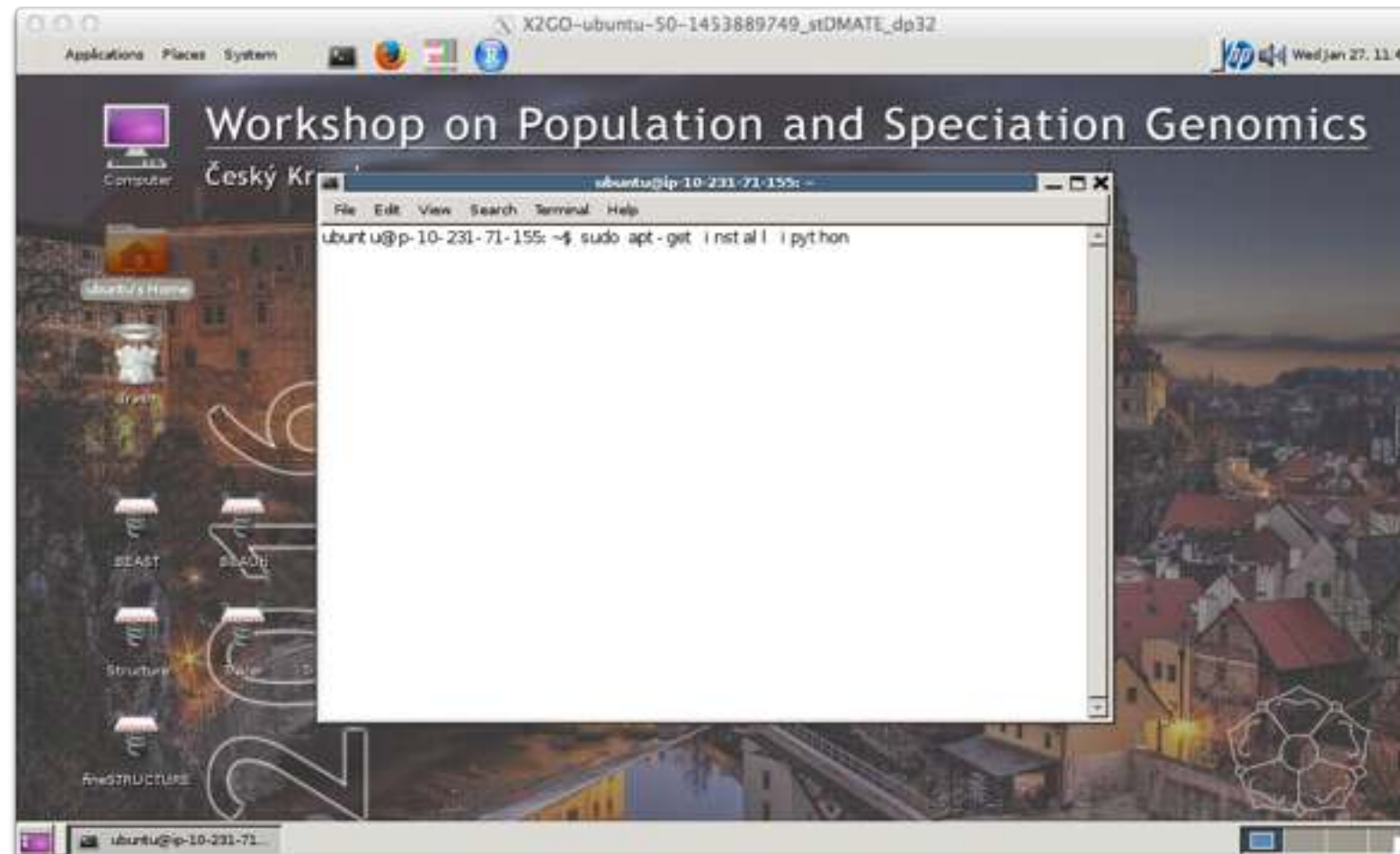


Introduction to Python



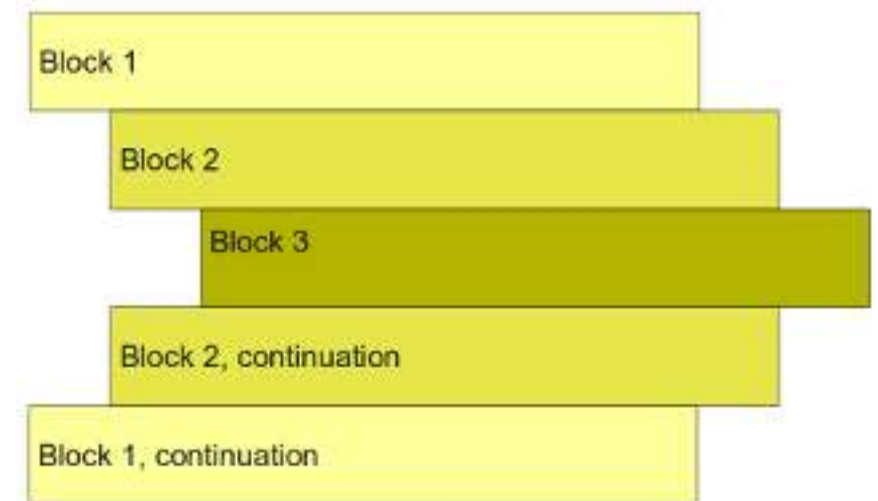
Ryan Gutenkunst
Molecular and Cellular Biology
University of Arizona

Before we start, fire up your Amazon instance, open a terminal, and enter the command
`sudo apt-get install ipython`



Why Python?

- Interpreted language
Use interactively, for fast development
- Clean syntax
Indentation matters
- Duck typing
Easy to write general functions
- Open Source with a large community



Why Python?

- Numpy
Efficient computation for arrays of data
- SciPy
Grab bag of scientific algorithms
- Matplotlib
Matlab-like plotting interface
- Biopython
Bioinformatics tools
- cPickle
Simple storage of arbitrary data to disk



Why not Python?

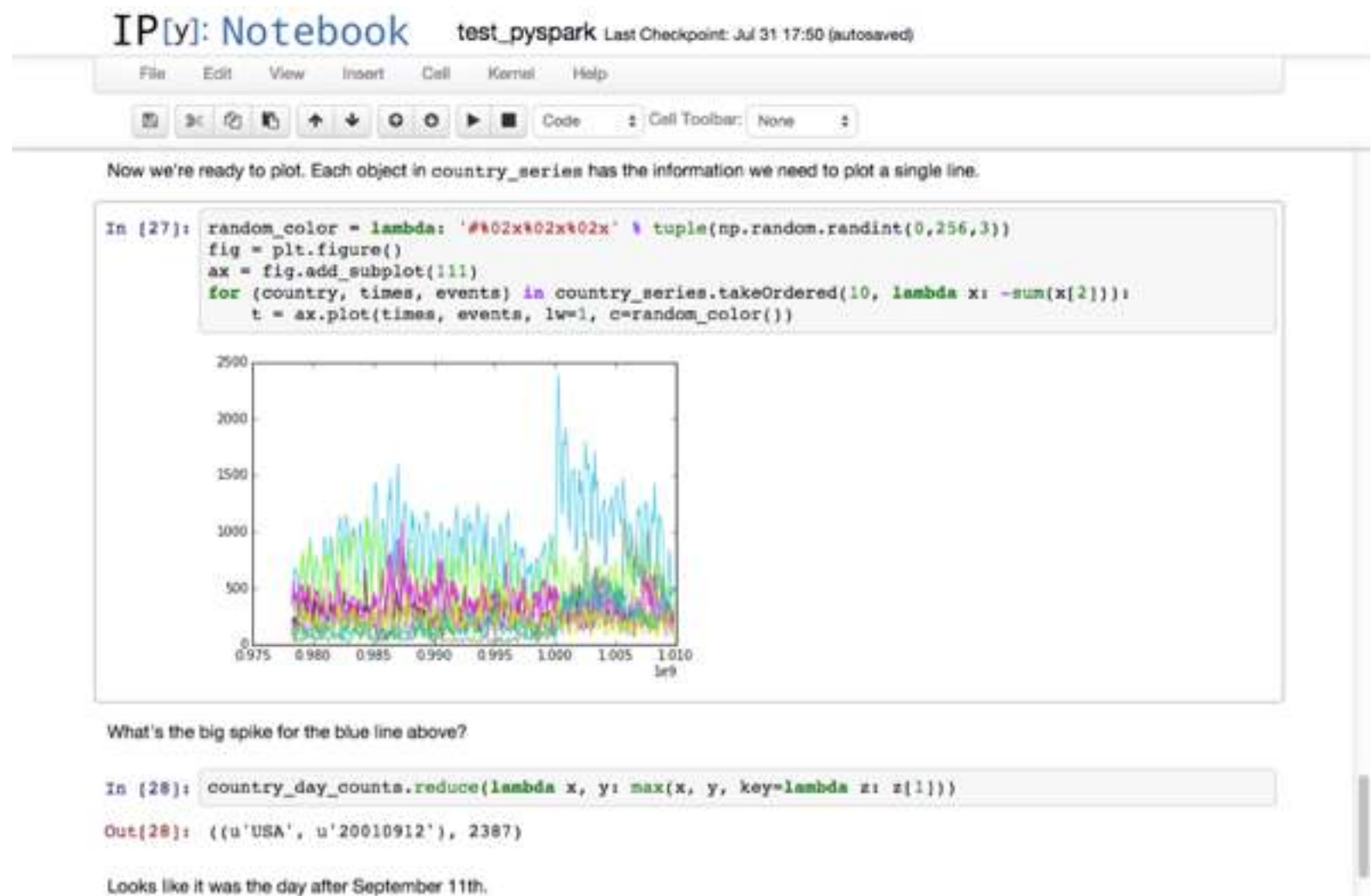
- Python is relatively slow (like R and Matlab)
 - But can speed up by interfacing directly with C
- Installation of libraries can be a headache



iPython

IP[y]:
IPython

- For me (and you today): a better Python shell
- Start with `ipython --pylab`
- Tab completion
 - `a = '1'`
 - `a.<tab>`
- Interactive help
 - `a.upper?`
- Command history
- The future: iPython notebooks



Python data structures

- Strings
- Lists for sequential data
- Dictionaries (hashes) for mapping keys to values



Strings

- For storing and manipulating textual data
- `a = 'b'`
- `'b' + 'c'`
- `'c'.upper()`
- `'a_b_c'.split('_')`
- `'{0}_{1}'.format(1, 'd')`
- `'\t'` - Tab character

Lists

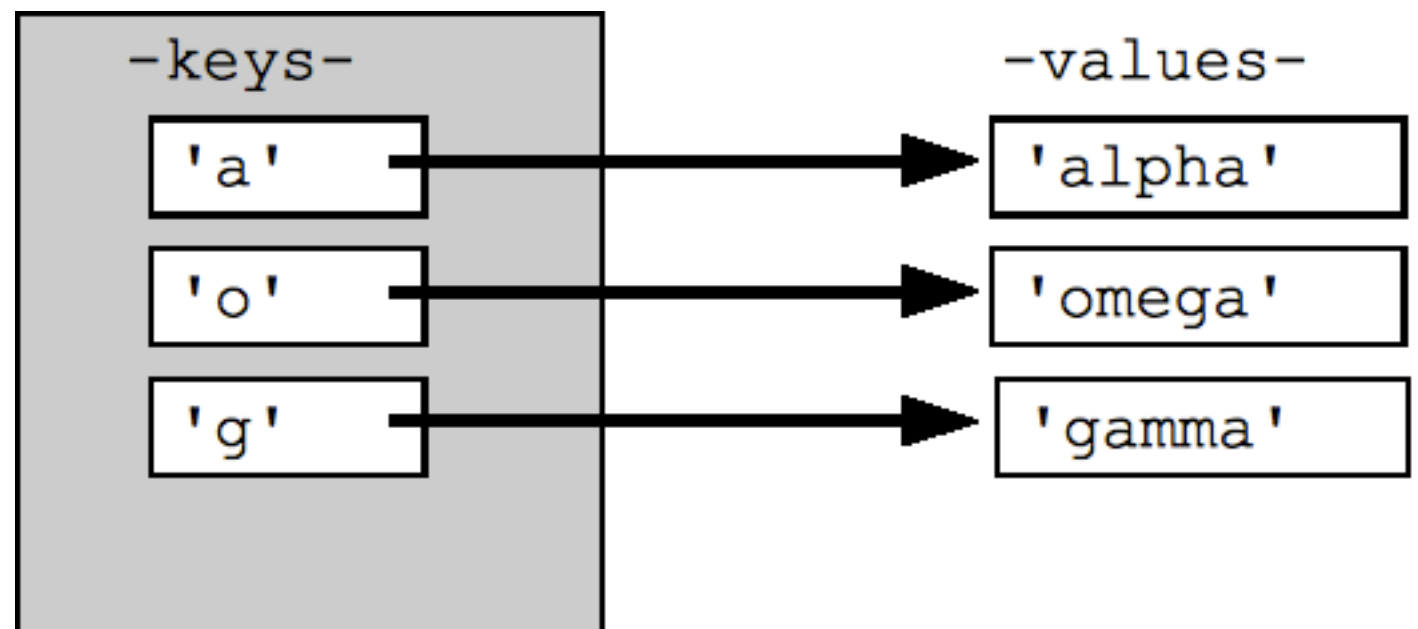
- Ordered sequences of potentially heterogeneous data
- `a = [1, 'a']`
- `a`
- `a.append(9)`
- `a`
- `a.extend(['b', 4])`
- `len(a)`

Indexing and Slicing Lists

- Accessing elements within lists
 - `a`
 - `a[0]`
 - `a[2:4]`
 - `a[:3]`
 - `a[-1]`
 - `a[1:]`

Dictionaries

- Key-value pairs of potentially heterogeneous data
- `b = { 'a':4, 'b':[1,2], 62:'c' }`
- `b[ 'a' ]`
- `b[ 'a' ] = 2`
- `b`
- `b.keys()`
- `b.values()`
- `b.items()`



Files

- Files read and written sequentially
- `fid = file('small.vcf', 'r')`
- `fid.readline()`

Using libraries

- `import numpy`
- `a = numpy.array([3,4,5])`
- `import numpy as np`
- `b = np.array([5,6,7])`
- `import this`

Plotting with Matplotlib

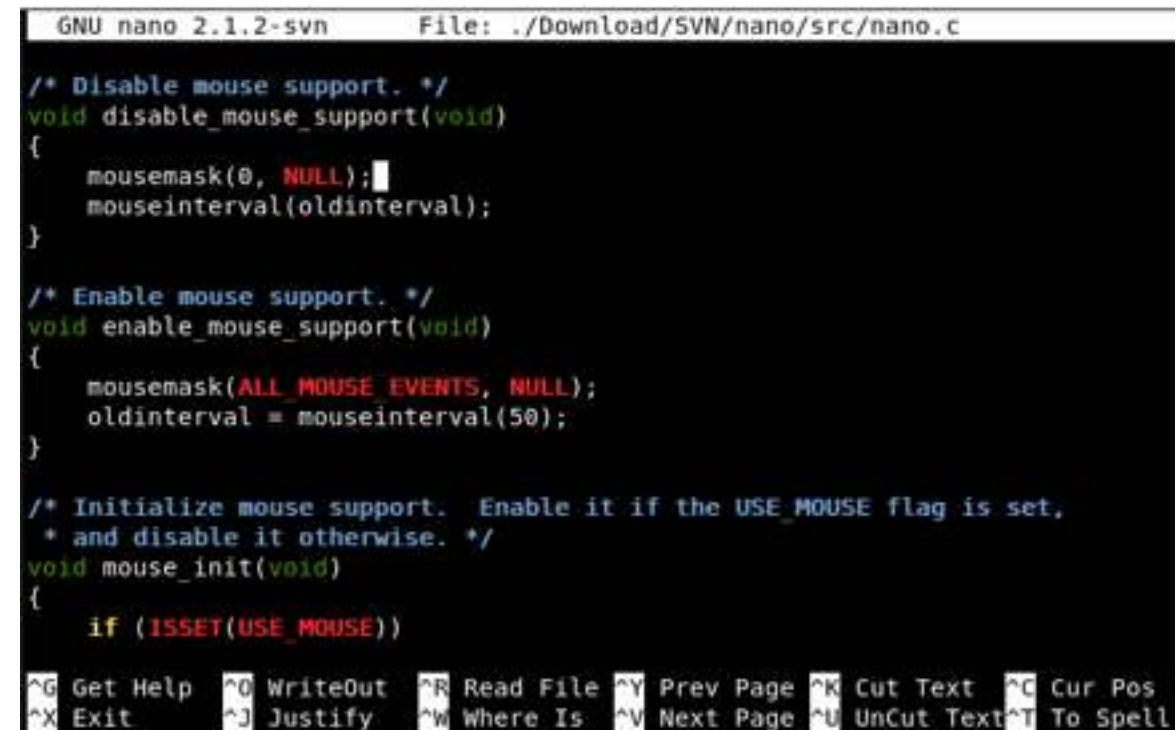
- `import matplotlib.pyplot as plt`
- `x = np.linspace(0, 4*np.pi, 1000)`
- `plt.plot(x, np.sin(x), '-r')`
- `plt.show()`
Not necessary in iPython, but needed in scripts.

My work style with iPython

- `%run <filename>`
 - Runs code in `filename`, with a clear namespace
- `%run -i <filename>`
 - Runs code in `filename`, with access to all currently defined variables
- So I have an editor window open with my script, which I'm continually re-running as I edit.
- If I have slow steps, I comment out the code that generates them and use `%run -i` to use live data.

nano

- Simple text editor available on most systems.
- Ctrl+o to save files
- Ctrl+x to exit
- If your favorite editor (vim, emacs, etc.) is available, feel free to use it.
- Start editing a new file
nano test.py



```
GNU nano 2.1.2-svn      File: ./Download/SVN/nano/src/nano.c

/* Disable mouse support. */
void disable_mouse_support(void)
{
    mousemask(0, NULL);
    mouseinterval(oldinterval);
}

/* Enable mouse support. */
void enable_mouse_support(void)
{
    mousemask(ALL_MOUSE_EVENTS, NULL);
    oldinterval = mouseinterval(50);
}

/* Initialize mouse support.  Enable it if the USE_MOUSE flag is set,
 * and disable it otherwise. */
void mouse_init(void)
{
    if (ISSET(USE_MOUSE))

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```


Functions

- `def adder(a, b):
 return a+b+2`
- `adder(2, 3)`
- `adder(np.array([4, 5, 6]),
 np.array([-1, 2, 3]))`
- `adder(np.array([4, 5, 6]), 2)`
- `adder(np.array([4, 5, 6,]),
 np.array([-1, 2]))`

Conditionals

- `if 5 < 10:`
 `print 'hello'`
- `if 5 < 10 and 10 < 7:`
 `print 'no'`
- `if 5 < 10 or 10 < 7:`
 `print 'hello'`

For loop

- `for ii in range(5):`
 `print ii`
- `for ii, val in enumerate('abcd'):`
 `print ii, val`
- `for val1, val2 in zip('abcd', 'wxyz'):`
 `print val1, val2`
- `for line in file('test.dat'):`
 `print line`

While loop

- `a = 0`
- `while a < 9:`
 `print a`
 `a = a + 4`

Exercises

- Two sets of exercises available:
 - DataProcessing.pdf: Parse a frequency spectrum (crudely) from 1000 Genomes Data.
 - Simulation.pdf: Simulate the Wright-Fisher model
- Start with the one that seems most interesting to you.
- Work together!
- Make a new directory, and switch to it.
`mkdir dadiExercise; cd dadiExercise`
- To get the data on your instance, run
`wget gutengroup.mcb.arizona.edu/temp/wspg2016.tgz`
`tar -xzf wspg2016.tgz`