Data Visualization - Matplotlib Tips

We'll cover the following

General Matplotlib Tips

Before we dive into the details of creating visualizations with Matplotlib, here are a few useful tips about using this package:

1. Importing Matplotlib

Just as we used the *np* shorthand for NumPy, and *pd* for Pandas, ***plt*** is the standard shorthand for Matplotlib:

```
import matplotlib.pyplot as plt
```

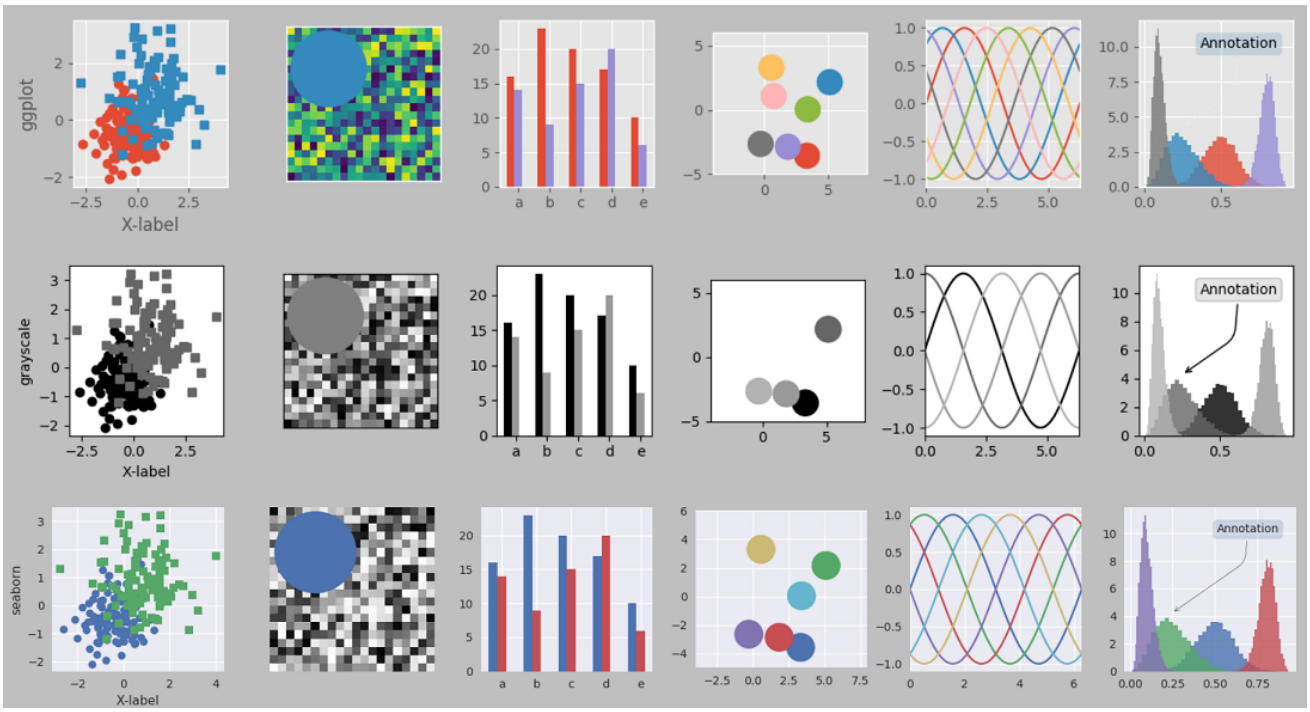
Note that Matplotlib is a huge library, so we are only importing the pyplot part of it. This is useful to save memory and speedup code. Otherwise we'll be importing gigabytes of libraries even when we are just interested in using it to perform some trivial tasks.

2. Setting Styles

`plt.style` directive can be used to choose different *prettyfying* styles for our figures. There a number of pre-defined styles provided by Matplotlib. For example, there's a pre-defined style called *ggplot*, which tries to copy the *look and feel* of ggplot, a popular plotting package for R. Below are some examples, both code and visual outputs, of the available styles; you can go through the

official reference sheet for a complete overview.

```
# Examples of available styles
plt.style.use('classic')
plt.style.use('ggplot')
plt.style.use('seaborn-whitegrid')
plt.style.use(['dark_background', 'presentation'])
```



Examples of available style. Image Credits: https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html

3. Displaying Plots

a. If you are using Matplotlib from within a script, the function `plt.show()` is the way to go. It triggers an event that looks for all currently active figure objects and opens one or more interactive windows to display them.

b. If you are working with a Jupyter notebook, plotting interactively within the notebook can be done with the *%matplotlib* command:

- `%matplotlib notebook` will create interactive plots embedded within the notebook.
- `%matplotlib inline` will create static images of your plots embedded in the notebook.

4. Saving Figures to File

Matplotlib provides a handy feature for saving figures in a wide range of formats using the `savefig()` command. For example, to save a figure called *fig* as a *png* file, you can run this:

```
fig.savefig('figure.png')
```

Notice that the file format is inferred from the extension of the given filename.

Useful introductory tips ✔ We are now ready to learn about some fun visualization techniques.