Visualization in Python

The Python Visualization Ecosystem

The Python visualization ecosystem is large and disparate. There are many packages and frameworks that have taken hold, but none of them are particularly dominant.

There are three technologies that serve as the foundation of this ecosystem:

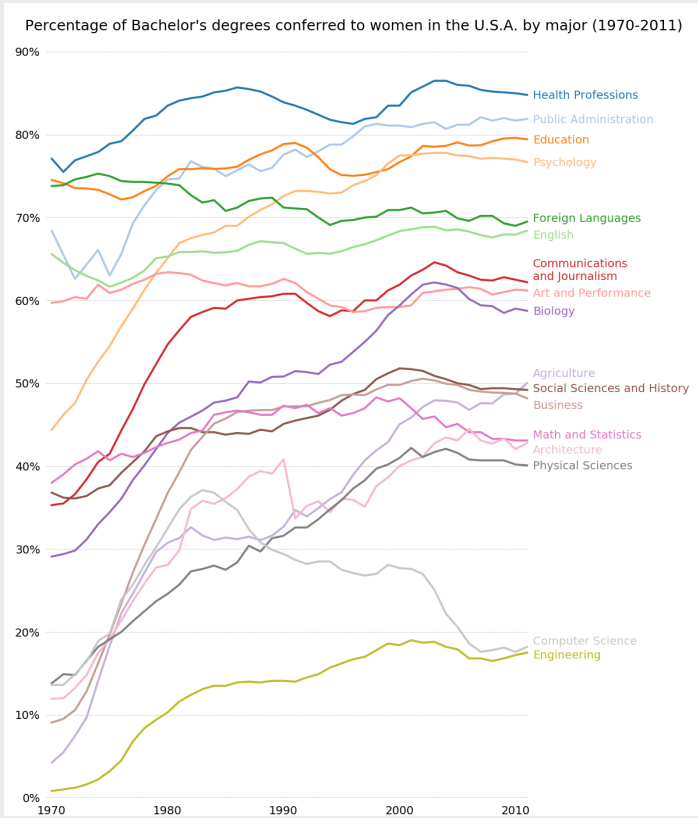
1. **matplotlib**
    - pandas
    - seaborn
2. **JavaScript**
    - bokeh
3. **d3.js**
    - plotly

Matplotlib

- Designed to mimic the functionality of Matlab, which helped to convert the scientific community.
- Can be verbose for simple plots.
- Styling looks a bit dated.
- With (sometimes a lot of) work you can create a wide variety of graphs.

Matplotlib

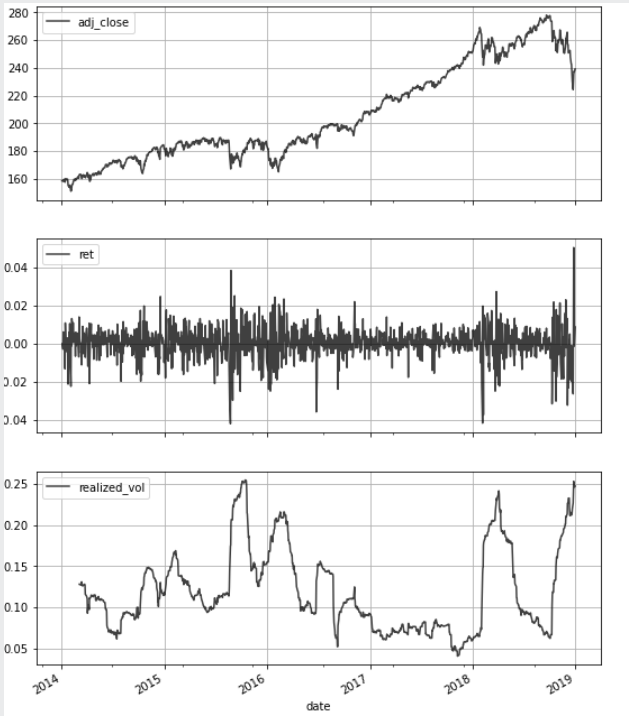
```
import matplotlib.pyplot as plt
from matplotlib.mlab import csv2rec
from matplotlib.cbook import get_sample_data

fname = get_sample_data('percent_bachelors_degrees_women_usa.csv')
gender_degree_data = csv2rec(fname)

# These are the colors that will be used in the plot
color_sequence = ['#1f77b4', '#aec7e8', '#ff7f0e', '#ffbb78', '#2ca02c',
                  '#98df8a', '#d62728', '#ff9896', '#9467bd', '#c5b0d5',
                  '#8c564b', '#c49c94', '#e377c2', '#f7b6d2', '#7f7f7f',
                  '#c7c7c7', '#bcbd22', '#dbdb8d', '#17becf', '#9edae5']

# You typically want your plot to be ~1.33x wider than tall. This plot
# is a rare exception because of the number of lines being plotted on it.
# Common sizes: (10, 7.5) and (12, 9)
fig, ax = plt.subplots(1, 1, figsize=(12, 14))

# Remove the plot frame lines. They are unnecessary here.
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)

# Ensure that the axis ticks only show up on the bottom and left of the plot.
# Ticks on the right and top of the plot are generally unnecessary.
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()

fig.subplots_adjust(left=.06, right=.75, bottom=.02, top=.94)
# Limit the range of the plot to only where the data is.
# Avoid unnecessary whitespace.
ax.set_xlim(1969.5, 2011.1)
ax.set_ylim(-0.25, 90)

# Make sure your axis ticks are large enough to be easily read.
# You don't want your viewers squinting to read your plot.
plt.xticks(range(1970, 2011, 10), fontsize=14)
plt.yticks(range(0, 91, 10), fontsize=14)
ax.xaxis.set_major_formatter(plt.FuncFormatter('{:.0f}'.format))
ax.yaxis.set_major_formatter(plt.FuncFormatter('{:.0f}%'.format))

# Provide tick lines across the plot to help your viewers trace along
# the axis ticks. Make sure that the lines are light and small so they
# don't obscure the primary data lines.
plt.grid(True, 'major', 'y', ls='--', lw=.5, c='k', alpha=.3)

# Remove the tick marks; they are unnecessary with the tick lines we just
# plotted.
plt.tick_params(axis='both', which='both', bottom='off', top='off',
                labelbottom='on', left='off', right='off', labelleft='on')

# Now that the plot is prepared, it's time to actually plot the data!
# Note that I plotted the majors in order of the highest % in the final year.
majors = ['Health Professions', 'Public Administration', 'Education',
          'Psychology', 'Foreign Languages', 'English',
          'Communications\nand Journalism', 'Art and Performance', 'Biology',
          'Agriculture', 'Social Sciences and History', 'Business',
          'Math and Statistics', 'Architecture', 'Physical Sciences',
          'Computer Science', 'Engineering']

y_offsets = {'Foreign Languages': 0.5, 'English': -0.5,
             'Communications\nand Journalism': 0.75,
             'Art and Performance': -0.25, 'Agriculture': 1.25,
             'Social Sciences and History': 0.25, 'Business': -0.75,
             'Math and Statistics': 0.75, 'Architecture': -0.75,
             'Computer Science': 0.75, 'Engineering': -0.25}

for rank, column in enumerate(majors):
    # Plot each line separately with its own color.
    column_rec_name = column.replace('\n', '_').replace(' ', '_').lower()

    line = plt.plot(gender_degree_data.year,
                    gender_degree_data[column_rec_name],
                    lw=2.5,
                    color=color_sequence[rank])

    # Add a text label to the right end of every line. Most of the code below
    # is adding specific offsets y position because some labels overlapped.
    y_pos = gender_degree_data[column_rec_name][-1] - 0.5

    if column in y_offsets:
        y_pos += y_offsets[column]

    # Again, make sure that all labels are large enough to be easily read
    # by the viewer.
    plt.text(2011.5, y_pos, column, fontsize=14, color=color_sequence[rank])

# Make the title big enough so it spans the entire plot, but don't make it
# so big that it requires two lines to show.

# Note that if the title is descriptive enough, it is unnecessary to include
# axis labels; they are self-evident, in this plot's case.
fig.suptitle('Percentage of Bachelor\'s degrees conferred to women in '
             'the U.S.A. by major (1970-2011)\n', fontsize=18, ha='center')

# Finally, save the figure as a PNG.
# You can also save it as a PDF, JPEG, etc.
# Just change the file extension in this call.
# plt.savefig('percent-bachelors-degrees-women-usa.png', bbox_inches='tight')
plt.show()
```



Pandas

- built on top of matplotlib
- basic plotting functionality for DataFrames
- very easy to use
- this is my visualization workhorse

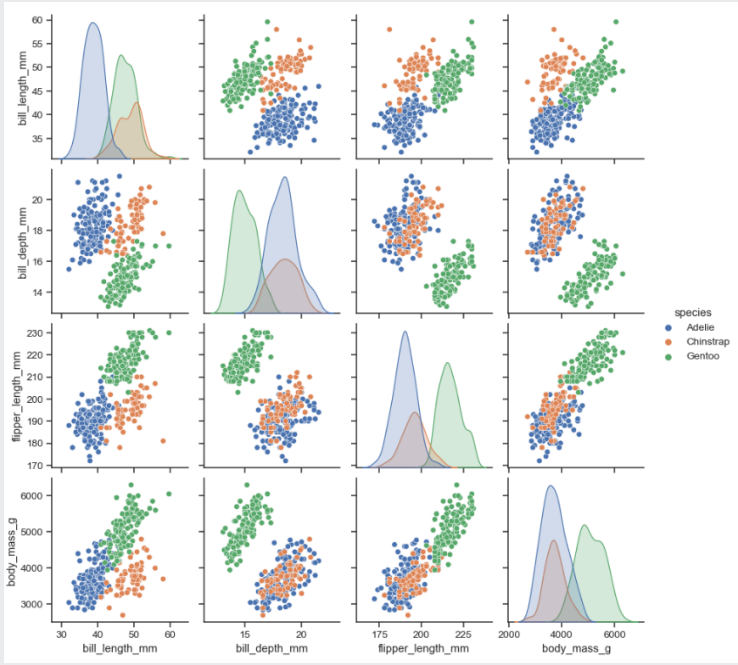
```
df_spy.plot(x = 'date',
            y = ['adj_close','ret','realized_vol',],
            subplots=True, style='k', alpha=0.75,
            figsize=(9, 12),
            grid=True);
```



Seaborn

- built on top of matplotlib
- emphasis on statistical visualizations
- nice styling out of the box

```
sns.pairplot(penguins, hue="species")
```



JavaScript and Visualization

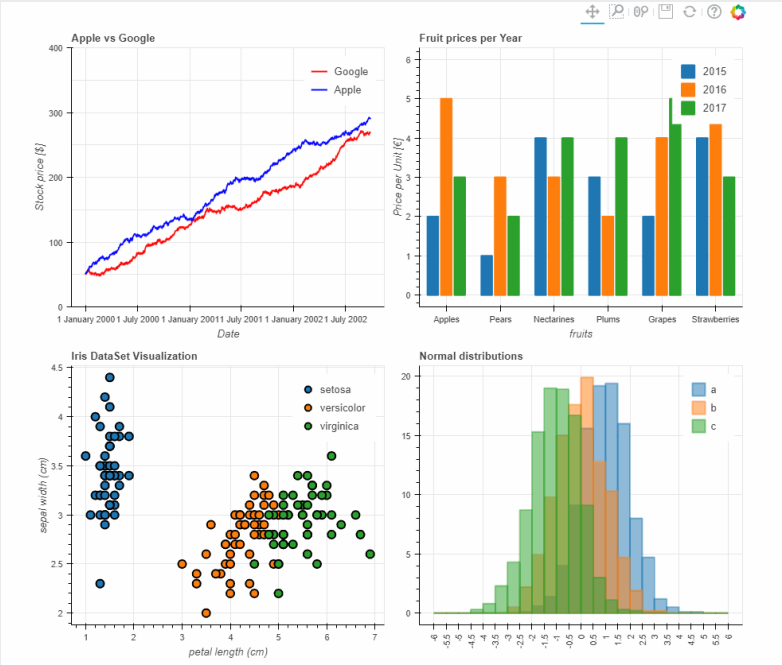
JavaScript is an interpreted programming language that is popular for client side web development.

If there is something cool (dynamic, non-static) going on in a webpage, it's probably JavaScript under the hood.

This same technology can be used for data visualization rendered in the browser.

Bokeh

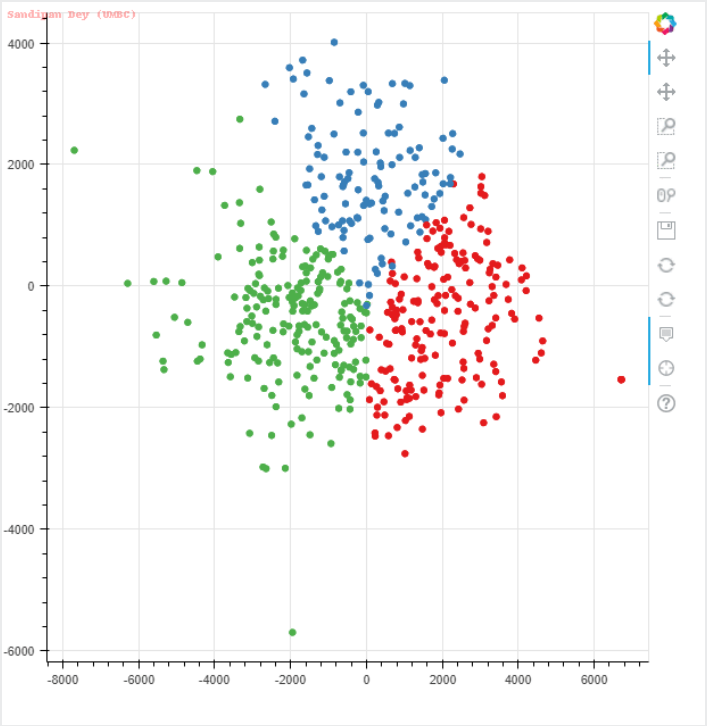
- High-level JavaScript interface for quickly producing interactive visualizations.
- It really shines when you want to make standard charts with typical interactivity (can be done without a ton of work)



D3.js and Plotly

D3.js is a JavaScript visualization kernel (low-level framework) for creating any manner of visualizations.

Plotly is an interactive plotting library that is built on top of D3.



References

These slides are a shell of this great talk:

Jake VanderPlas The Python Visualization Landscape PyCon 2017