Data Visualization in Python

Marc Garcia - @datapythonista

Data Visualisation Summit - London, 2017

About me

http://datapythonista.github.io

Python for data science

Why Python?

Python is the favorite of many:

- Fast to write: **Batteries included**
- Easy to read: **Readability** is KEY
- Excellent **community**: Conferences, local groups, stackoverflow...
- **Ubiquitous**: Present in all major platforms
- Easy to **integrate**: Implements main protocols and formats
- Easy to **extend**: C extensions for low-level operations

Python performance

Is Python fast for data science?

- Short answer: **No**
- Long answer: **Yes**
  - numpy
  - Cython
  - C extensions
  - Numba
  - etc.

Python is great for data science

A whole **ecosystem** exists:

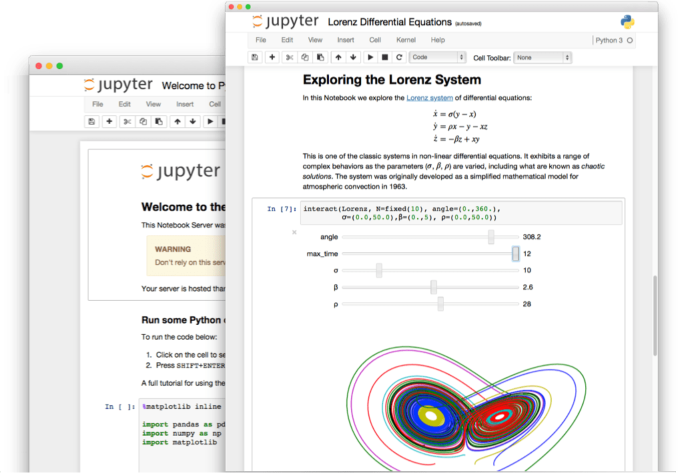
- numpy
- scipy
- pandas
- statsmodels
- scikit-learn
- etc.

Python environment

One ring to rule them all:

Python platform

Jupyter notebook



jupyter Welcome to P
File Edit View Insert Cell
jupyter

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this ser

Your server is hosted tha

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3 ○
Code
Cell Toolbar: None

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

angle 308.2

max_time 12

σ 10

β 2.6

ρ 28

Python for visualization

Main libraries:

- Matplotlib
  - Seaborn
- Bokeh
  - HoloViews
  - Datashader
- Domain-specific
  - Folium: maps
  - yt: volumetric data

Visualization tools

Matplotlib

- **First** Python visualization tool
- Still a de-facto **standard**
- Replicates **Matlab** API
- Supports many backends

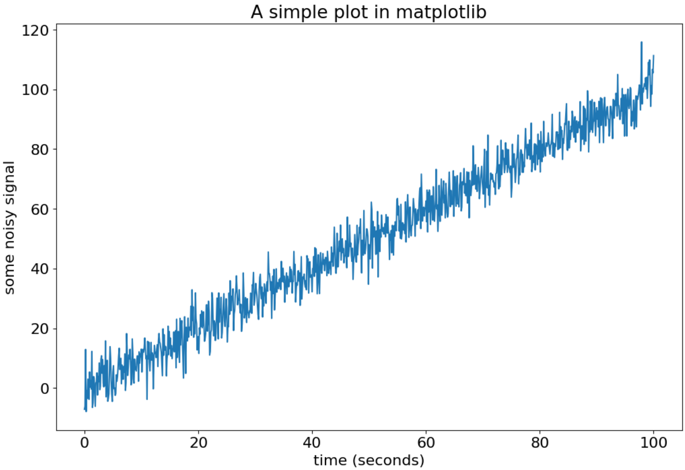
Matplotlib

```
import numpy
from matplotlib import pyplot

x = numpy.linspace(0., 100., 1001)
y = x + numpy.random.randn(1001) * 5

pyplot.plot(x, y)
pyplot.xlabel(’time (seconds)’)
pyplot.ylabel(’some noisy signal’)
pyplot.title(’A simple plot in matplotlib’)
```

Matplotlib


A simple plot in matplotlib
some noisy signal
time (seconds)
120
100
80
60
40
20
0
0
20
40
60
80
100

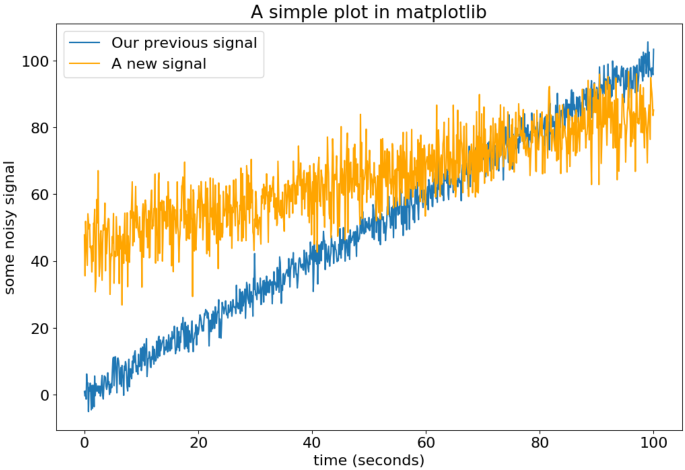
Matplotlib

```python
import numpy
from matplotlib import pyplot

x = numpy.linspace(0., 100., 1001)
y1 = x + numpy.random.randn(1001) * 3
y2 = 45 + x * .4 + numpy.random.randn(1001) * 7

pyplot.plot(x, y1, label='Our previous signal')
pyplot.plot(x, y2, color='orange', label='A new signal')
pyplot.xlabel('time (seconds)')
pyplot.ylabel('some noisy signal')
pyplot.title('A simple plot in matplotlib')
pyplot.legend()
```

Matplotlib


A simple plot in matplotlib
Our previous signal
A new signal
some noisy signal
100
80
60
40
20
0
0
20
40
60
80
100
time (seconds)

Seaborn

- Matplotlib **wrapper**
- Built-in **themes**
- Higher level plots:
  - Heatmap
  - Violin plot
  - Pair plot

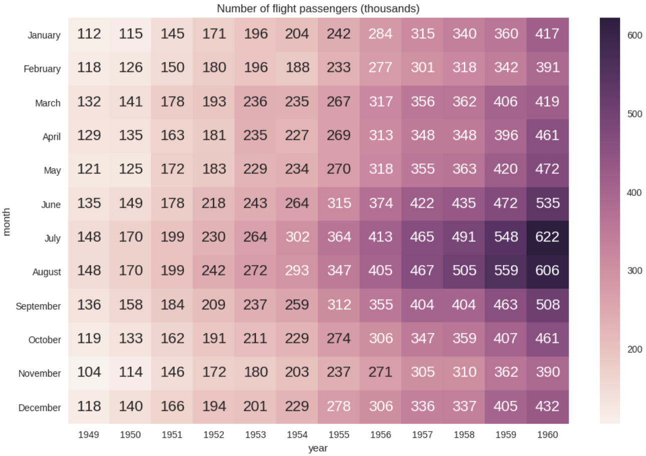
Seaborn

```
from matplotlib import pyplot
import seaborn

flights_flat = seaborn.load_dataset(’flights’)
flights = flights_flat.pivot(’month’, ’year’, ’passengers’)

seaborn.heatmap(flights, annot=True, fmt=’d’)
pyplot.title(’Number of flight passengers (thousands)’)
```

Seaborn


Number of flight passengers (thousands)

month	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
January	112	115	145	171	196	204	242	284	315	340	360	417
February	118	126	150	180	196	188	233	277	301	318	342	391
March	132	141	178	193	236	235	267	317	356	362	406	419
April	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
June	135	149	178	218	243	264	315	374	422	435	472	535
July	148	170	199	230	264	302	364	413	465	491	548	622
August	148	170	199	242	272	293	347	405	467	505	559	606
September	136	158	184	209	237	259	312	355	404	404	463	508
October	119	133	162	191	211	229	274	306	347	359	407	461
November	104	114	146	172	180	203	237	271	305	310	362	390
December	118	140	166	194	201	229	278	306	336	337	405	432

Bokeh

- **Client-server** architecture: JavaScript front-end
- Interactive
- Drawing **shapes** to generate plots

Bokeh

Demo

HoloViews

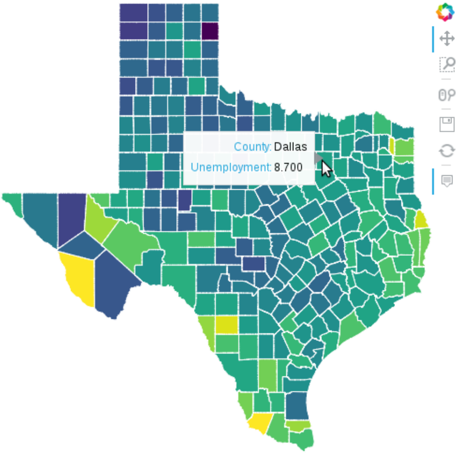
- Bokeh **wrapper**
- **Higher level** plots
- Mainly for Bokeh, but other backends supported

HoloViews

```
import numpy as np
import holoviews as hv
from bokeh.sampledata.us_counties import data as counties
from bokeh.sampledata.unemployment import data as unemployment

hv.extension('bokeh')
counties = {code: county for code, county in counties.items() if county['state'] == 'tx'}
county_xs = [county['lons'] for county in counties.values()]
county_ys = [county['lats'] for county in counties.values()]
county_names = [county['name'] for county in counties.values()]
county_rates = [unemployment[county_id] for county_id in counties]
county_polys = {name: hv.Polygons((xs, ys), level=rate, vdims=['Unemployment'])
                for name, xs, ys, rate in zip(county_names, county_xs, county_ys,
      county_rates)}
choropleth = hv.NdOverlay(county_polys, kdims=['County'])
plot_opts = dict(logz=True, tools=['hover'], xaxis=None, yaxis=None,
                 show_grid=False, show_frame=False, width=500, height=500)
style = dict(line_color='white')
choropleth({'Polygons': {'style': style, 'plot': plot_opts}})
```

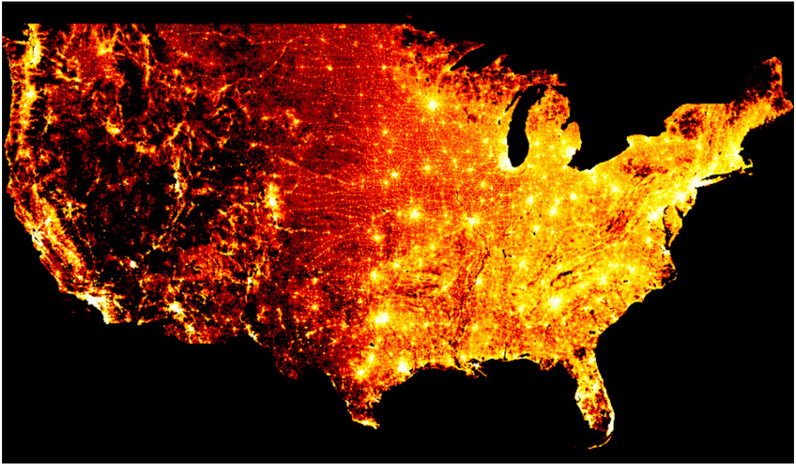
HoloViews



Datashader

- Bokeh **wrapper**
- Built for **big data**
- Advanced **subsampling** and **binning** techniques

Datashader



Folium

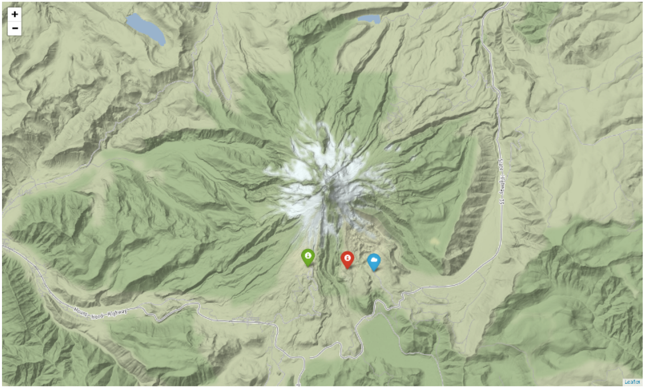
- Visualization of maps
- Compatible with **Google maps** and **Open street maps**
- Visualization of **markers**, **paths** and **polygons**

Folium

```
import folium

m = folium.Map(location=[45.372, -121.6972],
               zoom_start=12,
               tiles=’Stamen Terrain’)
folium.Marker(location=[45.3288, -121.6625],
              popup=’Mt. Hood Meadows’,
              icon=folium.Icon(icon=’cloud’)).add_to(m)
folium.Marker(location=[45.3311, -121.7113],
              popup=’Timberline Lodge’,
              icon=folium.Icon(color=’green’)).add_to(m)
folium.Marker(location=[45.3300, -121.6823],
              popup=’Some Other Location’,
              icon=folium.Icon(color=’red’, icon=’info-sign’)).add_to(m)
m
```

Folium



yt

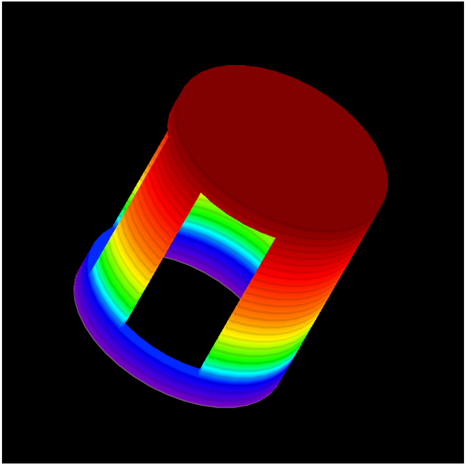
- Visualization of volumetric data
- Compatible with many formats
- Projects **multidimensional** data to a 2-D plane

yt

```
import yt

ds = yt.load(’MOOSE_sample_data/out.e-s010’)
sc = yt.create_scene(ds)
ms = sc.get_source()
ms.cmap = ’Eos A’
cam = sc.camera
cam.focus = ds.arr([0.0, 0.0, 0.0], ’code_length’)
cam_pos = ds.arr([-3.0, 3.0, -3.0], ’code_length’)
north_vector = ds.arr([0.0, -1.0, -1.0], ’dimensionless’)
cam.set_position(cam_pos, north_vector)
cam.resolution = (800, 800)
sc.save()
```

yt

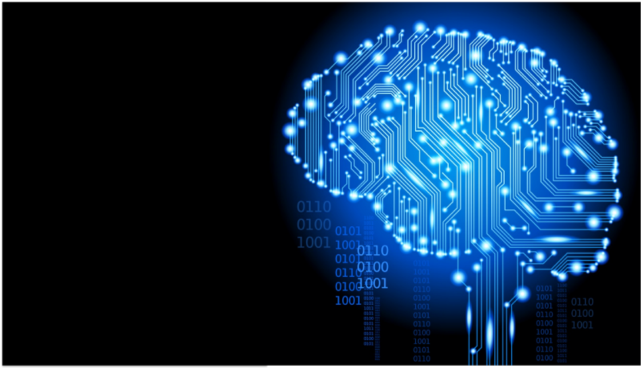


Conclusions

Conclusions

- Python is great as a **programming language**
- And is great for **data science**
- Plenty of **options** for visualization:
    - Standard plots
    - Ad-hoc plots
    - Interactive
    - 3D plots
    - Maps
    - Big data
    - Specialized

Questions?



@datapythonista