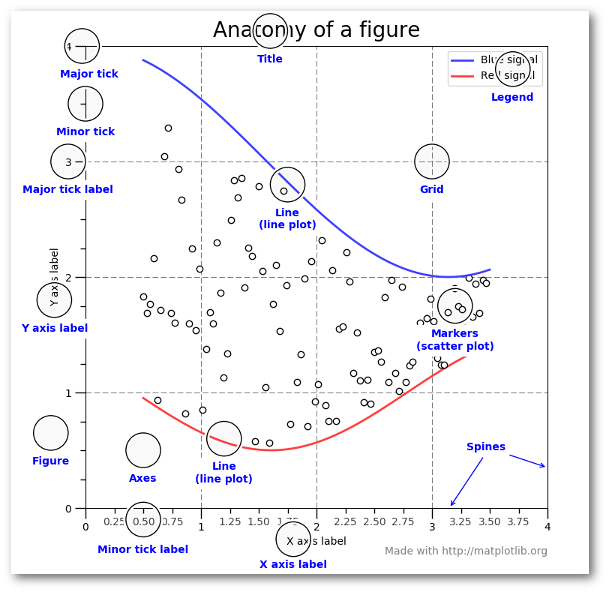
Matplotlib

1. Visualization library in python
2. Has two interfaces – Pyplot API and object-oriented API
3. [pyplot](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot) is mainly intended for interactive plots and simple cases of programmatic plot generation. Based on MATLAB.
4. Most commonly used is object-oriented API where we create objects with axes and figures
5. In object-oriented API, we utilize an instance of [axes.Axes](https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes) in order to render visualizations on an instance of [figure.Figure](https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure).
6. The Figure is the final image that may contain 1 or more Axes
7. The Axes represent and individual plot (do not confuse it with the word ‘axis’ x/y of a plot)
8. Methods are called that do plotting directly from the Axes, this gives much more flexibility.
9. Try to use object-oriented interface over pyplot interface
10. Problems with matplolib
    1. 2 interfaces
    2. Some default style choices are unattractive
    3. Confusion when to use pure matplotlib or a tool built over matplotlib like pandas and seaborn
11. In order to access the figure and axes in pyplot, we use object oriented API. This helps in adding future customizations



1. pyplot

X = …

Y = …

Plt.plot(X,Y) #can only add use few customization functionalities of matplotlib

Plt.title(…)

Plt.xlim()

Plt.ylim()

Plt.legend()

plt.gcf() allows you to get a reference to the current figure when using **pyplot**.

Same as with plt.gcf(), you can use plt.gca() to get a reference to the current axes, if you need to change the limits on the y-axis, for example.

1. Object oriented api

#create the figure and the axes

Fig, ax = plt.subplots()

Ax.plot(X,Y) #now we have access to the axes of plot, so we can add many visualizations

Ax.set\_xlim([...,…])

Ax.set\_ylim([…,…])

Ax.set\_xlabel(‘…’)

Ax.set\_ylabel(‘…’)

Or

Ax.set(title = ‘…’, xlabel = ‘…’, ylabel = ‘…’)

ax.set(title='2014 Revenue', xlabel='Total Revenue', ylabel='Customer')

can change the size of the image using the function plt.subplots()

Fig, ax = plt.subplots(figsize = (5,6))

Adding annotations

1. To add vertical line : ax.axvline()
2. To add custom text : ax.text()

Adding multiple plots in the figure

Pyplot:

Plt.figure(figsize = (10,10))

Plt.subplot(121)

Plt.plot(x1, y1)

Plt.subplot(122)

Plt.plot(x2, y2)

Object-oriented :

We can accomplish this using plt.subplots()

fig, (ax0, ax1) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize=(7, 4))

ax0.plot(X1, Y1)

ax0.set(title, xlabel, ylabel)

ax1.plot(X2, Y2)

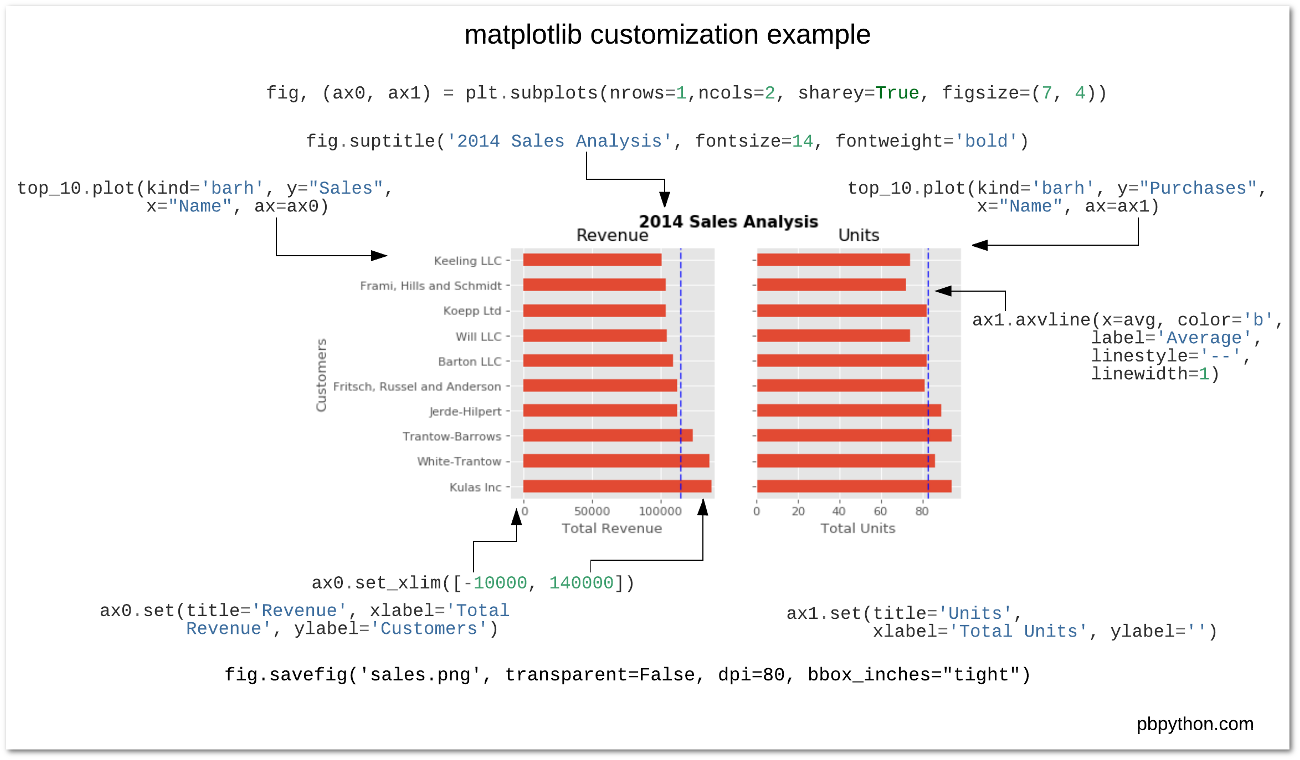
ax1.set(title, xlabel, ylabel)

fig.suptitle(‘’’)

Matplotlib supports many different formats for saving files. Since we have the fig object, we can save the figure using multiple options:

fig.savefig('sales.png', transparent=False, dpi=80, bbox\_inches="tight")

For labels, we can specify custom formatting guidelines in the form of functions by using the [ticker.FuncFormatter](https://matplotlib.org/api/ticker_api.html#matplotlib.ticker.FuncFormatter) class.



Another way to use object oriented API

figure = plt.figure()

new\_plot = figure.add\_subplot(111)

new\_plot.plot(x, y)

new\_plot = fig.add\_axes([0,0,1,1])

new\_plot.plot(x, y)

The difference between fig.add\_axes() and fig.add\_subplot() doesn’t lie in the result: they both return an Axes object. However, they do differ in the mechanism that is used to add the axes: you pass a list to add\_axes() which is the lower left point, the width and the height. This means that the axes object is positioned in absolute coordinates.

In contrast, the add\_subplot() function doesn’t provide the option to put the axes at a certain position: it does, however, allow the axes to be situated according to a subplot grid, as you have seen in the section above.

These methods are used to **clear** the current figure (plt.clf()) or the current axes (plt.cla()).

As I mentioned earlier, matplotlib has many different styles available for rendering plots. You can see which ones are available on your system using plt.style.available .

Using a style is as simple as:

plt.style.use('ggplot')

rc” is common for configuration files: they usually end in rc. It comes from the practice of having configs as executables: they are automatically **r**un and **c**onfigure settings, for example. You can read more about it [here](http://www.catb.org/jargon/html/R/rc-file.html). Matplotlib has such an rc file to which you can make adjustments dynamically and statically.

To dynamically change default rc settings, you can use the rcParams variable:

# Import the necessary packages and modules

import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

# Uncomment following line to see the effect

#mpl.rcParams['lines.linewidth'] = 5

# Prepare the data

x = np.linspace(0, 10, 100)

# Plot the data

plt.plot(x, x, label='linear')

# Add a legend

plt.legend()

# Show the plot

plt.show()

Other visualization libraries:

* [Pandas](http://pandas.pydata.org/)
* [Seaborn](http://stanford.edu/~mwaskom/software/seaborn/) –based on matplotlib
* [ggplot](http://ggplot.yhathq.com/) – based on matplotlib
* [Bokeh](http://bokeh.pydata.org/)
* [pygal](http://pygal.org/)
* [Plotly](https://plot.ly/)

Summary

1. Types of interface
2. Basic example of each interface
3. Anatomy of figure
4. Formatting with object oriented API
5. Subplots with each interface
6. Alternative ways to use object oriebtnted API
7. Saving a plot
8. Clearing a plot
9. Function formatter
10. Configuring plot Style and Rc parameters
11. Other visualization libraries

References

1. <https://pbpython.com/effective-matplotlib.html>
2. <https://matplotlib.org/api/api_overview.html>
3. <https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf>
4. Different typs of plots (<https://matplotlib.org/tutorials/introductory/sample_plots.html#sphx-glr-tutorials-introductory-sample-plots-py>)