matplotlib

Ben Bolker

18:50 14 March 2017

- matplotlib cheat sheet
- matplotlib gallery

Basic setup

- create a figure: `fig=plt.figure()`
- can include figure size `figsize=(w,h)`, background/edge color, resolution (dpi=dots per inch)
- add a *subplot* (or "axes"): `ax = fig.add_subplot(1,1,1)` (rows,columns,which plot)
- now can show or save the figure: `fig.show()` or `fig.savefig("filename")`
- your operating system probably knows what to do if you click on the saved figure (or you can stick it in a Word document, etc.)

```
import matplotlib.pyplot as plt  ## pyplot is a _sub-module_ of matplotlib
fig = plt.figure(figsize=(4,4))
print(plt.rcParams["figure.figsize"])  ## default figure size
ax = fig.add_subplot(1,1,1) ## number of rows, number of columns, plot num
fig.savefig("pix/empty.png")
```

```
## [8.0, 6.0]
```

Shortcut:

```
fig, ax = plt.subplots() ## single subplot
```

Basic plots

Basic things we can put on the plot: lines, scatter plots ...

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
## make two subfigures
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
x = np.array([1,2,3])
y = np.array([1,3,2])
ax1.plot(x,y)
ax1.text(2,1.1,"ax.plot()")
ax2.scatter(x,y)
ax2.text(2,0.6,"ax.scatter()")
fig.savefig("pix/basic.png")
```

Putting more than one thing on a plot

- You can do more than one `plot()` or `scatter()` on the same set of axes

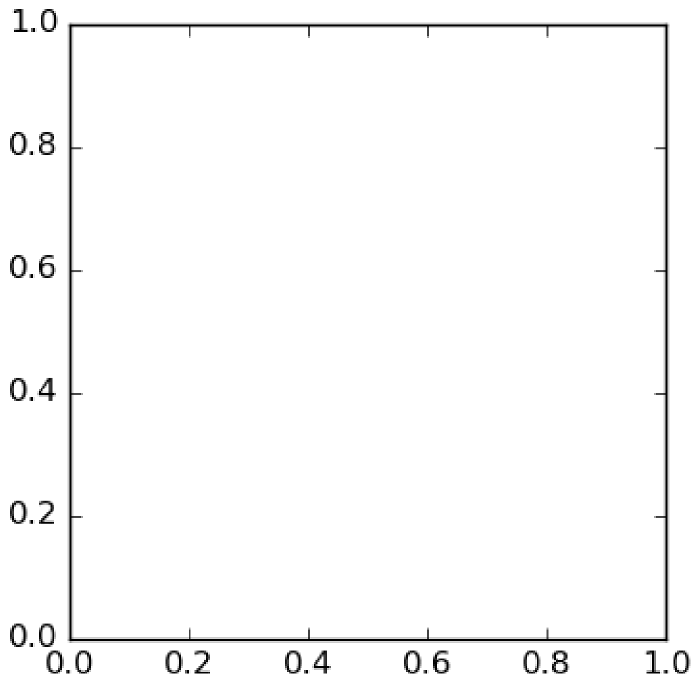

Figure 1:

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(4,4))
ax1 = fig.add_subplot(1,1,1)
x = np.arange(0,5*np.pi,0.1)
y = np.sin(x)
ax1.plot(x,y)
ax1.plot(x+np.pi/2,y,color="red")
fig.savefig("pix/basic2.png")
```

Distinguish lines:

- `color`
- `marker` (+, o, x, ...)
- `linewidth`
- `linestyle` (-, --, -., None, ...)

Decorating plots

- titles (`ax.set_xlabel()`, `ax.set_ylabel()`)
- change limits
- title: `fig.suptitle()` (refers to figure, not individual axes)
- legend: need to label plotted stuff. e.g.

```
ax1.plot(x,y,label="first")
ax1.plot(x+np.pi/2,y,color="red",label="second")
```

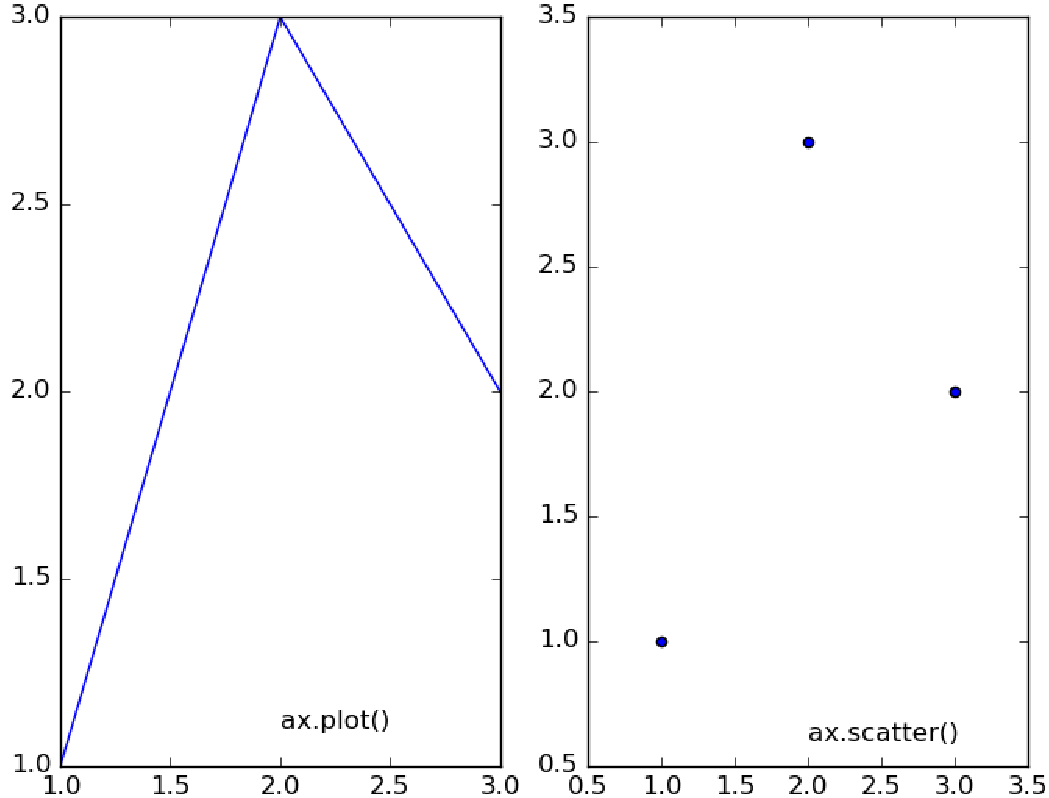
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
ax.plot()
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.5
1.0
1.5
2.0
2.5
3.0
3.5
ax.scatter()

Figure 2:

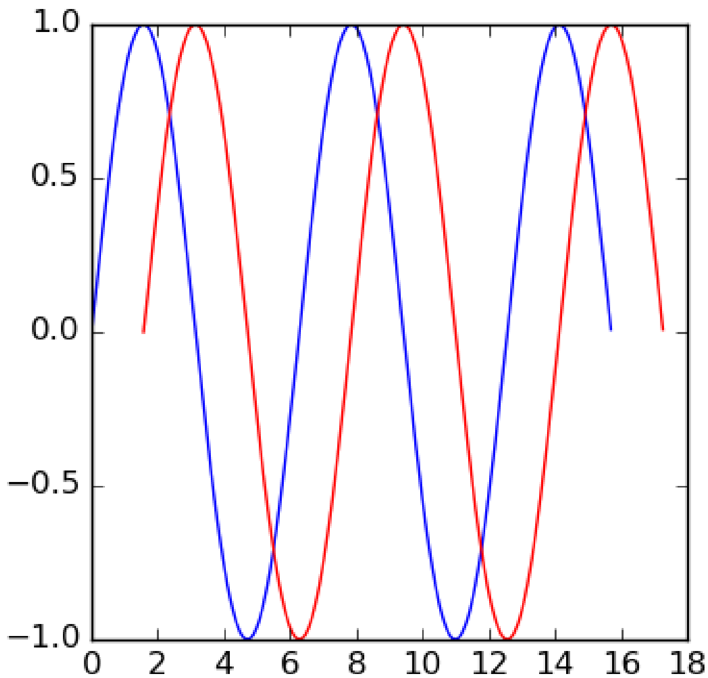

Figure 3:

```
ax1.set_xlim([0,25])
ax1.legend(fontsize=8)
fig.suptitle("my plot")
```

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(4,4))
ax1 = fig.add_subplot(1,1,1)
x = np.arange(0,5*np.pi,0.1)
y = np.sin(x)
ax1.plot(x,y,label="first")
ax1.plot(x+np.pi/2,y,color="red",label="second")
ax1.set_xlabel("X axis label")
ax1.set_ylabel("Y axis label")
ax1.set_xlim([0,25])
ax1.legend(fontsize=8)
fig.suptitle("my plot")
fig.savefig("pix/basic3.png")
```

The logistic map

The *discrete logistic map*, $x_{t+1} = rx_t(1-x_t)$, is a simple model for populations that has interesting dynamical properties. It has equilibria at 0 and $x^* = 1-1/r$. For $r > 1$ it mimics exponential (geometric) growth for $x_t \ll 1$.

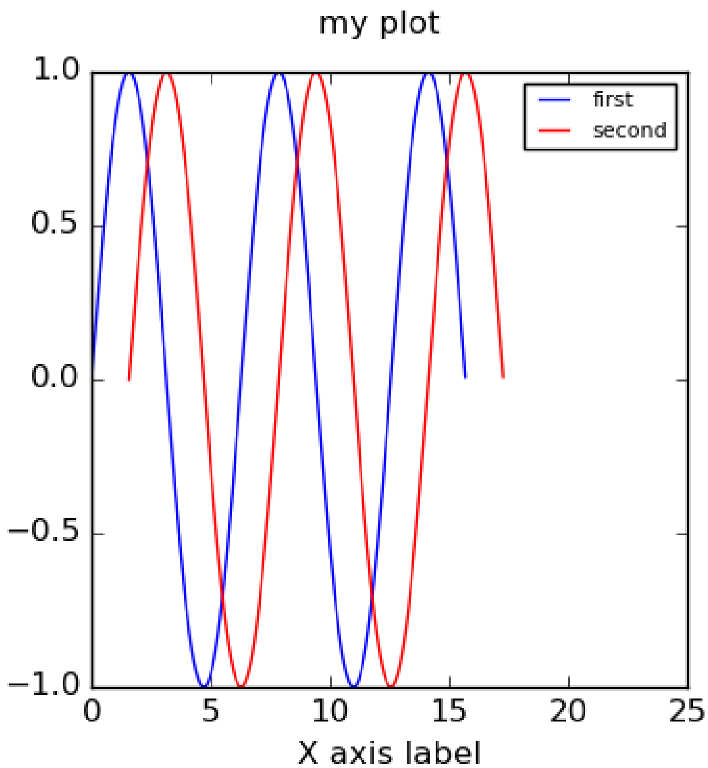

Figure 4:

```
from logist import *
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
y1 = logist_map(1.5)
y2 = logist_map(2)
y3 = logist_map(3)
ax1.plot(y1)
ax1.plot(y2)
ax1.plot(y3)
fig.savefig("pix/lm0.png")
```

```
from logist import *
import matplotlib.pyplot as plt
rvals = np.arange(1.1,3.9,0.05)
b = np.zeros((500,len(rvals)))
for i in range(len(rvals)):
   b[:,i] = logist_map(r=rvals[i],nt=500)
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
rmat = np.tile(rvals,(500,1))
ax1.scatter(rmat,b)
fig.savefig("pix/lm1.png")
## now without transient
rmat = np.tile(rvals,(250,1))
b2 = b[250:,]
```

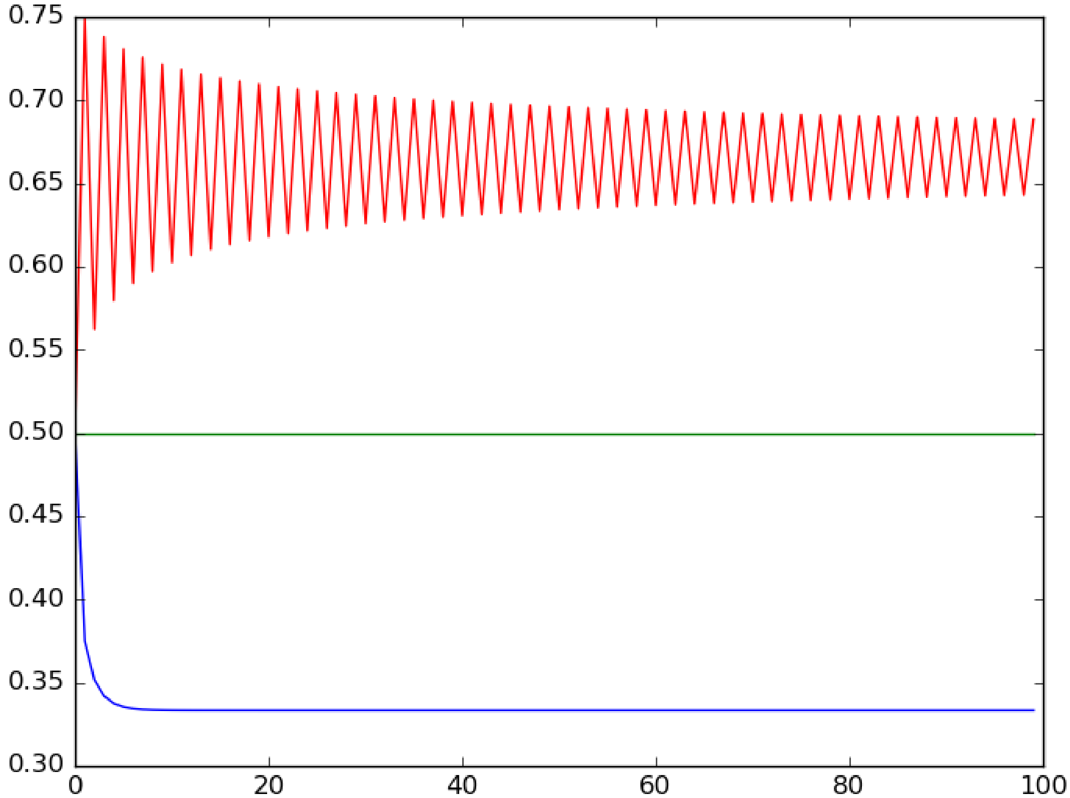

Figure 5:

```
fig,ax1 = plt.subplots()
ax1.scatter(rmat,b2)
fig.savefig("pix/lm2.png")
## now as an image plot
fig,ax1 = plt.subplots()
ax1.imshow(b2,aspect="auto",extent=[1.1,3.9,250,500],interpolation="none")
fig.savefig("pix/lm3.png")
```

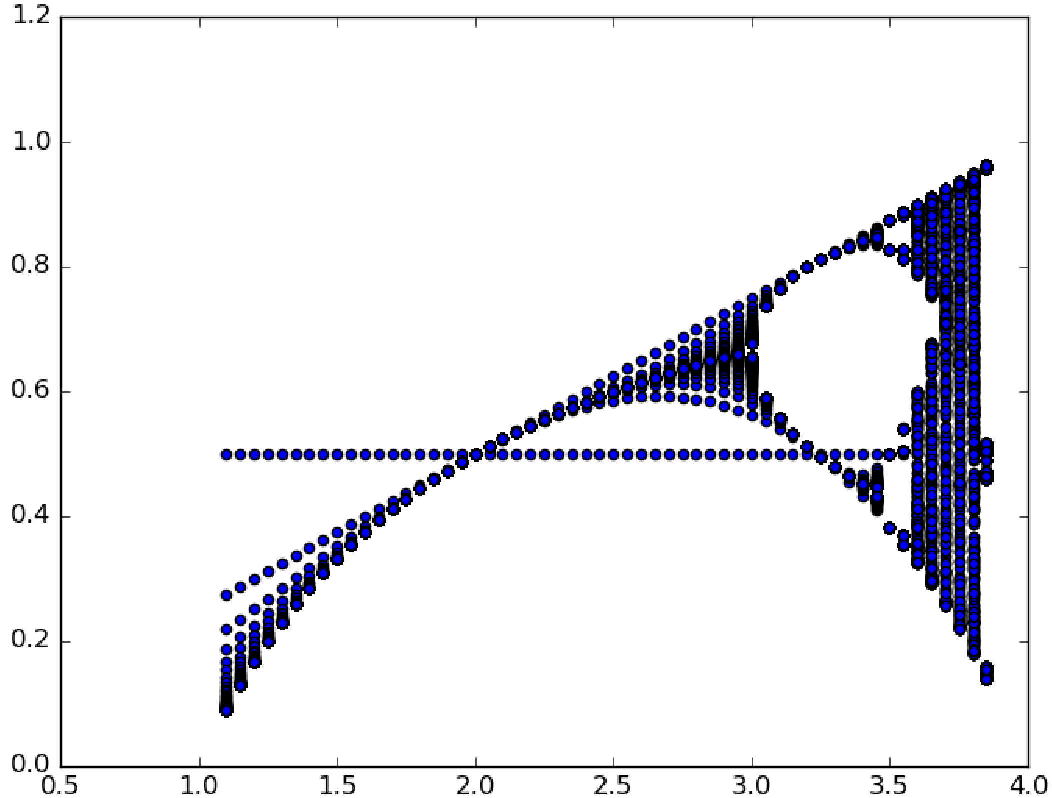

Figure 6:

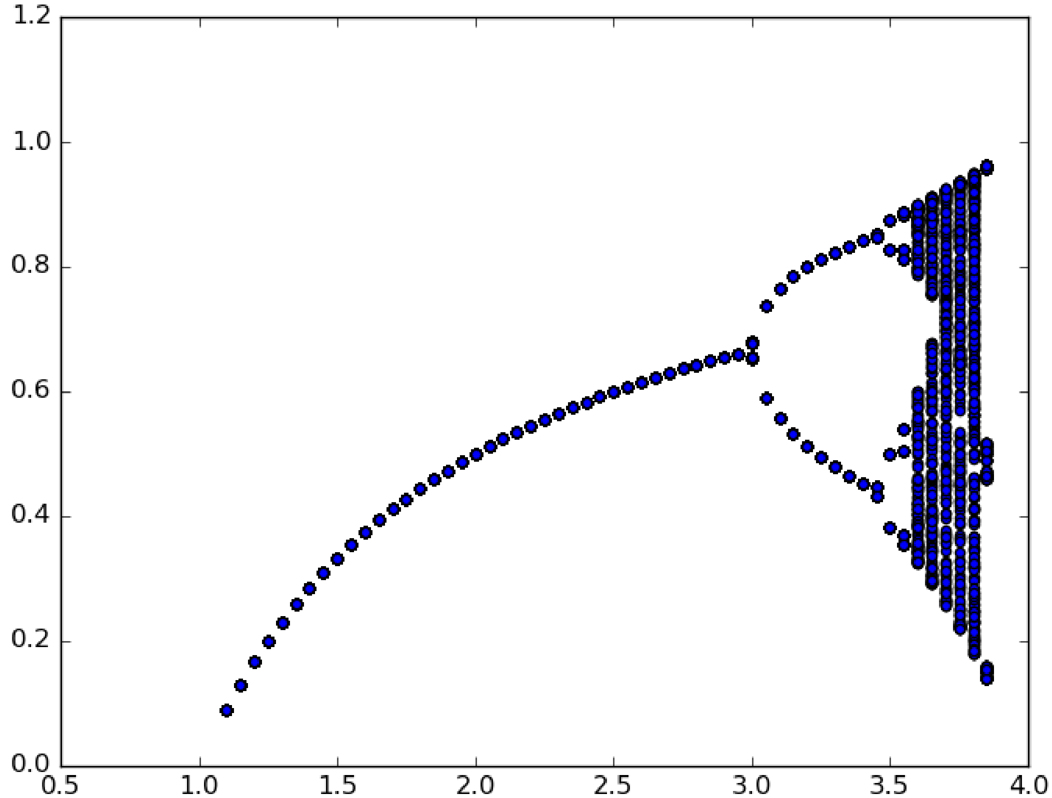

Figure 7:

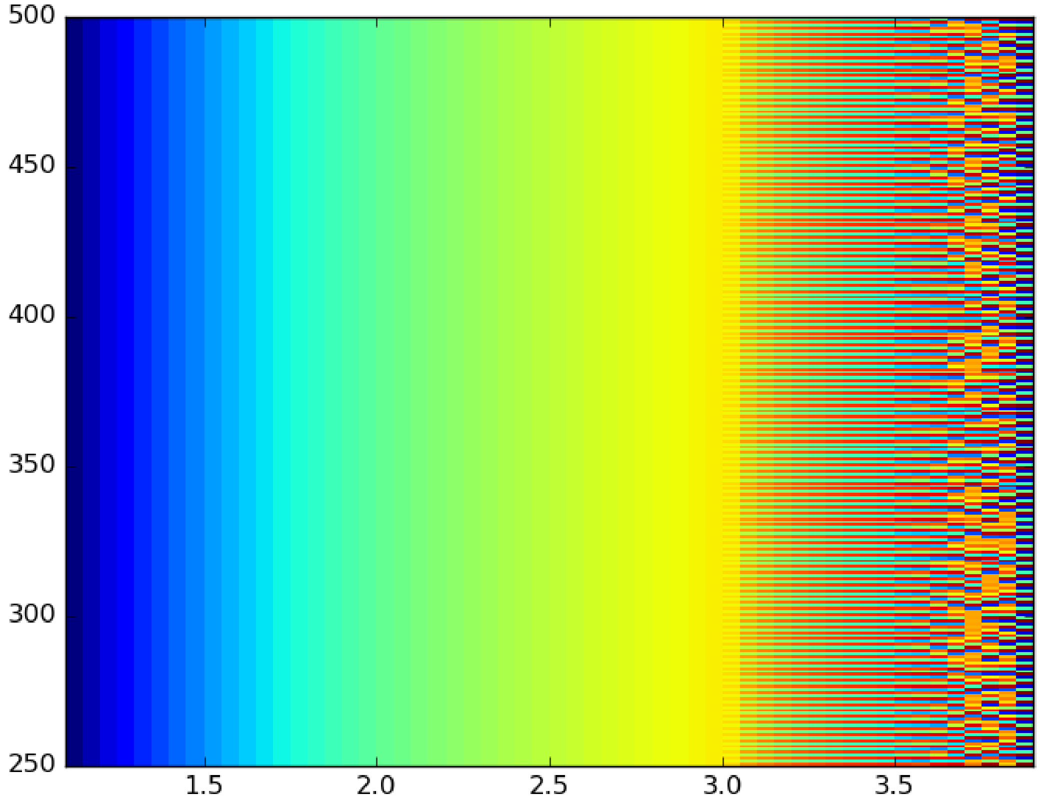

Figure 8: