CSE 6242 Assignment 1

Vincent La (Georgia Tech ID - vla6)

September 3, 2017

```
# Assignment 1: http://cse6242.gatech.edu/fall-2017/hw1/
# To compile R Markdown in terminal run: Rscript -e "rmarkdown::render('hw1.Rmd', clean=TRUE)"
# To create zip file: zip hw1.zip hw1.Rmd hw1.pdf

require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(rmarkdown)
```

```
## Loading required package: rmarkdown
```

```
options(expressions=500000)
```

Question 1: Getting Familiar with R

Below is a code snippet where I print hello world, and also run some code examining factor type. One thing I learned is how to work with factor types in R and how to set an ordered factor. See the `get_familiar_with_r` function defined below. In the example below, I create a factor variable with four levels: summer, fall, winter, and spring, and ordered them.

```
get_familiar_with_r <- function(){
    #' This function is for the first part of assignment 1
    #' Run some code examples and observe results
    #' Briefly describe one insight you learned about R in your observations
    #' Illustrate with a sample code snippet and observed output

    # Hello World in R
    print('hello world')

    # Display data sets in ggplot 2 library
    # print(data(package = 'ggplot2'))

    # Working with factors
    current.season = factor('summer',
              levels = c('summer', 'fall', 'winter', 'spring'),
              ordered = TRUE) # Ordered factor
    print(current.season)
    print(levels(current.season))

    print('goodbye world')
}
get_familiar_with_r()
```

```
## [1] "hello world"
## [1] summer
## Levels: summer < fall < winter < spring
## [1] "summer" "fall"   "winter" "spring"
## [1] "goodbye world"
```

Question 2: Log Gamma (Loop Implementation)

```
log_gamma_loop <- function(n){
    #' Computes and returns the natural logarithm of the gamma value of a positive integer
    #' using an iterative loop
    #' log gamma is defined as ln((n-1)!) = ln(n-1) + ln(n-2) + ... + ln(1)

    sum = 0
    for (i in seq(n - 1, 1, by = -1)) {
        sum = sum + log(i)
    }
    return(sum)
}
log_gamma_loop(5)
```

```
## [1] 3.178054
```

Question 3: Log Gamma (Recursive)

```
log_gamma_recursive <- function(n){
    #' Computers and returns the natural logarith mof the gamma value of a positive integer
    #' using recursion
    #' log gamma is defined as ln((n-1)!) = ln(n-1) + ln(n-2) + ... + ln(1)

    # Can stop at n == 2 since n-1 = 1 and log(1) equals 0
    if (n == 2){
        return(0)
    } else {
        sum = log(n - 1) + log_gamma_recursive(n - 1)
        return(sum)
    }
}
log_gamma_recursive(5)
```

```
## [1] 3.178054
```

Question 4: Sum of Log Gamma

```r
sum_log_gamma_loop <- function(n){
    #' Uses log_gamma_loop defined above to sum the log Gamma results over
    #' the range 1 to n
    sum = 0

    # Start at 2 because log(1 - 1) is undefined
    for (i in seq(2, n, by=1)) {
        sum = sum + log_gamma_loop(i)
    }
    return(sum)
}


sum_log_gamma_recursive <- function(n){
    #' Uses log_gamma_recursive defined above to sum the log Gamma results over
    #' the range 1 to n
    sum = 0

    # Start at 2 because log(1 - 1) is undefined
    for (i in seq(2, n, by=1)) {
        sum = sum + log_gamma_recursive(i)
    }
    return(sum)
}

sum_log_gamma_loop(5)
```

```
## [1] 5.66296
```

```r
sum_log_gamma_recursive(5)
```

```
## [1] 5.66296
```

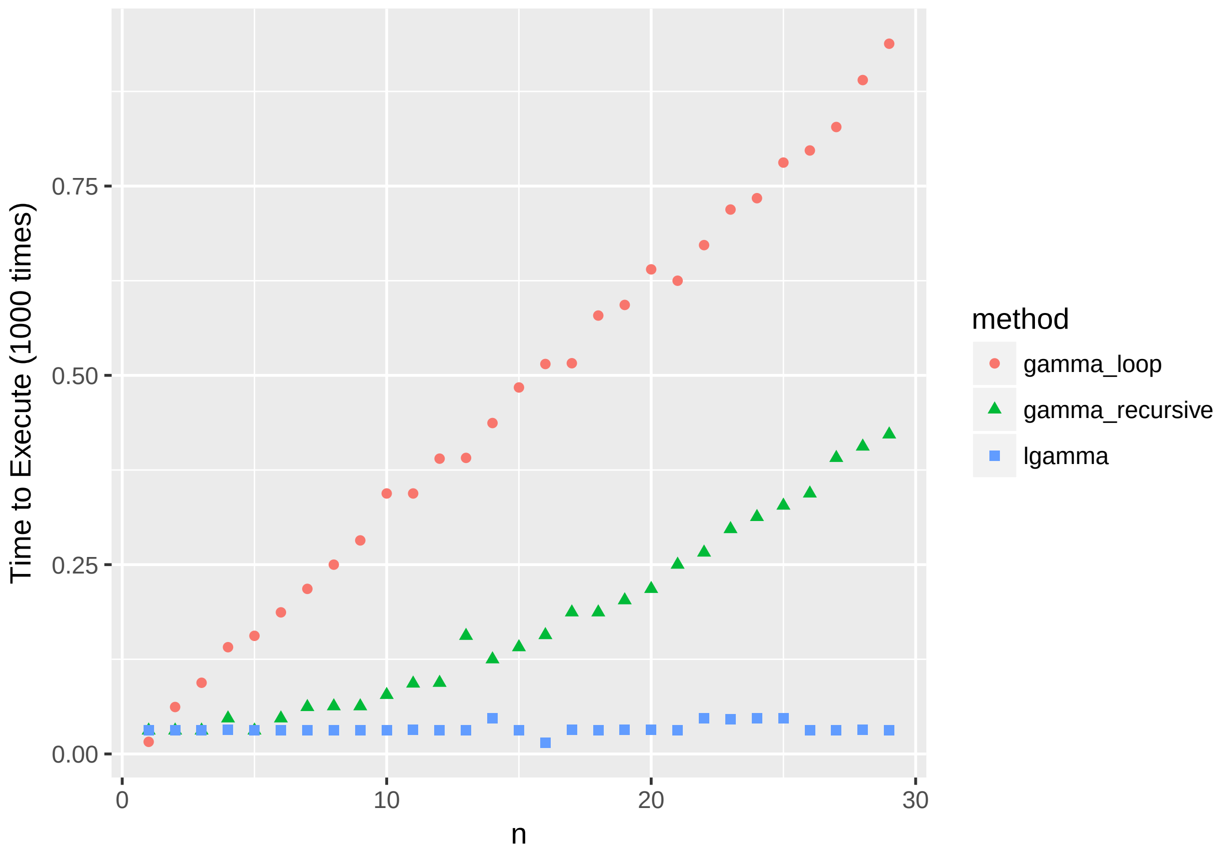
Question 5: Compare Results to Built-In R Function

```r
sum_lgamma <- function(n){
    #' Uses built in R function lgamma(n) to sum the log Gamma results over the
    #' range 1 to n
    sum = 0

    # Start at 2 because log(1 - 1) is undefined
    for (i in seq(2, n, by=1)) {
        sum = sum + lgamma(i)
    }
    return(sum)
}
sum_lgamma(5)
```

```
## [1] 5.66296
```

Next, we draw a graph using `ggplot2` to make comparisons between different methods. For better consistency, we repeat each implementation 1000 times. We then loop over values of n from n = 1 to 30.

```r
make_comparisons <- function(n) {
    #' Compare the execution times of the three implementations of finding the
    #' sum of log gamma defined above
    #' Replicate num_runs times for better consistency

    num_runs = 1000

    gamma_loop = vector('numeric', n - 1)
    gamma_recursive = vector('numeric', n - 1)
    lgamma = vector('numeric', n - 1)
    for (i in seq(2, n, by=1)){
        gamma_loop[i - 1] = system.time(replicate(num_runs, sum_log_gamma_loop(i)))[1]
        gamma_recursive[i - 1] = system.time(replicate(num_runs, sum_log_gamma_recursive(i)))[1]
        lgamma[i - 1] = system.time(replicate(num_runs, sum_lgamma(i)))[1]
    }

    index = c(seq(1, n-1), seq(1, n-1), seq(1, n-1))
    times = c(gamma_loop, gamma_recursive, lgamma)
    method = c(rep('gamma_loop', n-1), rep('gamma_recursive', n-1), rep('lgamma', n-1))

    # To plot: https://stackoverflow.com/questions/13837565/how-to-plot-one-variable-in-ggplot
    df = data.frame(index=index, times=times, method=method)
    ggplot(df, aes(x=index, y=times)) +
        geom_point(aes(color=method, shape=method, group=method)) +
        xlab('n') +
        ylab('Time to Execute (1000 times)')
}


make_comparisons(30)
```



Looks like Loop approach performs the worst. Recursive approach performs better, but the built in R function performs the best.