Lecture 13: ES 102
Plotting With `Matplotlib`

Introduction to Computing

IIT Gandhinagar, India

Oct, 2013

Simple Plotting

- `Matplotlib` provides the necessary functions for data visualization. `pylab` is an interface to matplotlib.
- Use `import pylab as pl` to use matplotlib.
- Matplotlib is generally used in conjunction with numpy. Use `import numpy as np` for that.
- This lecture is based on `http://scipy-lectures.github.io/intro/matplotlib/matplotlib.html`. Visit the site for more information.

```
1  import numpy as np
2  import pylab as pl
3
4  x=np.arange(-4,4,0.1)
5
6  s = np.sin(x) # sin defined in math and numpy
7                # behaves differently
8  q = x*x  # element-wise multiplication
9
10 pl.plot(x,s)
11 pl.plot(x,q)
12
13 pl.show()
```

Simple Plotting

- `plot()` generates an object that can be plotted.
- `show()` actually draws all the plottable objects created.

Customizing

```
1  import numpy as np
2  import pylab as pl
3
4  x=np.linspace(-np.pi,np.pi,200,endpoint=True)
5
6  s = np.sin(x)
7  c = np.cos(x)
8  pl.plot(x,s,color='red',linewidth=1,linestyle='--',label="sine")
9  pl.plot(x,c,color='blue',linewidth=1.5,linestyle='-',label="cosine")
10
11 pl.legend(loc='upper right')
12
13 pl.show()
```

- `linspace(start,end,no_of_elms,endpoint=True/False)`: generate `no_of_elms` many equally spaced elements from `start` to `end`. `end` will not be included only if `endpoint=False`.
- `color` color of the curve;`linewidth` thickness of the curve;`linestyle` curve pattern; `label` set the name of the curve.
- `legend` where to show the names of the curves.

Customizing: Axes Limit, Ticks and Grid

```
1  import numpy as np
2  import pylab as pl
3
4  x=np.linspace(-2*np.pi,2*np.pi,200,endpoint=True)
5
6  s = np.sin(x)
7  pl.plot(x,s,color='green',linewidth=1,linestyle='-')
8
9  pl.xlim(x.min()*1.1,x.max()*1.1)
10 pl.ylim(-1.1,1.1)
11
12 pl.xticks([-2*np.pi, -np.pi, 0, np.pi, 2*np.pi])
13 pl.yticks([-1, 0, +1])
14
15 pl.grid(True)
16 pl.show()
```

- `xlim` sets x-axis limit, `ylim` sets y-axis limit.
- `xticks,yticks` puts the ticks in the axes.
- `grid` sets a rectangular grid.

Customizing: Changing Axes

```
1  import numpy as np
2  import pylab as pl
3
4  x=np.linspace(-2*np.pi,2*np.pi,200,endpoint=True)
5
6  ax = pl.gca()  # gca stands for 'get current axis'
7  ax.spines['right'].set_color('none')     # No color
8  ax.spines['top'].set_color('none')       # No color
9  ax.xaxis.set_ticks_position('bottom')    # Where to put the ticks
10 ax.spines['bottom'].set_position(('data',0)) # set the bottom line to 0 in data-space
11 ax.yaxis.set_ticks_position('left')      # Where to put the ticks
12 ax.spines['left'].set_position(('data',0))   # set the left line to 0 in data-space
13
14 s = np.sin(x)
15 pl.plot(x,s,color='green',linewidth=1,linestyle='-')
16
17 pl.show()
```

More Customizing

```
1  import numpy as np
2  import pylab as pl
3
4  x=np.linspace(-2*np.pi,2*np.pi,200,endpoint=True)
5
6  s = np.sin(x)
7  pl.plot(x,s,color='green',linewidth=1,linestyle='-')
8
9
10 pl.xticks([-2*np.pi, -np.pi, 0, np.pi, 2*np.pi],
11           [r'$-2\pi$',r'$\pi$',r'$0$',r'$\pi$',r'$2\pi$'])
12 pl.yticks([-1, 0, +1])
13
14 pl.show()
```

- We can embed some LaTeXcode in the program to get fancy fonts.

Bar Plot

```
1  import numpy as np
2  import pylab as pl
3
4  x=np.array([0,1,2,3,4,5])
5  y=np.array([5,6,2,13,4,9])
6  z=np.random.rand(6)*10
7
8  pl.bar(x, y, facecolor='red', edgecolor='white')
9  pl.bar(x, -z, facecolor='blue', edgecolor='white')
10
11 pl.show()
```

Scatter Plot

```
1  import numpy as np
2  import pylab as pl
3
4  r1=np.random.rand(1000)
5  x=(r1-0.5)*10
6  r2=np.random.rand(1000)
7  y=10*r2
8
9  pl.scatter(x,y)
10
11 pl.show()
```

Polar Plot

```
1  import pylab as pl
2  import numpy as np
3
4  ax = pl.axes(polar=True) # Only change HERE
5
6  N = 200
7  theta = np.arange(0.0, 2 * np.pi, 2 * np.pi / N)
8  radii = 10 * np.sin(theta)
9
10 pl.plot(theta,radii,color='red')
11 pl.plot(theta,5*theta)
12
13 pl.show()
```

Plotting Data from File

- In many situations the data to be plotted is in a file.
- You can read the data, store it in arrays and plot the data.

```
1  import numpy as np
2  import pylab as pl
3
4  datafile = open("data.txt","r")
5
6  x=[]
7  y=[]
8  for line in datafile:
9      temp=line.split()
10     p=float(temp[0])
11     q=float(temp[1])
12     x+=[p]
13     y+=[q]
14
15 X=np.array(x)
16 Y=np.array(y)
17 pl.plot(X,Y,color='green',linewidth=1,linestyle='-')
18
19 pl.show()
```