1 Graphics

- Matplotlib
- Seaborn

- ***Matplotlib*** is a complete plotting library capable of high-quality graphics.
- Matplotlib contains both high level functions which produce specific types of figures, for example a simple line plot or a bar chart, as well as a low level API for creating highly customized charts.
- This chapter covers the basics of producing plots and only scratches the surface of the capabilities of matplotlib.
- Further information is available on the matplotlib website or in books dedicated to producing print quality graphics using matplotlib.
- Throughout this chapter, the following modules have been imported.

1.1 `matlibplot`

- Matplotlib is a complete plotting library capable of high-quality graphics. Matplotlib contains both high level functions which produce specific types of figures, for example a simple line plot or a bar chart, as well as a low level API for creating highly customized charts.
- This chapter covers the basics of producing plots and only scratches the surface of the capabilities of matplotlib.
- Further information is available on the matplotlib website or in books dedicated to producing print quality graphics using matplotlib.

1.2 `seaborn`

seaborn is a Python package which provides a number of advanced data visualized plots. It also provides a general improvement in the default appearance of matplotlib-produced plots, and so I recommend using it by default.

```
import seaborn as sns
```

All figure in this chapter were produced with seaborn loaded, using the default options. The dark grid background can be swapped to a light grid or no grid using sns.set(stype='whitegrid') (light grid) or sns.set(stype='nogrid') (no grid, most similar to matplotlib).

1.3 Histograms

Histograms can be produced using hist. A basic histogram produced using the code below is presented in Figure 15.5, panel (a). This example sets the number of bins used in producing the histogram using the keyword argument bins.

1.4 Adding a Title and Legend

Titles are added with title and legends are added with legend. legend requires that lines have labels, which is why 3 calls are made to plot – each series has its own label. Executing the next code block produces a the image in figure 15.8, panel (a).

```
>>> x = cumsum(randn(100,3), axis = 0)
>>> plot(x[:,0],'b',
label = 'Series 1')
>>> hold(True)
>>> plot(x[:,1],'g.',
label = 'Series 2')
>>> plot(x[:,2],'r:',label = 'Series 3')
>>> legend()
>>> title('Basic Legend')
```

legend takes keyword arguments which can be used to change its location (loc and an integer, see the docstring), remove the frame (frameon) and add a title to the legend box (title). The output of a simple example using these options is presented in panel (b).

```
>>> plot(x[:,0],'b',
label = 'Series 1')
>>> hold(True)
>>> plot(x[:,1],'g.',
label = 'Series 2')
>>> plot(x[:,2],'r:',label = 'Series 3')
>>> legend(loc = 0, frameon = False, title = 'The Legend')
>>> title('Improved Legend')
```

1.5 Plotting

```
close_px.plot(label='AAPL')
mavg.plot(label='mavg')
plt.legend()
```