Download and open the following in google colab :

creating plots.ipynb

labeling plots.ipynb

styling plots.ipynb

Fatalities.csv

oldfaithful

### What is matplotlib?

Data visualization plays a big part in the data science lifecycle. In particular, data scientists often create data visualizations to gain insight on data and develop a scope for a machine learning task. While several data visualization packages in Python are available, `matplotlib` is the most common. ***matplotlib*** is a package used to create static, dynamic, and interactive plots. `Seaborn`, another common data visualization package used primarily for statistical graphs, is based on `matplotlib`.

`matplotlib` uses figures to hold plot elements, such as axes, labels, tick marks, and legend. Each element can be adjusted manually. Ex: Positions of minor and major tick marks can be controlled as well as the font sizes for the labels.

Figures can be created using `pyplot`. The `pyplot` library is a state-based interface to the `matplotlib` package that uses a similar syntax to MATLAB. Each line of code adds a plot element to the figure one at a time, while preserving previously added elements in the figure. To learn more about common functions within the `pyplot` library, see the pyplot documentation.

## Advantages of matplotlib.

| | |
|---|---|
| Versatility | Can be used in Python scripts, shells, iPython, and Jupyter Notebooks |
| Familiarity | Uses MATLAB-style functions |
| Portability | Runs on MacOS, Linux, and Windows |
| Customizability | Can adjust details of a plot like axes, legends, and color |

## Plotting with matplotlib

The first step to create a plot with `pyplot` is to import the library. Since `pyplot` is a library within the `matplotlib` package, dot notation is needed. Ex: `import matplotlib.pyplot as plt`.

The `pyplot` library can display different objects within a figure. The most common object is `plt.plot(x, y)`, where x and y are arrays of the same size, which creates a line plot connecting consecutive x- and y- coordinates. Another common object is `plt.scatter(x, y)`, which creates a scatter plot showing all pairs of x- and y- coordinates.

Other useful functions in the `pyplot` library are:

- `plt.figure()` : creates a new figure
- `plt.show()`: displays the figure and all the object the figure contains
- `plt.savefig(fname)` : saves the figure in the current working directory with the filename `fname`

## Are the plt.figure() and plt.show() functions necessary?

*`plt.figure()`* is only needed when changing the default size of the figure. The size of the figure can be specified using the *`figsize`* parameter. Since a figure is implicitly created whenever a *`plt.plot(x,y)`* or *`plt.scatter(x,y)`* function is called, calling the *`plt.figure()`* function is not necessary if the default figure size is acceptable. Note: The default figure size is 6.4 inches by 4.8 inches.

Within Jupyter Notebooks or any interactive shell like IDLE, all elements within each figure are automatically displayed when cells or lines of code are run. However, the *`plt.show()`* function is necessary when *`matplotlib`* is used within a terminal or a script.

### Labeling plots

`matplotlib` has extensive support for adding titles, labeling axes, and adding text and annotation to different plots.

To add a title to a figure, the `plt.title()` function is used. Ex: `plt.title('Alcohol-related fatalities in highways')`

The following functions are used to add text for the axes:

- `plt.xlabel()`: adds text for the x-axis
- `plt.ylabel()`: adds text for the y-axis

Adding mathematical expressions using LaTeX is also supported. Typesetting mathematical expressions using LaTeX can be done by enclosing an expression within a string with dollar signs. The letter r should precede a string so that a backslash is not treated as a Python escape.
Ex: `plt.title(r'Standard normal distribution $f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$')`

The following functions can be used to add text within the figure:

- `plt.text(x, y, s)`: adds string s to the figure at coordinates (x, y)
- `plt.annotate(s, xy, xytext)`: links string s at coordinates given by xytext to a point given by xy
- `plt.legend()`: adds legend in the figure

**Styling plots**

By default, lines and scatter plots are assigned certain colors and style. The style of different plots can be changed manually by adding a character or characters that correspond to color, line style, and marker style.
Ex: `plt.plot(x, y, 'ro')` outputs a plot in red with circle markers. Style can also be defined using the parameters color, linestyle, and marker.

The table below gives a partial list of characters for color, line style, and marker style. If a marker style is not specified, the output will not show any markers.

Table 2.7.2: Characters for line color, line style, and marker style.

| Character(s) | Line color/style | Character(s) | Marker style | Character(s) | Marker style |
|---|---|---|---|---|---|
| b | Blue | . | Point marker | 1 | Tri-down marker |
| g | Green | , | Pixel marker | 2 | Tri-up marker |

| | | | | | |
|---|---|---|---|---|---|
| r | Red | o | Circle marker | 3 | Tri-left marker |
| w | White | + | Plus marker | 4 | Tri-right marker |
| k | Black | X | X marker | h | Hexagon1 marker |
| y | Yellow | v | Triangle-down marker | H | Hexagon2 marker |
| m | Magenta | ^ | Triangle-up marker | D | Diamond marker |
| - | Solid line | < | Triangle-left marker | d | Thin diamond marker |
| : | Dotted line | > | Triangle-right marker | \| | Vertical line marker |
| -- | Dashed line | * | Star marker | _ | Horizontal line marker |
| -. | Dashed-dot line | p | Pentagon marker | s | Square marker |