

Basic Course on R: Object-Oriented Programming Practical Answers

Elizabeth Ribble*

18-24 May 2017

Contents

1	Part A: Object Oriented Programming	2
2	Part B: Performance Enhancement: Speed	3

*emcclel3@msudenver.edu

1 Part A: Object Oriented Programming

1. The *geometric mean* can be defined as the n th root of the product of n positive numbers x_1, x_2, \dots, x_n , i.e.

$$\text{gm} = (x_1 \cdot x_2 \cdots x_n)^{\frac{1}{n}}$$

Write a function `gm()` that takes a vector argument `x` containing positive numbers and returns their geometric mean. Your function should include a `stop()` statement that returns an error message if any of the values in `x` are nonpositive. **Hint:** The function `prod()` can be used to compute the product of the values in `x`.

```
gm <- function(x) {  
  if(any(x <= 0)){  
    stop("All values in the vector must be positive.")  
  }  
  else{  
    y <- prod(x)  
    geom <- y^(1/length(x))  
    return(geom)  
  }  
}
```

2. Determine the class of your output by running the following: `class(gm(1:4))`

```
class(gm(1:4))  
  
## [1] "numeric"
```

3. Modify your geometric mean function, using `class()`, so that the return value has the class "geometric".

```
gm <- function(x){  
  if(any(x <= 0)){  
    stop("All values in the vector must be positive.")  
  }  
  else{  
    geom <- prod(x)^(1/length(x))  
    class(geom) <- "geometric"  
    return(geom)  
  }  
}
```

4. Verify the new class of your output is correct by running the following `class(gm(1:4))`

```
class(gm(1:4))  
  
## [1] "geometric"
```

2 Part B: Performance Enhancement: Speed

1. This problem concerns efficiency and timing code.

Using `system.time(expression)`, explore how time changes with size of the inputs. Plot time versus input size and see if algorithm is linear, polynomial, or exponential.

- Move expressions within loops that are invariant to compute just once and assigned to a variable.
- Avoid concatenating vectors by pre-allocating and assigning to the *i*th element. i.e.

```
for(i in seq(along = y)) x = c(x, g(y[i]))
```

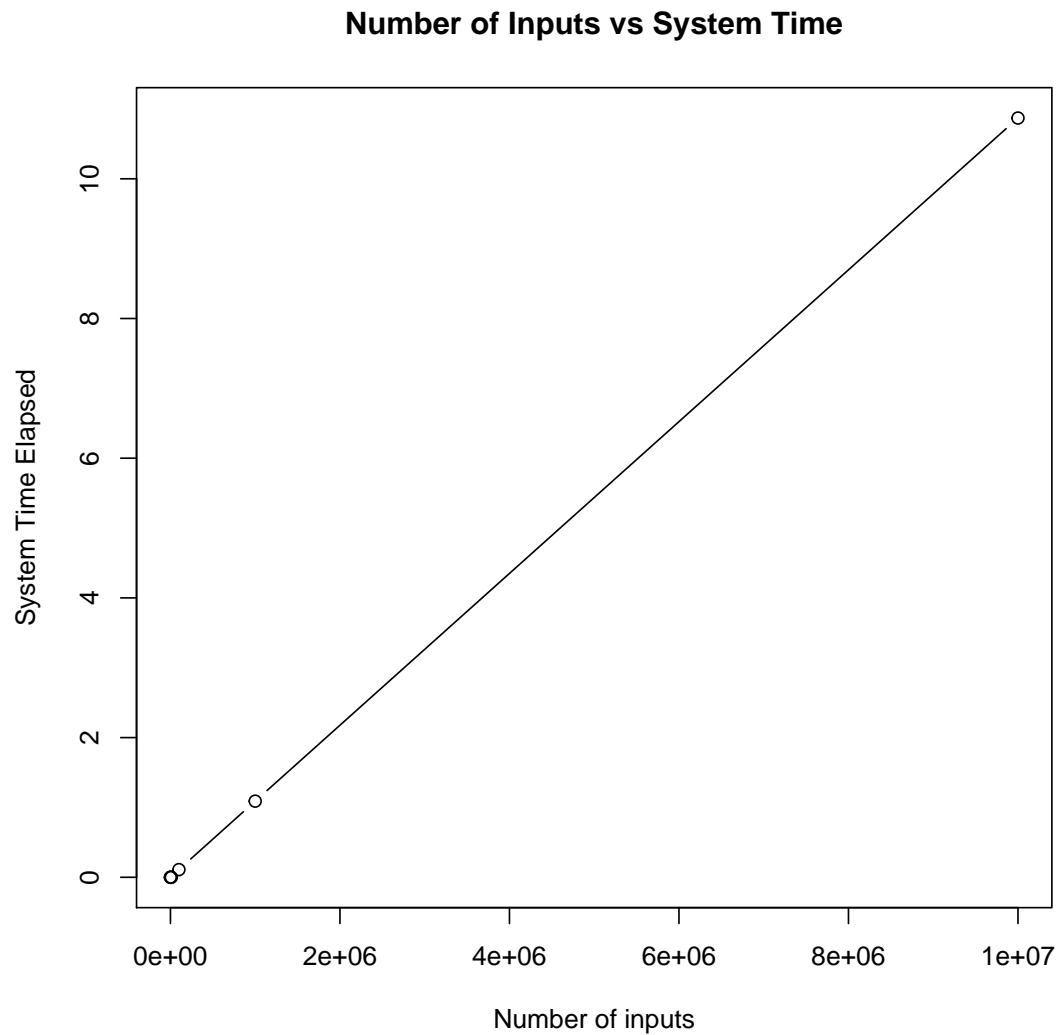
What are your conclusions?

Answers may vary.

```
time <- function(n){  
  x <- runif(n)  
  y <- sqrt(2)  
  t <- system.time( for(i in 1:n) {  
    x[i] <- x[i] + y  
  }  
)  
  return(t[["elapsed"]])  
}  
inputs <- c(100, 10^3, 10^4, 10^5, 10^6, 10^7)  
t1 <- time(100)  
t2 <- time(10^3)  
t3 <- time(10^4)  
t4 <- time(10^5)  
t5 <- time(10^6)
```

```
t6 <- time(10^7)
timeV <- c(t1, t2, t3, t4, t5, t6)
```

```
plot(x = inputs, y = timeV, main = "Number of Inputs vs System Time",
     xlab = "Number of inputs", ylab = "System Time Elapsed",
     type = "b")
```



The scatterplot of computation time versus size of inputs is linear.