Homework 3: STAT5400

Dev Narayan Baiju

19th September 2024

Submit your solutions as an .Rmd file and accompanying .pdf file. Include all the **relevant** R code and output. With the `echo` option in R markdown, you opt to hide some R code that is not very relevant (but still run it to generate the document).

Always comment on your result whenever it is necessary.

Problems

1. The map function in `tidyverse` Redo Question 4 in HW 1, say, write your own factorial function. At this time, explore `map_dbl` function in the R package `tidyverse`.

The function using map_dbl for computing the factorial is as follows:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
mult_func <- function(p){
  return(prod(1:p))
}
fact_func <- function(factnum){
  resulr <- map_dbl(1:factnum, mult_func)
  return(resulr)
}
library(microbenchmark)
print(microbenchmark(factorial(1:7), fact_func(7)))
```

```
## Unit: microseconds
##            expr    min      lq     mean median     uq      max neval
##  factorial(1:7)  1.290  1.4205  2.18753  1.765  1.883   41.033   100
##    fact_func(7) 41.179 42.0625 94.70450 42.502 42.997 2610.913   100
```

The factorial function using map_dbl is slower.

2. The skewness function. Download and install the R package `moments`. Find the following `skewness` function.

```r
"skewness" <-
function (x, na.rm = FALSE)
{
    if (is.matrix(x))
        apply(x, 2, skewness, na.rm = na.rm)
    else if (is.vector(x)) {
        if (na.rm) x <- x[!is.na(x)]
        n <- length(x)
      (sum((x-mean(x))^3)/n)/(sum((x-mean(x))^2)/n)^(3/2)
      }
    else if (is.data.frame(x))
        sapply(x, skewness, na.rm = na.rm)
    else skewness(as.vector(x), na.rm = na.rm)
}
```

Write your own skewness function and use `microbenchmark` library to compare the speed with the `skewness` function in `moment`. Can your function be faster than that? Also make sure your function handles both vectors, matrices, and dataframes, as well as NA values in as the `moment` package.

The following code gives a more optimized version of the skewness function:

```r
myskewness <- function (x, na.rm = FALSE) {
    if (is.matrix(x))
        apply(x, 2, skewness, na.rm = na.rm)
    else if (is.vector(x)) {
        if (na.rm)
            x <- x[!is.na(x)]
        n <- length(x)
        mx <- mean(x)
        diffmean <- (x-mx)
        diffmean2 <- diffmean*diffmean
        diffmean3 <- diffmean2*diffmean
        return((sum(diffmean3)/n)/(sum(diffmean2)/n)^(1.5))
        #Modifications are made in the skewness formula
    }
    else if (is.data.frame(x))
        sapply(x, skewness, na.rm = na.rm)
    else skewness(as.vector(x), na.rm = na.rm)
}
#Let us test both the functions
glam <- rnorm(1000, 30, 3)
library(microbenchmark)
microbenchmark(skewness(glam), myskewness(glam))
```

```
## Unit: microseconds
##              expr    min      lq     mean median      uq      max neval
##    skewness(glam) 62.840 63.4375 237.2015 63.651 64.0265 17255.10   100
##  myskewness(glam) 17.767 18.2140 181.8115 18.394 18.6235 16334.08   100
```

The lowest and median times it took it execute myskewness function is less significantly less than moments skewness function.

3. Monty Hall problem Back in 1970, a game show called “Let’s Make a Deal” was hosted by Monty Hall. Suppose there are three doors on the stage. Behind one door there is a nice prize, while behind the other two there are worthless prizes. A contestant selects one door at random, and then Monty Hall opens one of the other two door to reveal a worthless prize. Monty Hall then expresses the willingness to trade the curtain that the contestant has chosen for the other door that has not been opened. Should the contestant switch curtains or stick with the one that she has?

- Determine the probability that she wins the prize if she switches.
- Use a simulation to verify your results.
- What if there are two prizes and four doors? What about m prizes and n doors, where $m < n$?

This question was completed by the assistance of a classmate and AI.

```r
trials <- 100000
wins <- 0
for (i in 1:trials) {
  prize <- c(rep("good", 1), rep("bad", 2))
  choice <- sample(1:3, 1)

  remaining <- setdiff(1:3, choice)
  monty_opens <- remaining[prize[remaining] == "bad"][1]

  choice2 <- setdiff(1:3, c(choice, monty_opens))

  if(prize[choice2] == "good")
    wins = wins + 1
}
print(wins/trials)
```

```
## [1] 0.66699
```

Now for the case with more doors and prizes:

```r
#Number of prizes and doors as inputs
montyhall <- function(m, n){
  trials <- 100000
  wins <- 0
  for (i in 1:trials) {
    prize <- c(rep("good", m), rep("bad", n-m))
    choice <- sample(1:n, 1)

    remaining <- setdiff(1:n, choice)
    monty_opens <- remaining[prize[remaining] == "bad"][1]

    if(m == 1)
    {
      choice2 <- setdiff(1:n, c(choice, monty_opens))
    }else{
      choice2 <- sample(setdiff(1:n, c(choice, monty_opens)), 1)
    }

    if(prize[choice2] == "good")
      wins = wins + 1
  }
  return(wins/trials)
}
montyhall(2,4)
```

```
## [1] 0.75139
```

4. Coding practice I Check if a positive integer is power of 4.

```r
foo <- function (n) {
  if(n <= 0)
    stop("Number must be positive") #To stop the function and give error message
  else{
    t = log(n, base = 4) #To check if the the number is a power of 4
    return(t == round(t)) #Returns if the log to the base 4 is an integer or not
  }
}
print(foo(64))
```

```
## [1] TRUE
```

5. Coding practice II Write your own function to merge and sort two sorted vectors without using any sorting function such as `sort` or `order`.

```r
set.seed(5400)
foo2 <- function (a1, a2) {
  a3 <- append(a1, a2)
  for(i in 1:length(a3)){
    for(j in i:length(a3)) #Use of nested for loop
      if(a3[i]>a3[j]){
        temp = a3[i] #Using a temporary variable to swap the elements of a3
        a3[i] = a3[j]
        a3[j] = temp
      }
  }
  return(a3)
}
a1 <- sort(sample(8, 8, replace=TRUE))
a2 <- sort(sample(10, 8, replace=TRUE))
print(foo2(a1, a2))
```

```
##  [1]  1  2  2  3  3  3  4  5  5  5  6  8  8  9 10 10
```