A Gentle Introduction to Python

Phuong Vu

Department of Biostatistics
University of Washington

December 6, 2017

Acknowledgment

What is Python?

- A programming language
- Quite similar to R
- Take spacing **very seriously**

Why Use Python?

According to Noah Simon:

- Hacking around with a robot
- Building a website
- Good machine learning libraries
- Better/cleaner for building big systems
- Inference is pretty meh...

My reason for learning `Python`?...

Why Use Python?

Sometimes Python is just faster than R, especially with loops...

In R

```
ptm1 <- proc.time()
for(i in 1:100000) {
  print("Hi")
}
print("Finished!")
ptm2 <- proc.time()

print(ptm2 - ptm1, digits = 4)
```

Result:

```
user   system elapsed
2.67    0.08    2.63
```

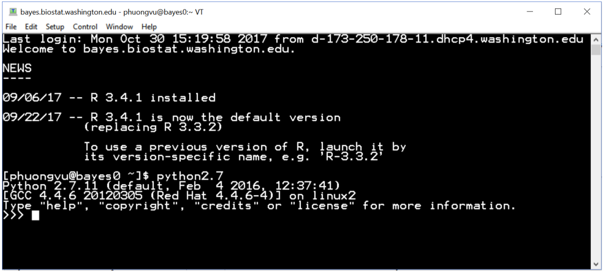
In Python

```
import time

timer_start = time.time()
for i in xrange(1, 100000):
  print "Hi!"
print "Finished!"
timer_end = time.time()

print timer_end - timer_start
```

Result:

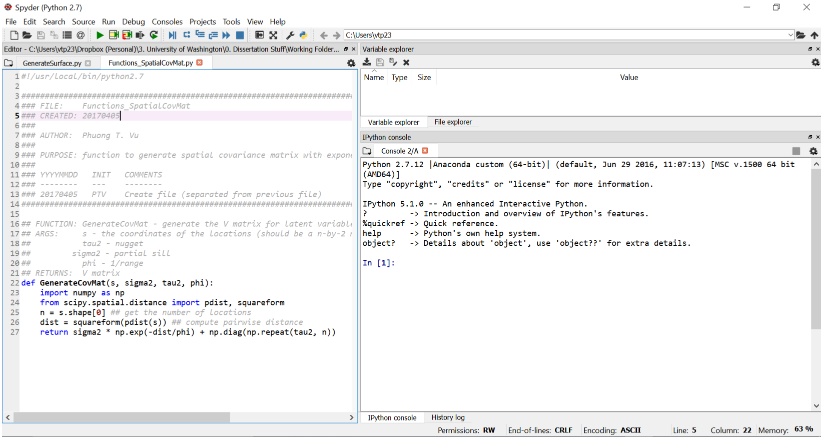
```
0.946000099182
```

How to Run Python?



```
Last login: Mon Oct 30 15:19:58 2017 from d-173-250-178-11.dhcp4.washington.edu
Welcome to bayes.biostat.washington.edu.

NEWS
----

09/06/17 -- R 3.4.1 installed

09/22/17 -- R 3.4.1 is now the default version
            (replacing R 3.3.2)

            To use a previous version of R, launch it by
            its version-specific name, e.g. 'R-3.3.2'

[phuongvu@bayes0 ~]$ python2.7
Python 2.7.11 (default, Feb  4 2016, 12:37:41)
[GCC 4.4.6 20120305 (Red Hat 4.4.6-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

How to Run Python?


Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\vtp23
Editor - C:\Users\vtp23\Dropbox (Personal)\3. University of Washington\0. Dissertation Stuff\Working Folder...
GenerateSurface.py
Functions_SpatialCovMat.py
Variable explorer
Name Type Size Value
Variable explorer File explorer
IPython console
Console 2/A
IPython console History log

```
#!/usr/local/bin/python2.7

#########################################################################
### FILE:     Functions_SpatialCovMat
### CREATED:  20170405
###
### AUTHOR:   Phuong T. Vu
###
### PURPOSE:  function to generate spatial covariance matrix with expone
###
### YYYYMMDD   INIT   COMMENTS
### --------   ---    --------
### 20170405   PTV    Create file (separated from previous file)
#########################################################################

## FUNCTION: GenerateCovMat - generate the V matrix for latent variabl
## ARGS:      s - the coordinates of the locations (should be a n-by-2 r
##            tau2 - nugget
##          sigma2 - partial sill
##            phi - 1/range
## RETURNS:   V matrix
def GenerateCovMat(s, sigma2, tau2, phi):
    import numpy as np
    from scipy.spatial.distance import pdist, squareform
    n = s.shape[0] ## get the number of locations
    dist = squareform(pdist(s)) ## compute pairwise distance
    return sigma2 * np.exp(-dist/phi) + np.diag(np.repeat(tau2, n))
```

```
Python 2.7.12 |Anaconda custom (64-bit)| (default, Jun 29 2016, 11:07:13) [MSC v.1500 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Permissions: RW   End-of-lines: CRLF   Encoding: ASCII   Line: 5   Column: 22   Memory: 63 %

Python 2.x or Python 3.x?

- Official on Python Wiki: *Python 2.x is legacy, Python 3.x is the present and future of the language*
- Final version of 2.x is 2.7, and for 3.x is 3.6 released in 2016
- Pros and cons:
    - 3.x is on the cutting edge, i.e. moving forward new features will be implemented on 3.x but not added to 2.x
    - 2.x has much better library support and documentation
    - 2.x has much more extensive and specific third-party packages or utility that may not have been released on 3.x yet
    - Some Linux distributions and Macs still use 2.x as default
- Me? Python 2.7 ¯_(ツ)_/¯- it's just a tool...

Python modules

- To do any serious (statistical) work in Python, you need to extend it with modules, which are similar to `R` libraries
- Frequently used modules often come with the Anaconda distribution or are already installed on the department clusters: `numpy`, `scipy`, `pandas`, `scikit-learn`
- Other stuff you may like: `CvxPy` (convex optimization), `TensorFlow` by Google (deep learning)

Python modules

- To do any serious (statistical) work in Python, you need to extend it with modules, which are similar to `R` libraries
- Frequently used modules often come with the Anaconda distribution or are already installed on the department clusters: `numpy`, `scipy`, `pandas`, `scikit-learn`
- Other stuff you may like: `CvxPy` (convex optimization), `TensorFlow` by Google (deep learning)
- How much of these cool modules will we cover today? Not much!

How Similar is Python to R?

- `Python` is a lot like `R` - there are lists, strings, arrays, and functions which should all seem similar, although some syntax is different
- Indentation/spacing are taken **very seriously**!
- Indexing starts at **0, not 1**!
- Help files are very well-documented:
  - For `python2.x`: `https://docs.python.org/2/tutorial/`
  - For `python3.x`: `https://docs.python.org/3/tutorial/`

Methods and Objects

- Everything in Python is an object
- Objects have classes:

```
In [1]: x = "statistics"
In [2]: x.__class__
Out[2]: str
```

- Each class has methods and attributes associated with objects of that type. Access these with the "." operator. Find out what methods and object has with dir.

```
In [3]: dir(x)
Out[3]: <omit for space limit>
In [4]: x.capitalize()
Out[4]: ’Statistics’
In [5]: print x.__doc__
Out[5]: str(object=’’) -> string
Return a nice string representation of the object.
If the argument is a string, the return value is the same object.
```

- You can get a long way just using the built-in methods

Example: Loops in Python

Silly example:

```
import numpy as np

x = np.zeros(shape = (11, 2)) ## create a 11x2 array of zeros
x[:, 1] = np.linspace(start = 0, stop = 10, num = 11) ## replace
the 2nd column with some non-zeros

## this is just to show how a for loop in python looks like...
for i in range(11):
    x[i, 0] = x[i, 1]/3 + 1

## you should probably do this instead...
x[:, 0] = x[:, 1]/3 + 1
```

Example: Matrix Algebra in Python

```
import numpy as np

A = np.array([[1, 2, 3], [4, 5, 6]]) ## create a 2x3 array
A.T ## transpose of A

## this gives you error
b1 = np.array([0.5, 1])
A.dot(b1) ## mismatched shapes

## this runs, but final answer is a (2, ) array
b2 = np.array([0.5, 1, 2])
A.dot(b2) ## shapes are not aligned

## this runs, and final answer is a (2, 1) array (column vector)
A.dot(np.reshape(b2, (3, 1)))
```

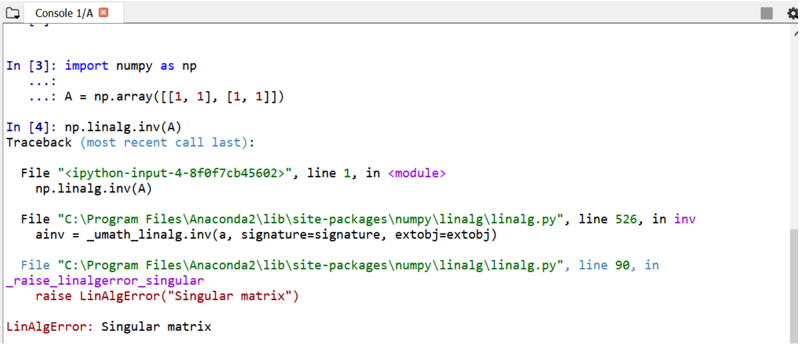
- Arrays and matrices are two different objects in python
- You should almost always stick with arrays
- Make sure you always know/check the dimensions

Example: Data manipulation in Python

```
import numpy as np
import pandas as pd

## read a csv file
data = pd.read_csv(’exposure_surface.csv’)
data[’covariate.1’] ## see the variable named "covariate.1"

## convert an array into a pandas dataframe and print to a .csv file
n = 100
p = 5
fakeX = np.reshape(np.random.normal(0, 1, n*p), (n, p))
fakeY = np.random.normal(0, 1, n)

fakedata = np.column_stack( (fakeX, fakeY) )
cols = [’var.’ + str(i+1) for i in range(p)] + [’outcome’]
fakedata_pd = pd.DataFrame(fakedata, columns = cols)
fakedata_pd.to_csv("fakedata.csv", index = False)
```

Excellent Error Traceback

Python tells you exactly what and where (i.e. on which line of which function/module) the error is!



```
In [3]: import numpy as np
   ...:
   ...: A = np.array([[1, 1], [1, 1]])

In [4]: np.linalg.inv(A)
Traceback (most recent call last):

  File "<ipython-input-4-8f0f7cb45602>", line 1, in <module>
    np.linalg.inv(A)

  File "C:\Program Files\Anaconda2\lib\site-packages\numpy\linalg\linalg.py", line 526, in inv
    ainv = _umath_linalg.inv(a, signature=signature, extobj=extobj)

  File "C:\Program Files\Anaconda2\lib\site-packages\numpy\linalg\linalg.py", line 90, in
_raise_linalgerror_singular
    raise LinAlgError("Singular matrix")

LinAlgError: Singular matrix
```

Alias - Friend or Foe?

- When two identifiers refer to the same variable (and therefore value), this is known as an **alias**
- In `R`, `X <- f(X)` generally makes a copy of `X`, which sometimes mean slower performance due to making too many copies of data... In `Python`, `f(X)` needs not...

Alias - Friend or Foe?

Sometimes aliasing is bad... For example:

- In `R`, you sometimes do this to generate placeholders:

  ```
  >>> output1 <- output2 <- rep(0, nsim)
  ```

  and changing `output2` does not change `output1`

- In `Python`, you will end up with 2 output arrays that look exactly the same...

  ```
  In [1]: import numpy as np

  In [2]: output1 = output2 = np.zeros((5))

  In [3]: print id(output1), id(output2)
  Out[3]: 204788400 204788400

  In [4]: output1[1] = 0.5 ## change output1

  In [5]: output2 ## check output2...
  Out[5]: array([ 0. ,  0.5,  0. ,  0. ,  0. ])
  ```

Run as Script?

- Similar to R, you will need this line for the `.py` file to be executed on Cox or Bayes/Gosset:

  ```
  #!/usr/local/bin/python2.7
  <your python code>
  ```

- Submit on Cox:

  ```
  [phuongvu@cox ~]$ /usr/local/bin/python2.7 Sim.py &
  ```

Run as Script?

- Submit on Bayes/Gosset:

```
[phuongvu@bayes0 Dissertation]$ ./submit_Sim.sh

[phuongvu@bayes0 Dissertation]$ cat submit_Sim.sh
#!/bin/sh
qsub -t 1-22 -cwd -e Trashfiles/ -o Trashfiles/ -q s-normal.q
-M phuongvu@uw.edu -m e call_Sim.sh

[phuongvu@bayes0 Dissertation]$ cat call_Sim.sh
#!/bin/sh
<extra stuff omitted>
/usr/local/bin/python2.7 Sim.py
```

- The current underlying BLAS libraries for `python2.7` on Bayes/Gosset are NOT optimized. If you seriously consider `python` for your research, you should consider link your modules with a more optimized BLAS library (ask BITE!)

Final Remarks

- The best way to learn Python is diving into some coding projects and using google/stackoverflow/etc.
- Python2.x or 3.x? Just pick one!
- OOP, by definition, is designed in a manner to minimize the amount of new code that needs to be written. There maybe a well-written and well-tested module somewhere that can do the job you want.
- For those who are interested in spatial statistics or GIS stuff: Python is the primary scripting language for ArcGIS
- Don't hesitate to ask your fellow students for tips and tricks!

Homework ¯_(ツ)_/¯

Write a script in `Python` to execute the following:

1. Choose a vector $\boldsymbol{\beta} \in \mathbb{R}^p$ for $p = 3$, and generate data:

$$
\begin{aligned}
\mathbf{y} &= \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \\
\mathbf{X} \in \mathbb{R}^{n \times p},\ \mathbf{y} &\in \mathbb{R}^n,\ \boldsymbol{\epsilon} \in \mathbb{R}^n,\ n = 300 \\
\mathbf{X}_{i1} &= 1 \text{ for } i = 1, \dots, n \\
\mathbf{X}_{i2} &\sim \text{Bernoulli}(0.7) \text{ for } i = 1, \dots, n \\
\mathbf{X}_{i3} &\sim \text{Uniform}(-1, 1) \text{ for } i = 1, \dots, n \\
\boldsymbol{\epsilon}_i &\sim \mathcal{N}(0, 1) \text{ for } i = 1, \dots, n
\end{aligned}
$$

2. Find $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$
3. Save your generated $\mathbf{X}$ and $\mathbf{y}$ to a `.csv` file

In `R`, read in this .csv file, and confirm that your result, either by doing the algebra or using `lm()`

You will only need to google functions from `numpy` and `pandas`.