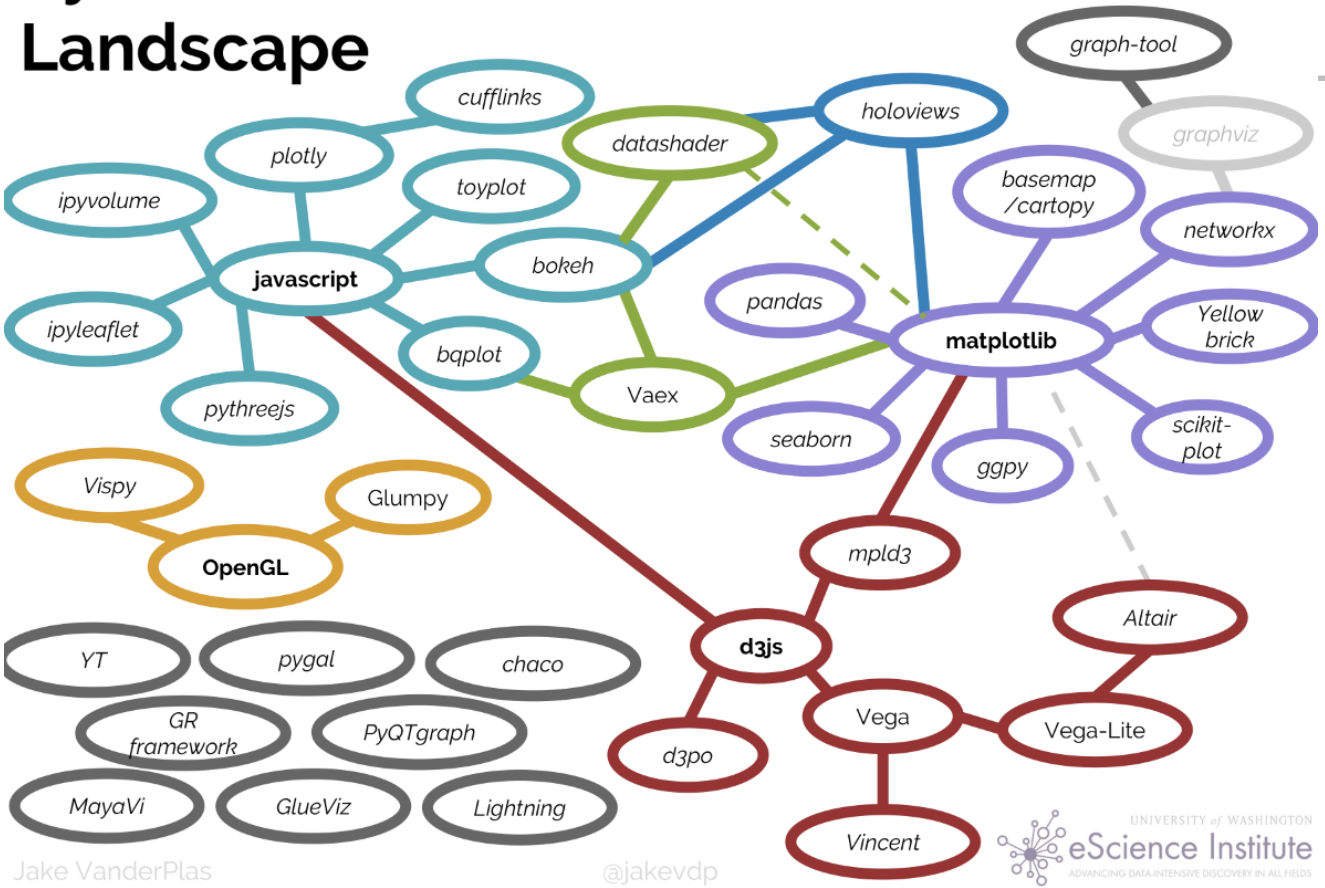
Interactive data visualization in Python

Alexey Zaytsev,

Skoltech, CDISE

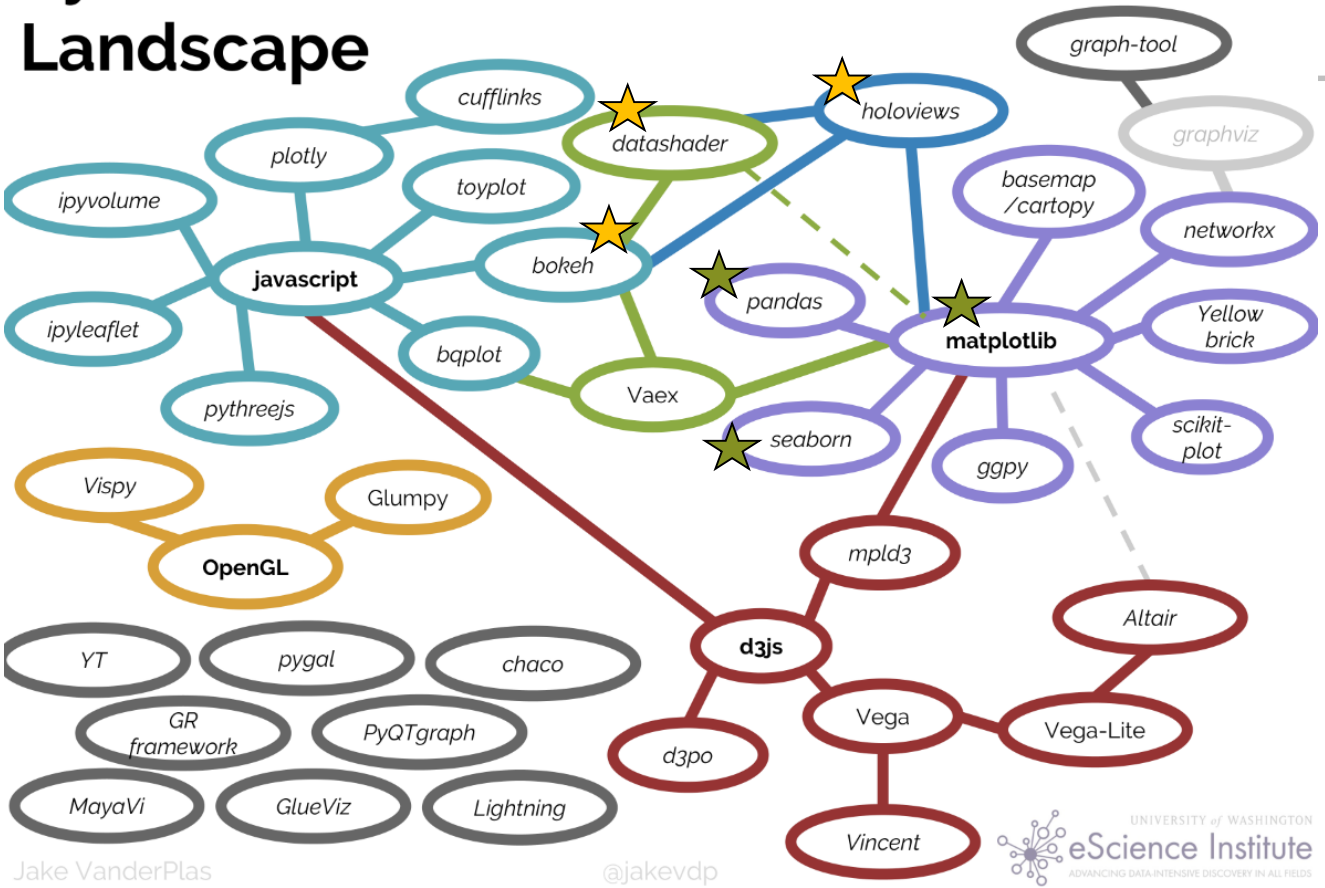
15 January

Python's Visualization Landscape



Jake VanderPlas, The Python Visualization Landscape PyCon 2017, https://www.youtube.com/watch?v=FytuB8nFHPQ

Python's Visualization Landscape



Jake VanderPlas, The Python Visualization Landscape PyCon 2017, https://www.youtube.com/watch?v=FytuB8nFHPQ

Plotting with Matplotlib

Strengths:

- Designed like Matlab: switching was easy

Plotting with Matplotlib

Strengths:

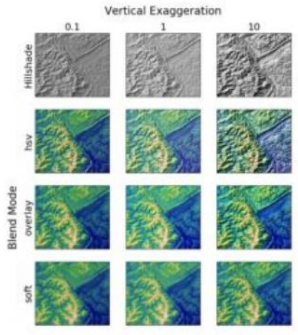
- Designed like Matlab: switching was easy
- Many rendering backends

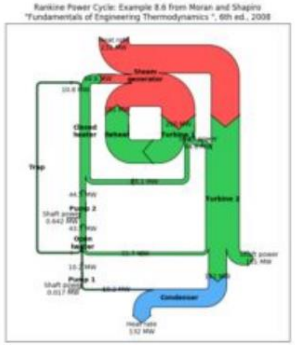
```
In [26]: from matplotlib import rcsetup
         rcsetup.all_backends

Out[26]: ['GTK',
          'GTKAgg',
          'GTKCairo',
          'MacOSX',
          'Qt4Agg',
          'Qt5Agg',
          'TkAgg',
          'WX',
          'WXAgg',
          'GTK3Cairo',
          'GTK3Agg',
          'WebAgg',
          'nbAgg',
          'agg',
          'cairo',
          'gdk',
          'pdf',
          'pgf',
          'ps',
          'svg',
          'template']
```

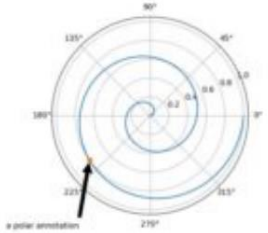
Plotting with Matplotlib

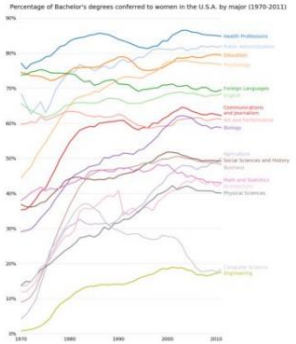
Strengths:

- Designed like Matlab: switching was easy
- Many rendering backends
- Can reproduce just about any plot (with a bit of effort)









Plotting with Matplotlib

Strengths:

- Designed like Matlab: switching was easy
- Many rendering backends
- Can reproduce just about any plot (with a bit of effort)
- Well-tested, standard tool for over a decade

It can be too complex sometimes!

Example: Iris Data

```
import pandas as pd
iris = pd.read_csv('iris.csv')
iris.head()
```

	petalLength	petalWidth	sepalLength	sepalWidth	species
0	1.4	0.2	5.1	3.5	setosa
1	1.4	0.2	4.9	3.0	setosa
2	1.3	0.2	4.7	3.2	setosa
3	1.5	0.2	4.6	3.1	setosa
4	1.4	0.2	5.0	3.6	setosa

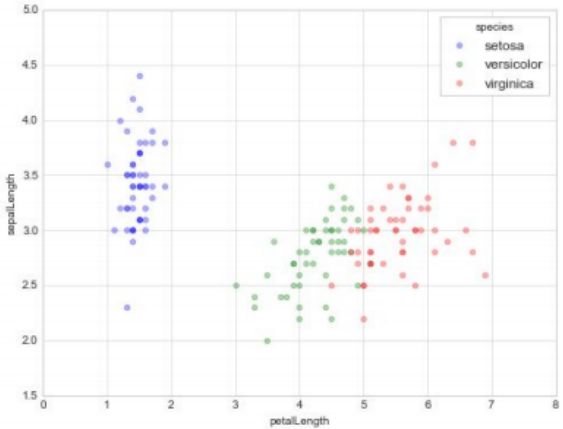
It can be too complex sometimes!

“I want to scatter petal length vs. sepal length, and color by species”

	petalLength	petalWidth	sepalLength	sepalWidth	species
0	1.4	0.2	5.1	3.5	setosa
1	1.4	0.2	4.9	3.0	setosa
2	1.3	0.2	4.7	3.2	setosa
3	1.5	0.2	4.6	3.1	setosa
4	1.4	0.2	5.0	3.6	setosa

It can be too complex sometimes!

```
color_map = dict(zip(iris.species.unique(),
                     ['blue', 'green', 'red']))

for species, group in iris.groupby('species'):
    plt.scatter(group['petalLength'], group['sepalLength'],
                color=color_map[species],
                alpha=0.3, edgecolor=None,
                label=species)

plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```



Jake VanderPlas @jakevd

Plotting with Matplotlib

Strengths:

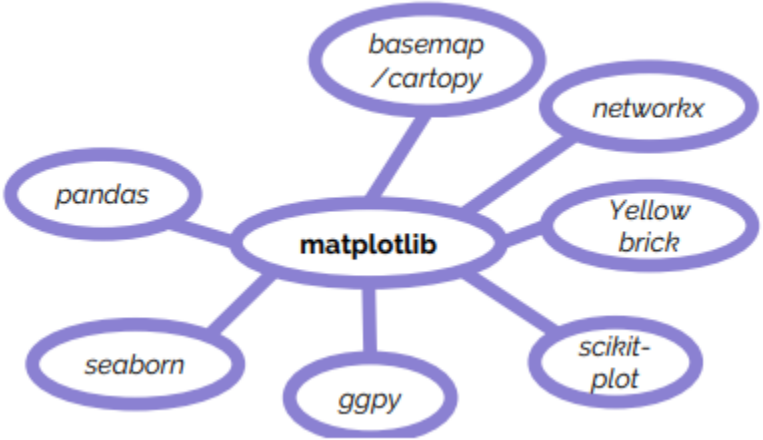
- Designed like Matlab: switching was easy
- Many rendering backends
- Can reproduce just about any plot (with a bit of effort)
- Well-tested, standard tool for over a decade

Weaknesses:

- API is imperative & often overly verbose
- Sometimes poor stylistic defaults
- Poor support for web/interactive graphs
- Often slow for large & complicated data

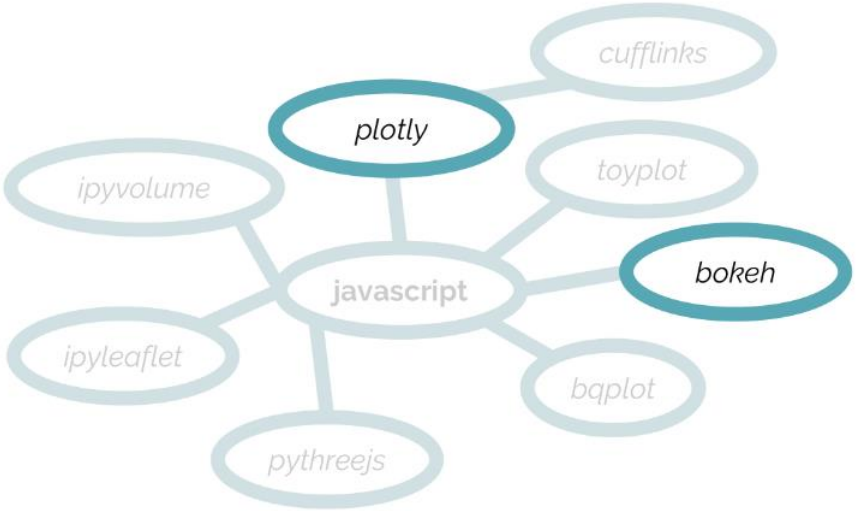
Everyone’s Goal: Improve on the weaknesses of matplotlib (without sacrificing the strengths!)

Building on Matplotlib



Common Idea: Keep matplotlib as **a versatile, well-tested backend**, and provide a new domain-specific API.

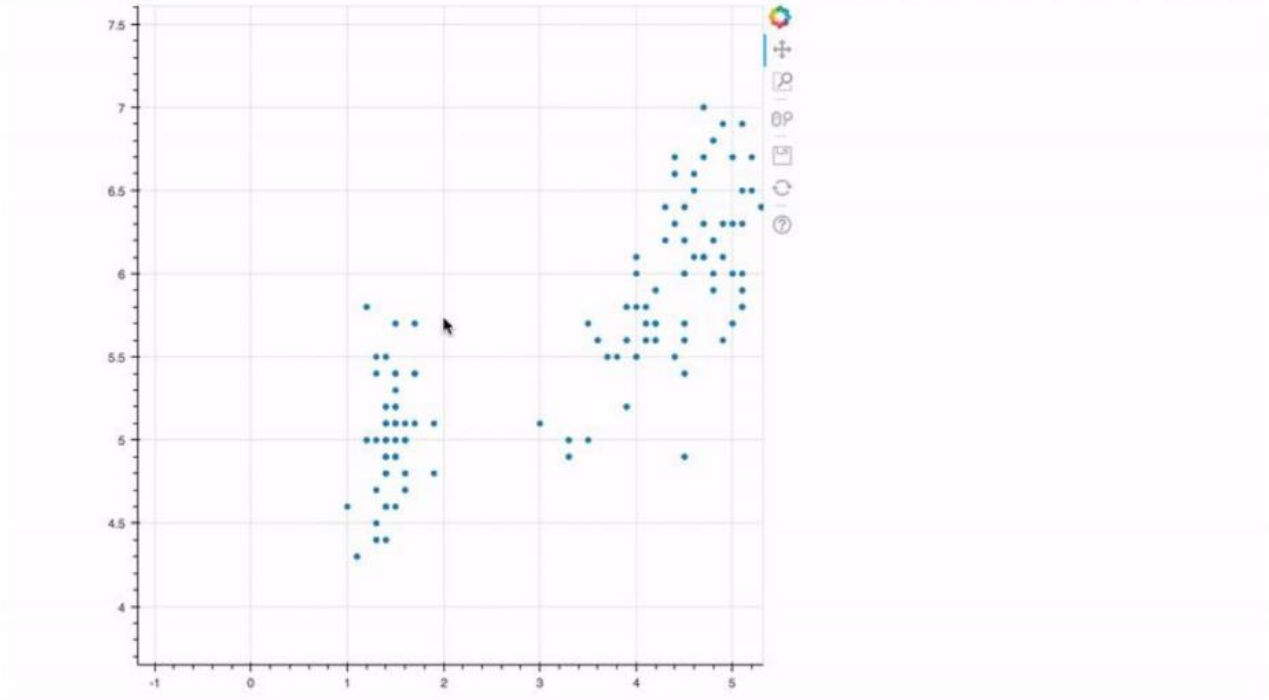
Javascript based Viz



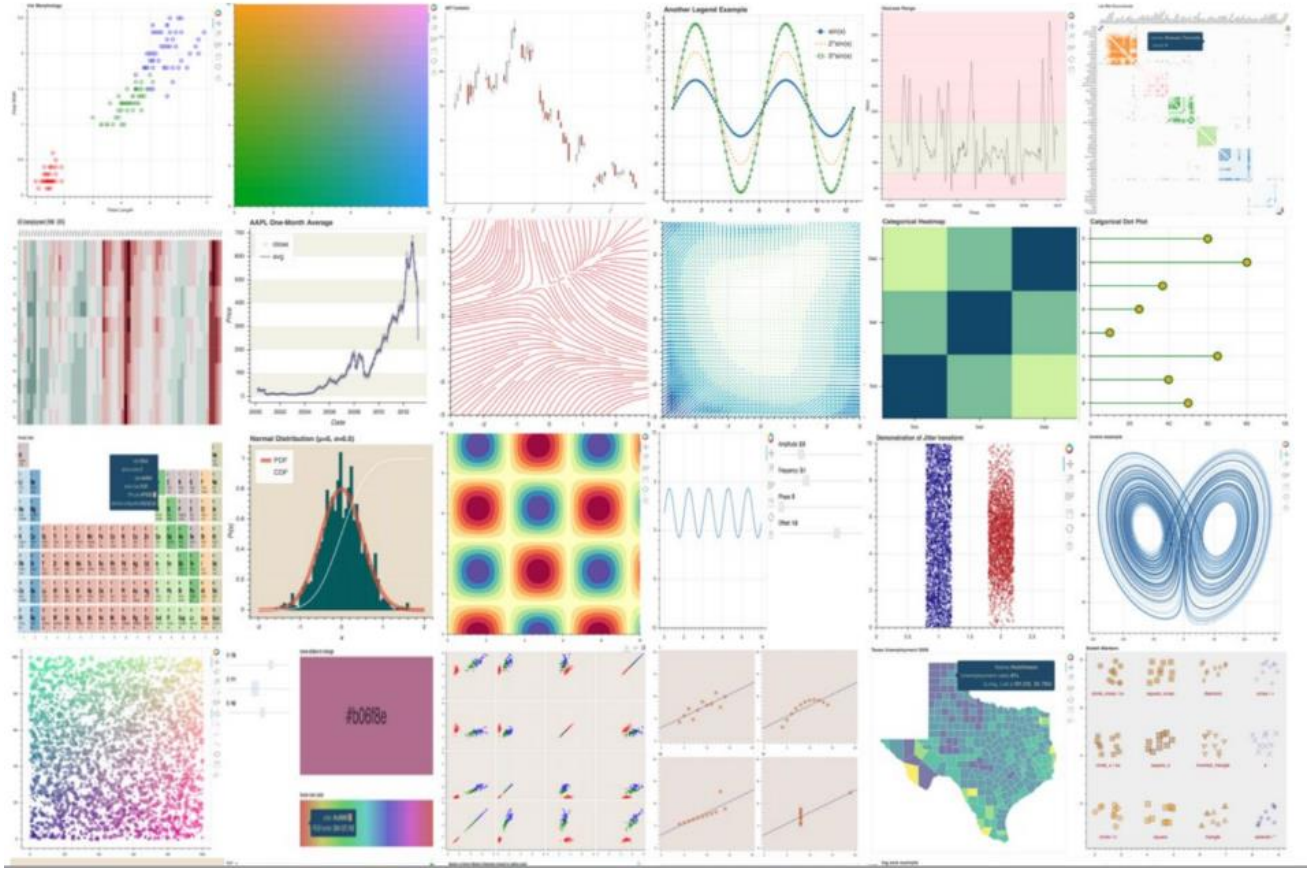
Common Idea: build a new API that produces a plot serialization (often JSON) that can be displayed in the browser (often in Jupyter notebooks)

Plotting with Bokeh

```
In [10]: p = figure()
         p.circle(iris.petalLength, iris.sepalLength)
         show(p)
```



Bokeh gallery



Skoltech
Skolkovo Institute of Science and Technology

Plotting with Bokeh

Strengths:

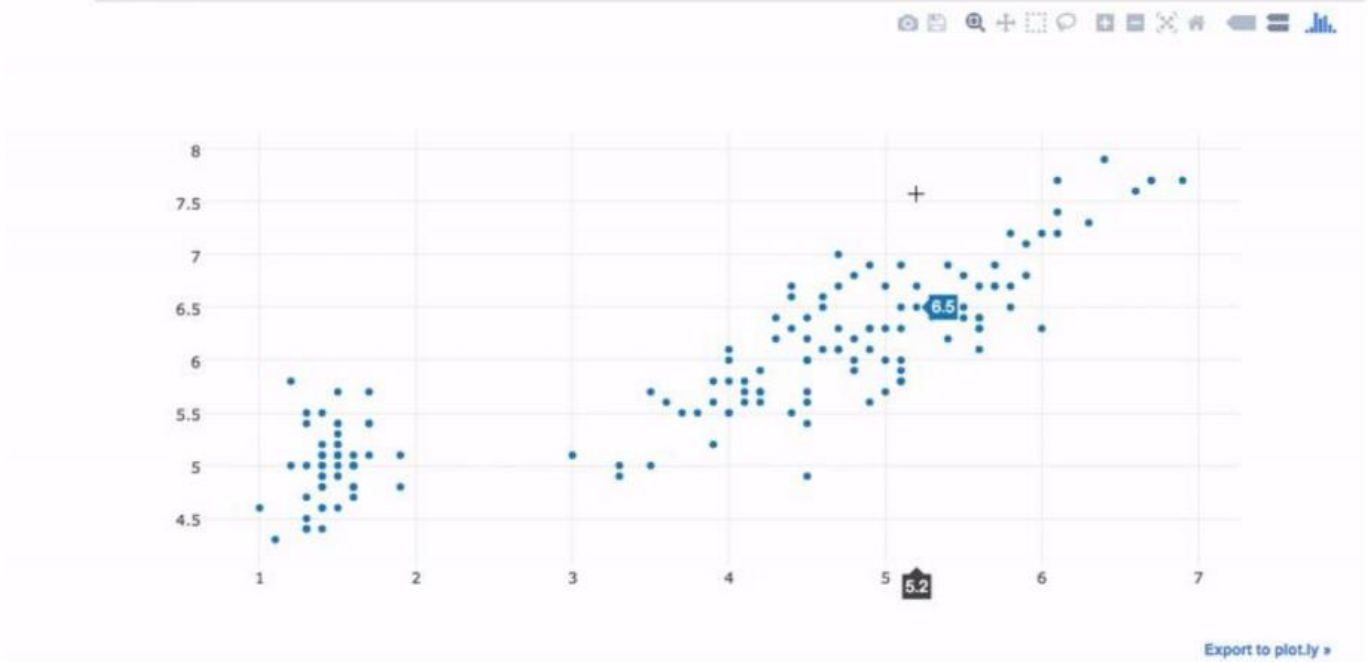
- Web view/interactivity
- Imperative and Declarative layer
- Handles large and/or streaming datasets
- Geographical visualization
- Fully open source

Weaknesses:

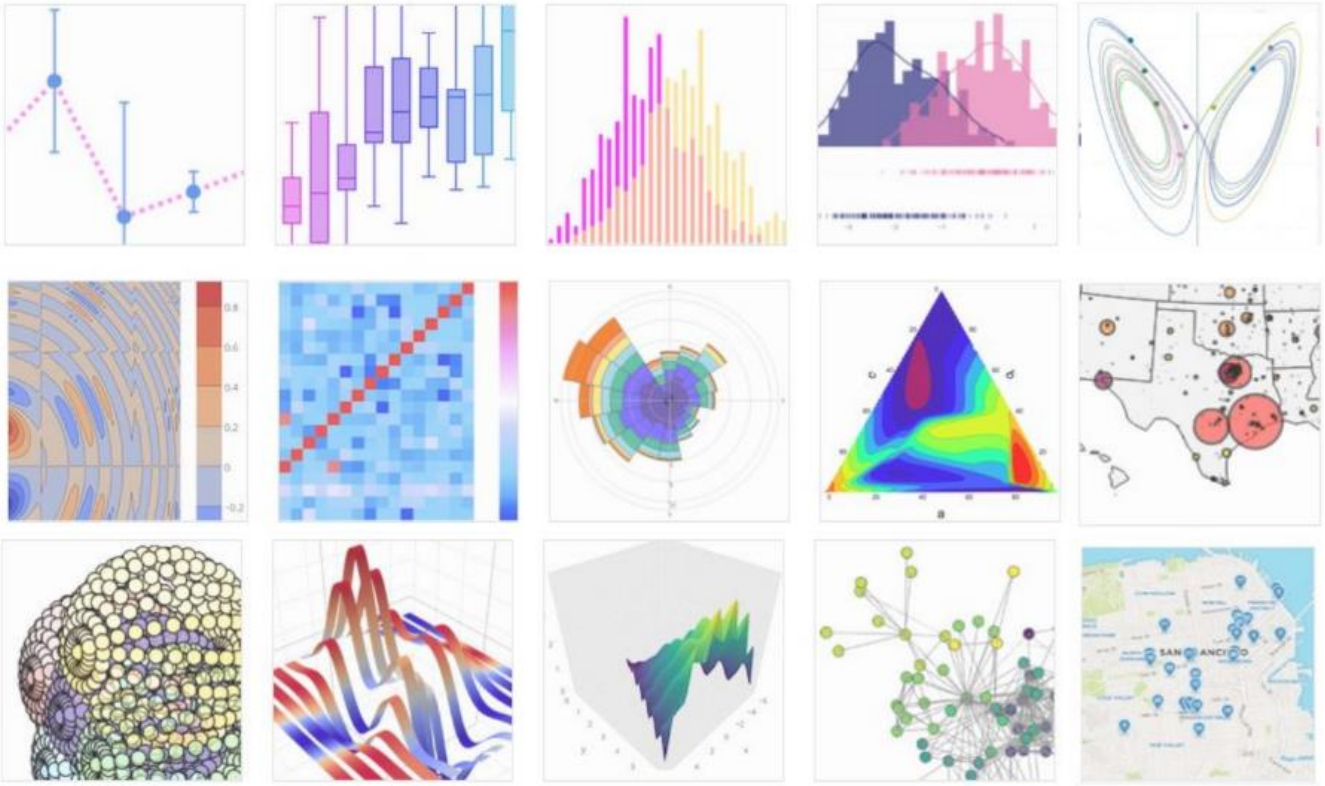
- No vector output (need PDF/EPS? Sorry)
- Newer tool with a smaller user-base than matplotlib
- Slow for large data

Plotting with Plotly

```
In [8]: from plotly.graph_objs import Scatter
        from plotly.offline import iplot

        p = Scatter(x=iris.petalLength,
                    y=iris.sepalLength,
                    mode='markers')

        iplot([p])
```



Plotly gallery

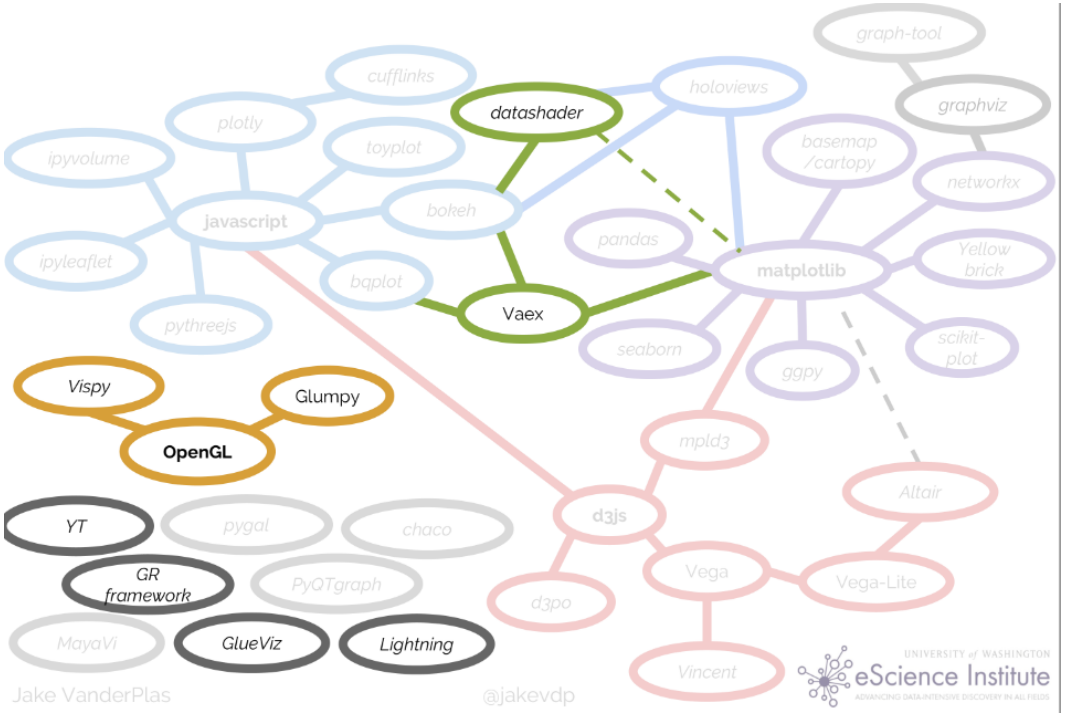


Plotting with Plotly

Strengths:
- Web view/interactivity
- Multi-language support
- 3D plotting capability
- Animation capability
- Geographical visualization

Weaknesses:
- Some features require a paid plan

Visualization of Larger data



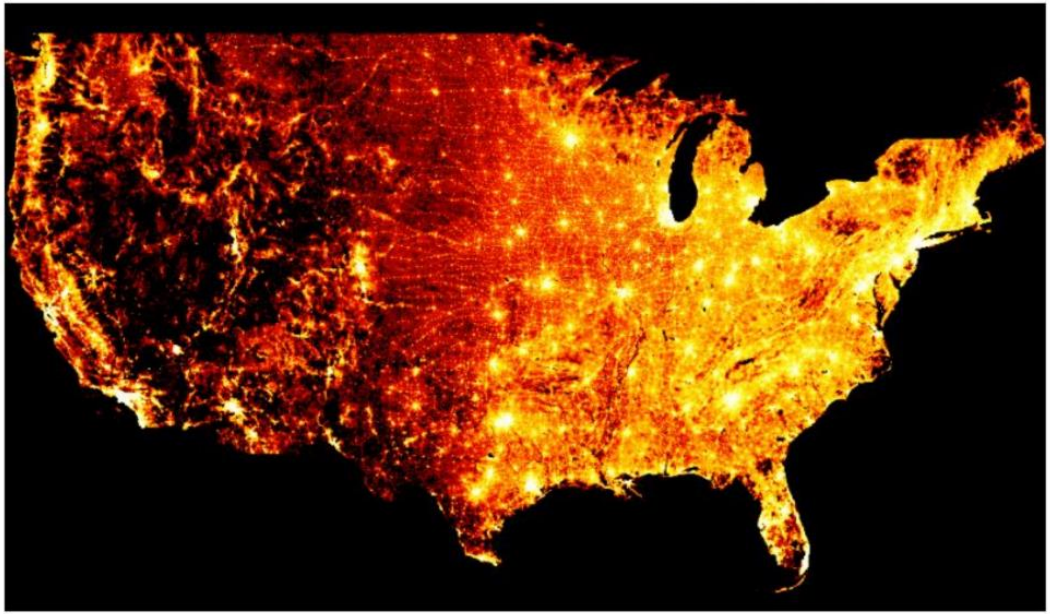
Jake VanderPlas, The Python Visualization Landscape PyCon 2017, https://www.youtube.com/watch?v=FytuB8nFHPQ

Datashader example

Fast server-side engine for dynamic data aggregation

```
In [12]: from colorcet import fire
         export(tf.shade(agg, cmap = cm(fire,0.2), how='eq_hist'),"census_ds_fire_eq_hist")
```

Out[12]:

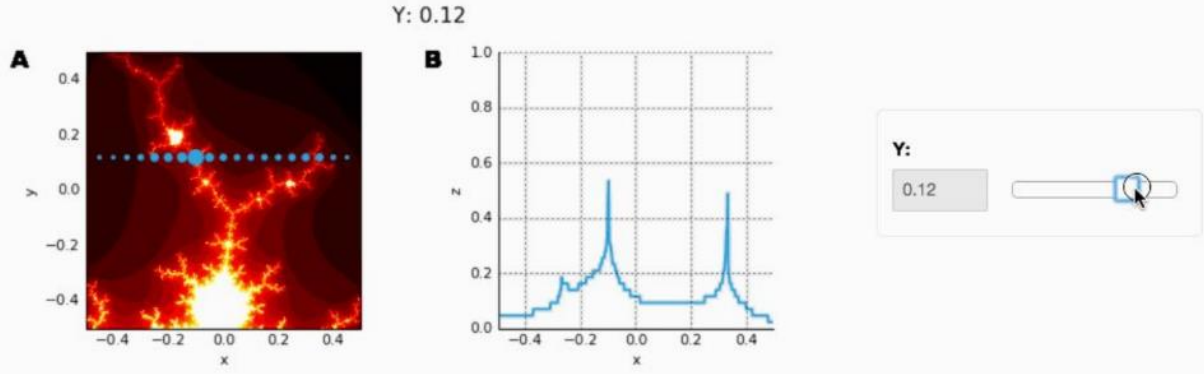


Holoviews

- Datasets themselves stored in objects that automatically produce intelligent visualizations
- Composition & Interactivity via operator overloading
- Renders to Bokeh, DataShader, and Matplotlib

```
In [3]: %%opts Points [scaling_factor=50] Contours (color='w')
        dots = np.linspace(-0.45, 0.45, 19)
        hv.HoloMap({y: (fractal * hv.Points(fractal.sample([(i,y) for i in dots])) + fractal.sample(y=y))
                    for y in np.linspace(-0.3, 0.3, 21)}, kdims=['Y']).collate().cols(2)
```

Out[3]:



Let’s have a look at Bokeh

https://hub.mybinder.org/user/bokeh-bokeh-notebooks-lr6k0kge/notebooks/tutorial