Graphing with Matplotlib

Thomas Schwarz, SJ

Matplotlib

- Matplotlib was conceived in 2002 to enable Matlab style graphics
- Very universal
- Upgraded with Seaborn etc.

Matplotlib

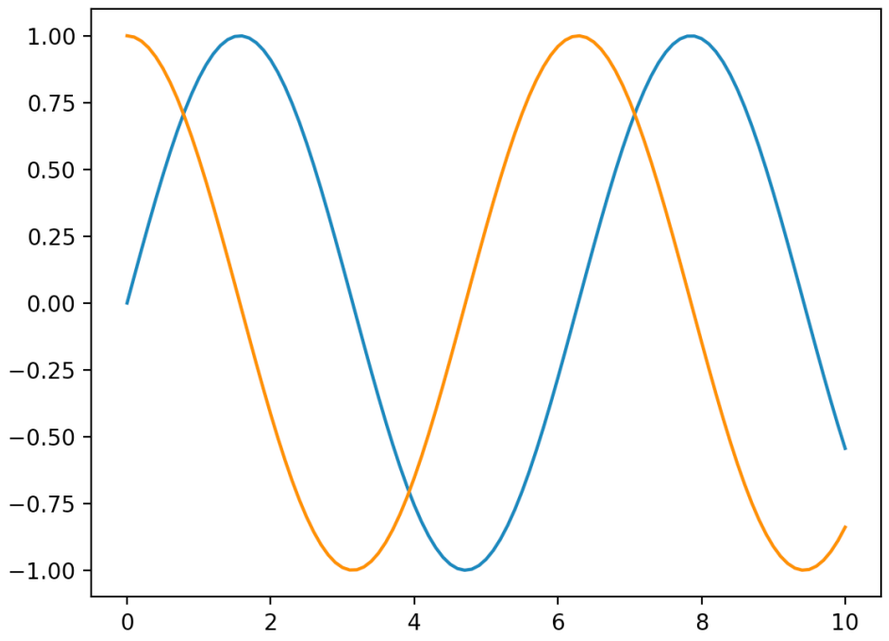
- Simple example

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

x = np.linspace(0,10, 101)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```

Create a sample of x-values: 0, 0.1, 0.2, … , 9.8, 9.9, 10.0



Matplotlib

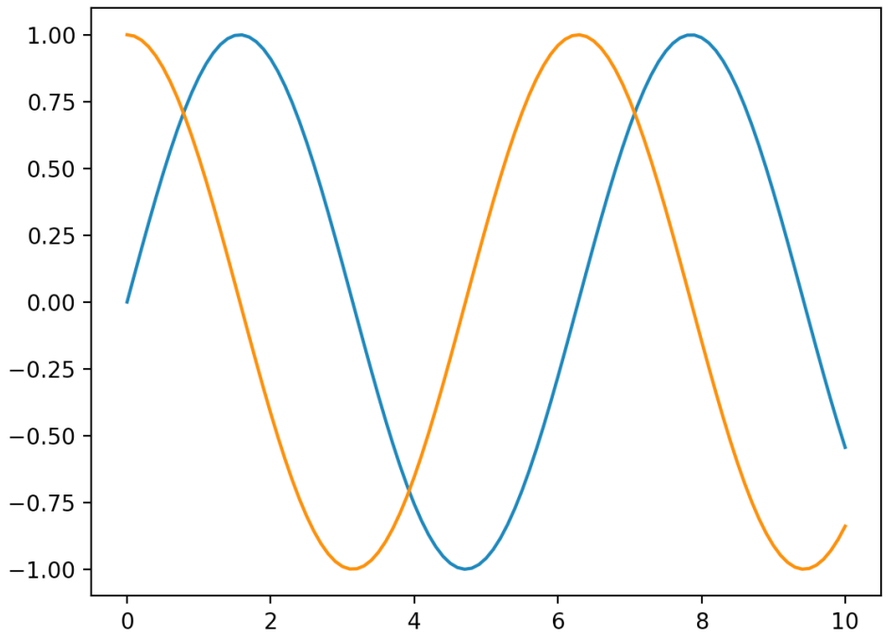
- Simple example

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

x = np.linspace(0,10, 101)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```

np.sin applies the sine function to all elements in x



Matplotlib

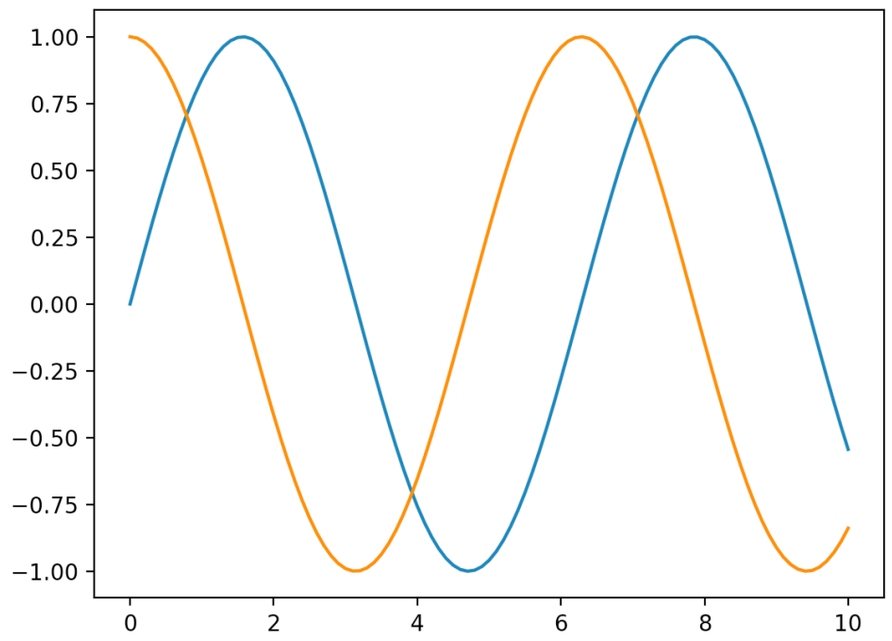
- Simple example

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

x = np.linspace(0,10, 101)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```

plot draws a line graph



Matplotlib

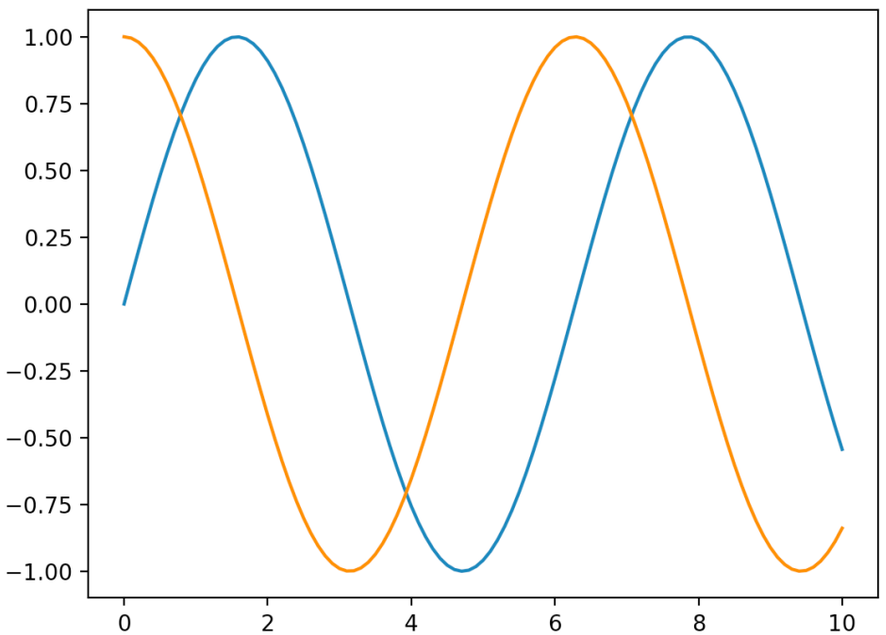
- Simple example

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

x = np.linspace(0,10, 101)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```

plt show is necessary
Consult manual for
IPython



- plt.show does a lot of work

Matplotlib

- We can use different plotstyles

```
>>> plt.style.available
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid'
, 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'gra
yscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'sea
born-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seabor
n-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk'
, 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

Matplotlib

- Matplotlib creates one figure
  - Each plot in the figure is called an axes
    - `fig, ax = plt.subplots(2,1)`
      - `ax` is an array of axes
      - parameters are nr. of rows and nr. of columns
- To make a title for the figure, use `suptitle`
  - `plt.suptitle('Some trig functions', fontsize='x-large')`

Plotting

- For each axes, use plot
  - Can set color, line-style, line-width, and label for a legend
  - Can also set title for the sup-plot

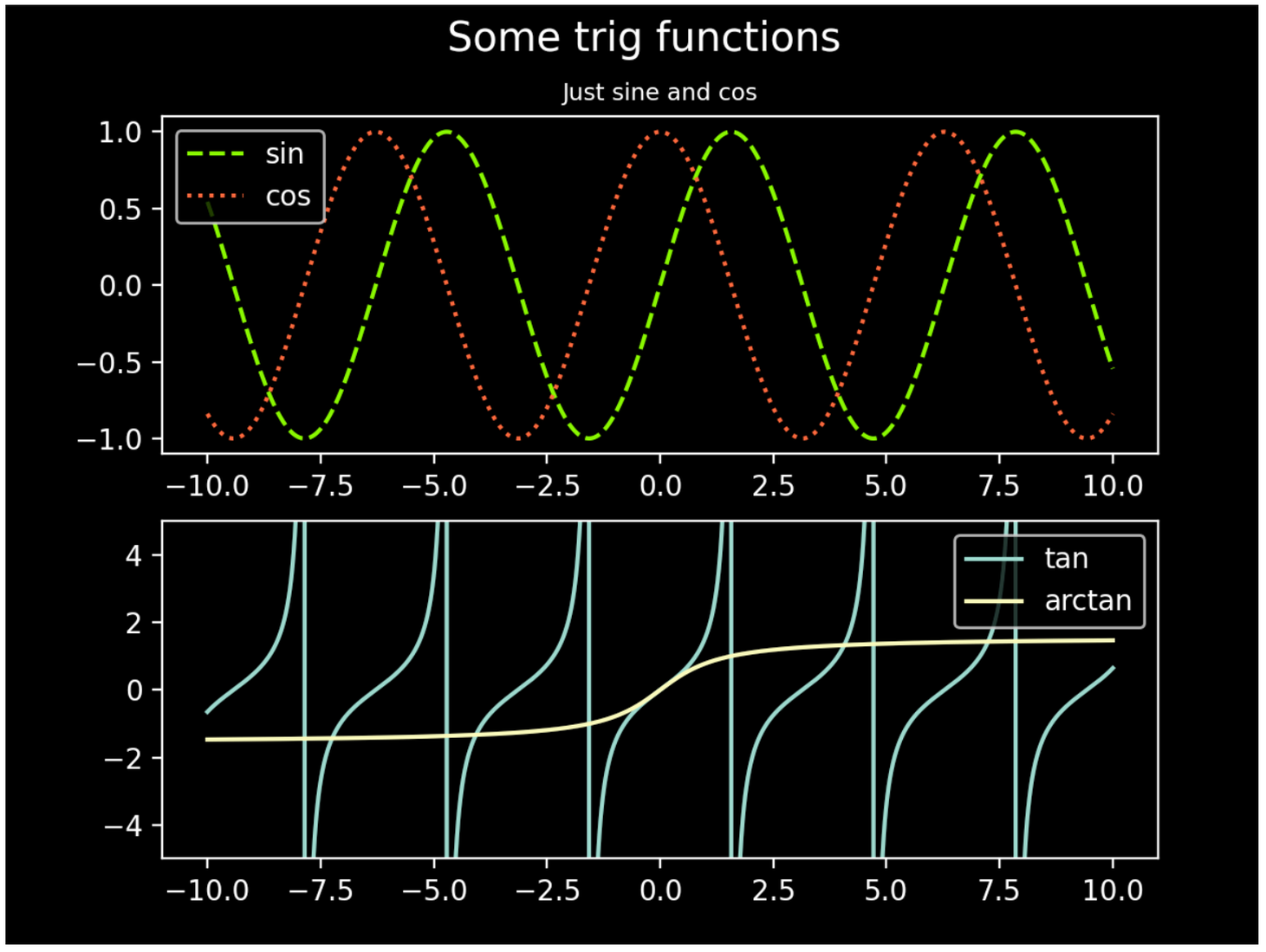
```
ax[0].plot(x, np.sin(x), color='chartreuse', linestyle='--', label='sin')
ax[0].plot(x, np.cos(x), color='#FF5533', linestyle = ':', label='cos')
ax[0].set_title('Just sine and cos', fontsize='small')
ax[0].legend(loc='best')

ax[1].plot(x, np.tan(x), label='tan')
ax[1].plot(x, np.arctan(x), label='arctan')
ax[1].set_ylim(-5,5)
ax[1].legend(loc='best')
```

y-range from -5 to 5

Plotting

- AND DO NOT FORGET TO INCLUDE
  - `plt.show()`



Scatter Graphs

- Let’s get some data
  - Indian cities file `in.csv`
  - Import with Pandas because it is easier

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

cities = pd.read_csv('in.csv')
```

Scatter Graphs

- We need to look at the data frame
  - Indexed with a default index
  - First line defines headers in the default

```
>>> cities
           city        lat  ...  population_proper markersize
0        Mumbai  18.987807  ...          12691836.0      197.0
1         Delhi  28.651952  ...           7633213.0      166.0
2       Kolkata  22.562627  ...           4631392.0      155.0
3       Chennai  13.084622  ...           4328063.0       78.0
4     Bengalūru  12.977063  ...           5104047.0       75.0
..          ...        ...  ...                 ...        ...
207     Calicut  11.248016  ...                 NaN        NaN
208  Kagaznāgār  19.331589  ...                 NaN        NaN
209      Jaipur  26.913312  ...                 NaN        NaN
210  Ghandinagar  23.216667  ...                 NaN        NaN
211   Panchkula  30.691512  ...                 NaN        NaN

[212 rows x 10 columns]
```

Scatter Graphs

- We use latitude (y-axis) and longitude (x-axis) to plot ovals
  - Calculate the size of the ovals from the population
  - In Pandas, we can just add a new column markersize

```
cities['markersize'] = np.round(np.log10(cities.population)
+cities.population/100000)
```

- We can also create a new data frame for cities with population > 500000

```
bigcities = cities[cities.population>500000]
```

Scatter Graphs

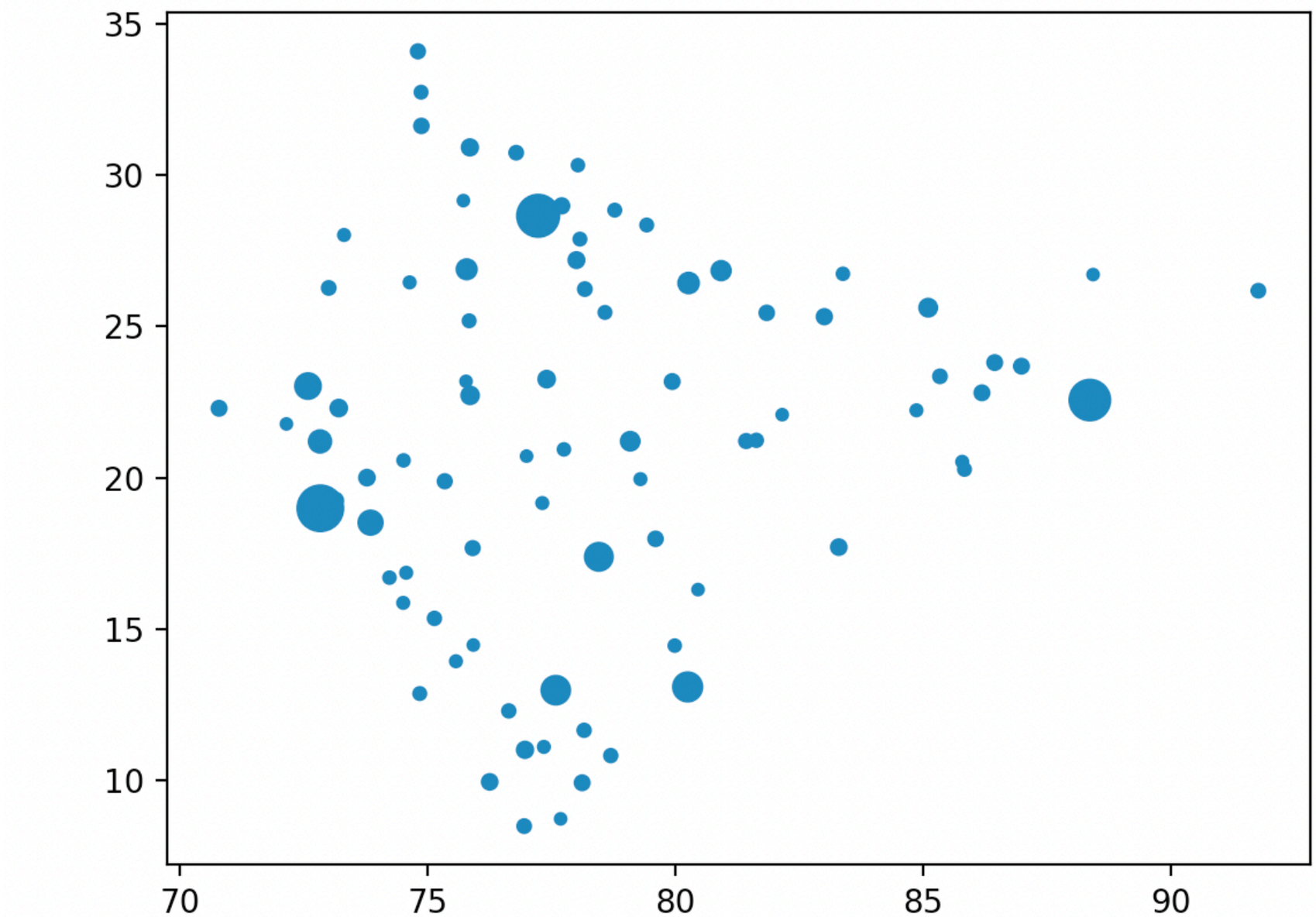
- And now we use scatter
  - X-values as an np.array-ish thing
  - Y-values as an np.array-ish thing
  - c for color values
  - s for marker size

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

cities = pd.read_csv('in.csv')
cities['markersize'] = np.round(np.log10(cities.population)+cities.population/100000)
bigcities = cities[cities.population>500000]

plt.scatter(bigcities.lng, bigcities.lat, s=bigcities.markersize)
plt.show()
```

Scatter Graphs



Scatter

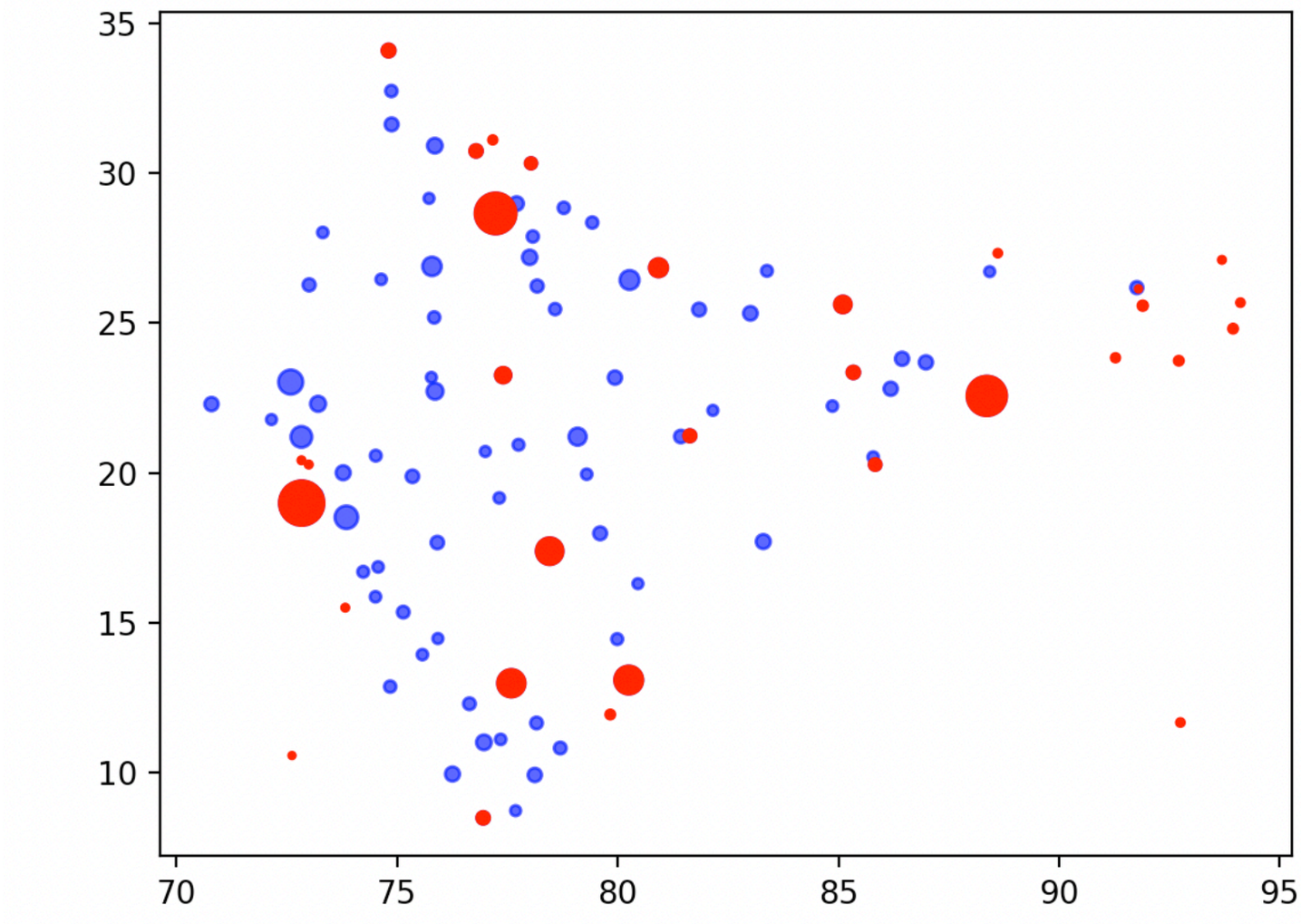
- We can create a different list of capital cities
  - `capitals = cities[cities.capital=='admin']`
  - The big cities are plotted in blue, but with a transparency of 70% (`alpha=0.7`)

```
plt.scatter(bigcities.lng, bigcities.lat,
            s=bigcities.markersize, c='blue', alpha=0.7)
```

- The capitals are plotted in red

```
plt.scatter(capitals.lng, capitals.lat,
            s=capitals.markersize, c='red')
```

Scatter



Using Maps

- There exist several projects to allow you to draw maps
  - Goes beyond this class