Assignment 1

Abdullah Ejaz

September 29, 2017

1 mapply() Function

1.1 Definition

mapply is a function that is applied to multiple list or vector arguments. mapply is a multivariate version of s-apply function which means it applies FUN to each element of every argument. mapply also has the functionality of recycling the arguments if necessary.

1.2 Syntax

```
mapply(FUN,, MoreArgs = NULL, SIMPLIFY = TRUE,
USE.NAMES = TRUE)
```

1.3 Code Snippet for mapply()

```
Mapply(rep, 1:4, 4:1)

[[1]]
[1] 1 1 1 1

[[2]]
[1] 2 2 2

[[3]]
[1] 3 3

[[4]]
[1] 4
```

1.4 Equivalent "for" loop

```
> nn<- function(x){
+ list = list()
+ for(i in 1:x){
+ length<- x-i+1
+ ww<-rep(i,length)
+ list[[i]]<-ww
+ }
+ print(list)
+ }
> nn(4)
```

1.5 Performance Comparison

```
#install differert packages to compare the performance.
> install.packages("microbenchmark")
#install package ggplot2 to plot graph
> install.packages("ggplot2")
> library("microbenchmark")
> library("ggplot2")
# using microbenchmark function to compare
> microbenchmark::microbenchmark(nn(5),mapply(rep, 1:5, 5:1),10)
#Saving microbenchmark to pp variable
> pp<-microbenchmark::microbenchmark(nn(5),mapply(rep, 1:5, 5:1),10)
#replicating the "pp" 100 times using the replicate function
> replicate(100, pp)
#plot the graph
> plot(pp)
> pp<-microbenchmark::microbenchmark(nn(5),mapply(rep, 1:5, 5:1),100)
#Rplot02
> plot(pp)
```

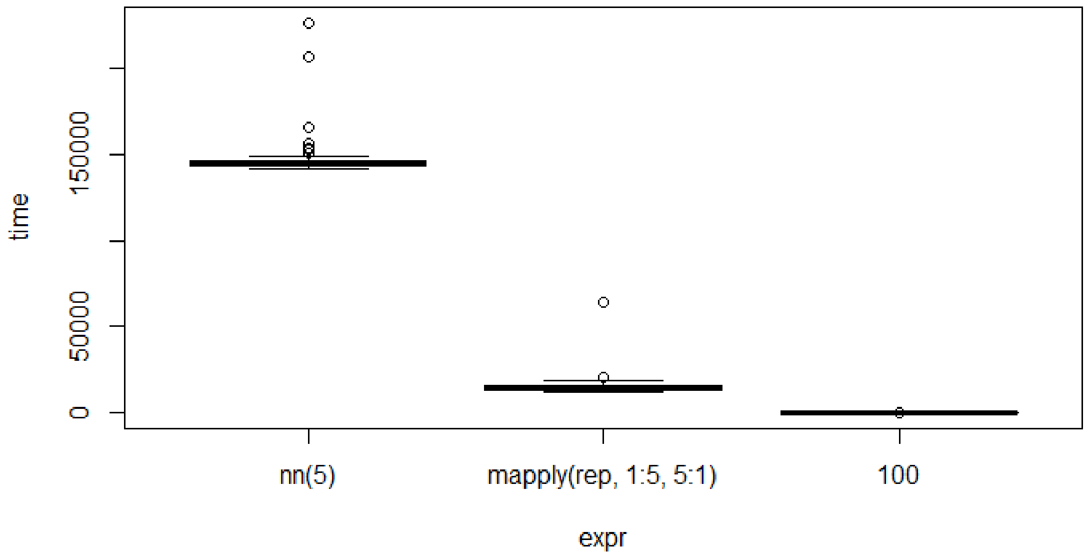

Figure 1: Graph to show that mapply works better in terms of time

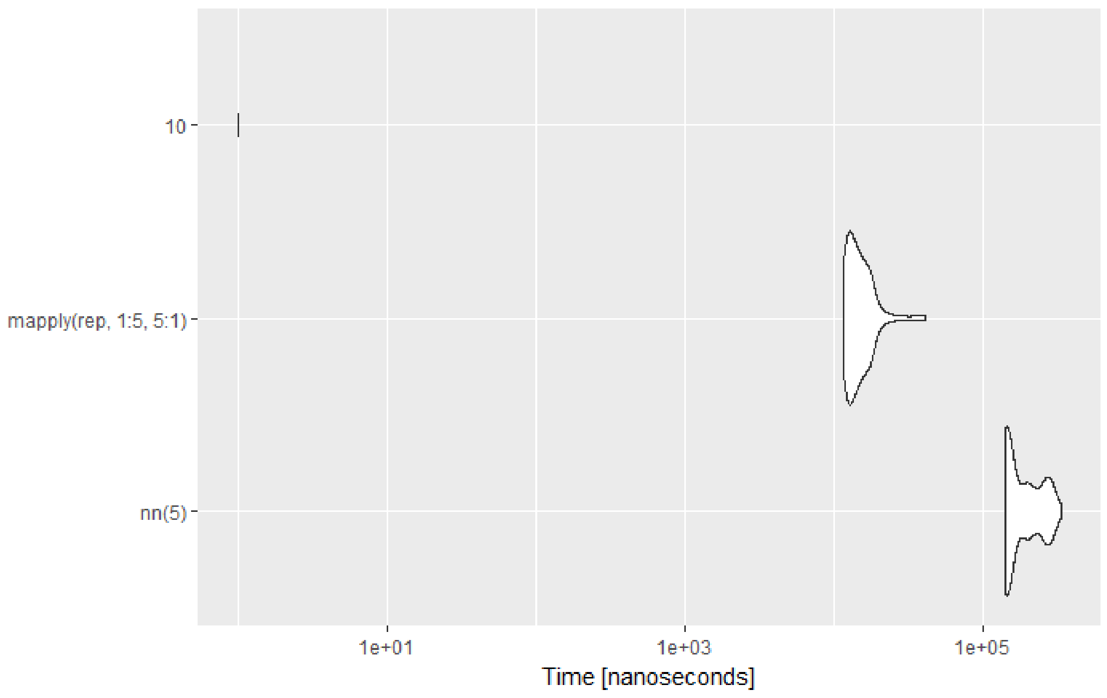

Figure 2: Autoplot graph in terms of nanoseconds

2 rapply() Function

2.1 Definition

rapply is also one of the powerful apply functions in R. r in rapply stands for recursive. According to the name it performs also in a recursive fashion like it applies function to all the elements in the list recursively.
For example:-

```
> x<- list(1,2,3,4)
> rapply(x, function(x){x^2}, class=c("numeric"))
[1]  1  4  9 16
> #doing sublisting
> x <- list(1,2,list(3,4,5),6,list(7,list(8,9)))
> rapply(x, function(x){x^2}, class=c("numeric"))
[1]  1  4  9 16 25 36 49 64 81
```

2.2 Syntax

```
rapply(object, f, classes = ANY, deflt = NULL, how = c(unlist, replace, list),)
```

2.3 Code Snippet for rapply()

```
> x <- list(1,2,list(3,4,5),6,list(7,list(8,9)))
> rapply(x, function(x){x^2}, class=c("numeric"))
[1]  1  4  9 16 25 36 49 64 81
```

2.4 Equivalent ”for” loop

```
> qq <- function(list){
+ list1 <- list(2,3)
+ list2 <- list(4,6)
+ list3 <- list(8,7)
+ list4 <- list(9,10)
+ list <- list(list1,list2,list3,list4)
+ for(i in list){
+ for(x in i){
+ www <- x^2
+ print(www)
+ }
+ }
+ }
> qq(list)
[1] 4
[1] 9
[1] 16
[1] 36
[1] 64
[1] 49
[1] 81
[1] 100
```

2.5 Performance Comparison

Comparing the rapply function and the for loop for the same task of printing squares values of the elements in the list.

```
> microbenchmark::microbenchmark(qq(list),rapply(x, function(x){x^2},
class=c("numeric")),5)
```

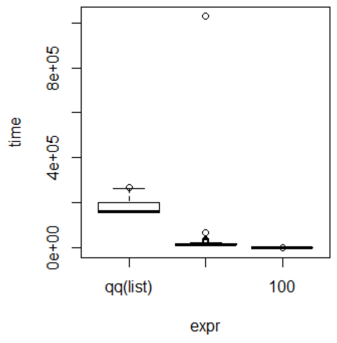

Figure 3: Plotted graphs to show that rapply works better in terms of time

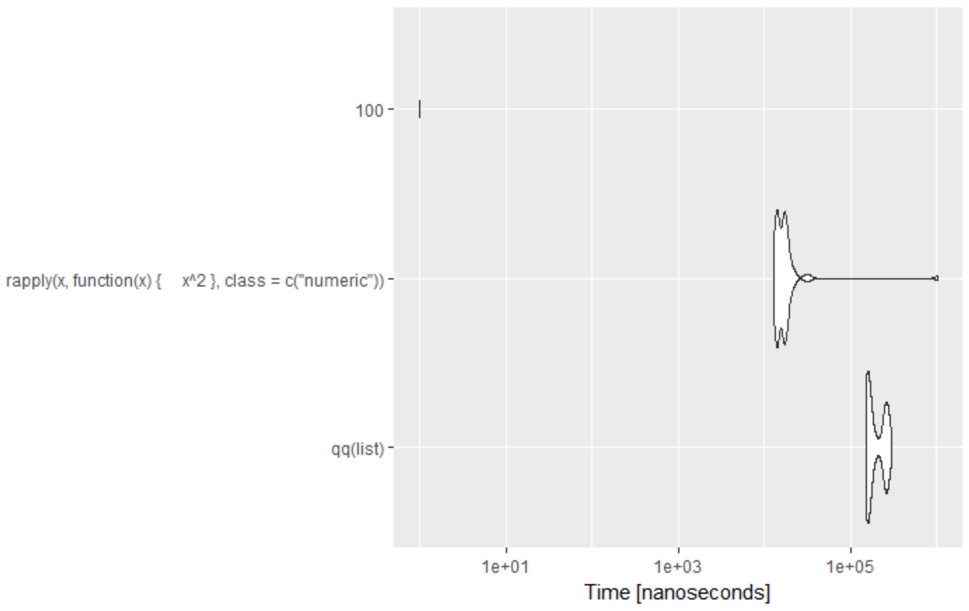

Figure 4: Autoplot graph showing running time(nanoseconds)

3 tapply() Function

3.1 Definition

Most of the time when we deal with statistics, the most common operations that we perform is to calculate the summary of the variables which means calculating the mean, average etc. This is the place where tapply comes in handy. It is very useful when we have to break the data into groups through some factors. For example the case which I will be showing in the further sections is about data frames which consist some cricket teams and those teams are consists of different players. And we want to calculate the max value or mean value of a particular team or all the team. We can do that through tapply.

3.2 Syntax

```
tapply(summary_variable, group_variable, FUN)
```

3.3 Code Snippet for tapply()

```
> cricket<- data.frame(team= gl(5,5,labels= paste("Team",
 LETTERS[1:5])),player = sample(letters, 25), batting_avg = runif(25.45, 50, 100))

> tt<- tapply(cricket$batting_avg, cricket$team, mean)
> tt
  Team A   Team B   Team C   Team D   Team E
74.31707 67.29457 73.37558 81.73795 88.35800
```

3.4 Equivalent ”for” loop

```
> y <- function(x){
```

```
+ for(i in unique(cricket$team)){
+ ttt <- mean(cricket[which(cricket$team == i), "batting_avg"])
+ print(c(ttt,i))
+ }
+ }
> y(x)
[1] "74.317069221288" "Team A"
[1] "67.2945735882968" "Team B"
[1] "73.3755815424956" "Team C"
[1] "81.7379524279386" "Team D"
[1] "88.3579969662242" "Team E"
```

3.5 Performance Comparison

```
> microo <- microbenchmark::microbenchmark(y(x),tapply(cricket$batting_avg,
cricket$team, mean),10)
> plot(microo)
> autoplot(microo)
```

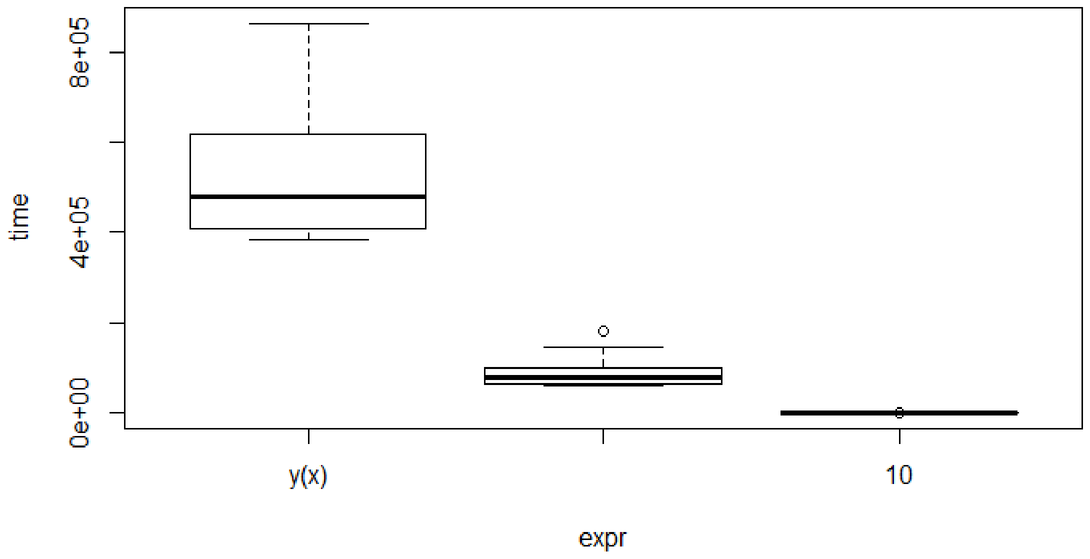

Figure 5: Graph to show that tapply works better than ”for” loop in terms of time

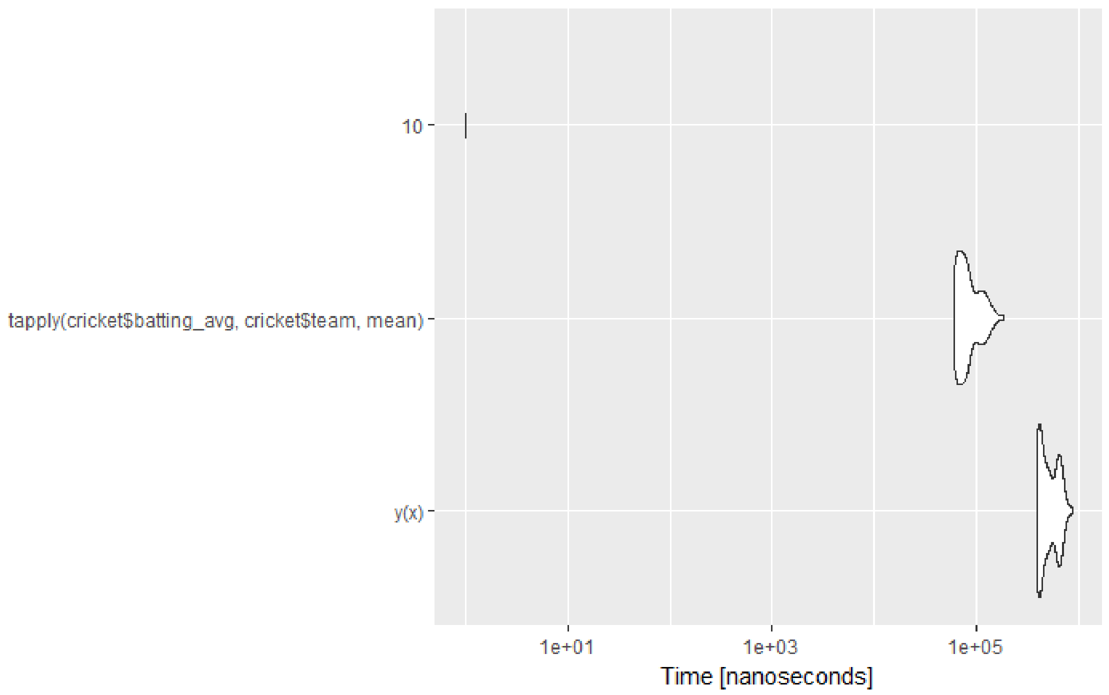

Figure 6: Autoplot graph to show running time(nanoseconds)

4 GitHub Links

- The GitHub Link : https://github.com/Abdullahejaz/R/blob/master/mapply.R
- The GitHub Link: https://github.com/Abdullahejaz/R/blob/master/rapply.R
- The GitHub Link: https://github.com/Abdullahejaz/R/blob/master/tapply.R