East West University
Department of Computer Science and Engineering

CSE 303: Statistics for Data Science
LAB 06
Course Instructor: Dr. Mohammad Rezwanul Huq

Intermediate Plotting using Matplotlib Libraries

Lab Objective

Introducing Matplotlib libraries for different types of plotting.

Lab Outcome

After completing this lab successfully, students will be able to:

1. **Understand** Matplotlib functions for basic and intermediate-level plotting.
2. **Apply** Matplotlib functions to generate different types of plotting.

Psychomotor Learning Levels

This lab involves activities that encompass the following learning levels in psychomotor domain.

Level	Category	Meaning	Keywords
P1	Imitation	Copy action of another; observe and replicate.	Relate, Repeat, Choose, Copy, Follow, Show, Identify, Isolate.
P2	Manipulation	Reproduce activity from instruction or memory	Copy, response, trace, Show, Start, Perform, Execute, Recreate.

Required Applications/Tools

- Anaconda Navigator (Anaconda3)
  - Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.
  - Popular Tools/IDEs: Spyder, Jupyter Notebook
- Google Colab: Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Lab Activities

1. Basic Plotting

```
import matplotlib.pyplot as plt

# x axis values
x = np.array([1,2,3])
# corresponding y axis values
y = np.array([2,4,1])

# plotting the points
plt.plot(x, y)
```

```
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('My first graph!')

# function to show the plot
plt.show()
```

```
import matplotlib.pyplot as plt

# x-coordinates of left sides of bars
left = [1, 2, 3, 4, 5]

# heights of bars
height = [10, 24, 36, 40, 5]

# labels for bars
tick_label = ['one', 'two', 'three', 'four', 'five']

# plotting a bar chart
plt.bar(left, height, tick_label = tick_label, width = 0.8, color =
['red', 'green'])

# naming the x-axis
plt.xlabel('x - axis')
# naming the y-axis
plt.ylabel('y - axis')
# plot title
plt.title('My bar chart!')

# function to show the plot
plt.show()
```

2. Useful Line Properties

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1,1,1)
X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
C,S = np.cos(X), np.sin(X)
# Plot cosine using blue color with a continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-", label="cosine")
# Plot sine using green color with a continuous line of width 1(pixels)
plt.plot(X, S, color="green", linewidth=1.0, linestyle="-", label="sine")
# Set x limits
plt.xlim(-4.0,4.0)
# Set x ticks
plt.xticks(np.linspace(-4,4,9,endpoint=True))
# Set y limits
```

```
plt.ylim(-1.0,1.0)
# Set y ticks
plt.yticks(np.linspace(-1,1,5,endpoint=True))
# showing x and y labels
plt.xlabel("x-axis")
plt.ylabel("y-axis")
# showing legend
plt.legend(loc = "upper left")
# showing title
plt.title("Sine and Cosine Graph")
# Save figure using 72 dots per inch
plt.savefig("sample1.png",dpi=72)
# Show result on screen
plt.show()
```

3. Changing Tick Labels

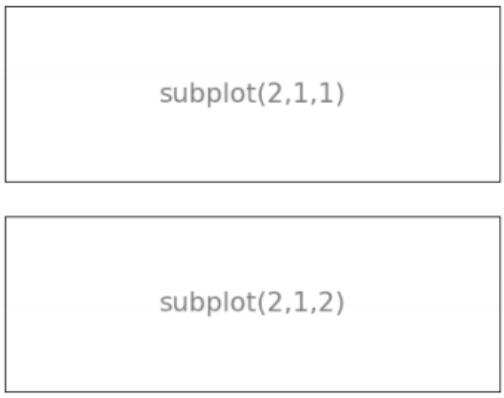
```
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],
           [r'$-1$', r'$0$', r'$+1$'])
```

4. Annotating Graphs

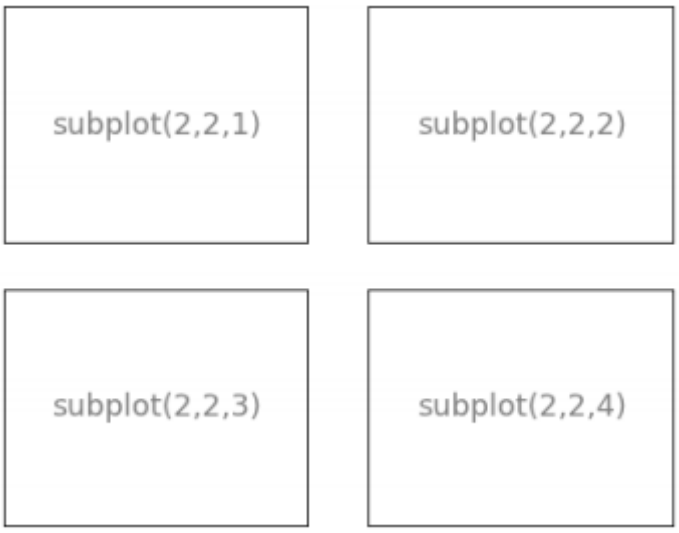
```
t = 2*np.pi/3
plt.plot([t,t],[0,np.cos(t)],color ='blue', linewidth=2.5, linestyle="--")
plt.scatter([t,],[np.cos(t),], 50, color ='blue')
plt.annotate(r'$\sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
 xy=(t, np.sin(t)), xycoords='data',
 xytext=(+10, +30), textcoords='offset points', fontsize=16,
 arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
plt.plot([t,t],[0,np.sin(t)], color ='red', linewidth=2.5, linestyle="--")
plt.scatter([t,],[np.sin(t),], 50, color ='red')
plt.annotate(r'$\cos(\frac{2\pi}{3})=-\frac{1}{2}$',
 xy=(t, np.cos(t)), xycoords='data',
 xytext=(-90, -50), textcoords='offset points', fontsize=16,
 arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
```

5. Subplots

With subplot you can arrange plots in a regular grid. You need to specify the number of rows and columns and the number of the plot.



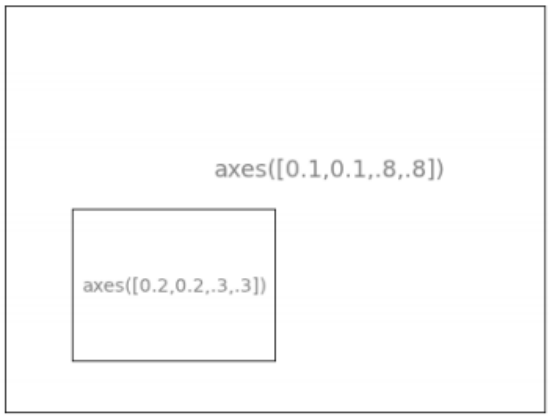
Subplot with 2 rows and 1 column



Subplot with 2 rows and 2 columns

6. Axes

Axes are very similar to subplots but allow placement of plots at any location in the figure. So if we want to put a smaller plot inside a bigger one we do so with axes.



Putting a smaller figure inside bigger figure

7. Other Types of Plotting

- Scatter Plot: `plt.scatter(X, Y)`
- Bar Plot: `plt.bar(X,Y)`
- Pie Chart: `plt.pie(Z)`
- Showing Grids:

```
axes = gca()
axes.set_xlim(0,4)
axes.set_ylim(0,3)
axes.set_xticklabels([])
axes.set_yticklabels([])
show()
```