HW1: R Programming

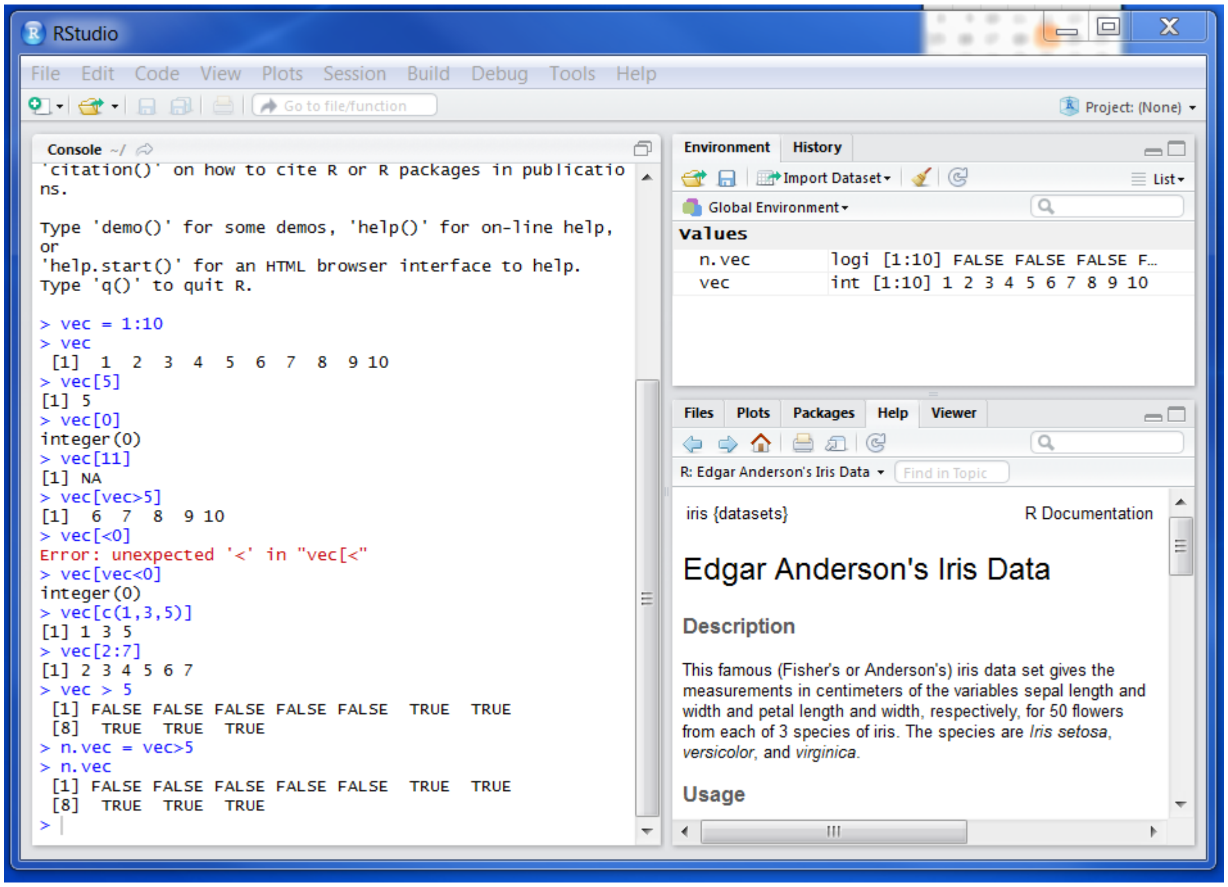
Xiaodong Gu (xgu60)

1. Get Familiar with R [10 points]

Familiarize yourself with the R code from the first reading assignment. Run the code in an R environment of your choice and observe the results. Briefly describe one insight you learned about R in your observations. Illustrate it with a sample code snippet and observed output.

One impression that I have is that R has very flexible operations on vectors/arrays/lists. Elements of vectors can be identified by their indexes, or logical operations. New vector of Booleans can be generated just by a simple comparison. One difference between R and java/python is that the index starts from 1 not 0. It is hard to tell which one is more reasonable (start from 0 or 1). Some small syntax differences may be hard to debug for python/java programmers, e.g. in a function, the return values must be located in parentheses.

Figure 1. Snippet of my R working environment.



```
'citation()' on how to cite R or R packages in publicatio
ns.

Type 'demo()' for some demos, 'help()' for on-line help,
or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> vec = 1:10
> vec
 [1]  1  2  3  4  5  6  7  8  9 10
> vec[5]
[1] 5
> vec[0]
integer(0)
> vec[11]
[1] NA
> vec[vec>5]
[1]  6  7  8  9 10
> vec[<0]
Error: unexpected '<' in "vec[<"
> vec[vec<0]
integer(0)
> vec[c(1,3,5)]
[1] 1 3 5
> vec[2:7]
[1] 2 3 4 5 6 7
> vec > 5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
 [8]  TRUE  TRUE  TRUE
> n.vec = vec>5
> n.vec
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
 [8]  TRUE  TRUE  TRUE
> 
```

```
Values
  n.vec     logi [1:10] FALSE FALSE FALSE F…
  vec       int [1:10] 1 2 3 4 5 6 7 8 9 10
```

iris {datasets} R Documentation

Edgar Anderson's Iris Data

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

Usage

5. Compare Results to Built-In R Function [30 points]

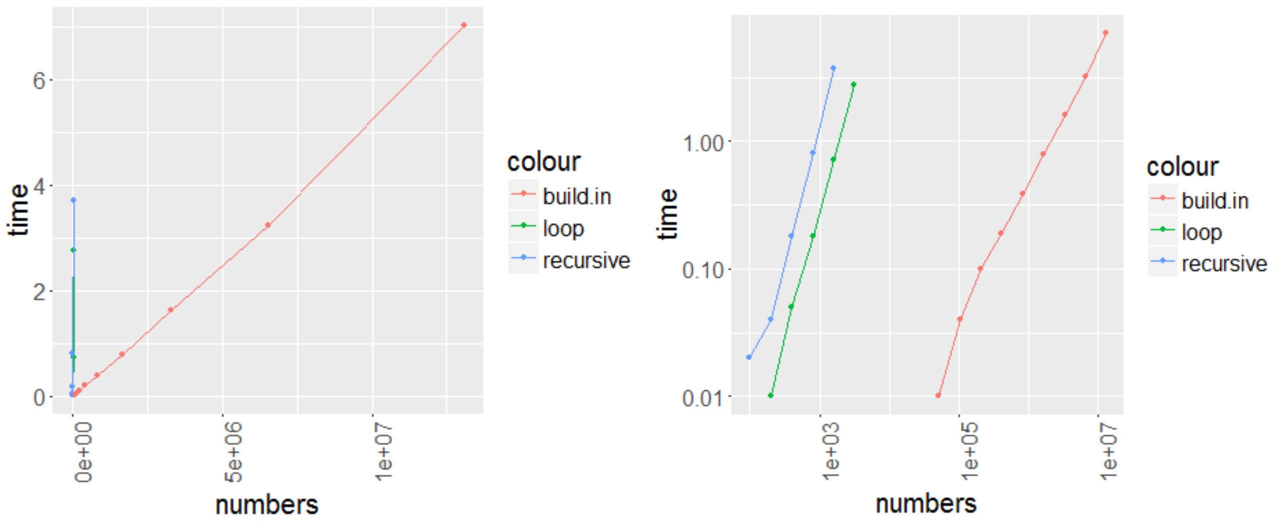
Compare the execution times of your two implementations from problem 4 with an implementation based on the built-in R function lgamma(n) . You may use the function system.time() to measure execution time. What are the growth rates of the three implementations as n increases?

Note: Use the command options(expressions=500000) to increase the number of nested recursions allowed. Compare the timing of the recursive implementation as much as possible (until an overflow occurs), and continue beyond that for the other two implementations.

Optional: You may want to collect the running times and plot them using R. Note that system.time() returns a 5-element vector, the first of which is the user time that we are interested in (see proc.time).

The implementations in problem 4 deliver the sum of log_gamma values. For better comparison, I wrote a new function sum_lgamma(n) which directly calls the build-in function lgamma(n) in a for loop. I also wrote functions running.time(num.seq) to do the experiment. Num.seq is a vector of numbers, e.g. (100, 200, 400, 800, 1600, 3000) , and the function return a data frame of running time of different implementations as shown in the following figure. It seems that the recursive implementation costs much more running time compared with loop implementation or the one with build-in lgamma function. By using the build-in function lgamma(), system can compute much larger number in linear time (Fig 2, left). The running times for both loop and recursive functions are exponentially increased with numbers as shown in log-log plot (Fig2, right). The build-in lgamma() function must be implemented in C or C++, thus has high efficiency in computing.

Figure 2. the plot of running time versus n using different implementations (calling loop, recursive or build-in log gamma functions). Left: normal scale, Right: both x and y use log10 scales.


time
0e+00
5e+06
1e+07
numbers
colour
build.in
loop
recursive
time
1e+03
1e+05
1e+07
numbers
colour
build.in
loop
recursive