Project organization: hierarchies, doc, notebook

WS Noble (2009) PLoS Comp. Biol.

A Quick Guide to Organizing Computational Biology Projects

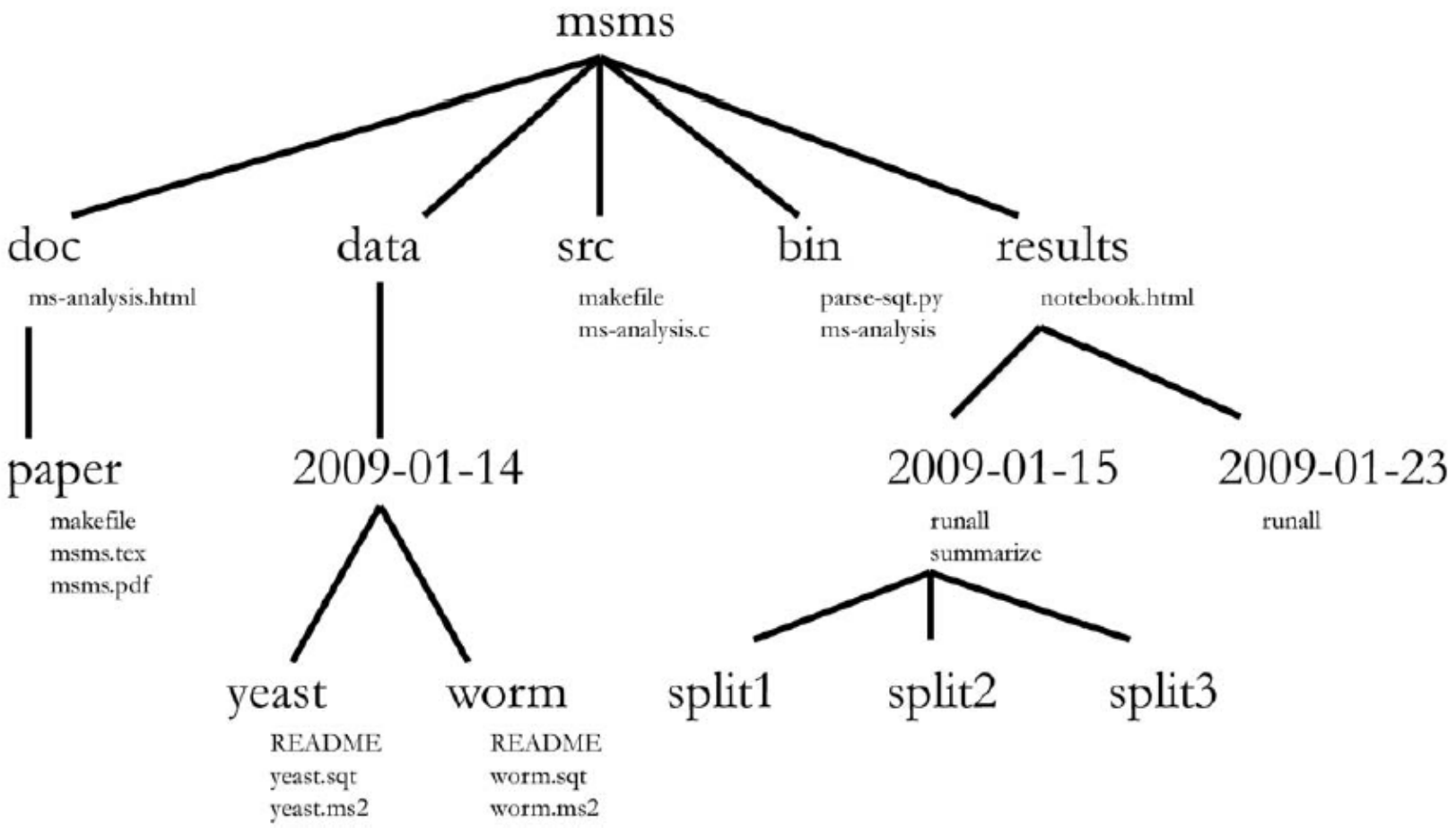


Figure 1. Directory structure for a sample project. Directory names are in large typeface, and filenames are in smaller typeface. Only a subset of the files are shown here. Note that the dates are formatted `<year>-<month>-<day>` so that they can be sorted in chronological order. The

Write a Python program: base composition of a DNA seq.

- Open new text file (Text Wrangler, or other enriched text editor)
- Save as dnacalcMyInitials.py in your Scripts directory
- Open Terminal : (Mac: Terminal.app)
  - Mac Terminal – Preferences – resize default window
- $ cd ….. < scripts directory >
- $ which python
- /usr/bin/python
- First line is shebang for python (recall Bash script: #!/bin/bash)
- **#!/usr/bin/env python (Bold type are lines for program.)**
  - preferable instead of: /usr/bin/python
  - env is a program – assures portability
  - “env python” : find and run Python, wherever it is located

Constructing the dnacalc.py program

- **#!/usr/bin/env python** # Except for shebang, # begins a comment.
- **DNASeq = 'ATGAAC'** # A value is assigned to a variable.
  - Variable: simultaneously created and assigned its initial value. Automatic type.
- **print 'Sequence: ', DNASeq** # print command: Build output.

  Compose texts, labels, etc., and values contained in variables.
- $ ls -l to check for permission to execute
  - -rwxr-xr-x@ …….. dnacalcPB.py
  - $ chmod u+x dnacalcPB.py
- $ dnacalcPB.py
  - Sequence: ATGAAC
- Get string length: len() function
- **SeqLength = len(DNASeq)**
- **print 'Sequence Length:', SeqLength**

Built-in functions of variables

- In Terminal, open new screen (Mac: use cmd-T or cmd-N).
- Open a Python Interactive prompt (chapter 9)
- $ python
  - Python 2.6.1 (r261:67515, Aug 2 2010, 20:10:18)
    [GCC 4.2.1 (Apple Inc. build 5646)] on darwin
    Type "help", "copyright", "credits" or "license" for more information.
    >>> 8/4
    2
- >>> dna = 'atcgatc'
- >>> dir(dna)   shows all “methods” for use with strings
  - Methods are functions built into each type of variable. “dot notation”
- >>> dna.count('atc')
  - 2
- **NumberA = DNASeq.count ('A')**
- **print "Number of A's:", NumberA**
  - Double quotes enables internal single quote. print 'Les Ch'tis' => error

Arithmetic

Addition	+	`35 + 22`	`57`
		`'Py' + 'thon'`	`'Python'`
Subtraction	-	`35 - 22`	`13`
Multiplication	*	`3 * 2`	`6`
		`'Py' * 2`	`'PyPy'`
Division	/	`3.0 / 2`	`1.5`
		`3 / 2`	`1`
Exponentiation	**	`2 ** 0.5`	`1.41421356...`
Remainder	%	`13 % 5`	`3`

Converting between types of variables

- Python “types” variables dynamically.
- Change type with str(), int() or float()
- >>> s = '5'; n = 3; s + n => Error
- >>> print int(s) + n, s + str(n)
  - 8 53
- >>> a=7; b=3; a/b
  - 2 Integer / Integer => truncation of result
- >>> a = float(a); a/b
  - 2.33333333333 e.g. print float('2.454e-2')
- **SeqLength = float(len(DNASeq))**
- **print 'A:', NumberA/SeqLength**
- check output in Terminal

Controlling string formatting with the % operator

- Flexible string formatting.
- Controlling the number of significant digits; padding to align digits.
- String formatting operator: % => 'a %d b %f c' **%** (varInt1, varFlt2)
  - “Placeholders” mark positions in the text.
  - integer digit (%d), floating point number (%f), or string (%s)
  - Values of variables are inserted at placeholder positions.
  - placeholders also control type conversion
- print "A is in %.2f of %d bases." % (NumberA/SeqLength, SeqLength)
  - A is in 0.50 of 6 bases.
- Also use for building strings in variables
  - pctA = “%.1f" % (100 * NumberA/SeqLength)
- **print "A: %.1f" % (100 * NumberA/SeqLength)**

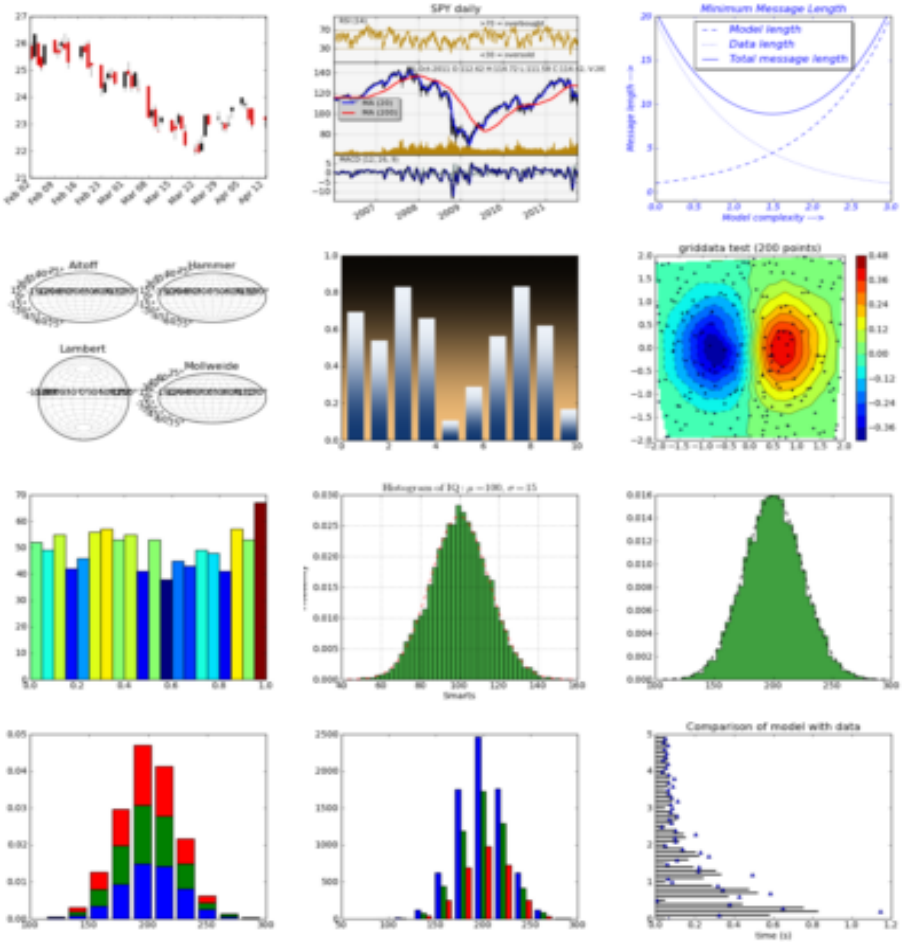
Getting input from keyboard

- Data input, 3 ways: entered in script, screen prompt or from file
- **DNASeq = raw_input("Enter a DNA sequence: ")**
- Convert string input into format used in program.
  - Use built in methods – e.g. .upper(), .replace()
- **DNASeq = DNASeq.upper()**
  - # convert to uppercase for .count() function
- **DNASeq = DNASeq.replace(" ","")**
  - # remove spaces and line returns as \n

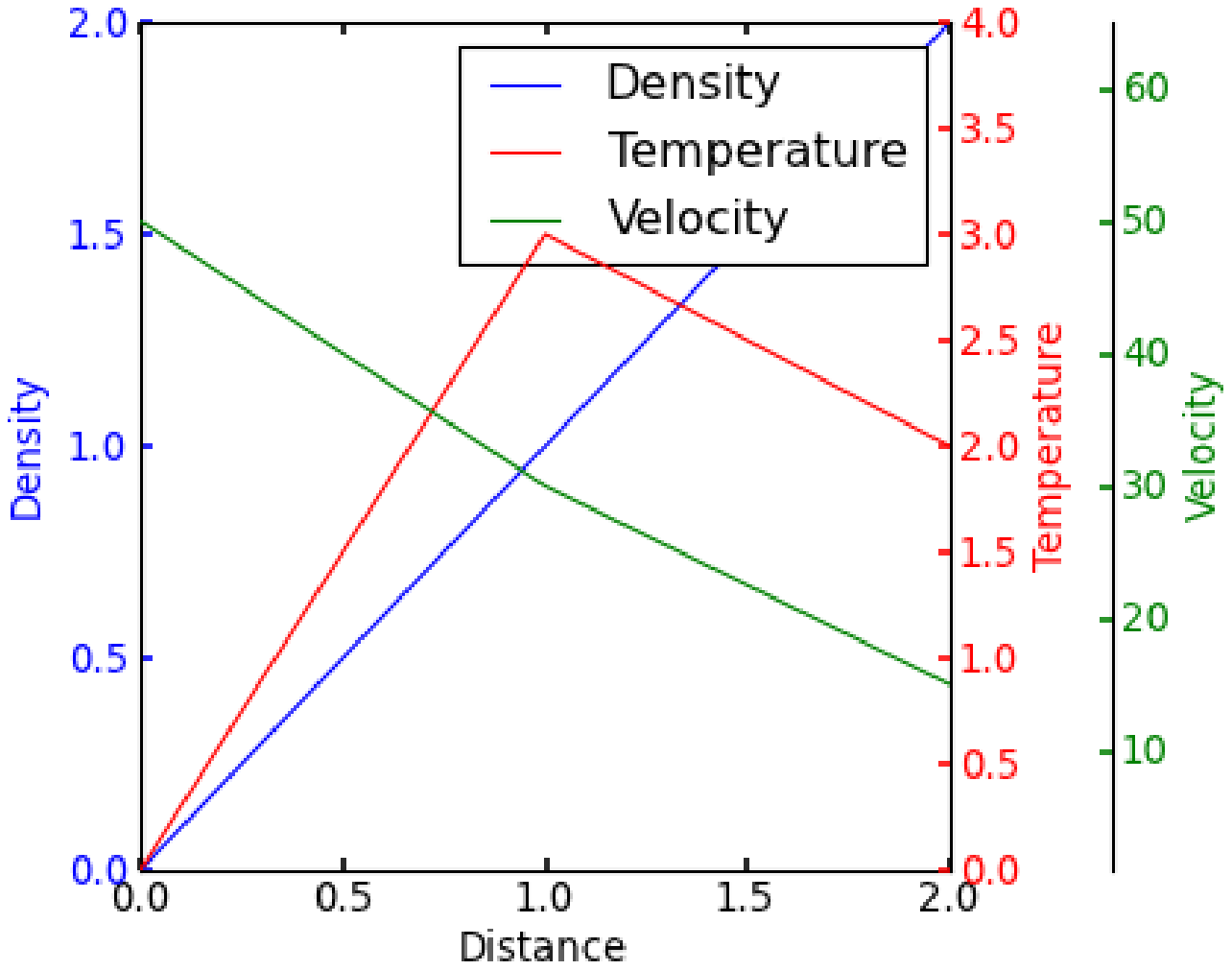
Why learn and use Python as a scripting language?

- relative clarity of the code (see Appendix 5 for comparisons)
- ease of manipulating text data
- native support for advanced features
  - e.g., object-oriented programming
- easy to learn and get running quickly
- rapid iteration as script evolves (as for Perl or Ruby)
- fully functional and mature language suitable for complex tasks
- large and growing user base
  - vast source of existing code available to copy and adapt.

python matplotlib graphics gallery
http://matplotlib.sourceforge.net/gallery.html



Multiple axis plot example



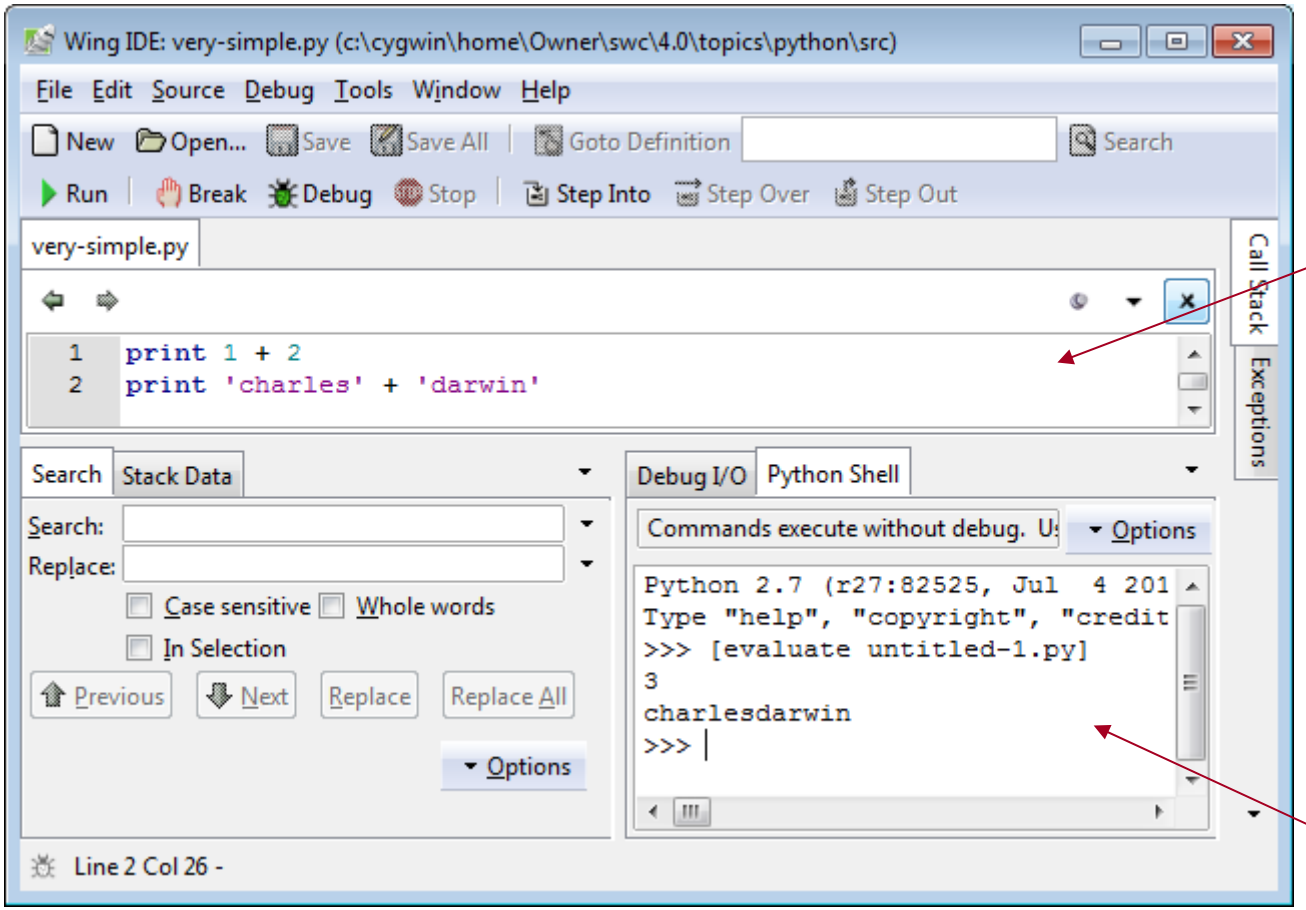
Multiple axes code (1)

```
import matplotlib.pyplot as plt

def make_patch_spines_invisible(ax):
    ax.set_frame_on(True)
    ax.patch.set_visible(False)
    for sp in ax.spines.itervalues():
        sp.set_visible(False)

fig = plt.figure()
fig.subplots_adjust(right=0.75)

host = fig.add_subplot(111)
par1 = host.twinx()
par2 = host.twinx()

# Offset the right spine of par2.  The ticks and label have already been
# placed on the right by twinx above.
par2.spines["right"].set_position(("axes", 1.2))
# Having been created by twinx, par2 has its frame off, so the line of its
# detached spine is invisible.  First, activate the frame but make the patch
# and spines invisible.
make_patch_spines_invisible(par2)
# Second, show the right spine.
par2.spines["right"].set_visible(True)

p1, = host.plot([0, 1, 2], [0, 1, 2], "b-", label="Density")
p2, = par1.plot([0, 1, 2], [0, 3, 2], "r-", label="Temperature")
p3, = par2.plot([0, 1, 2], [80, 30, 15], "g-", label="Velocity")
```

Multiple axes code (2)

```
host.set_xlim(0, 2)
host.set_ylim(0, 2)
par1.set_ylim(0, 4)
par2.set_ylim(1, 100)

host.set_xlabel("Distance")
host.set_ylabel("Density")
par1.set_ylabel("Temperature")
par2.set_ylabel("Velocity")

host.yaxis.label.set_color(p1.get_color())
par1.yaxis.label.set_color(p2.get_color())
par2.yaxis.label.set_color(p3.get_color())

tkw = dict(size=4, width=1.5)
host.tick_params(axis='y', colors=p1.get_color(), **tkw)
par1.tick_params(axis='y', colors=p2.get_color(), **tkw)
par2.tick_params(axis='y', colors=p3.get_color(), **tkw)
host.tick_params(axis='x', **tkw)

lines = [p1, p2, p3]

host.legend(lines, [l.get_label() for l in lines])

plt.show()
```

Use an *integrated development environment* (IDE)



Source file

Execution shell