Visualization: Part 1

CISC 3225
Fall 2024
PDSH 5.9, DSFS 10.8

Matplotlib

- A time-tested, solid library for data visualization
- Biggest strength: Deals with many platform's graphical backends to create consistent visualizations anywhere.
- Biggest weakness: Older, low-level, requires you to do a lot of work to make graphics look good.
    - Primarily deals with NumPy arrays
    - Not aware of Pandas objects!

Seaborn

- Built on top of Matplotlib
- Adds functionality
  - Stylistic opinions
  - Complex graph types
  - Focus on Pandas data types
- Still need familiarity with Matplotlib
  - Underlying concepts leak into Seaborn
  - Use Matplotlib functionality for control over the graph

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

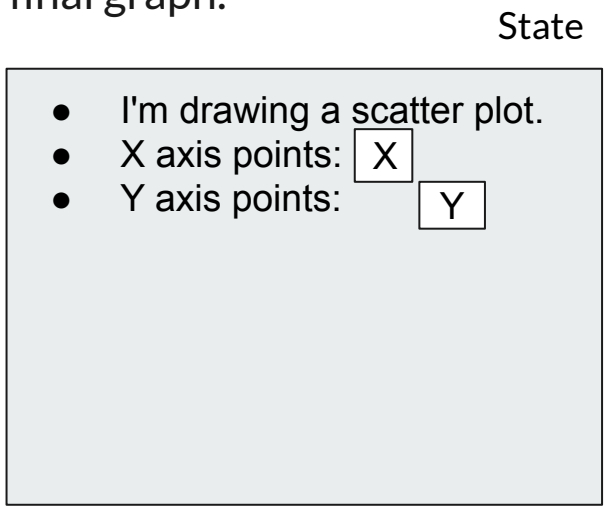
```
from matplotlib import pyplot as plt
```

State

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

```
from matplotlib import pyplot as plt

plt.scatter(X, Y)
```



The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

```
from matplotlib import pyplot as plt

plt.scatter(X, Y)
plt.title("Test graph")
```

State

- I'm drawing a scatter plot.
- X axis points: X
- Y axis points: Y
- My title is "Test graph"

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

```
from matplotlib import pyplot as plt

plt.scatter(X, Y)
plt.title("Test graph")
plt.xlabel("Thing 1")
```

State

- I'm drawing a scatter plot.
- X axis points: X
- Y axis points: Y
- My title is "Test graph"
- X-axis label: "Thing 1"

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

```
from matplotlib import pyplot as plt

plt.scatter(X, Y)
plt.title("Test graph")
plt.xlabel("Thing 1")
plt.ylabel("Thing 2")
```

State

- I'm drawing a scatter plot.
- X axis points: X
- Y axis points: Y
- My title is "Test graph"
- X-axis label: "Thing 1"
- Y-axis label: "Thing 2"

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

```
from matplotlib import pyplot as plt

plt.scatter(X, Y)
plt.title("Test graph")
plt.xlabel("Thing 1")
plt.ylabel("Thing 2")
plt.show()
```

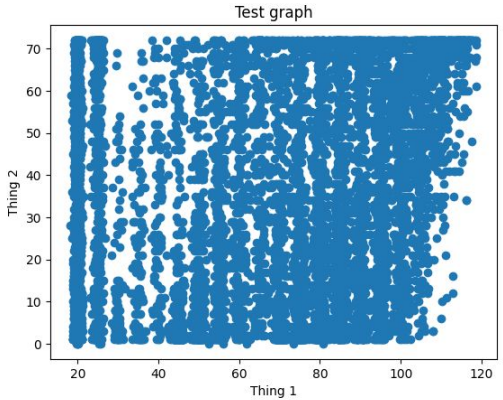
State

- I'm drawing a scatter plot.
- X axis points: X
- Y axis points: Y
- My title is "Test graph"
- X-axis label: "Thing 1"
- Y-axis label: "Thing 2"

Time to render!

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.



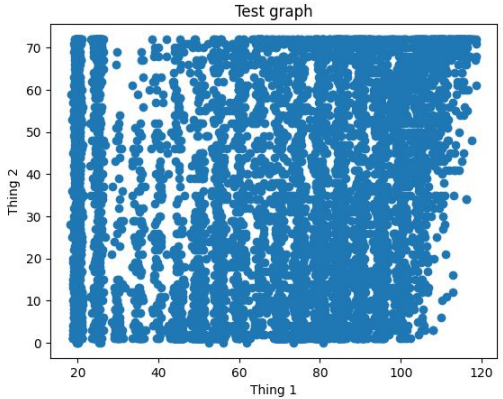
State

- I'm drawing a scatter plot.
- X axis points: X
- Y axis points: Y
- My title is "Test graph"
- X-axis label: "Thing 1"
- Y-axis label: "Thing 2"

Time to render!

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.



Drawing a graph clears any accumulated state

State

The Matplotlib Global State Machine

While there are other ways to interact with Matplotlib, typical usage *accumulates sequences of commands* before drawing the final graph.

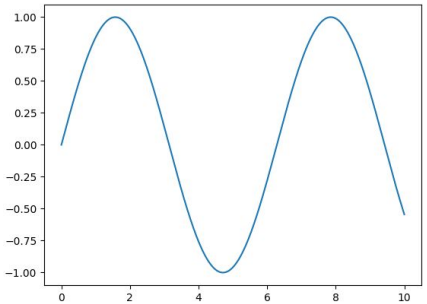
```
from matplotlib import pyplot as plt

plt.scatter(X, Y)
plt.title("Test graph")
plt.xlabel("Thing 1")
plt.ylabel("Thing 2")
plt.show()
```

Note: `plt.show()` is required in a script, but unnecessary in a notebook environment. Leftover accumulated state in a notebook cell will always be rendered and cleared when a cell executes.

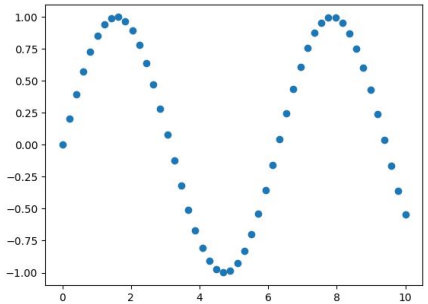
However, it is good practice to call `plt.show()` explicitly.

Types of Plots

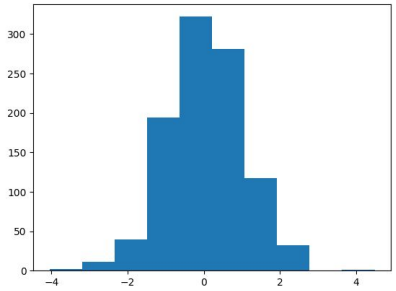
Line



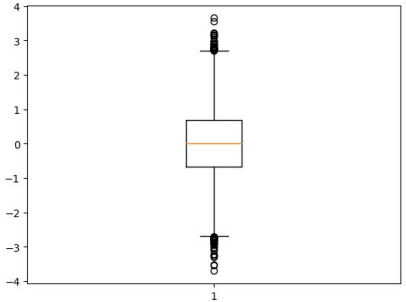
Scatter



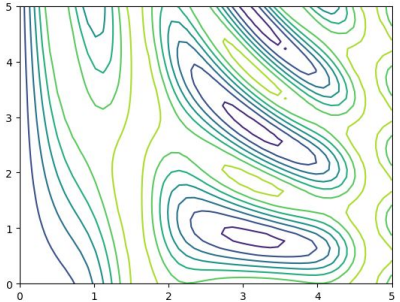
Histogram/Bar Chart



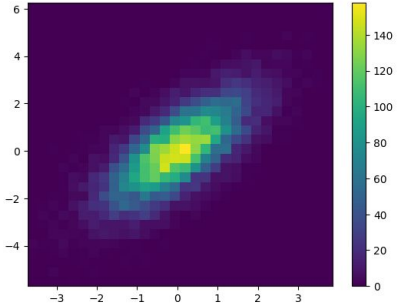
Box Plot



Contour

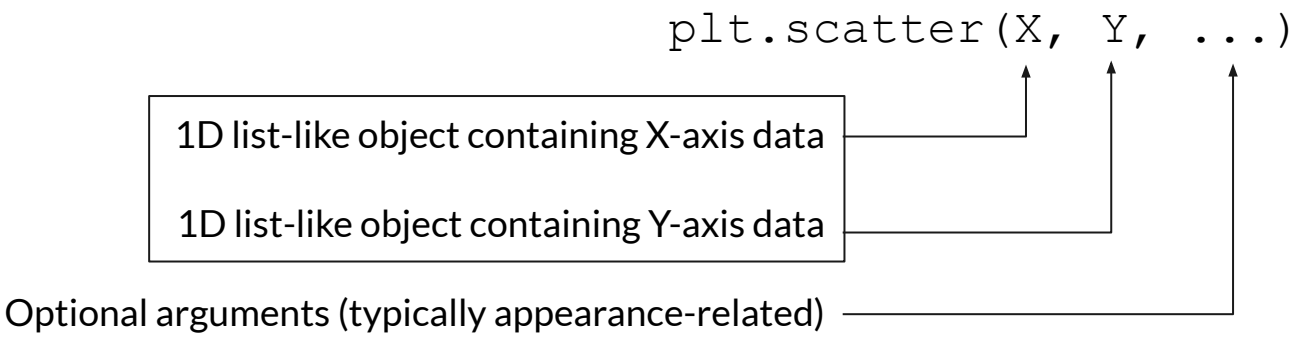


2D Histogram / Image



Many others listed here: https://matplotlib.org/stable/api/pyplot_summary.html

Typical Matplotlib call



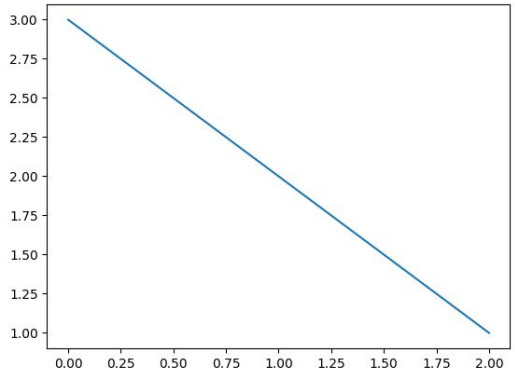
Note: The initial arguments to graphing functions are, unless indicated otherwise:

- Always 1D list-like objects
- Numeric
- The number of list-like arguments depends on the style of graph

Typical Matplotlib call: Optional inputs

```
A = [3, 2, 1]
B = [0, 1, 10]
```

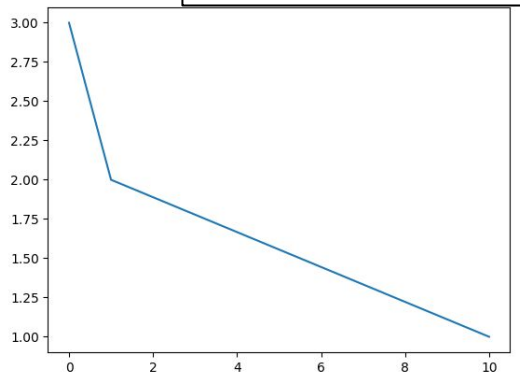
```
plt.plot(A)
```



A contains y-axis points
x-axis assumed to be 0, 1, 2, …

A contains y-axis points
B contains x-axis points

```
plt.plot(B, A)
```



Matplotlib demo

- Axis labels
- Axis ranges
- Titles
- Plotting multiple things at once
- Colors
- Labels
- Legends
- Complex plotting example with California cities