

제4강

조건문,반복문,함수

Section 03

apply() 함수

3. apply() 함수

1. apply() 함수의 개념

- 반복 작업이 필요한 경우에는 반복문을 적용
- 반복 작업의 대상이 매트릭스나 데이터프레임의 행(row) 또는 열(column)인 경우는 **for문**이나 **while문** 대신에 **apply()** 함수를 이용하는 것이 속도가 좋다.
- **apply()** 함수의 문법

apply(데이터셋, 행/ 열방향 지정, 적용 함수)

데이터셋 : 반복 작업을 적용할 대상 매트릭스나 데이터 프레임 입력

행/열방향 : **행방향 1**을 입력, **열방향 2**를 입력

적용함수 : 반복작업의 내용을 알려주는 것으로, R함수나 다음 절에서 배울 사용자 정의 함수를 지정한다.

3. apply() 함수


2. apply() 함수의 적용

코드 4-15

```
apply(iris[,1:4], 1, mean)    # row 방향으로 함수 적용  
apply(iris[,1:4], 2, mean)    # col 방향으로 함수 적용
```

```
> apply(iris[,1:4], 1, mean)    # row 방향으로 함수 적용  
[1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225  
[10] 2.400 2.700 2.500 2.325 2.125 2.800 3.000 2.750 2.575  
[19] 2.875 2.675 2.675 2.675 2.350 2.650 2.575 2.450 2.600  
[28] 2.600 2.550 2.425 2.425 2.675 2.725 2.825 2.425 2.400  
[37] 2.625 2.500 2.225 2.550 2.525 2.100 2.275 2.675 2.800  
...(중간 생략)  
[136] 4.775 4.425 4.200 3.900 4.375 4.450 4.350 3.875 4.550  
[145] 4.550 4.300 3.925 4.175 4.325 3.950
```

3. apply()함수



| Sepal.Length | Sepal.Width | Petala.Length | Petal.Width | |
|--------------|-------------|---------------|-------------|--------|
| 5.1 | 3.5 | 1.4 | 0.2 | mean() |
| 4.9 | 3.0 | 1.4 | 0.2 | mean() |
| 4.7 | 3.2 | 1.3 | 0.2 | |
| 4.6 | 3.1 | 1.5 | 0.2 | |
| 5.0 | 3.6 | 1.4 | 0.2 | |
| 5.4 | 3.9 | 1.7 | 0.4 | |
| 4.6 | 3.4 | 1.4 | 0.3 | |
| 5.0 | 3.4 | 1.5 | 0.2 | |
| 4.4 | 2.9 | 1.4 | 0.2 | |
| 4.9 | 3.1 | 1.5 | 0.1 | |
| 5.4 | 3.7 | 1.5 | 0.2 | |
| 4.8 | 3.0 | 1.4 | 0.1 | |
| 4.8 | 3.4 | 1.6 | 0.1 | |
| 4.3 | 3.0 | 1.1 | 0.1 | |
| 5.8 | 4.0 | 1.2 | 0.2 | mean() |

그림 4-2 `apply(iris[,1:4], 1,
mean)`

3. apply()함수

```
> apply(iris[,1:4], 2, mean)           # col 방향으로 함수 적용  
Sepal.Length Sepal.Width Petal.Length Petal.Width  
5.843333     3.057333     3.758000     1.199333
```

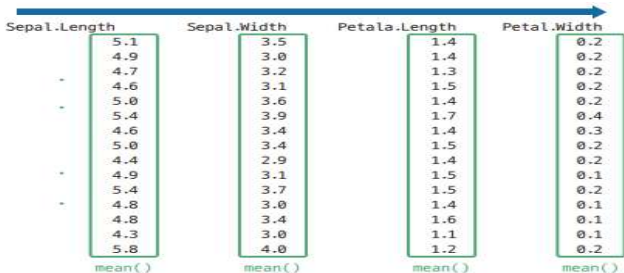


그림 4-3 `apply(iris[,1:4], 2,`

Section 04

사용자 정의 함수

4. 사용자 정의 함수

1. 사용자 정의 함수 만들기

- R은 사용자들도 자신만의 함수를 만들어 사용할 수 있는 기능을 제공하는데, 이를 사용자 정의 함수라고 함
- 사용자 정의 함수 문법

```
함수명 <- function(매개변수 목록) {  
  실행할 명령문(들)  
  return(함수의 실행 결과)  
}
```

4.1 사용자 정의 함수를 만들고 사용하기

코드 4-16

```
mymax <- function(x, y) {  
  num.max <- x  
  if (y > x) {  
    num.max <- y  
  }  
  return(num.max)  
}
```


4. 사용자 정의 함수

4.1 사용자 정의 함수를 만들고 사용하기

코드 4-17

```
mymax(10,15)  
a <- mymax(20,15)  
b <- mymax(31,45)  
print(a+b)
```

```
> mymax(10,15)  
[1] 15  
> a <- mymax(20,15)  
> b <- mymax(31,45)  
> print(a+b)  
[1] 65
```

4. 사용자 정의 함수

4.2 사용자 정의 함수의 매개변수에 초기값 설정하기

코드 4-18

함수를 정의하는데, 매개변수 y는 2로 초기값이 설정되어 있다.

```
mydiv <- function(x, y=2) {  
  result <- x/y  
  return(result)  
}
```

```
mydiv(x=10, y=3) # 매개변수 이름과 매개변수값을 쌍으로 입력  
mydiv(10, 3)     # 매개변수값만 입력  
mydiv(10)        # x에 대한 값만 입력(y 값이 생략됨)
```

4. 사용자 정의 함수

```
> mydiv <- function(x,y=2) {  
+   result <- x/y  
+   return(result)  
+ }
```

```
>
```

```
> mydiv(x=10,y=3)
```

```
[1] 3.333333
```

```
> mydiv(10,3)
```

```
[1] 3.333333
```

```
> mydiv(10)
```

```
[1] 5
```

매개변수 이름과 매개변수값을 쌍으로 입력

매개변수값만 입력

x에 대한 값만 입력(y 값이 생략됨)

4. 사용자 정의 함수

4.3 함수가 반환하는 결과값이 여러 개일 때의 처리

코드 4-19

```
myfunc <- function(x, y) {  
  val.sum <- x+y  
  val.mul <- x*y  
  return(list(sum=val.sum, mul=val.mul))  
}
```

```
result <- myfunc(5, 8) # 여기서 result는 list데이터 타입이다.(리턴 값이 리스트니깐...)  
s <- result$sum        # 5, 8의 합  
m <- result$mul        # 5, 8의 곱  
cat('5+8=', s, '\n')  
cat('5*8=', m, '\n')
```

4. 사용자 정의 함수

```
> myfunc <- function(x,y) {  
+   val.sum <- x+y  
+   val.mul <- x*y  
+   return(list(sum=val.sum, mul=val.mul))  
+ }  
>  
> result <- myfunc(5,8)  
> s <- result$sum           # 5, 8의 합  
> m <- result$mul           # 5, 8의 곱  
> cat('5+8=', s, '\n')  
5+8= 13  
> cat('5*8=', m, '\n')  
5*8= 40
```

4. 사용자 정의 함수

2. 사용자 정의 함수의 저장 및 호출

코드 4-20

```
setwd("d:/source")      # myfunc.R이 저장된 폴더
source("myfunc.R")       # myfunc.R 안에 있는 함수 실행

# 함수 사용
a <- mydiv(20,4)         # 함수 호출
b <- mydiv(30,4)         # 함수 호출
a+b
mydiv(mydiv(20,2),5)     # 함수 호출
```

사용자 정의 함수의 사용절차

1. 함수를 작성(정의)
2. 함수를 실행하여 R에 함수를 등록함
3. 필요한 곳에서 함수를 호출함
4. 결과값을 받아서 코드 처리함.

4. 사용자 정의 함수

```
> setwd("d:/source")      # myfunc.R이 저장된 폴더
> source("myfunc.R")      # myfunc.R 안에 있는 함수 실행

# 함수 사용
> a <- mydiv(20,4)         # 함수 호출
> b <- mydiv(30,4)         # 함수 호출
> a+b
[1] 12.5
> mydiv(mydiv(20,2),5)     # 함수 호출
[1] 2
```

Section 05

조건에 맞는 데이터의 위치 찾기

5. 조건에 맞는 데이터의 위치 찾기

5. 조건에 맞는 데이터 위치 찾기

데이터 분석을 하다 보면 자신이 원하는 데이터가 벡터나 매트릭스, 데이터 프레임 안에서 어디에 위치하고 있는지를 알기 원하는 때가 있음

예를 들어, 50명의 학생 성적이 저장된 벡터가 있는데 가장 성적이 좋은 학생은 몇 번째에 있는지를 알고 싶은 경우

이런 경우 편리하게 사용할 수 있는 함수가 **which()**, **which.max()**, **which.min()** 함

코드 4-21

which()함수는 조건에 해당하는 인덱스를 리턴하는데, 한 개거나 여러 개를 리턴할수 있음을 주목하자.(중요)

```
score <- c(76, 84, 69, 50, 95, 60, 82, 71, 88, 84)
which(score==69)      # 성적이 69인 학생은 몇 번째에 있나
which(score>=85)      # 성적이 85 이상인 학생은 몇 번째에 있나
max(score)            # 최고 점수는 몇 점인가
which.max(score)       # 최고 점수는 몇 번째에 있나
min(score)            # 최저 점수는 몇 점인가
which.min(score)       # 최저 점수는 몇 번째에 있나
```

5. 조건에 맞는 데이터의 위치 찾기 사용자 정의 함수

```
> score <- c(76, 84, 69, 50, 95, 60, 82, 71, 88, 84)
> which(score==69)           # 성적이 69인 학생은 몇 번째에 있나
[1] 3
> which(score>=85)          # 성적이 85 이상인 학생은 몇 번째에 있나
[1] 5 9
> max(score)                # 최고 점수는 몇 점인가
[1] 95
> which.max(score)          # 최고 점수는 몇 번째에 있나
[1] 5
> min(score)                # 최저 점수는 몇 점인가
[1] 50
> which.min(score)          # 최저 점수는 몇 번째에 있나
[1] 5
```

5. 조건에 맞는 데이터의 위치 찾기 사용자 정의 함수

코드 4-22

```
score <- c(76, 84, 69, 50, 95, 60, 82, 71, 88, 84)
idx <- which(score<=60)      # 성적이 60 이하인 값들의 인덱스
score[idx] <- 61             # 성적이 60 이하인 값들은 61점으로 성적 상향 조정
score                       # 상향 조정된 성적 확인

idx <- which(score>=80)      # 성적이 80 이상인 값들의 인덱스
score.high <- score[idx]     # 성적이 80 이상인 값들만 추출하여 저장
score.high                  # score.high의 내용 확인

> score <- c(76, 84, 69, 50, 95, 60, 82, 71, 88, 84)
> idx <- which(score<=60)    # 성적이 60 이하인 값들의 인덱스
> score[idx] <- 61          # 성적이 60 이하인 값들은 61점으로 성적 상향 조정
> score                     # 상향 조정된 성적 확인
[1] 76 84 69 61 95 61 82 71 88 84
>
> idx <- which(score>=80)    # 성적이 80 이상인 값들의 인덱스
> score.high <- score[idx]  # 성적이 80 이상인 값들만 추출하여 저장
> score.high                # score.high의 내용 확인
[1] 84 95 82 88 84
```

5. 조건에 맞는 데이터의 위치 찾기 사용자 정의 함수

코드 4-23

```
idx <- which(iris$Petal.Length>5.0) # 꽃잎의 길이가 5.0 이상인 값들의 인덱스
idx
iris.big <- iris[idx,]              # 인덱스에 해당하는 값만 추출하여 저장
iris.big
```

5. 조건에 맞는 데이터의 위치 찾기 사용자 정의 함수

```
> idx <- which(iris$Petal.Length>5.0)      # 꽃잎의 길이가 5.0 이상인 값들의 인덱스
> idx
[1] 84 101 102 103 104 105 106 108 109 110 111 112 113 115 116 117 118
[18] 119 121 123 125 126 129 130 131 132 133 134 135 136 137 138 140 141
[35] 142 143 144 145 146 148 149 150
> iris.big <- iris[idx,]                  # 인덱스에 해당하는 값만 추출하여 저장
> iris.big
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|------------|--------------|-------------|--------------|-------------|------------|
| 84 | 6.0 | 2.7 | 5.1 | 1.6 | versicolor |
| 101 | 6.3 | 3.3 | 6.0 | 2.5 | virginica |
| 102 | 5.8 | 2.7 | 5.1 | 1.9 | virginica |
| ...(중간 생략) | | | | | |
| 148 | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| 149 | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| 150 | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

5. 조건에 맞는 데이터의 위치 찾기 사용자 정의 함수

코드 4-24

```
# 1~4열의 값 중 5보다 큰 값의 행과 열의 위치  
idx <- which(iris[,1:4]>5.0, arr.ind =TRUE)  
idx
```

**which()함수를 이용하여 매트릭스나 데이터프레임 안에 있는 특정 조건의 값의
행과 열의 위치를 알고 싶다면, 매개변수로 arr.ind속성을 TRUE로 주면된다.**

```
> idx <- which(iris[,1:4]>5.0, arr.ind =TRUE )  
> idx
```

```
      row col  
[1,]   1   1  
[2,]   6   1  
[3,]  11   1  
[4,]  15   1  
[5,]  16   1  
[6,]  17   1  
[7,]  18   1  
...(중간 생략)  
[155,] 144   3  
[156,] 145   3  
[157,] 146   3  
[158,] 148   3  
[159,] 149   3  
[160,] 150   3
```

감사합니다.