

Principles of Programming in Econometrics

Introduction, structure, and advanced programming techniques

Charles S. Bos

Vrije Universiteit Amsterdam
Tinbergen Institute

`c.s.bos@vu.nl`

August 2020 – Version Python

Separate lecture slides

Compilation: August 15, 2020

Overview

Principles of Programming in Econometrics

D0: Syntax, example 2⁸

D1: Structure, scope

D2: Numerics, packages

D3: Optimisation, speed

Day 2: Numerics and flow

9.30 Numbers and representation

- ▶ Steps, flow and structure
- ▶ Floating point numbers
- ▶ Practical Do's and Don'ts
- ▶ Packages
- ▶ Graphics

13.30 Practical

- ▶ Cleaning OLS program
- ▶ Loops
- ▶ Bootstrap OLS estimation
- ▶ Handling data: Inflation

A module: matplotlib.pyplot

Several options available, here we focus on `pyplot`.

Listing 1: matplotlib/plot1.py

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

# Initialisation
mY= np.random.randn(100, 3)

# Output
plt.figure(figsize=(8,4))           # Choose alternate size (def= (6.4,4.8))
plt.subplot(2, 1, 1)                # Work with 2x1 grid, first plot
plt.plot(mY)                        # Simply plot the white noise
plt.legend(["a", "b", "c"])          # Add a legend
plt.title("White noise")            # ... and a title

plt.subplot(2, 1, 2)                # Start with second plot
plt.plot(mY[:,0], mY[:,1:], ".")    # Plot here some cross-plots
plt.ylabel("b,c")
plt.xlabel("a")
plt.title("Unrelated data")         # ... and name the graph
plt.savefig("graphs/plot1.png");    # Save the result
plt.show()                          # Done, show it
```

Details: [matplotlib documentation](#), or e.g. Kevin Sheppard's
[Python Introduction](#)

A module: matplotlib.pyplot II

Basic plot:

- ▶ Initialise the plot with `plt.figure()`
- ▶ (Optionally) also set the size with `plt.figure(figsize=(8,4))` (I prefer a wider shape)
- ▶ Graphing appears in *subplots*, choose i 'th plot out of $R \times C$ using `plt.subplot(iR, iC, i)` (counting starts at 1, following matlab customs)
- ▶ Plot either y values against x -axis (`plt.plot(mY)`)
- ▶ ... or plot x against y , `plt.plot(mY[:,0], mY[:,1:])`

A module: matplotlib.pyplot III

Embellish plot:

- ▶ Place a legend for multiple lines using `plt.legend(['a', 'b', 'c'])`
- ▶ Alternatively, specify the label with the plot, `plt.plot(vY, label='y')`; `plt.legend()`. In the latter case, don't forget to turn on the legend.
- ▶ Plot takes extra arguments specifying line types, colours etc: `plt.plot(vX, vY, 'r+')` for red crosses
- ▶ *After drawing the graph, and before showing it, possibly save the figure, as .eps, .png, .pdf, .jpg, .svg or others, `plt.savefig('graphs/plot1.png')`*

A module: matplotlib.pyplot IV

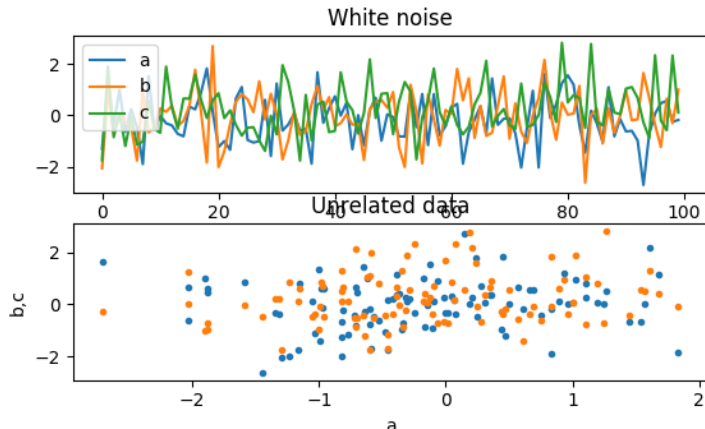


Figure: The resulting plot1.png

A module: matplotlib.pyplot V

All plotting is done against the last *figure* and/or *axes*. This one can make explicit as well:

Listing 2: matplotlib/plot1b.py

```
fig= plt.figure(figsize=(8, 6))           # Choose alternate size
ax=fig.add_subplot(2, 1, 1)               # Work with 2x1 grid, first plot
ax.plot(mY)                               # Simply plot the white noise
ax.legend(["a", "b", "c"])                # Add a legend
ax.set_title("White noise")               # ... and a title

ax2=fig.add_subplot(2, 1, 2)              # Start with second plot
ax2.plot(mY[:,0], mY[:,1:], ".")          # Plot here some cross-plots
ax2.set_ylabel("b,c")
ax2.set_xlabel("a")
ax2.set_title("Unrelated data")           # ... and name the graph
fig.savefig("graphs/plot1b.png")          # Save the result
fig.show()                                # Done, show figure
```


A module: matplotlib.pyplot + \LaTeX

For inclusion in \LaTeX , true formulas might be nice.

Example:

Listing 3: plot_latex.py

```
plt.rc('text', usetex=True)           # Start using latex text

plt.figure()
plt.plot(mY, '.')                     # Simply plot the white noise, with dots
plt.legend([r'$E=m\ C^2$', r'$s=\sum_{i=1}^n y_j$']) # Add a legend
plt.title(r'Use \textbf{(most)} \LaTeX\ commands {\em at will}')
plt.savefig('graphs/plot_latex1.png')
plt.show()
```

Note: Without the `usetex=True`, you can still use simple \LaTeX commands, but get different fonts.

A module: matplotlib.pyplot + \LaTeX II

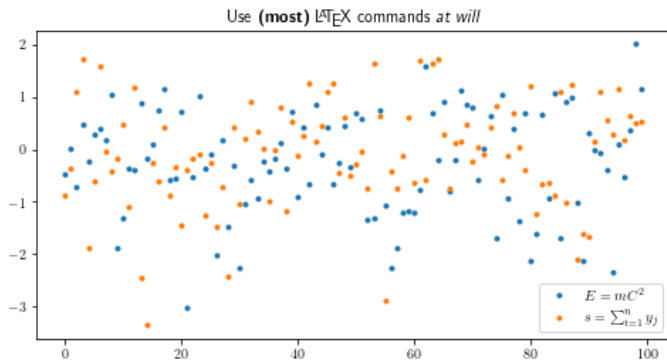


Figure: The resulting plot_latex1.png

A module: matplotlib.pyplot + ???

Other options:

- ▶ Zillions...
- ▶ Check the [examples](#)
- ▶ Use google, get some practice!