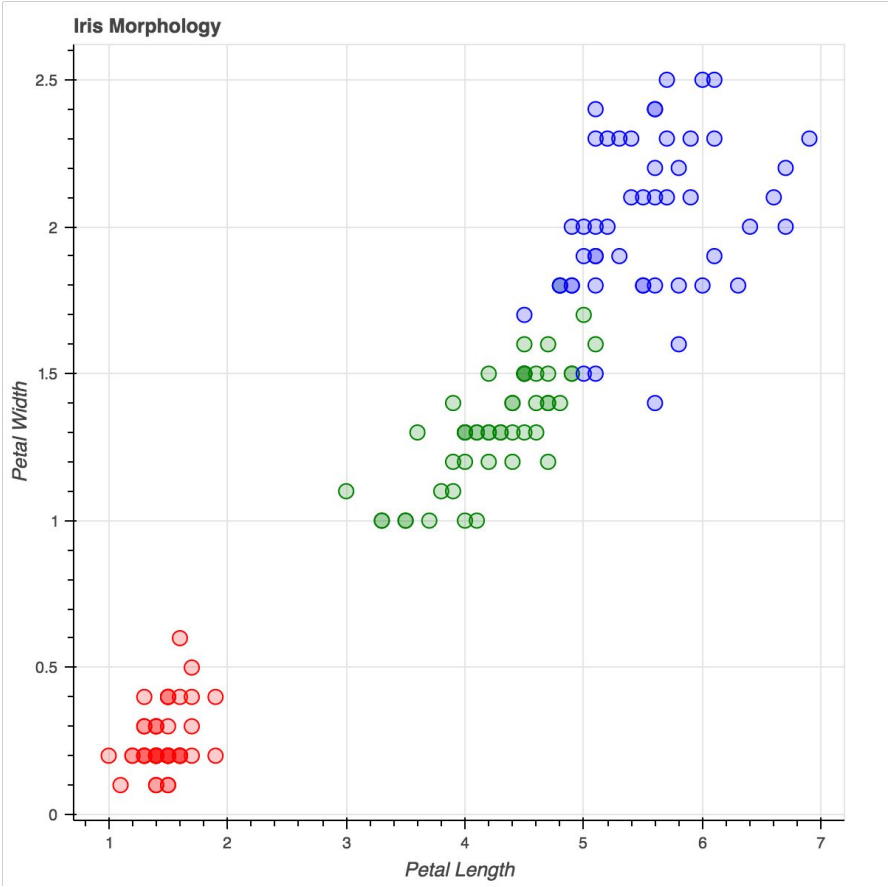
Data Visualization in Python

Matplotlib vs. Seaborn vs. Bokeh vs. Plotly

Why visualize?

- Because words are boring (ex: this slide)
- Reduces time to understanding (if used correctly)
- Portability and ease of sharing



But how?

- There are **TONS** of visualization libraries in Python alone. (1394 unique packages alone on github)
- What do we want out of a visualization tool?
  - Easy
  - Fast
  - Flexible

Top 5 Python Visualization Libraries*:

- Matplotlib: 10,901,586
- Plotly: 2,954,343
- Seaborn: 2,552,274
- Bokeh: 1,321,885
- GGplot: 730,180

* By downloads from PyPI since 08/2016

Criteria

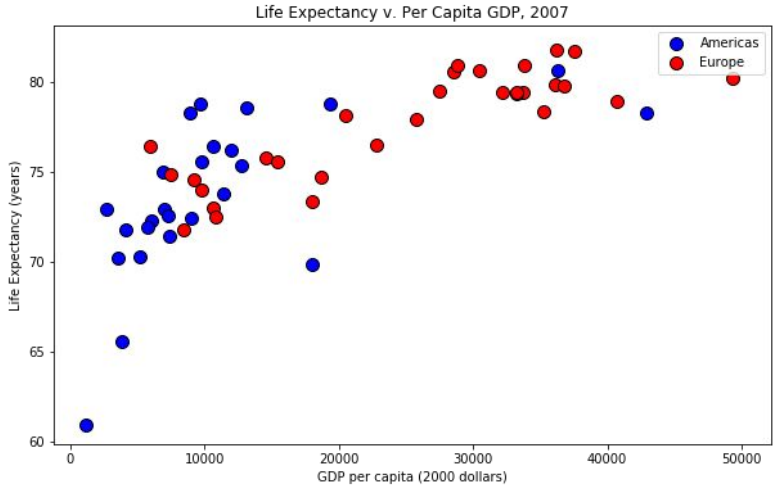
- Fast
  - Ability to “ad-hoc” plot
  - Low render time
- Easy
  - Learning startup cost is low
  - Building plots is intuitive
- Flexible
  - Can be displayed in many different formats
  - Doesn’t choke on large data

	Fast	Easy	Flexible
matplotlib			
seaborn			
bokeh			
plotly			

matplotlib

matplotlib

```
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
for con in df.continent.unique():
    x = df.query("continent=='{0}'".format(con)).gdp_percap.values
    y = df.query("continent=='{0}'".format(con)).life_exp.values
    if con == 'Americas':
        color = 'blue'
    else:
        color = 'red'

    ax.scatter(x,y,c=color,label=con,s=100,edgecolors='black', )

ax.legend()
ax.set_xlabel('GDP per capita (2000 dollars)')
ax.set_ylabel('Life Expectancy (years)')
ax.set_title("Life Expectancy v. Per Capita GDP, 2007")
plt.show()
```



matplotlib

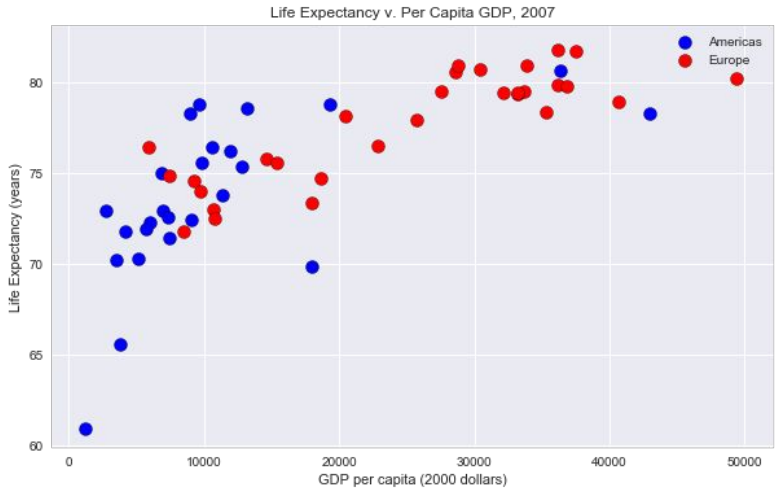
- Pros
  - Ad-hoc plotting is great for very simple plots
  - Has native jupyter plugins
  - Most popular, huge ecosystem
- Cons
  - Complicated plots get exponentially harder
  - Streaming, interactive and web plots are second-class citizens or non-existent
  - High memory usage

	Fast	Easy	Flexible
matplotlib	B+	C+	C
seaborn			
bokeh			
plotly			

seaborn

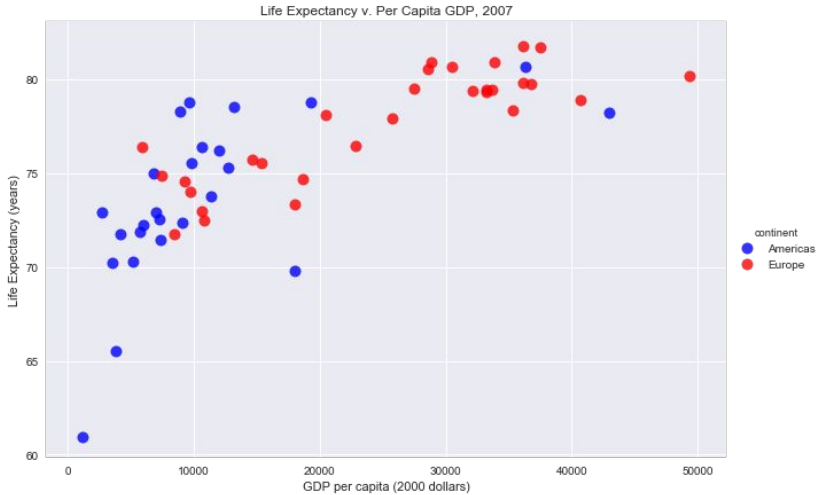
seaborn (replace)

```
import seaborn as sns
sns.set()

import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
for con in df.continent.unique():
    x = df.query("continent=='{0}'".format(con)).gdp_percap.values
    y = df.query("continent=='{0}'".format(con)).life_exp.values
    if con == 'Americas':
        color = 'blue'
    else:
        color = 'red'

    ax.scatter(x,y,c=color,label=con,s=100,edgecolors='black', )

ax.legend()
ax.set_xlabel('GDP per capita (2000 dollars)')
ax.set_ylabel('Life Expectancy (years)')
ax.set_title("Life Expectancy v. Per Capita GDP, 2007")
ax.grid(True)
plt.show()
```



seaborn (native)

```
import seaborn as sns
sns.set()

import matplotlib.pyplot as plt
%matplotlib inline

sns.lmplot('gdp_percap',
           'life_exp',
           data=df,
           size=6,
           aspect=1.5,
           fit_reg=False,
           hue='continent',
           palette={'Americas':'blue',
                    'Europe':'red'},
           scatter_kws={"edgecolors":"black",
                        "s":100})

plt.xlabel('GDP per capita (2000 dollars)')
plt.ylabel('Life Expectancy (years)')
plt.title("Life Expectancy v. Per Capita GDP, 2007")
plt.show()
```



seaborn

- Pros
  - Ad-hoc plotting is great for statistical models
  - Plot object model makes more sense
  - Jupyter native with interactivity added
- Cons*
  - Non-supported plots can't be created from primitives
  - Streaming and web still missing
  - Large data is still a problem

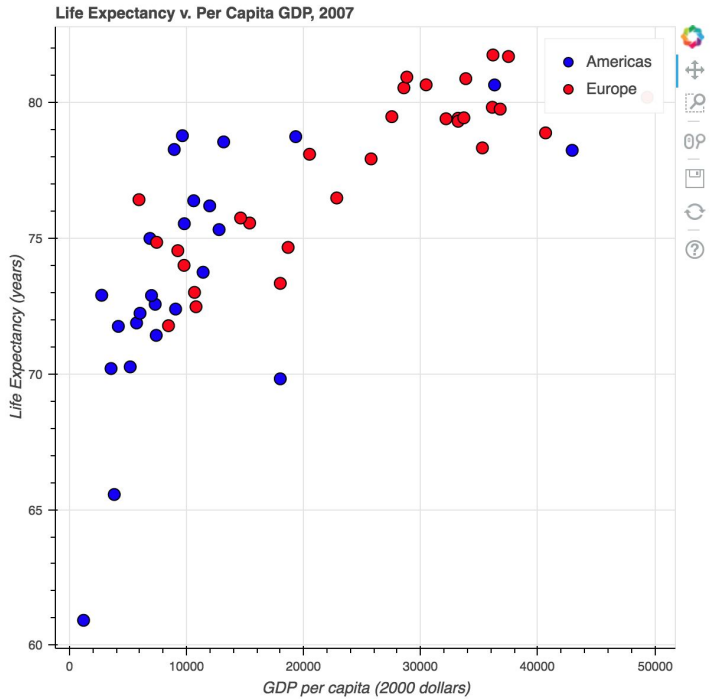
*If you are doing statistical modelling, and you aren't planning on ever using your plots on the web, then seaborn is the right choice 95% of the time

	Fast	Easy	Flexible
matplotlib	B+	C+	C
seaborn	B+	B	C+
bokeh			
plotly			

bokeh

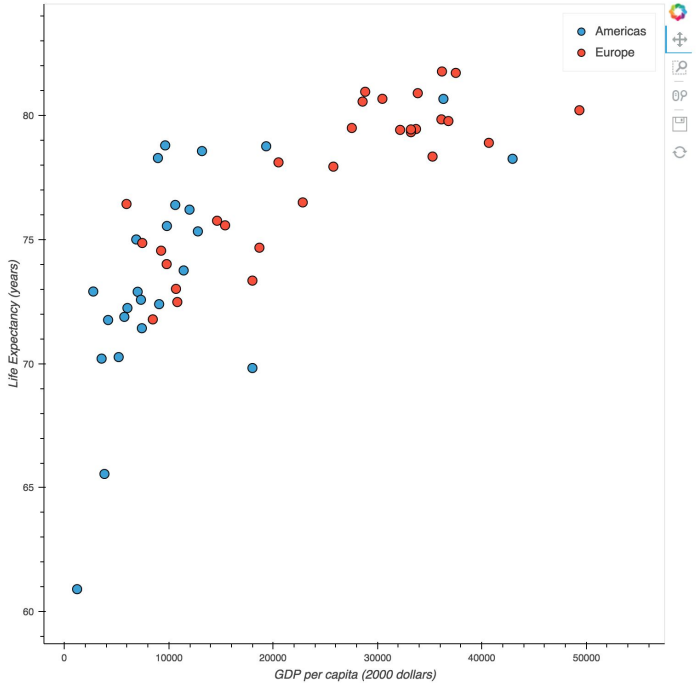
bokeh (glyph)

```
from bokeh.io import output_notebook
from bokeh.plotting import figure, show

output_notebook()

p = figure(x_axis_label="GDP per capita (2000 dollars)",
           y_axis_label="Life Expectancy (years)",
           title="Life Expectancy v. Per Capita GDP, 2007")

for con in df.continent.unique():
    x = df.query("continent=='{0}'".format(con)).gdp_percap.values
    y = df.query("continent=='{0}'".format(con)).life_exp.values
    if con == 'Americas':
        color = 'blue'
    else:
        color = 'red'

    p.circle(x,y,legend=con,color=color,size=10, line_color='black')

show(p)
```



bokeh (charts/HoloViews)

```
import holoviews as hv
hv.extension('bokeh')

%output size=250
%opts Scatter (size=10, line_color='black')

vdims = [('gdp_percap', 'GDP per capita (2000 dollars)'),
         ('life_exp', 'Life Expectancy (years)')]
ds = hv.Dataset(df, ['continent'], vdims)

grouped = ds.select(Continent=['Americas', 'Europe']).to(hv.Scatter,
                                                         'gdp_percap',
                                                         'life_exp')
grouped.overlay('continent')
```



bokeh

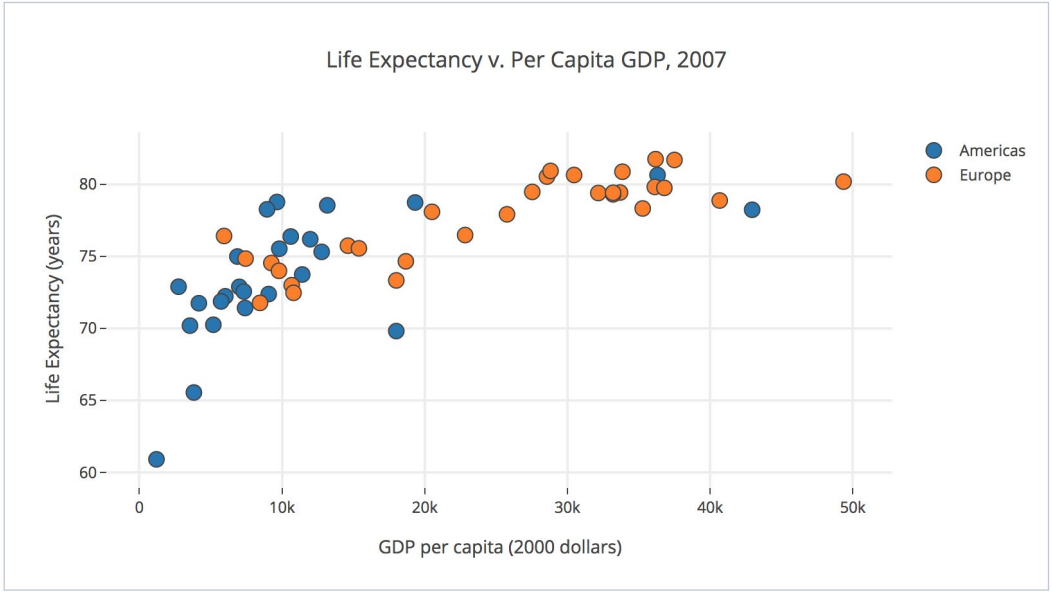
- Pros
  - Ad-hoc plotting is very fast and customizable
  - Variety of plot models (object or figure) allows diversity of choice
  - Interacts with Jupyter, natively interactive, compiles to javascript and accepts streaming data
- Cons
  - Learning startup cost is higher
  - Relies on JS backed, so render has a ‘warm up’ time

	Fast	Easy	Flexible
matplotlib	B+	C+	C
seaborn	B+	B	C+
bokeh	A-	B-	A
plotly			

plotly

plotly

```
import plotly.plotly as py
import plotly.graph_objs as go

americas = df[(df.continent=='Americas')]
europe = df[(df.continent=='Europe')]

trace_comp0 = go.Scatter(
    x=americas.gdp_percap,
    y=americas.life_exp,
    mode='markers',
    marker=dict(size=12,
                line=dict(width=1),
                color="navy"
               ),
    name='Americas',
    text=americas.country,
    )

trace_comp1 = go.Scatter(
    x=europe.gdp_percap,
    y=europe.life_exp,
    mode='markers',
    marker=dict(size=12,
                line=dict(width=1),
                color="red"
               ),
    name='Europe',
    text=europe.country,
        )

data_comp = [trace_comp0, trace_comp1]
layout_comp = go.Layout(
    title='Life Expectancy v. Per Capita GDP, 2007',
    hovermode='closest',
    xaxis=dict(
        title='GDP per capita (2000 dollars)',
        ticklen=5,
        zeroline=False,
        gridwidth=2,
    ),
    yaxis=dict(
        title='Life Expectancy (years)',
        ticklen=5,
        gridwidth=2,
    ),
)
fig_comp = go.Figure(data=data_comp, layout=layout_comp)
py.plot(fig_comp, filename='life-expectancy-per-GDP-2007')
```



plotly

- Pros
  - Flexibility is first class, backends for web, streaming interactivity and dashboards are native.
  - Language agnostic backend, making charts portable between R and python etc.
- Cons
  - Verbose
  - Learning curve is high
  - Tightly integrated into the pay side of the plotly company.

	Fast	Easy	Flexible
matplotlib	B+	C+	C
seaborn	B+	B	C+
bokeh	A-	B-	A
plotly	A-	C-	A+

So What?

- The python visualization ecosystem is super strong.
- Picking a visualization library often feels like a holy war.
- Picking one and getting comfortable is the most important aspect!

	Recommended For
matplotlib	MATLAB, General
seaborn	Statistical Analysis
bokeh	General, Web, Streaming
plotly	Web, Dashboards*

*Remember there is a company trying to get you to pay for it eventually