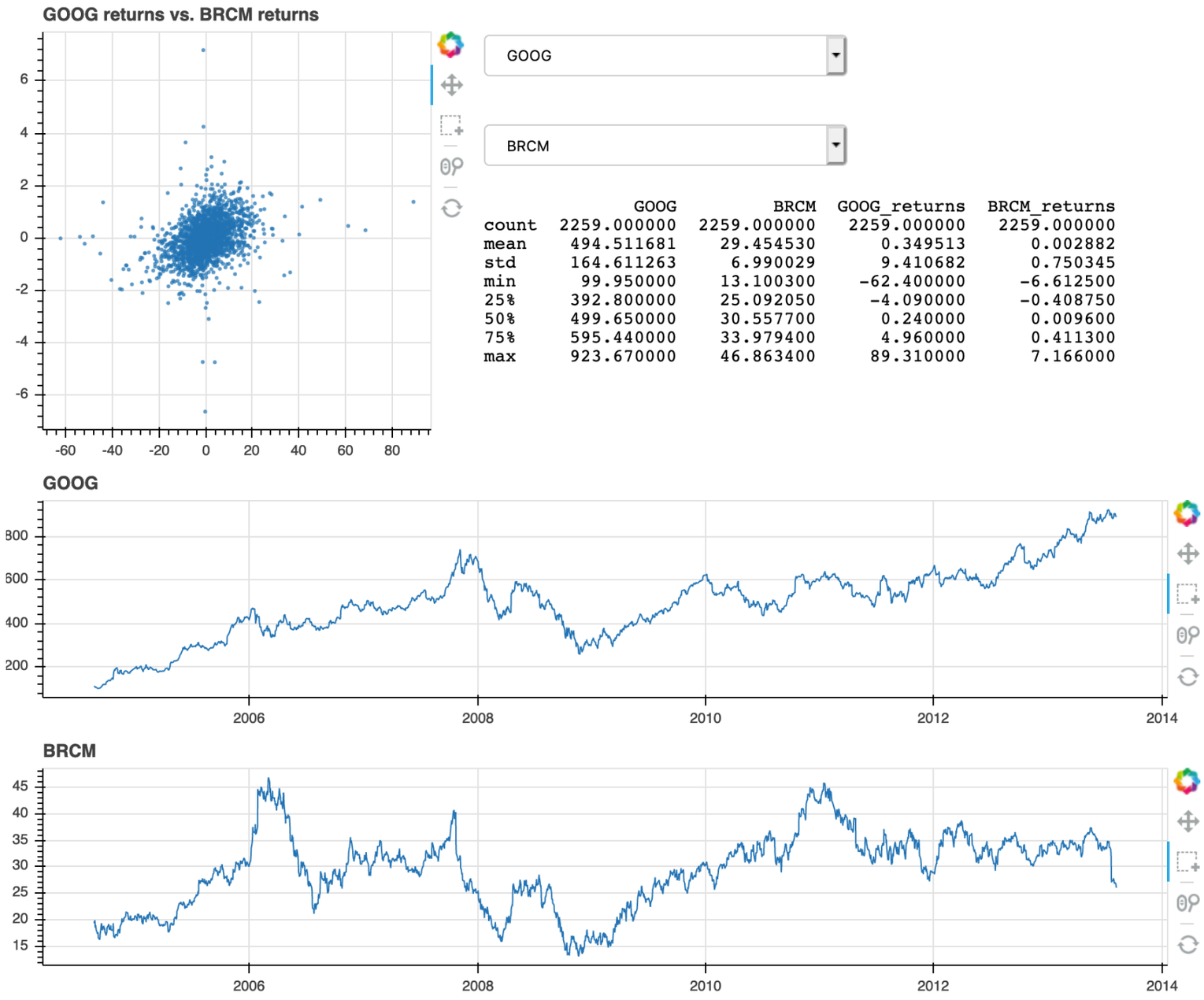
```
from bokeh.layouts import column
from bokeh.models import CustomJS, ColumnDataSource, Slider
from bokeh.plotting import figure, output_file, show

output_file("slider.html")

x = [x*0.005 for x in range(0, 200)]
y = x

source = ColumnDataSource(data=dict(x=x, y=y))

plot = figure(plot_width=400, plot_height=400)
plot.line('x', 'y', source=source, line_width=3, line_alpha=0.6)

callback = CustomJS(args=dict(source=source), code="""
        var data = source.data;
        var f = cb_obj.value
        var x = data['x']
        var y = data['y']
        for (var i = 0; i < x.length; i++) {
            y[i] = Math.pow(x[i], f)
        }
        source.change.emit();
    """)

slider = Slider(start=0.1, end=4, value=1, step=.1, title="power", callback=callback)

layout = column(slider, plot)

show(layout)
```

bokeh_slider.py

```
from bokeh.layouts import column
from bokeh.models import CustomJS, ColumnDataSource, Slider
from bokeh.plotting import figure, output_file, show

output_file("slider.html")

x = [x*0.005 for x in range(0, 200)]
y = x

source = ColumnDataSource(data=dict(x=x, y=y))

plot = figure(plot_width=400, plot_height=400)
plot.line('x', 'y', source=source, line_width=3, line_alpha=0.6)

callback = CustomJS(args=dict(source=source), code="""
        var data = source.data;
        var f = cb_obj.value
        var x = data['x']
        var y = data['y']
        for (var i = 0; i < x.length; i++) {
            y[i] = Math.pow(x[i], f)
        }
        source.change.emit();
    """)

slider = Slider(start=0.1, end=4, value=1, step=.1, title="power", callback=callback)

layout = column(slider, plot)

show(layout)
```

what about even more complicated?

cd stocks
python download_sample_data.py
bokeh serve main.py



```
# set up callbacks

def ticker1_change(attrname, old, new):
    ticker2.options = nix(new, DEFAULT_TICKERS)
    update()

def ticker2_change(attrname, old, new):
    ticker1.options = nix(new, DEFAULT_TICKERS)
    update()

def update(selected=None):
    t1, t2 = ticker1.value, ticker2.value

    data = get_data(t1, t2)
    source.data = source.from_df(data[['t1', 't2',
't1_returns', 't2_returns']])
    source_static.data = source.data
    update_stats(data, t1, t2)

    corr.title.text = '%s returns vs. %s returns' % (t1, t2)
    ts1.title.text, ts2.title.text = t1, t2

def update_stats(data, t1, t2):
    stats.text = str(data[[t1, t2, t1+'_returns',
t2+'_returns']].describe())

ticker1.on_change('value', ticker1_change)
ticker2.on_change('value', ticker2_change
```

we can set up arbitrary python to respond to user interaction!

lots of freedom, some serious drawbacks

- **custom styling**

- **WebGL acceleration**

```
p = figure(output_backend="webgl")
```

  - **only partially supported**

- **if your code starts getting complicated, think of ways to simplify!**

- or you may need to start writing js

questions?