

채널 G

2016125077 최재혁

2016126049 박희재

데이터 사이언스

- myapply, mylapply, mymapply 함수 구현



목 차

- 과제 설명

R강의 영상을 보고 myapply, mylapply, mymapply 함수를 작성.

- 과제 목표

myapply, mylapply, mymapply 함수를 작성하며, 각 apply 함수들에 대한 이해를 높인다.

- 사용 Tool

분석용 언어 : R

목 차

-
- 과제1 - myapply 작성

-
- 과제2 - mylapply 작성

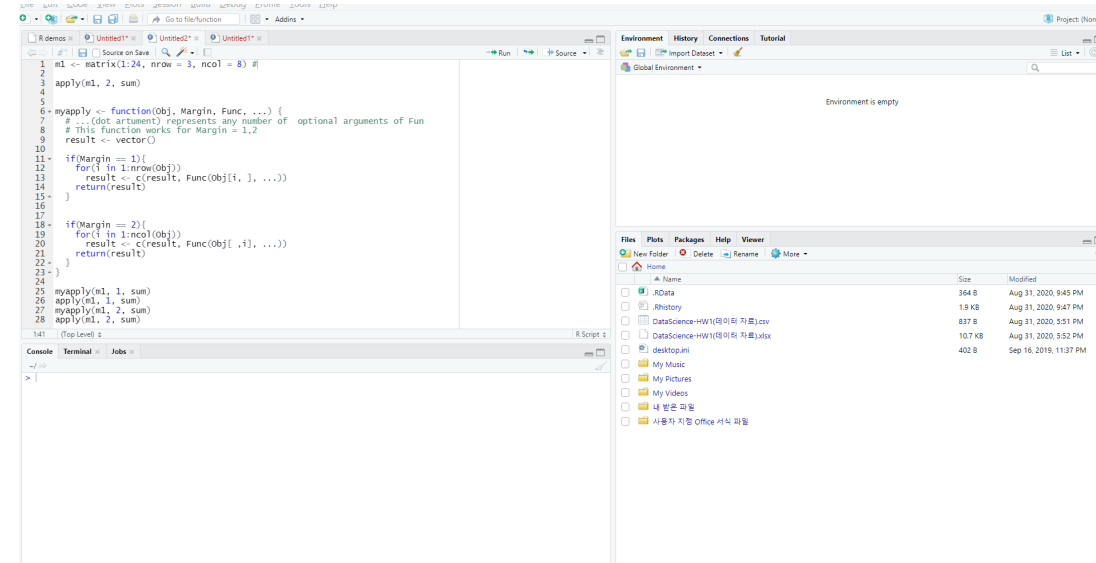
-
- 과제3 - mymapply 작성
-

과제1 - myapply

- 기존 apply 함수 설명

- apply(Object, Margin, Function) 코드 형태

- Object를 Matrix 즉 행렬로 받으며, Margin 값 (1 : 행, 2 : 열)에 따라 Function을 수행하여 벡터로 반환한다.



<사진 1>

- myapply - Rstudio

- <사진 1> 참조

과제1 - myapply 작성

- 코드

```
1 m1 <- matrix(1:24, nrow = 3, ncol = 8) #
2
3 apply(m1, 2, sum)
4
5
6 myapply <- function(Obj, Margin, Func, ...) {
7   # ...(dot argument) represents any number of optional arguments of Fun
8   # This function works for Margin = 1,2
9   result <- vector()
10
11   if(Margin == 1){
12     for(i in 1:nrow(Obj))
13       result <- c(result, Func(Obj[i, ], ...))
14     return(result)
15   }
16
17
18   if(Margin == 2){
19     for(i in 1:ncol(Obj))
20       result <- c(result, Func(Obj[, i], ...))
21     return(result)
22   }
23 }
24
25 myapply(m1, 1, sum)
26 apply(m1, 1, sum)
27 myapply(m1, 2, sum)
28 apply(m1, 2, sum)
```

- If문을 통해 Margin 값이 1이면, 같은 행 끼리 function에 함수대로 계산하여 벡터로 만들어 result로 출력하고
- Margin 값이 2이면, 같은 열끼리 function에 함수대로 계산하여 벡터로 만들어 result로 출력한다
- 밑에 4줄은 기존 apply와 작성한 myapply를 비교하기 위해 추가했다.

과제1 - myapply 작성

- 코드 실행

```
Console Terminal x Jobs x
~/
+ result <- c(result, Func(Obj[, j], ...))
+ return(result)
+ }
+
+ if(Margin == 2){
+   for(i in 1:ncol(Obj)){
+     result <- c(result, Func(Obj[, i], ...))
+   }
+ }
+ }
> myapply(m1, 1, sum)
[1] 92 100 108
> apply(m1, 1, sum)
[1] 92 100 108
> myapply(m1, 2, sum)
[1] 6 15 24 33 42 51 60 69
> apply(m1, 2, sum)
[1] 6 15 24 33 42 51 60 69
> |
```

- Margin 값이 1일 때, apply와 myapply가 출력한 벡터가 같은 행들의 원소들을 sum하여 같은 값으로 출력된 것을 알 수 있다.
- Margin 값이 2일 때, apply와 myapply가 출력한 벡터가 같은 열들의 원소들을 sum하여 같은 값으로 출력된 것을 알 수 있다.
- myapply가 apply와 같이 잘 작동함을 알 수 있다.

과제2 - lapply 함수

- 기존 lapply 함수 설명

정의 : Apply a function over a list or vector

각각의 elements 들에 Function을 적용하여 list형태로 반환하는 함수이다.

형태 : lapply (input : list or vector, output : list)

입력

출력

```
Console Terminal Jobs
C:/Users/user/Desktop/3학년2학기/데이터사이언스/과제/2주/

> m1 <- matrix(1:24, nrow = 3, ncol = 8)
> m2 <- c(1,2,3,4,5,6,10)
> result <- lapply(m1, sum)
> result2 <- lapply(m2, sum)
> result
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 4

[[5]]
[1] 5

.

.

[1] 20
[[21]]
[1] 21

[[22]]
[1] 22

[[23]]
[1] 23

[[24]]
[1] 24
> result2
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 4

[[5]]
[1] 5

[[6]]
[1] 6

[[7]]
[1] 10

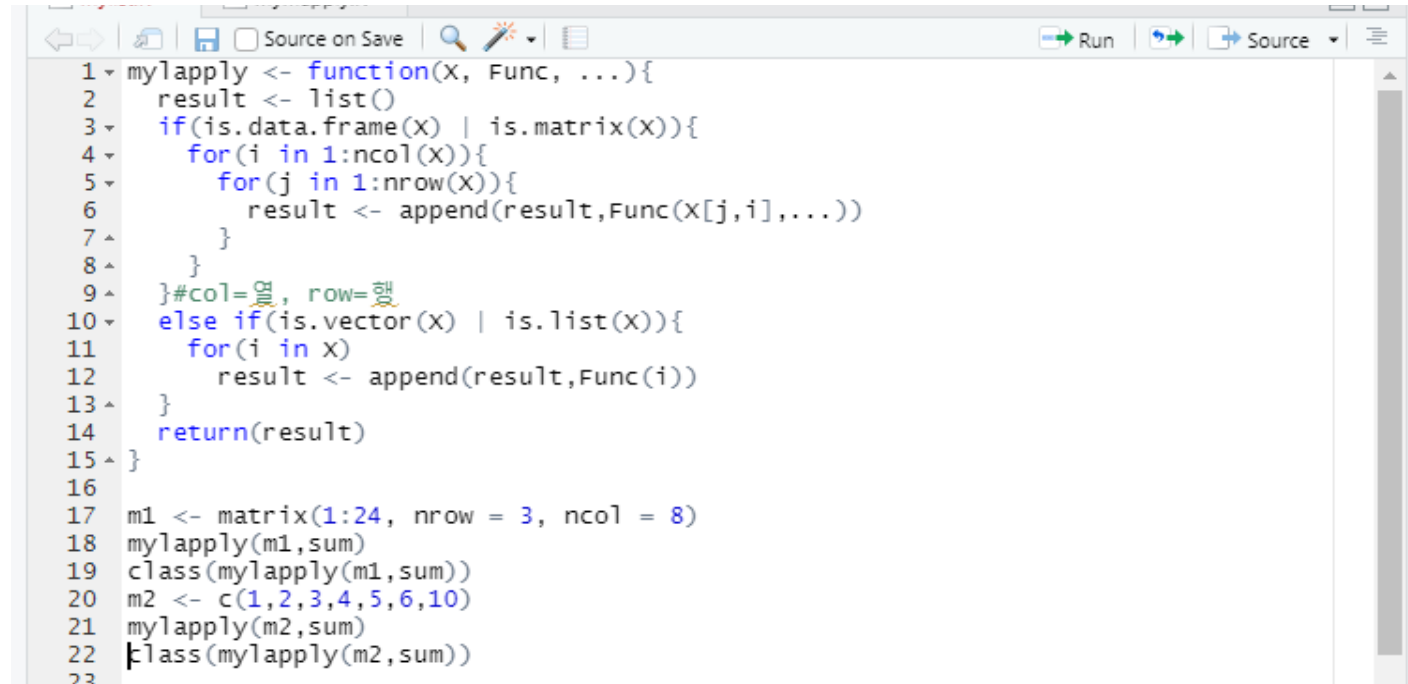
~
```

과제2 - mylapply 작성

- 코드 설명

우선 lapply함수에 들어가는 argument로는 list, vector, matrix, data frame이 있는데 이를 구분 짓기 위해서 예외처리를 해주었습니다.

이전 슬라이드에서도 보았듯이 각각의 element들에게 Func를 적용하는 것을 확인할 수 있습니다.
따라서 각 원소마다 Func을 적용하여 list() 객체에 append 시켰습니다.



```
1 mylapply <- function(X, Func, ...){
2   result <- list()
3   if(is.data.frame(X) | is.matrix(X)){
4     for(i in 1:ncol(X)){
5       for(j in 1:nrow(X)){
6         result <- append(result,Func(X[j,i],...))
7       }
8     }
9     }#col=열, row=행
10  else if(is.vector(X) | is.list(X)){
11    for(i in X)
12      result <- append(result,Func(i))
13  }
14  return(result)
15 }
16
17 m1 <- matrix(1:24, nrow = 3, ncol = 8)
18 mylapply(m1,sum)
19 class(mylapply(m1,sum))
20 m2 <- c(1,2,3,4,5,6,10)
21 mylapply(m2,sum)
22 class(mylapply(m2,sum))
23 }
```


과제2 - mylapply 출력

- 동일한 결과

```
Console Terminal Jobs
C:/Users/user/Desktop/3학년2학기/데이터사이언스/과제/2주/
> mylapply <- function(X, Func, ...){
+   result <- list()
+   if(is.data.frame(X) | is.matrix(X)){
+     for(i in 1:ncol(X)){
+       for(j in 1:nrow(X)){
+         result <- append(result,Func(X[j,i],...))
+       }
+     }
+   }#col=열, row=행
+   else if(is.vector(X) | is.list(X)){
+     for(i in X)
+       result <- append(result,Func(i))
+   }
+   return(result)
+ }
> m1 <- matrix(1:24, nrow = 3, ncol = 8)
> mylapply(m1,sum)
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 4

[[5]]
[1] 5

[[6]]
[1] 6
```

```
Console Terminal Jobs
C:/Users/user/Desktop/3학년2학기/데이터사이언스/과제/2주/
[[1]]
[1] 22

[[23]]
[1] 23

[[24]]
[1] 24

> class(mylapply(m1,sum))
[1] "list"
> m2 <- c(1,2,3,4,5,6,10)
> mylapply(m2,sum)
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 4

[[5]]
[1] 5

[[6]]
[1] 6

[[7]]
[1] 10

~
```

* 결과 타입을 보여 드리기 위해서 class()함수를 이용했습니다.

과제3 - mapply 함수

- 기존 mapply 함수 설명

정의 : Apply a function to multiple list or vector arguments

return 값은 vector 형태이며, sapply의 multi 버전으로 볼 수 있습니다. R 문서에 정의되어 있는 대로 list와 vector 여러 개를 argument로 받아 각각의 element들을 합쳐서 Func을 적용합니다.

형태 : mapply (input : list or vector, output : vector or array)

입력

```
24  
25 a <- c(1:5)  
26 b <- c(10:14)  
27 d <- c(1:5)  
28 result <- mapply(sum,a,b,d)  
29 result  
30
```

출력

```
Console Terminal x Jobs x  
C:/Users/user/Desktop/3학년2학기/데이터사이언스/과제/2주/ ↗  
> a <- c(1:5)  
> b <- c(10:14)  
> d <- c(1:5)  
> result <- mapply(sum,a,b,d)  
> result  
[1] 12 15 18 21 24  
> |
```

과제3 - mymapply 작성

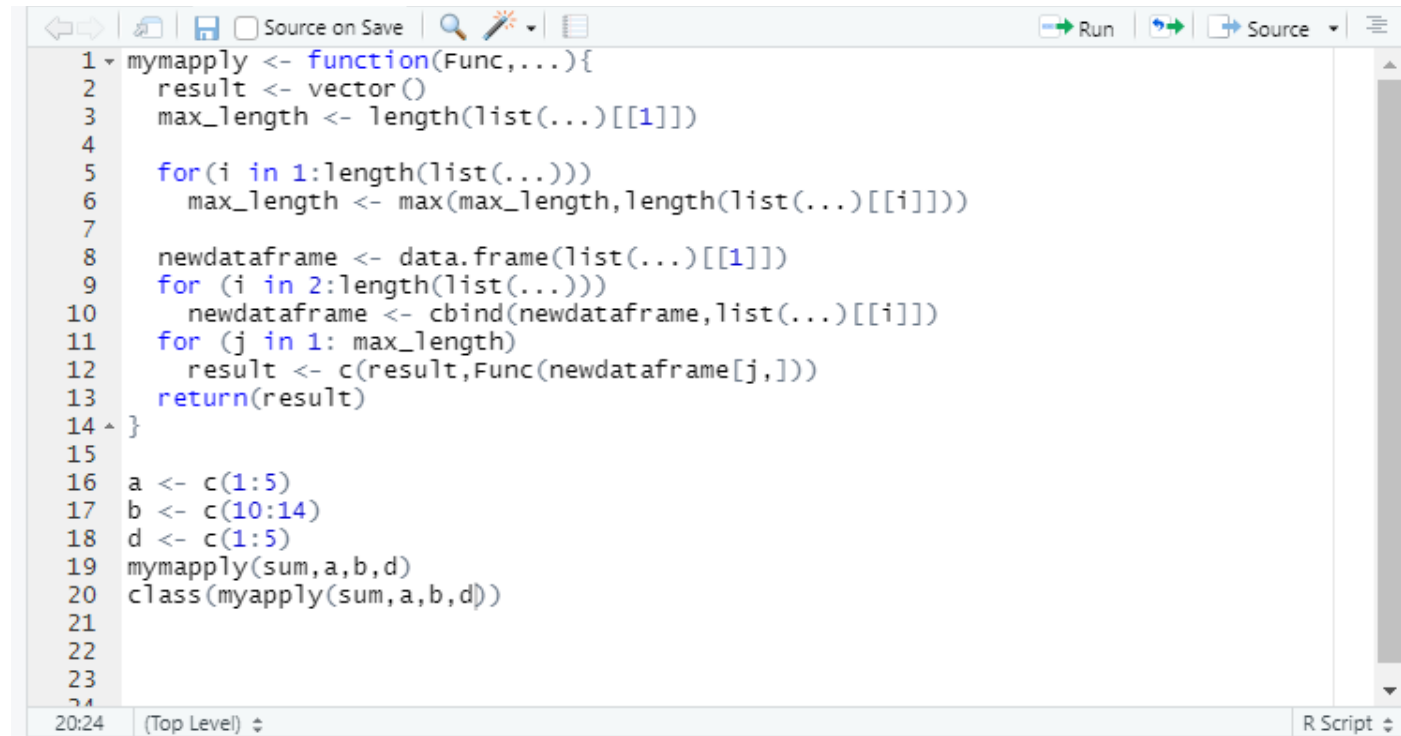
- 코드 설명

우선 정의되어 있는 대로 list와 vector여러 개를 argument로 받기 위해서, ...을 사용하였습니다. 이는 몇 개가 되었든 상관 없이 읽을 수 있는 장점이 있습니다.

이후 vector객체에 원소를 하나씩 추출하여 Func()을 적용시킵니다. 이때 max_length는 recycle하면서 계산하는 긴 자료에 맞춰 주기 위해서 정해 주었습니다.

...을 접근하는 방법은 list(...)[[index]]이런 문법으로 접근하면 원소에 접근할 수 있었습니다.

마지막으로 data frame형태로 변환 후, 열 결합하여 연산을 마무리한 코드입니다.



```
1 mymapply <- function(Func,...){
2   result <- vector()
3   max_length <- length(list(...)[[1]])
4
5   for(i in 1:length(list(...)))
6     max_length <- max(max_length,length(list(...)[[i]]))
7
8   newdataframe <- data.frame(list(...)[[1]])
9   for (i in 2:length(list(...)))
10     newdataframe <- cbind(newdataframe,list(...)[[i]])
11   for (j in 1: max_length)
12     result <- c(result,Func(newdataframe[j,]))
13   return(result)
14 }
15
16 a <- c(1:5)
17 b <- c(10:14)
18 d <- c(1:5)
19 mymapply(sum,a,b,d)
20 class(mymapply(sum,a,b,d))
21
22
23
24
```

과제3 – mymapply 출력

- 동일한 결과

```
15
16 a <- c(1:5) |
17 b <- c(10:14)
18 d <- c(1:5)
19 mymapply(sum,a,b,d)
20 class(mymapply(sum,a,b,d))
21
22
23
```



The screenshot shows an R console window with the following content:

```
C:/Users/user/Desktop/3학년2학기/데이터사이언스/과제/2주/
> mymapply <- function(Func,...){
+   result <- vector()
+   max_length <- length(list(...)[[1]])
+
+   for(i in 1:length(list(...)))
+     max_length <- max(max_length,length(list(...)[[i]]))
+
+   newdataframe <- data.frame(list(...)[[1]])
+   for (i in 2:length(list(...)))
+     newdataframe <- cbind(newdataframe,list(...)[[i]])
+   for (j in 1: max_length)
+     result <- c(result,Func(newdataframe[j,]))
+   return(result)
+ }
> a <- c(1:5)
> b <- c(10:14)
> d <- c(1:5)
> mymapply(sum,a,b,d)
[1] 12 15 18 21 24
> class(mymapply(sum,a,b,d))
[1] "integer"
>
```

* 결과 타입을 보여 드리기 위해서 class()함수를 이용했습니다.