Section 9: Python vs. R

Zoe Vernon

10/30/2020

This week, we will be exploring the behavior of Python for several common actions that we have previously covered in R. The final product of today's work will be a pdf which is uploaded as a group submission to Gradescope.

There is some example python code in this file

Instructions

1) We will separate into groups of (ideally) 3 or (if necessary) 4.
2) Groups should try to answer every question in the Main Questions section. There are 2 ways to go about this:
   - Everyone can work on a different question, until all of the questions are answered
   - The group can work together, answering each question in turn
   - Feel free to use a mixture of the above methods, helping if members become stuck on any questions.
   - Your group is expected to work on this for ~50 minutes. If you do not make it through all the questions that is okay, as long as it is clear that you all made an effort.

3) Combine the solutions into one document that one group member should upload to the problem set 6 group work assignment on Gradescope.
   - Make sure to add your fellow group members to the assignment on Gradescope when you submit.
   - How you create the pdf to submit is up to you but some options are (1) create an IPython notebook and save as pdf. (2) create an R Markdown file with the code and knit to pdf.

Questions

Main Questions

(Ideally, you will make it through all of these, although there if you run out of time that is okay.)

1) Do Python functions behave like pass-by-value or pass-by-reference
   i.e., if you pass in an object and modify it, does that affect the value of the object in the environment from which the function was called?
   Check this for a scalar, a list, and a numpy array.
2) If you copy a list, dictionary, or numpy array in Python, are the values copied or does the new object just use the same memory as the original object?
3) How are `NA`s handled in Python lists? What about in numpy arrays?
4) Do Python functions use promises/lazy evaluation?
5) Assess whether it is inefficient to grow a list in Python, as it is in R. Consider whether a copy is made when the object grows.

6) How does variable scoping work in Python - does it use lexical scoping and look for variables in the environment where a function was defined?

7) Are the maximum and minimum sizes of integers and real-valued numbers the same as they are in R?

8) Consider the relative efficiency of for loops versus vectorized calculations for numpy arrays and see how it compares to the equivalent operation in R.

9) Can lists and numpy arrays be modified in place, without copying the object?

10) Consider whether Python allows you to have functions and variables in the global environment that have the same names as functions/variables in packages or in modules (e.g., a file test.py in your working directory that you can import using 'import test'). E.g. consider math.cos and create your own 'cos' function. How does this compare to how R finds objects?

Additional Questions

(Work on these if you finish quickly/are curious)

12) Can you create a closure with embedded data, like we did in R?

13) Can you determine if the speed of looking up values in a dictionary varies with the size of the dictionary (this will indicate if something like hashing is going on or if the look up has to scan through all the elements).

14) Compare the Python debugger to R's debugger.

15) If you create classes and objects in Python's object-oriented system, what are the similarities and differences relative to R's R6 system? There is a brief section on object-oriented programming in Python in the lab 7 materials.