Basic Plots with Matplotlib

Topics to cover…

- Basic Plots with Matplotlib
- Other applications

Basic Plots

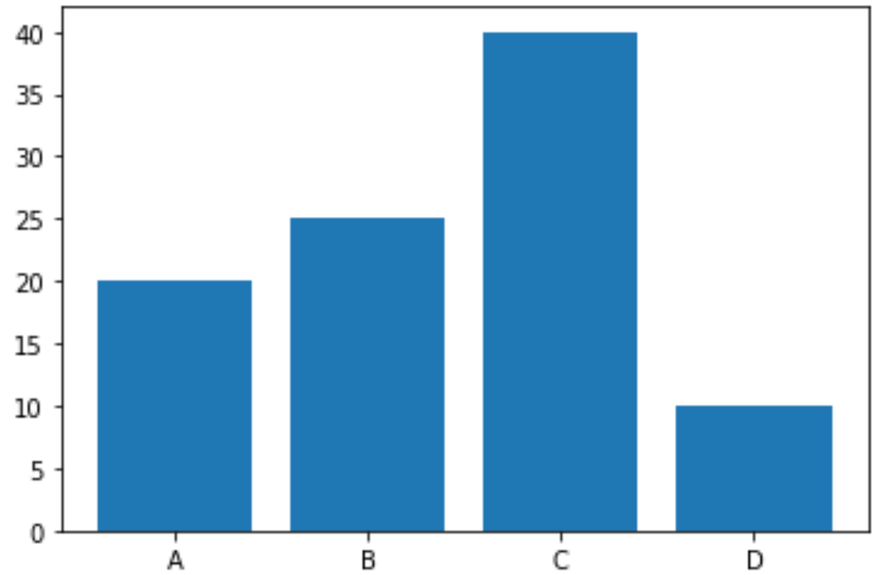
Bar chart

- `plt.bar(x, height, [width])` creates a vertical bar plot. For horizontal bars, use the `plt.barh()` function
- Important parameters:
  - `x`: Specifies the x coordinates of the bars
  - `height`: Specifies the height of the bars
  - `width` (optional): Specifies the width of all bars; the default is 0.8

Bar chart

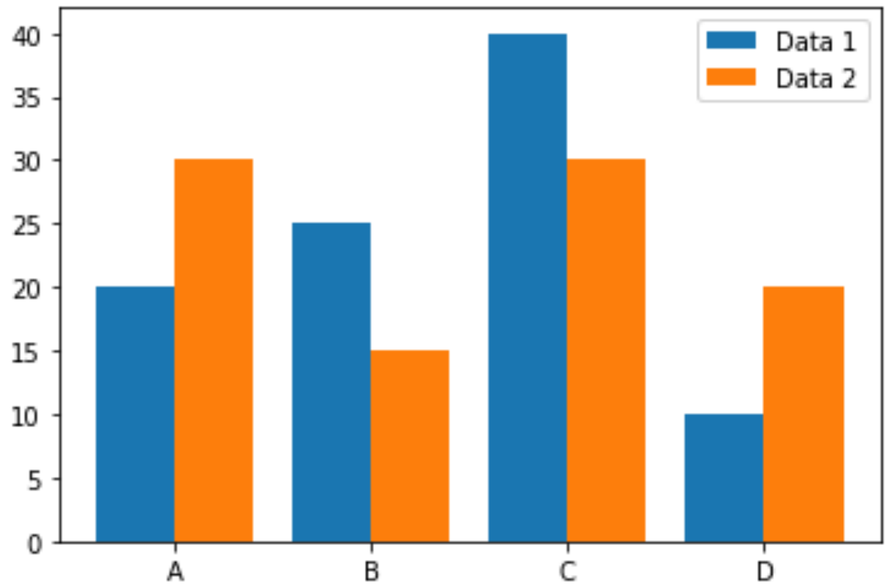
```
plt.bar(['A', 'B', 'C', 'D'], [20, 25, 40, 10])
```

```
<BarContainer object of 4 artists>
```



Bar chart

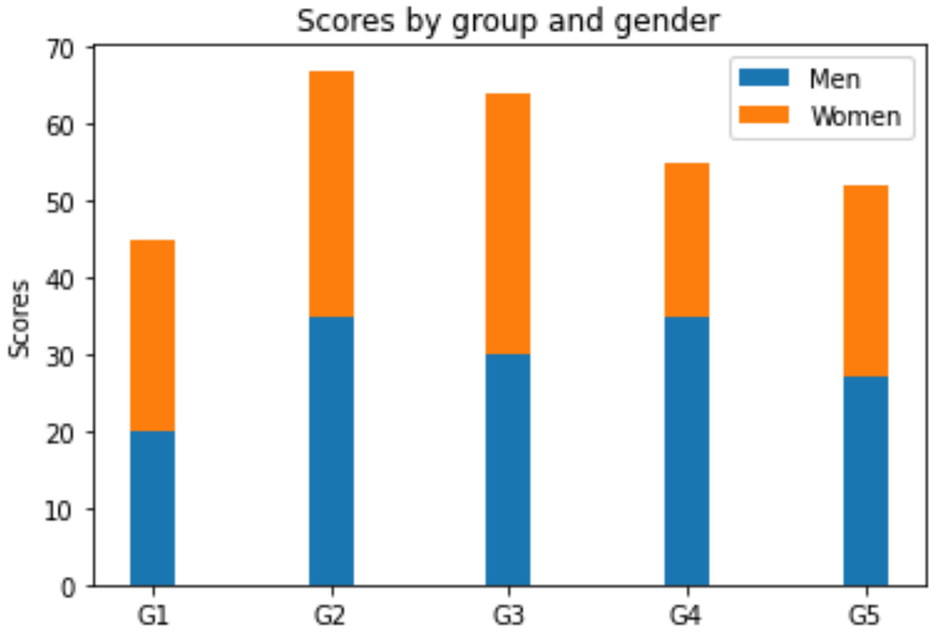
```python
import numpy as np

labels = ['A', 'B', 'C', 'D']
x = np.arange(len(labels))
width = 0.4

# Manually position the bars
plt.bar(x-width / 2, [20, 25, 40, 10], width=width, label='Data 1')
plt.bar(x+width / 2, [30, 15, 30, 20], width=width, label='Data 2')

# Get the current Axes
ax = plt.gca()

ax.legend()

# Set the xticks to be x
ax.set_xticks(x)
# Set the xticklabels
ax.set_xticklabels(labels)

plt.show()
```



Stacked bar chart

- The parameter `bottom` must be specified starting from the second stacked bar

```
import matplotlib.pyplot as plt

labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 35, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]
width = 1/len(x)       # the width of the bars

fig, ax = plt.subplots() # Create 1 Axes

ax.bar(labels, men_means, width, label='Men')
ax.bar(labels, women_means, width, bottom=men_means,label='Women')

ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.legend()

plt.show()
```

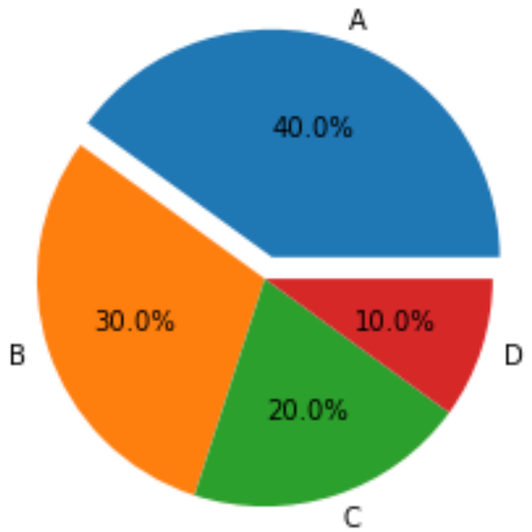


Pie chart

- `plt.pie(x, [explode], [labels], [autopct])` function creates a pie chart
- Important parameters:
  - `x`: Specifies the slice sizes
  - `explode` (optional): Specifies the fraction of the radius offset for each slice. The
    explode-array must have the same length as the x-array
  - `labels` (optional): Specifies the labels for each slice
  - `autopct` (optional): Shows percentages inside the slices according to the specified format string (e.g.,`'%1.1f%%'`)

Pie chart

```
plt.pie([0.4, 0.3, 0.2, 0.1], explode=(0.1, 0, 0, 0),
        labels=['A', 'B', 'C', 'D'], autopct='%1.1f%%')
plt.show()
```

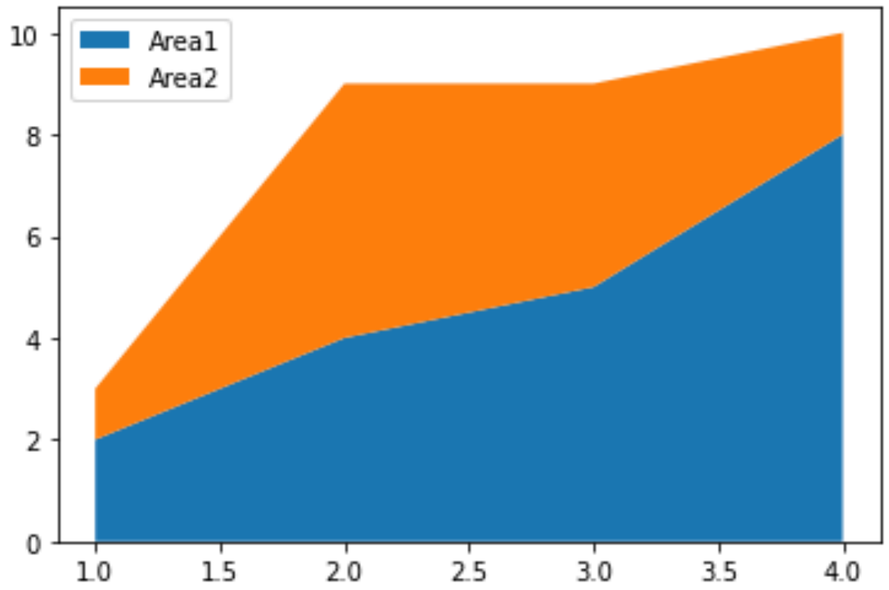


Stacked area chart

- `plt.stackplot(x, y)` creates a stacked area plot
- Important parameters:
  - `x`: Specifies the x-values of the data series
  - `y`: Specifies the y-values of the data series. For multiple series, either as a 2D array, or any number of 1D arrays, use `plt.stackplot(x, y1, y2, y3, …)`
  - `labels` (Optional): Specifies the labels as a list or tuple for each data series

Stacked area chart

```
# plt.stackplot(x, y1, y2)
plt.stackplot([1, 2, 3, 4], [2, 4, 5, 8], [1, 5, 4, 2], labels=['Area1', 'Area2'])
plt.legend(loc='upper left')
plt.show()
```

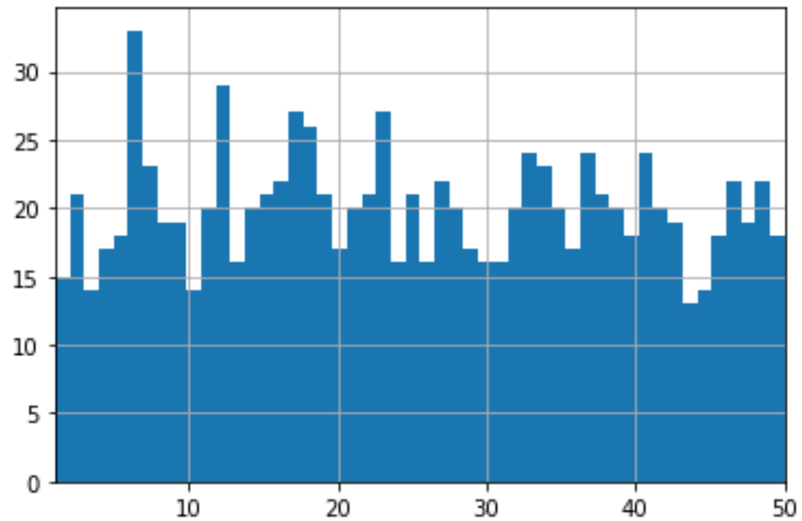


Histogram

- `plt.hist(x)` creates a histogram
- Important parameters:
  - `x`: Specifies the input values
  - `bins`: (optional): Either specifies the number of bins as an integer or specifies the bin edges as a list
  - `range`: (optional): Specifies the lower and upper range of the bins as a tuple
  - `density`: (optional): If true, the histogram represents a probability density

Histogram

```
import random

# generate a list of 100 random numbers with range from 1 to 50 (inclusive)
x = [random.randint(1,50) for x in range(1000)]

plt.hist(x, bins=50, density=False)
plt.xlim(1,50)
plt.grid()
plt.show()
```

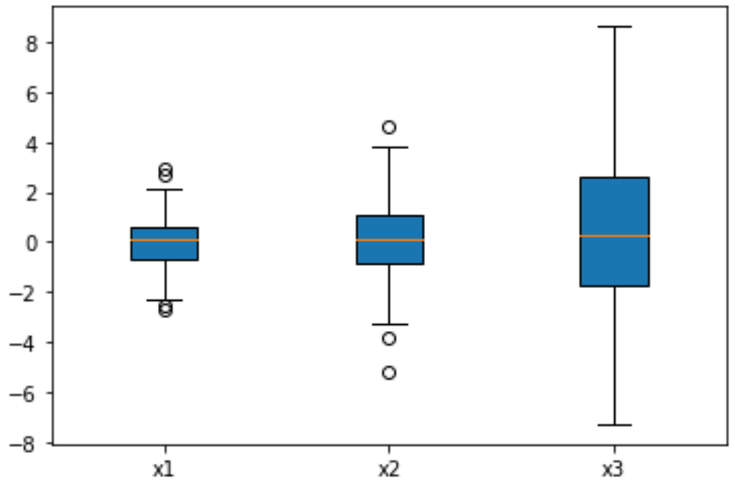


Box plot

- `plt.boxplot(x)` creates a box plot
- Important parameters:
  - `x`: Specifies the input data. It specifies either a 1D array for a single box or a sequence of arrays for multiple boxes.
  - `notch` (Optional): If true, notches will be added to the plot to indicate the confidence interval around the median
  - `labels`: Optional: Specifies the labels as a sequence
  - `showfliers`: Optional: By default, it is true, and outliers are plotted beyond the caps
  - `showmeans`: Optional: If true, arithmetic means are shown

Box plot

```
import random
import numpy as np

# generate some random test data
all_data = [np.random.normal(0, std, size=100) for std in range(1, 4)]
labels = ['x1', 'x2', 'x3']

plt.boxplot(all_data, vert=True,  # vertical box alignment
        patch_artist=True,        # fill with color
        labels=labels)            # will be used to label x-ticks
plt.show()
```

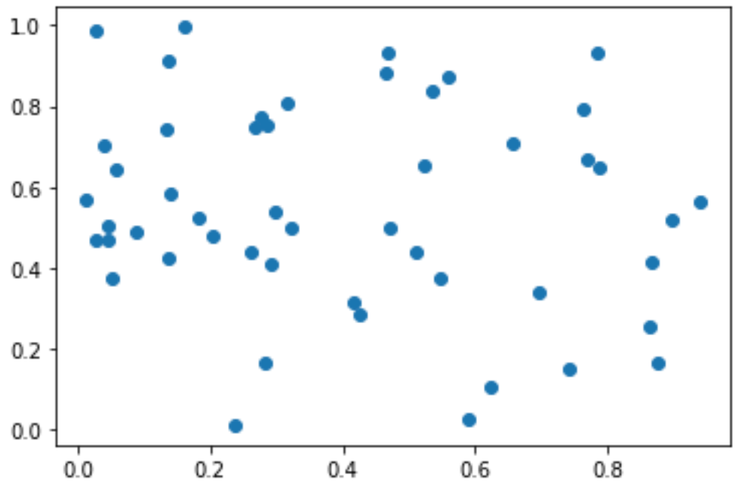


Scatter plot

- `plt.scatter(x, y)` creates a scatter plot of y versus x with optionally varying marker size and/or color
- Important parameters:
  - `x`, `y`: Specifies the data positions
  - `s` (Optional): Specifies the marker size in points squared
  - `c` (Optional): Specifies the marker color. If a sequence of numbers is specified, the numbers will be mapped to colors of the color map

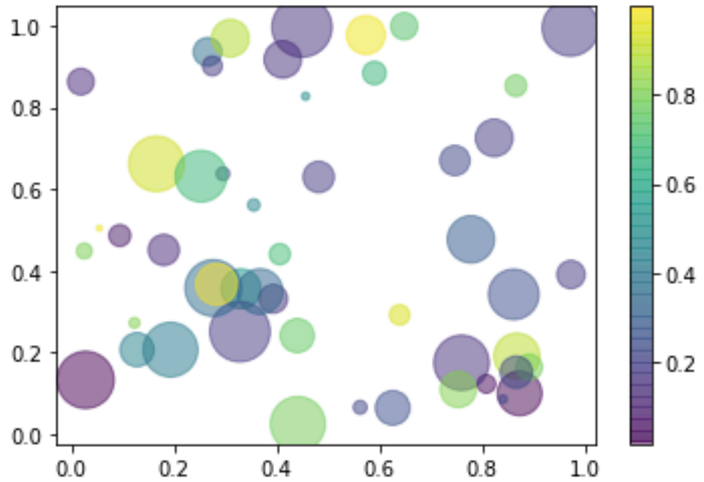
Scatter plot

```
import numpy as np

N = 50

# np.random.rand(N): Return a sample (or samples) from the “standard normal” distribution
x = np.random.rand(N)
y = np.random.rand(N)

plt.scatter(x, y)
plt.show()
```



Bubble plot

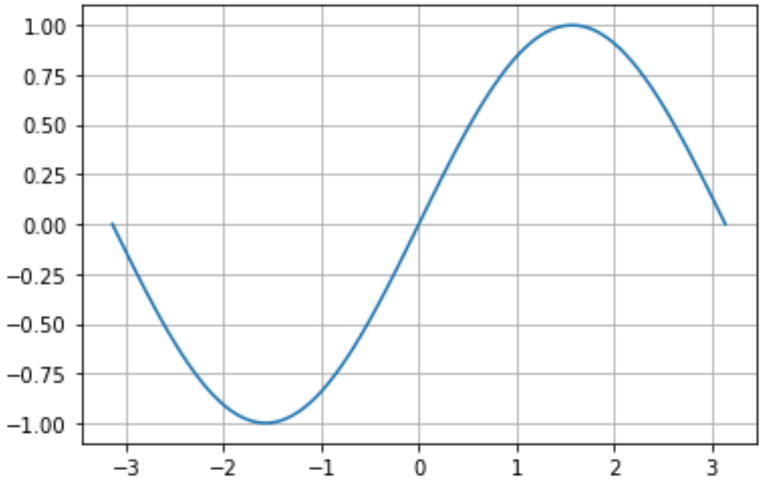
```
N = 50

# np.random.rand(N): Return a sample (or samples) from the “standard normal” distribution
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.colorbar()
plt.show()
```



Other applications

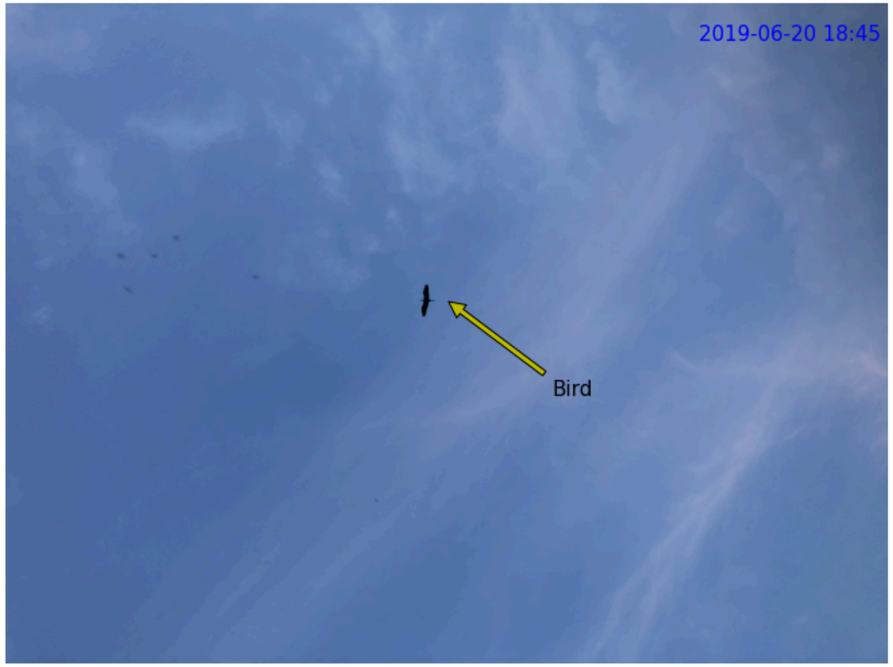
Plotting mathematical functions

```
import numpy as np
import math

# Creating vectors X and Y
x = np.linspace(-math.pi, math.pi, 100)
y = np.sin(x)

plt.plot(x, y)
plt.grid()
plt.show()
```



Loading images

```
import matplotlib.image as mpimg

# Create a figure object
fig = plt.figure(figsize=(12,9))

# Add one subplot
ax1 = fig.add_subplot(1,1,1)

# Read the image
img = mpimg.imread('data/20190620185840.JPG')

# Annotate on the figure
ax1.annotate('2019-06-20 18:45', xy=(4000, 100), va='top', ha='right', size=15, color='b')
ax1.annotate('Bird', xy=(2000, 1350), xytext=(2500,1800), size=15, ha='left',
             arrowprops=dict(facecolor='y', shrink=0.05))

plt.imshow(img)
plt.axis('off')

plt.show()
```



References

- Part of this slide set is prepared or/and extracted from the following resources:
  - Mario Dobler and Tim Gromann (2019): “Data Visualization with Python: Create an impact with meaningful data insights using interactive and engaging visuals”, Packt Publishing
  - Matplotlib: https://matplotlib.org/
- This set of slides is for teaching purpose only