Homework 1 Report

Mohammad Hassan Salim

Msalim7

Part 1: Get Familiar with R

In familiarizing myself with R, I noticed that it's very robust Pythonic-like functional programing language. When calling R robust, I'm referring to its vast built in functionality. It has a function for almost everything. If you want to query a dataset, you can pass a conditional within indexing the dataset:

```
w = seq(0, 1, length.out = 11)
print(w)
print(w[w < .5])
```

This will access all elements from w that has the value less than .5. You can also do conditional checks across the entire dataset:

```
print(w <= 0.5)
print(any(w <= 0.5))
print(all(w <= 0.5))
print(which(w <= 0.5))
```

This is very useful for quickly hacking up code to check the properties of datasets. I've mentioned that R is Pythonic-like. R and Python share similar coding techniques such as slicing. You can easily access subsets of datasets by indexing specific rows and columns.

```
data("iris")
print(names(iris))
print(iris[1:4,])
print(iris[1,])
print(iris$Sepal.Length[1:10])
```

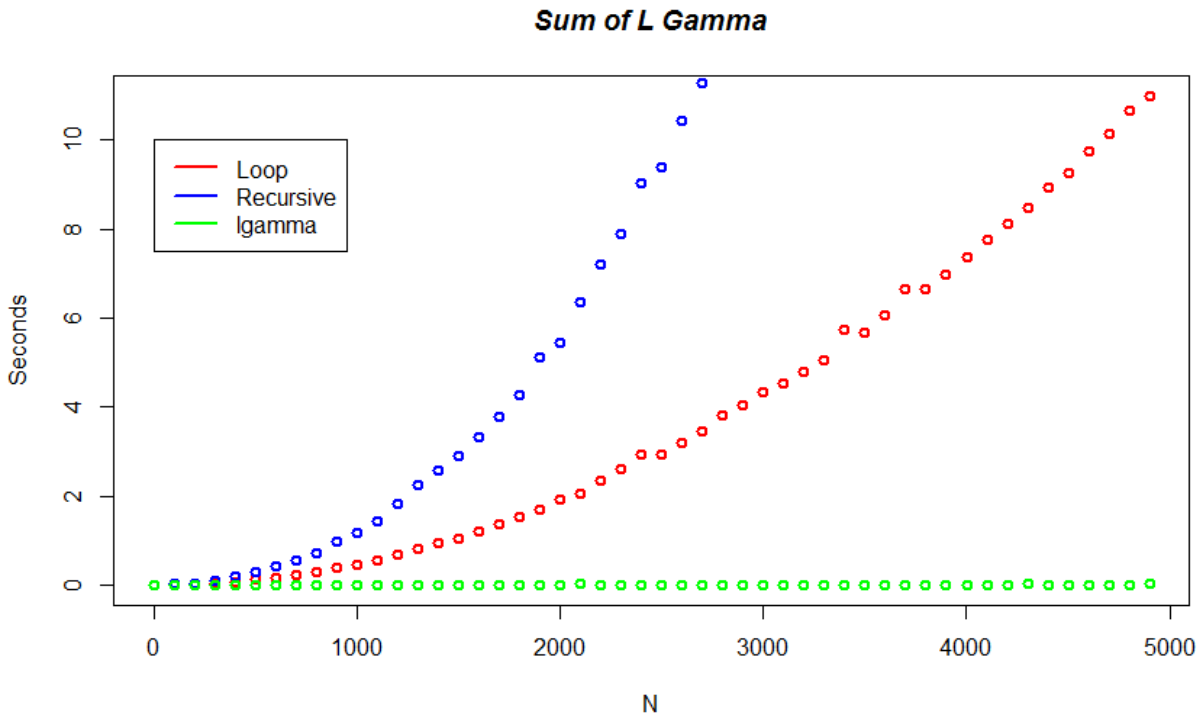
Finally, R seems to be functional. This is may seem obvious, but I want to note that you can do object oriented programming in R. However, after writing some sample code, R naturally writes as functional and probably should be used as so.

Part 5: Compare Results to Built-In R Function

I ran 3 functions that computed the sum of log gammas given an N. I did no use vectorization to give speed boost, because I couldn't get it to work for the loop and recursive variations, so to be fair I didn't vectorize the built-in lgamma function. I ran from 1 to N where N is 5000 with a step count of 100. With any N greater and any step count lower, the code ran severely slow. I believe this N and step count is enough to show the differences in speed between the 3 functions.

Somewhere around 2500 iterations, the recursive function stopped working. The recursion nested too deeply even though I set expressions to 500000. The recursive algorithm is by far the slowest. This could be because it's constantly adding nested function calls to the stack. This will maximize memory allocation and forcing the machine to not take advantage of RAM as well as it could if it were the loop function. The loop function was slow too, but it had a more linear relationship than the recursive function did (which seemed far more exponential). The built-in function is crazy fast. It seems to be just as fast when no matter the range of N (without going over 5000). I'd also like to note, all 3 functions had very similar speeds up until N was 500.

I suppose the lesson learned is that to use built-in functions when you can. I'd imagine the lgamma function uses C/C++ code underneath the hood and mathematically optimizes.



Function	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Loop	0.000	0.722	2.935	3.779	6.500	10.990	4851
Recursive	0.000	1.238	5.900	8.805	14.710	26.640	4059
LGamma	0.000	0.000	0.000	0.002	0.000	0.020	4851