Python’s Visualization Landscape

Jake VanderPlas
@jakevdp #PyCon2017
[Python’s Visualization Landscape]

From the abstract:
“In this talk I’ll give an overview of the landscape of dataviz tools in Python . . .”
[Python’s Visualization Landscape]

From the abstract:
“In this talk I’ll give an overview of the landscape of dataviz tools in Python . . .”
From the abstract:

"In this talk I’ll give an overview of the landscape of dataviz tools in Python . . ."
From the abstract:

"In this talk I’ll give an overview of the Python’s Visualization Landscape..."
From the abstract:
"In this talk I'll give an overview of the landscape of dataviz tools in Python . . ."
From the abstract:

"In this talk I'll give an overview of the landscape of dataviz tools in Python . . ."

[Python's Visualization Landscape]
From the abstract:
"In this talk I'll give an overview of the landscape of dataviz tools in Python..."
In this talk I'll give an overview of the landscape of dataviz tools in Python.
From the abstract:
"In this talk I'll give an overview of the landscape of dataviz tools in Python..."
[Making Sense of the Deluge]
matplotlib
matplotlib

seaborn

ggpy

networkx

Yellow brick

basemap/cartopy

scikit-plot
plotly

javascript

bokeh

basemap / cartopy
networkx
Yellow brick
scikit-plot
pandas
tplotlib
seaborn
ggpy
Python’s Visualization Landscape
How did we get here?
In the beginning was matplotlib*

* well, actually… Python visualization existed before matplotlib, but was not very mature.
Plotting with Matplotlib

Strengths:
- Designed like MatLab: switching was easy
Plotting with Matplotlib

Strengths:
- Designed like MatLab: switching was easy
- Many rendering backends

```python
In [26]: from matplotlib import rc
rc().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
Matplotlib Gallery
Example: Statistical Data

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

<table>
<thead>
<tr>
<th>petalLength</th>
<th>petalWidth</th>
<th>sepalLength</th>
<th>sepalWidth</th>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.4</td>
<td>5.1</td>
<td>3.5</td>
<td>setosa</td>
</tr>
<tr>
<td>1</td>
<td>1.4</td>
<td>4.9</td>
<td>3.0</td>
<td>setosa</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>4.7</td>
<td>3.2</td>
<td>setosa</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>4.6</td>
<td>3.1</td>
<td>setosa</td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td>5.0</td>
<td>3.6</td>
<td>setosa</td>
</tr>
</tbody>
</table>

Tidy data: i.e. rows are samples, columns are features
Just a simple visualization . . .

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

<table>
<thead>
<tr>
<th>petalLength</th>
<th>petalWidth</th>
<th>sepalLength</th>
<th>sepalWidth</th>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.4</td>
<td>0.2</td>
<td>5.1</td>
<td>3.5</td>
</tr>
<tr>
<td>1</td>
<td>1.4</td>
<td>0.2</td>
<td>4.9</td>
<td>3.0</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>0.2</td>
<td>4.7</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>0.2</td>
<td>4.6</td>
<td>3.1</td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td>0.2</td>
<td>5.0</td>
<td>3.6</td>
</tr>
</tbody>
</table>
Just a simple visualization . . .

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')
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...
Building on Matplotlib...

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

- Pandas provides a DataFrame object
- Also provides a simple API for plotting DataFrames
```python
iris.plot.scatter('petalLength', 'petalWidth')
```
- More sophisticated statistical visualization tools have recently been added

```python
from pandas.tools.plotting import andrews_curves
andrews_curves(iris, 'species')
```
Seaborn: statistical data visualization

Key Features:
- Like Pandas, wraps matplotlib
- Nice set of color palettes & plot styles
- Focus on statistical visualization & modeling

http://seaborn.pydata.org
import seaborn as sns
sns.lmplot('petalLength', 'sepalWidth', iris,
           hue='species', fit_reg=False)
Seaborn examples

```python
sns.pairplot(iris, hue='species')
```
Javascript-based Viz:

- plotly
- cufflinks
- toyplot
- bokeh
- bqplot
- ipyvolume
- ipyleaflet
- pythreejs
- cufflinks
- ipyleaflet
- pythreejs
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)
Javascript-based Viz:
Plotting with Bokeh

```python
In [10]:
p = figure()
p.circle(iris.petalLength, iris.sepalLength)
show(p)
```
Bokeh Gallery
Plotting with Bokeh

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

Disadvantages:
- No vector output (need PDF/EPS? Sorry)
- Newer tool with a smaller user-base than matplotlib
Basic Plotting with Plotly

```python
import scatter = plotly.graph_objs import Scatter
from plotly.offline iplot

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

iplot([p])
```
Plotly Gallery
Plotting with Plotly

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

Disadvantages:
- Some features require a paid plan
Visualization for Larger Data
Visualization for Larger Data . . .
Datashader

Fast server-side engine for dynamic data aggregation
Datashader

- Compute layer that works with Bokeh
- Rather than sending data to the client, it aggregates data and sends pixels.
- Can handle interactive visualization of billions of rows.

```
In [19]: export(create_image(*LakeMichigan),"Zoom 1 - Lake Michigan")
Out[19]:
```
Datashader

- Compute layer that works with Bokeh
- Rather than sending *data* to the client, it aggregates data and sends *pixels*.
- Can handle interactive visualization of billions of rows.

In [20]: export(create_image(*Chicago),"Zoom 2 - Chicago")
Out[20]:
Toward Declarative Visualization . . .
Toward Declarative Visualization . . .
Holoviews

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

```python
In [3]: plot_opt = dict(scaling_factor=50)  
contours = hv.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]:
```

![Image of visualizations showing two graphs: one with a complex fractal pattern and the other with a histogram.](image)

A: Fractal visualization with dots distributed across the x-axis.

B: Histogram showing data distribution with a slider interface.

Y: 0.12
Holoviews

- Also can handle geographic data & time-series
Altair

What if instead of passing around *pixels*, we pass around *visualization specifications* plus *data*?
Altair

What if instead of passing around pixels, we pass around visualization specifications plus data?

“Declarative Visualization”
Altair

What if instead of passing around pixels, we pass around visualization specifications plus data?

“Declarative Visualization”
**Declarative Visualization:**
Viz for data science

**Imperative**
- Specify *How* something should be done.
- Must manually specify plotting steps
- Specification & Execution intertwined.

**Declarative**
- Specify *What* should be done
- Details determined automatically
- Separates Specification from Execution

Declarative visualization lets you think about **data** and **relationships**, rather than incidental details.
From D3 to Altair . . .

Drought and Deluge in the Lower 48

Last summer’s drought, one of the worst in a century, has continued through the winter. This chart shows the proportion of what is now the contiguous U.S. in various stages of drought over 118 years of record-keeping. Roll mouse over individual months to see what percentage of the lower 48 was in drought. Related Article »

During January, 56 percent of the contiguous U.S. was in moderate to extreme drought, the highest January level since 1955.

2010 Jul 2011 2012 2013

Dryness based on the Palmer Drought Index

2000

Large, intense regional droughts that migrated across the country. The West saw persistent drought in the first half of the decade.


The drought of 1988 destroyed at least half the crops on the Great Plains and was one of the costliest U.S. natural disasters.


(link to live version)

#JSM2016
But working in D3 can be challenging . . .
Bar Chart: d3

D3 is a Javascript package that streamlines manipulation of objects on a webpage.

var margin = {top: 20, right: 20, bottom: 30, left: 40},
    width = 960 - margin.left - margin.right,
    height = 500 - margin.top - margin.bottom;

var x = d3.scale.ordinal()
    .rangeRoundBands([0, width], .1);

var y = d3.scale.linear()
    .range([height, 0]);

var xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom");

var yAxis = d3.svg.axis()
    .scale(y)
    .orient("left")
    .ticks(10, "%");

var svg = d3.select("body").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", "translate(0,0)"");

d3.tsv("data.tsv", type,
    function(error, data) {
      if (error) throw error;
      x.domain(data.map(function(d) { return d.letter; }));
      y.domain([0, d3.max(data, function(d) { return d.frequency; })]);

      svg.append("g")
        .attr("class", "x axis")
        .call(xAxis);

      svg.append("g")
        .attr("class", "y axis")
        .call(yAxis)
        .append("text")
        .attr("transform", "rotate(-90)"
            .attr("y", 6)
            .attr("dy", ".71em")
            .style("text-anchor", "end")
            .text("Frequency");

      svg.selectAll(".bar")
        .data(data)
        .enter().append("rect")
        .attr("class", "bar")
        .attr("x", function(d) { return x(d.letter); })
        .attr("width", x.rangeBand())
        .attr("y", function(d) { return y(d.frequency); })
        .attr("height", function(d) { return height - y(d.frequency); });

    });

function type(d) {
    d.frequency = +d.frequency;
    return d;
}
Vega is a detailed declarative specification for visualizations, built on D3.
Bar Chart: Vega-Lite

Vega-Lite is a simpler declarative specification aimed at statistical visualization.
Altair is a Python API for creating Vega-Lite specifications.
From Declarative API to declarative Grammar

```python
chart = Chart(data).mark_circle(
    opacity=0.3
).encode(
    x='petalLength:Q',
    y='sepalWidth:Q',
    color='species:N',
)
chart.display()
```
From Declarative API to declarative Grammar

```python
>>> chart.to_dict()

{'config': {'mark': {'opacity': 0.3}},
 'data':
  {'url': 'https://vega.github.io/vega-datasets/data/iris.json'},
 'encoding': {'color': {'field': 'species', 'type': 'nominal'},
               'x': {'field': 'petalLength', 'type': 'quantitative'},
               'y': {'field': 'sepalWidth', 'type': 'quantitative'}},
 'mark': 'circle'
```
(Visualizations from jakevdp/altair-examples).
Coming Very Soon: Altair 2.0
- Includes a Grammar of Interaction
Try Altair:

$ conda install altair --channel conda-forge

or

$ pip install altair
$ jupyter nbextension install --sys-prefix --py vega

For a Jupyter notebook tutorial, type

import altair
altair.tutorial()

http://github.com/ellisonbg/altair/
Python’s Visualization Landscape

- matplotlib
- seaborn
- pandas
- ggpy
- scikit-plot
- Yellowbrick
- networkx
- basemap/cartopy
- graph-tool
- graphviz
- pythreejs
- bqplot
- bokeh
- cufflinks
- plotly
- toyplot
- datashader
- ipyvolume
- ipyleaflet
- pythreexjs
- javascript
- Glumpy
- OpenGL
- Vispy
- GlumPy
- OpenGl
- pygal
- chaco
- pyqtgraph
- MayaVi
- MayaVi
- Lightning
- YT
- GR framework
- Vega
- Vega
- Vega-Lite
- Alair
- d3po
- d3js
- Vincent
- MayaVi
- GlueViz
- Lightning
- University of Washington
- eScience Institute
- ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

Jake VanderPlas
@jakevdp
Thank You!

Email: jakevdp@uw.edu
Twitter: @jakevdp
Github: jakevdp
Web: http://vanderplas.com/
Blog: http://jakevdp.github.io/