Fairfield University
DOLAN
School of Business

Python for Analytics

The Matplotlib Library

[The Matplotlib Docs]

Learning Objectives

- **Theory:** You should be able to explain ...
  - The basic kinds of plots
  - Matplotlib's 3 step plotting process
  - How plots are decorated with various options
  - The limitations of Matplotlib's 2D plotting model
- **Skills:** You should know how to ...
  - Create and display common types of plots
  - Pass data and set options for each plot type
  - Find your way around the documentation
  - Manipulate and Plot Time Series data

Overview

Making professional-quality plots from 2D arrays
(vanilla Python, NumPy, Pandas)

What's Matplotlib?

From the docs ...

> “Matplotlib is a library for making **2D plots of arrays** in Python … designed with the philosophy that **you should be able to create simple plots with just a few commands**, or just one! If you want to see a histogram of your data, you shouldn’t need to instantiate objects, call methods, set properties, and so on; **it should just work**.”

Goal: Easy, Professional Plots

More from the docs …

> “Plots should look great - publication quality. One important requirement … is that the text looks good (antialiased, etc.)
>
> “Code should be easy enough that [anyone] can understand it and extend it.
>
> “Making plots should be easy.”

Powerful and Flexible API

Input is 2D data in many possible formats:

- Lists, tuples, … arrays from NumPy or Pandas

Can produce a wide variety of plots

- Line, Bar, Pie, and Scatter plots
- Histograms and Spectrograms
- Error plots, Box and Whisker plots, and Violin plots
- Polar plots and Hexagonal Binning ("heatmap") plots
- ...

Standard Imports

The remaining slides assume that we have already imported NumPy, Pandas, and Matplotlib in the standard way.

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

The `pyplot` module

It all starts here

Intuitive 3 Step Process

1. Make a new plot

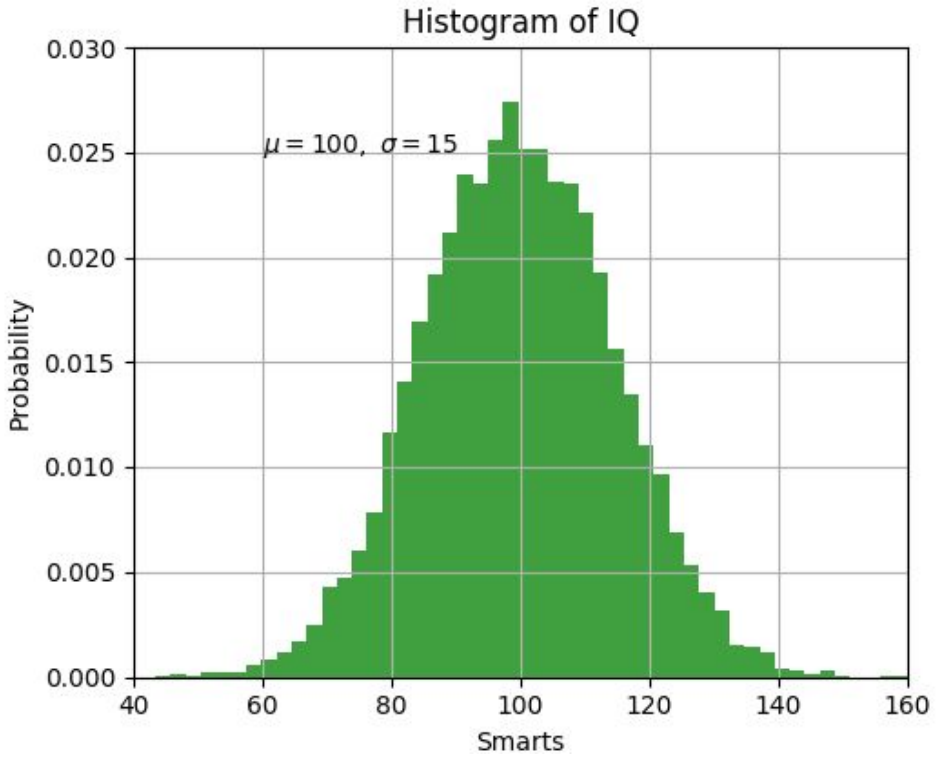
```
plt.hist(...)
```

2. Set a few options

```
plt.xlabel('Smarts')
...
```

3. Display the plot

```
plt.show()
```



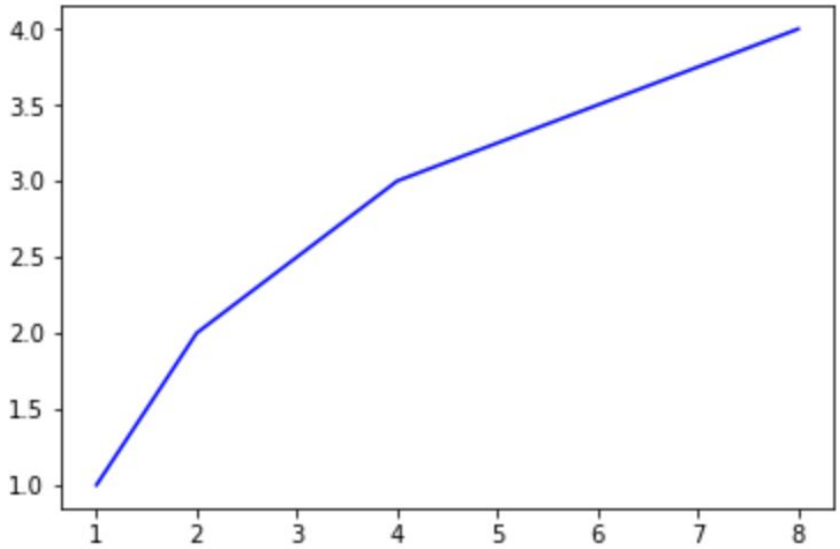
Note: Data is in Columns

`pyplot` generally assumes that data is passed as 1D sequences (lists, Pandas Series, NumPy arrays, etc.)

If using DataFrames or other 2D arrays, then **slice column-wise** to define x-coordinates and y-coordinates.

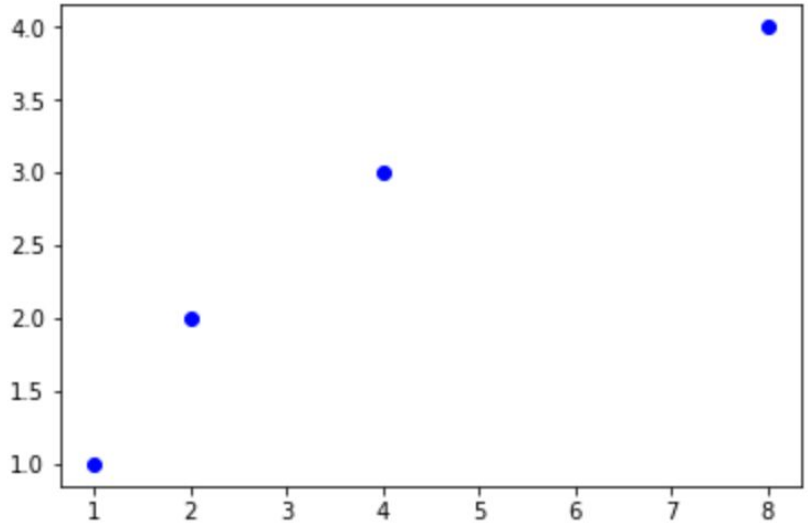
Line Chart

```
# A basic 2D plot
plt.plot(
    [1,2,4,8],   # x
    [1,2,3,4],   # y
    "b-"         # blue line
)
plt.show()
```



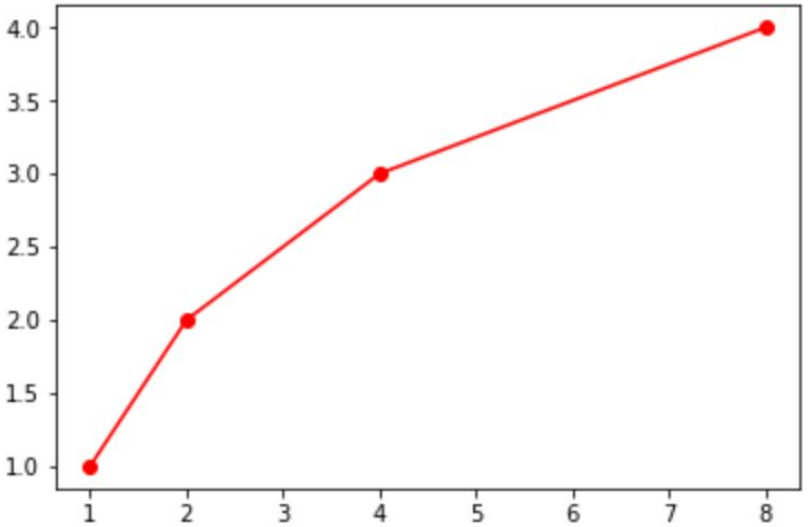
Simple Point Plot

```
# A basic 2D plot
plt.plot(
    [1,2,4,8],   # x
    [1,2,3,4],   # y
    "bo"         # blue dots
)
plt.show()
```



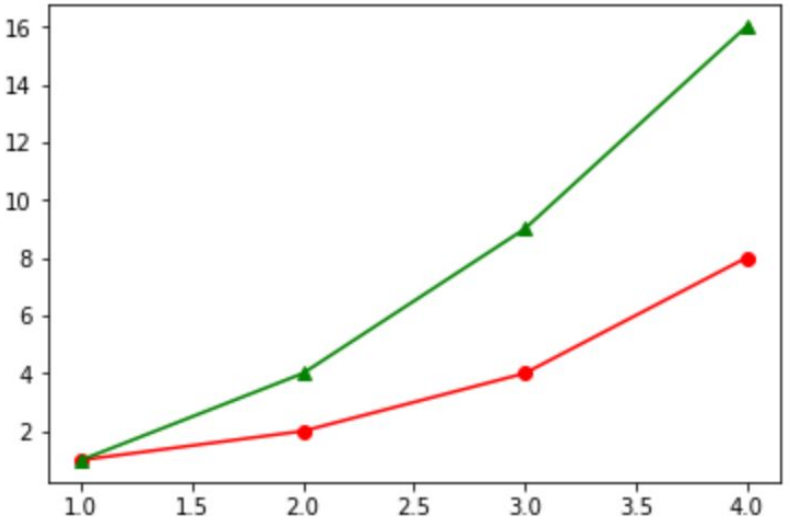
Line Chart with Markers

```
# A basic 2D plot
plt.plot(
    [1,2,4,8],   # x
    [1,2,3,4],   # y
    "r-o"        # line/dot
)
plt.show()
```



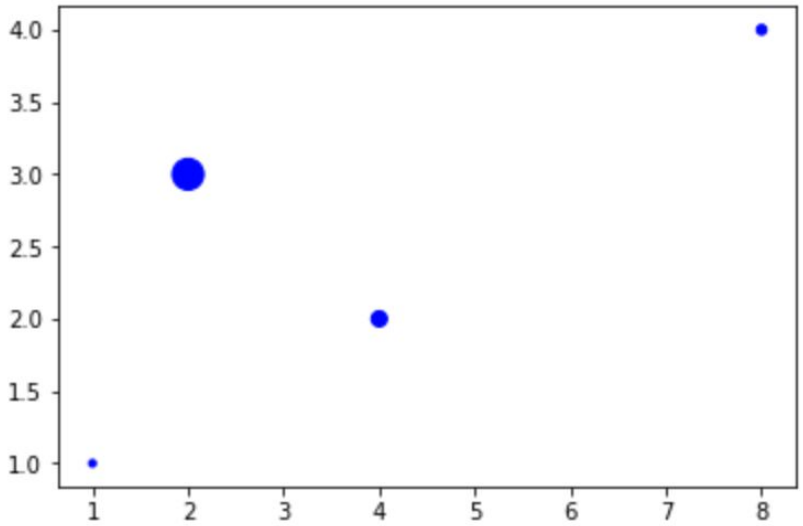
Multiple Lines

```
plt.plot(
    [1,2,3,4],   # series 1
    [1,2,4,8],
    "r-o"
    [1,2,3,4],   # series 2
    [1,4,9,16],
    "g-^"
)
plt.show()
```



Scatter Plot

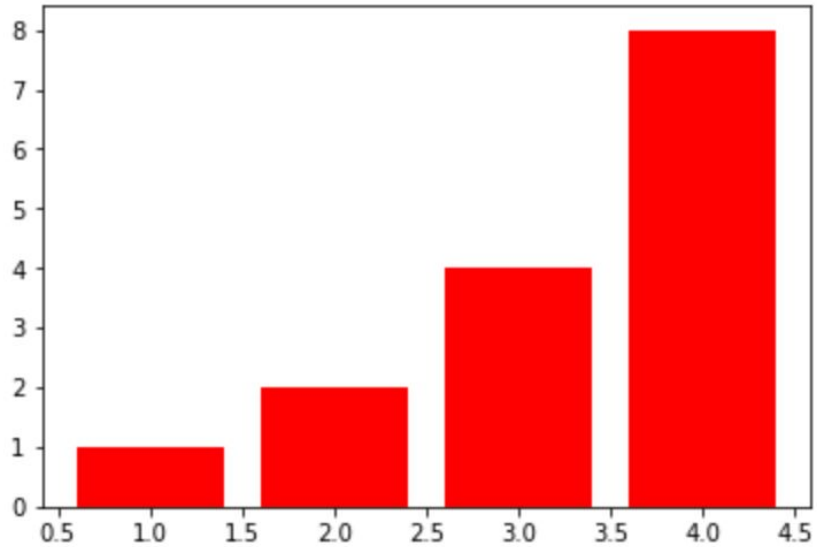
```
plt.scatter(
    [1,4,2,8],          # x
    [1,2,3,4],          # y
    c='b',              # blue
    s=[10,50,200,20] #sizes
)
plt.show()
```



Bar Chart

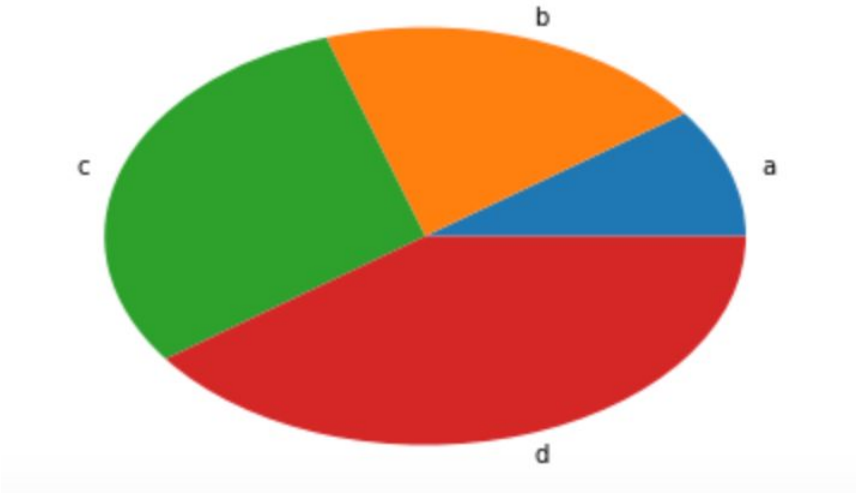
```
plt.bar(
    [1,2,3,4],   # x
    [1,2,4,8],   # y
    color = "r"
)
plt.show()

# more options in the docs
```



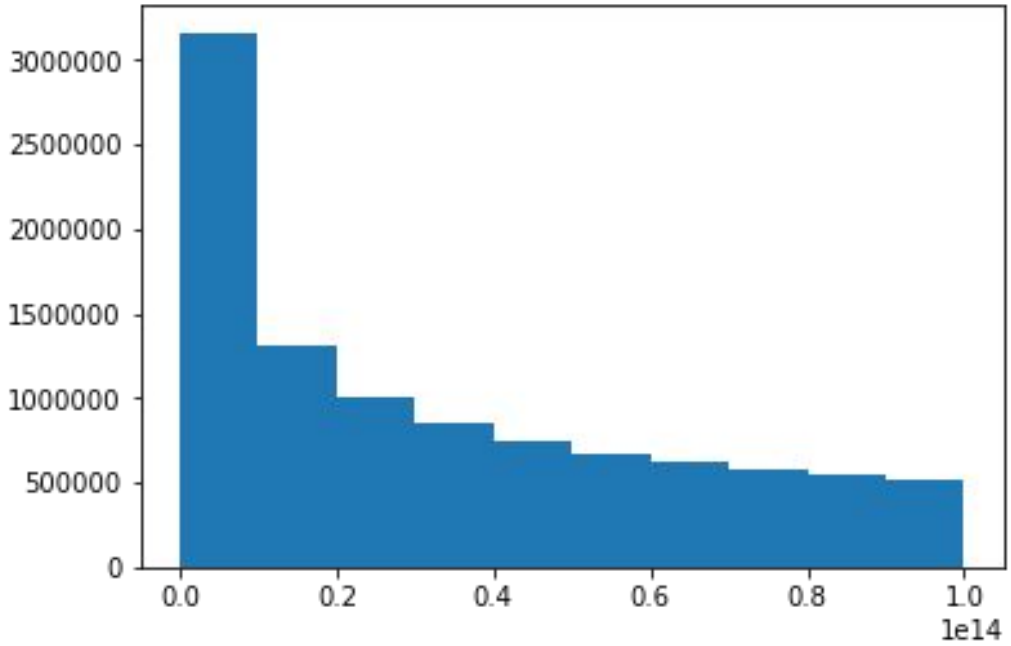
Pie Chart

```
plt.pie(
    [1,2,3,4], #x
    labels=['a','b','c','d']
)
plt.show()
```



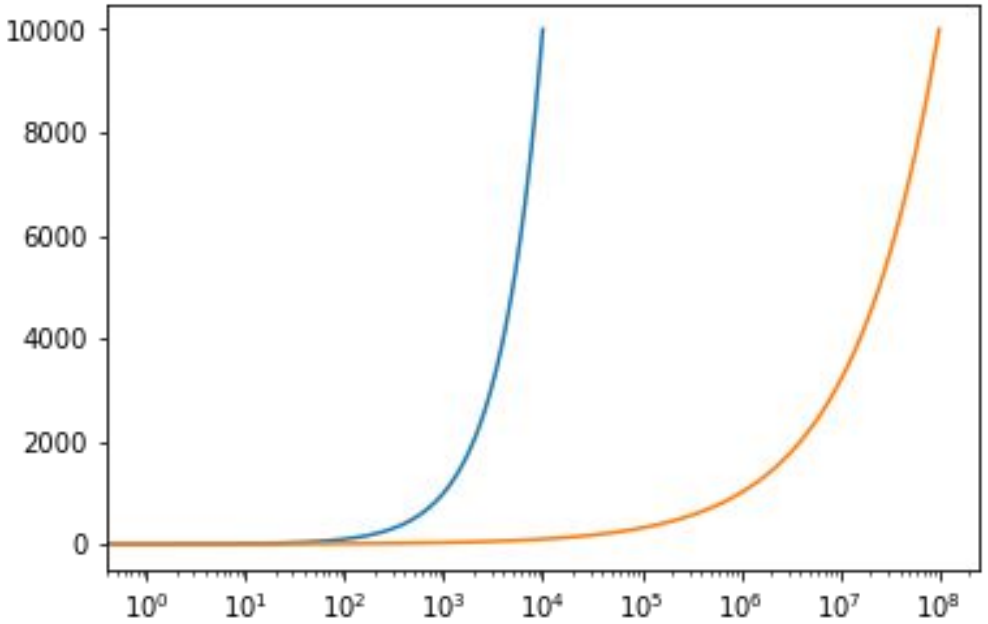
Histogram

```
x=[x**2 for x in
    range(10000000)]
plt.hist(x, bins=10)
plt.show()
```



Scaling Axes

```
x=np.arange(10000)
plt.plot(x,x,x**2,x)
plt.xscale('log')
plt.show()
```



Labels, Gridlines, etc.

We can use plotting commands (functions) to decorate our plots with x-labels, y-labels, gridlines, annotations, etc.

Each decoration can have styling properties for the text, lines, color fills, etc.

When all else fails, RTFM:

https://matplotlib.org/api/pyplot_summary.html

Time Series Data

With an assist from Pandas

Time Series docs: https://pandas.pydata.org/pandas-docs/stable/timeseries.html

Time Series in Pandas

Pandas arrays (Series or DataFrame) can be indexed using timestamps instead of numbers or string labels.

```
aapl_5d = pd.Series(
        [158.67,158.73,156.07,153.39,151.89],
        index=pd.date_range('2017-9-18',periods=5,freq='D'))
```

```
aapl_5d  →   2017-09-18    158.67
             2017-09-19    158.73
             2017-09-20    156.07
             2017-09-21    153.39
             2017-09-22    151.89
             Freq: D, dtype: float64
```

The `DatetimeIndex` Type

When a date range is used as an index, Pandas automatically converts it to a **`DatetimeIndex`**.

```
appl_5d.index

→ DatetimeIndex(['2017-09-18', '2017-09-19',
                 '2017-09-20', '2017-09-21',
                 '2017-09-22'],
                dtype='datetime64[ns]', freq='D')
```

Time Series Slicing

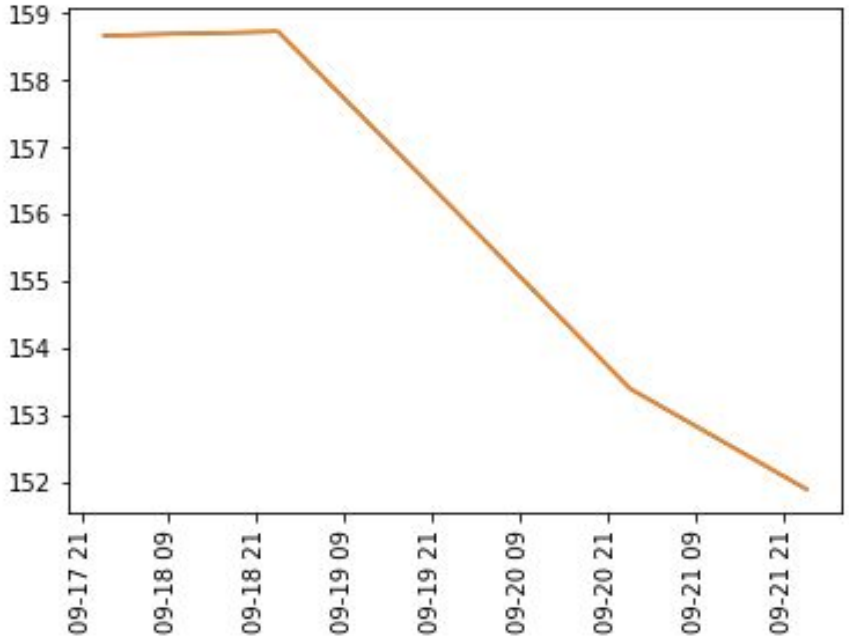
`DatetimeIndex` is optimized for slicing date ranges.

```
appl_5d[:'2017-09-20']
→  2017-09-18    158.67
   2017-09-19    158.73
   2017-09-20    156.07
   Freq: D, dtype: float64
appl_5d['2017-09-20':]
→  2017-09-20    156.07
   2017-09-21    153.39
   2017-09-22    151.89
   Freq: D, dtype: float64
```

Dates strings are treated like labels, so slices include the end points.

A Time Series in pyplot

```
plt.plot(aapl_5d)
plt.xticks(rotation='vertical')
plt.show()
```

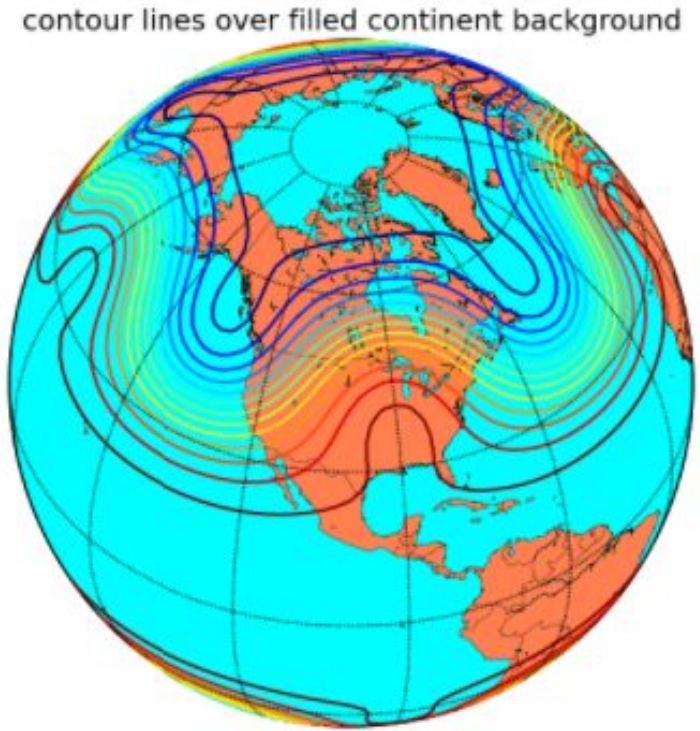


Toolkits

Extensions that go beyond `pyplot`'s 2D model

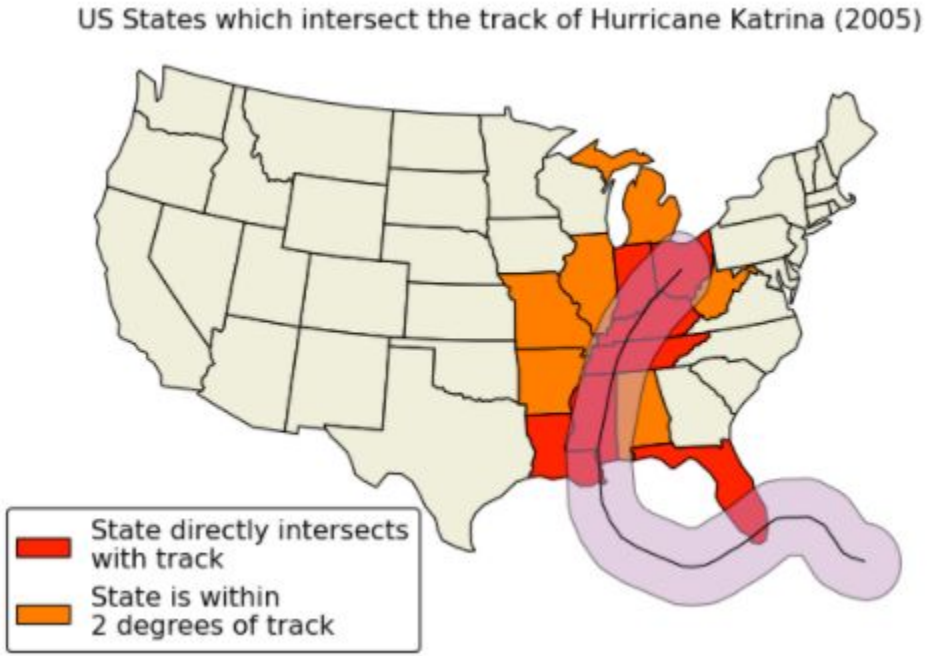
Basemap

Creates map projections with political boundaries.



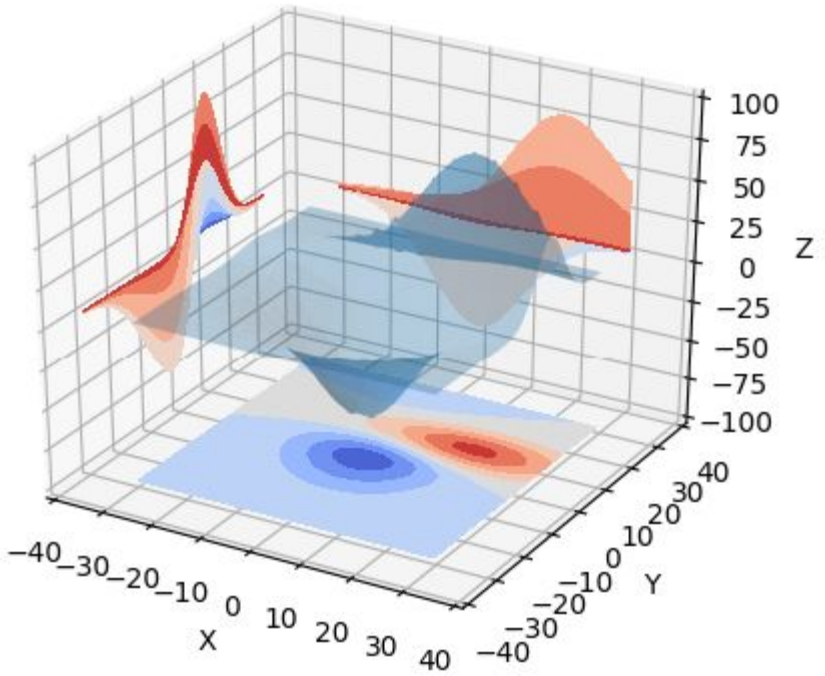
Cartopy

Yet another mapping library.



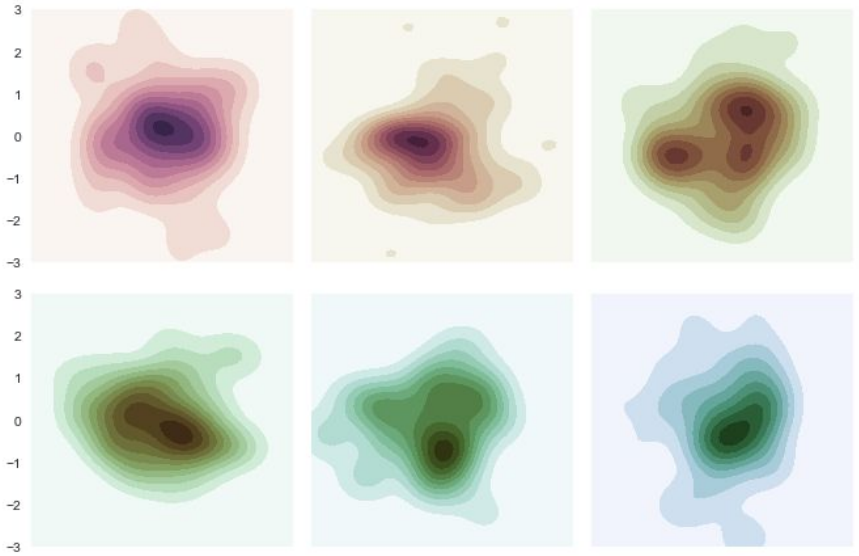
Matplot3d

Provides 3D extensions of the basic plots.



Seaborn

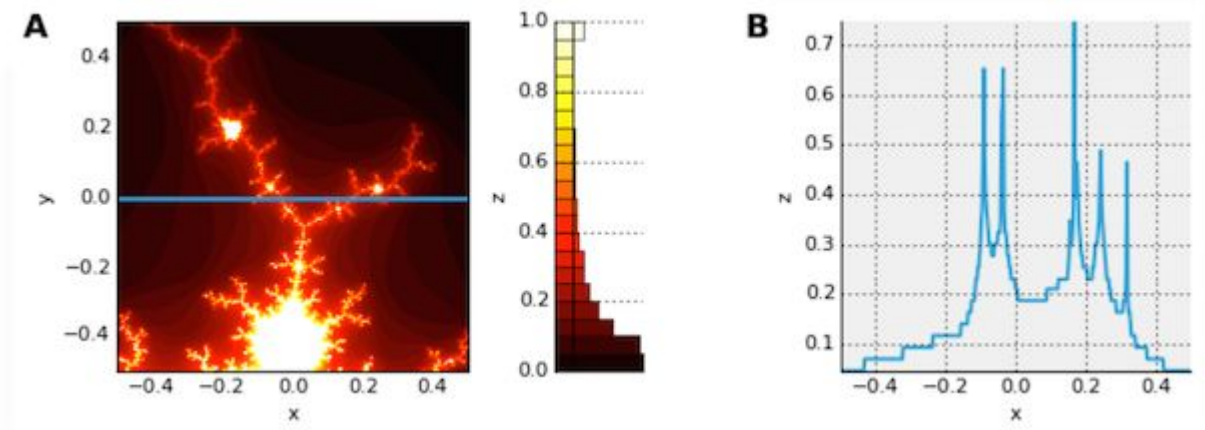
Provides deeper integration with Pandas, NumPy, and Scipy to simplify plots from linear regression, clustering, time series, and other common analyses.



Holoviews

Make plots into objects that can be recalled and shown in a variety of layouts.

Very handy for Jupyter notebooks.



Fairfield University
DOLAN
School of Business

Python for Analytics

The Matplotlib Library

The Matplotlib Docs