


Python Data Visualization

What now?

Hi. I'm Rob Story.

wrobstory.github.io

 github.com/wrobstory/pdxdatasci2014

 @oceankidbilly

I work @simple



We're hiring Data Engineers,
Scientists, and Analysts.

Great company. Great team.
Interesting Data.

Question:

I have data.

It's November 2014.

I want to make a chart.

What library should I use?

As with programming languages,
databases, or web frameworks,

**It depends on what
you're trying to do**

Quick Web Visualization

D3 Wrappers: NVVD3, C3, Dimple, Vega

Custom Web Visualization

"Raw" D3

Custom Python Visualization

"Raw" Matplotlib or Bokeh

Exploratory Visualization

R → ggplot, RCharts

Python → Pandas, Seaborn, ggplot, Bokeh

What about creating journal-quality figures?

Libraries built on Matplotlib and Bokeh renderers give you the flexibility to use them for data exploration and customization for publishing.

First rule of Python data vis:
Use the IPython Notebook.

Second rule of Python data vis:
Use the IPython Notebook.

IPython Notebook:

Flexible

Reproducible

Customizable

Publishable

Interactive-able

all-the-other-ibles-and-ables

The Toolset:

IPython Notebook

Pandas

Matplotlib

Seaborn

ggplot

Bokeh?

NOTEBOOK DEMO!

Q: What happens when data exploration reveals that your data is boring?

A: Make a bar chart and move on.

Statistical Visualization

With Seaborn and the ggplot port, we now
have a great toolset

DEMO!

Q: Is it more likely to snow on Mt.
Hood in December or March?

What now?

Q: How should developers interested in building Python data vis libraries proceed? Why didn't you bring up Vincent.py, the thing you made?

***We should be building on
common visualization
“kernels”.***

Building rendering engines is a lot of
work.

High Level Tool -> kernel

Bokeh —> Bokeh (Bokeh.js)

Seaborn —> Matplotlib/Bokeh

ggplot —> Matplotlib/Bokeh

Vincent —> Vega —> D3

The first three can leverage everything that the Bokeh and Matplotlib team are working on. Publishing to different formats, rendering widgets, etc.

Let's build better abstractions around common toolsets

1. Build on kernel “primitives” (Bokeh glyphs, Matplotlib line/bar, etc)
- 2. Document your data interfaces**

That one is worth repeating:

***DOCUMENT YOUR DATA
INTERFACES!***

If users struggle to get data into a chart, either your API is broken, your docs are broken, or **both** are broken.

Ingest common data formats

lists, dicts, numpy arrays, Pandas
DataFrames & Series

Document the shape of those formats

Should my data be “long” or “wide”? Should
I have a list of dicts, or a dict of lists? Does
my DataFrame need a specific structure?

Q: Should I just contribute to MPL/Bokeh core?

Bokeh: Yes!

Bokeh has high-level Chart interfaces built on their glyphs. I'm sure they would appreciate PRs building on those interfaces.

MPL: ...maybe!

Seaborn has shown that a lib built on MPL with a focused API can work **really** well. Model for other libraries?

What does the Future look like?

Bokeh

Write once, render everywhere

Out of core “big data” visualization

Crossfilter-like linked brushing

but...

Can't quite recommend it yet for everyday analysis.

Moving very fast, documentation hard to follow,
data input inconsistent. It's still a young project.

DEMO!

Last Thing: is there a place for D3 in the IPy Notebook?

Yes. I think so.

One topic that wasn't covered at all here was interactivity outside of IPython widgets. Data transitions, etc. Bokeh is working on it, but D3 already has many libs with these features.

Sticky.py? Maybe one day...

FIN!

THANK YOU!