

# 분석프로그래밍 I.08

## 제어와 함수 1

국민대학교 경영학부 빅데이터경영통계학 전공

2018. 4. 30(월)

### 1 시험 문제 풀이, 지난 시간 복습 (데이터 시각화)

### 2 Scripts

- *R Studio - File - New File - R Script*
- 순차적으로 실행하게 될 함수들을 모아 놓은 텍스트 (재연 가능한 분석)
  - 스크립트 실행하기 : `source()`<sup>1</sup>
  - 재연 가능한 연구를 위한 환경 설정

*R Studio - Options - General - Restore .Rdata into workspace at startup* [ ]<sup>2</sup>

```
rm(list=ls())
```

### 3 제어문 : 조건 또는 반복

- 일련의 함수들 중 일부를 **조건에 따라**, 혹은 **반복적으로** 실행해야 할 경우가 있다.
- 조건문 : `if/else`, `ifelse`, `switch`

---

<sup>1</sup>현재의 환경 (변수, 함수)를 재구성하는 방법은 R 스크립트를 다시 실행하는 방법과 현재의 변수, 함수를 스크립트 (`dump()`), 또는 데이터 파일 (`save.image()`)로 저장하는 방법이 있다. 다시 불러들일 경우 `source()`, `load()`를 사용한다.

<sup>2</sup>*Options - General - Save workspace to .RData on exit [Never]* 설정을 사용할 수도 있다.

```
1 if (s == "here") {  
2   print("Kookmin Univ.")  
3 } else {  
4   print("Somewhere else than Kookmin Univ.")  
5 }
```

Listing 1: if/else 구문

```
1 s2 <- ifelse(s=="here",  
2             "Kookmin Univ.", # if s=="here"  
3             "Somewhere else than Kookmin Univ") # if s!="here"
```

Listing 2: ifelse 함수

```
1 x <- "two"  
2 switch(x,  
3   one = 1,  
4   two = 2,  
5   3)  
6  
7 x <- "two"; z <- 3  
8 switch(x,  
9   one=1,  
10  two={  
11    y <- paste(x,"is entered.")  
12    z + 1},  
13  three=z^2)
```

Listing 3: switch 함수

- 반복문 : for, while, repeat / next, break

```
1 s=0  
2 for (i in 1:10)  
3   s = s + i  
4 print(s)
```

Listing 4: for 구문

```
1 s=0; i=1
2 while (i <= 10) {
3   s=s+i;
4
5   i=i+1
6 }
7 print(s)
```

Listing 5: while 구문

```
1 s=0; i=1
2 repeat {
3   s=s+i; i=i+1
4   if (i>10) break
5 }
6 print(s)
```

Listing 6: repeat 구문

- 연습문제

- 길이 1의 numeric 벡터를 받아 1이면 "one", 2이면 "two", 그 밖에는 "other than one or two"를 출력하세요. 이때 if를 활용한 방법과 switch를 활용한 방법을 모두 구현하세요.
- 1부터 100까지 모든 짝수를 합하는 스크립트를 for (i in 1:100), while, repeat를 활용하여 작성하세요.
- 1부터 시작하여 '3, 7, 23의 배수이며 2의 약수가 아닌 수'까지 모든 정수를 더하는 스크립트를 for (i in 1:1000), while, repeat를 활용하여 작성하세요.

## 4 R 반복문 다시 보기

### 4.1 반복문의 기본적인 구조

1. 결과를 저장할 장소(변수)를 마련한다.

2. 특정한 값을 따라, 혹은 주어진 조건이 만족할 때까지 반복하도록 한다.

3. 반복되는 내용

```
1 s=0                # 결과를 저장할 변수 s
2 for (i in 1:10)    # 변수 가i 부터1 까지10 반복된다.
3   s = s + i        # 반복되어 수행되는 작업
4 print(s)
```

Listing 7: 반복 구문의 일반적인 구조

```
1 x = c(1, 3, 5, 9, 15)
2 s = rep(NA, 5)      # 결과를 저장할 변수 s
3 #for (i in 1:5)      # 변수 가i 벡터 의x 길이에 맞춰 변한다.
4 #for (i in 1:length(x))
5 for (i in seq_along(x))
6   s[i] = x[i]^2      # 반복되어 수행되는 작업
7 print(s)
```

Listing 8: 벡터 x의 모든 원소에 대해 제곱한 값을 구하기

## 4.2 반복문의 대체

```
1 #01. Vectorized function
2 x <- c(1, 3, 5, 9, 15)
3 s <- sqrt(x)
4
5 #02. sapply
6 x <- c(1, 3, 5, 9, 15)
7 s <- sapply(x, sqrt)
```

Listing 9: 반복문을 대체하여 벡터 x의 모든 원소에 대해 제곱근을 구하기

```
1 #01. if the function is not vectorized
2 tonum = function(x) {
3   switch(x,
4     one = 1,
```

```

5     two = 2,
6     3)
7   }
8 x <- c("one","three","two","four","two")
9 s <- tonum(x)
10 #Error in switch(x, one = 1, two = 2, 3) : EXPR must be a length 1
    vector
11
12 tonumV = Vectorize(tonum)
13 s <- tonumV(x)
14
15 #02. sapply with 'Vectorize'-d function
16 x <- c("one","three","two","four","two")
17 s <- sapply(x, tonumV)

```

Listing 10: Vectorize 함수

### 4.3 for의 몇 가지 변형

- for (x in xs), for (nm in names(xs))
- 결과의 길이가 가변적일 때

```

1 x <- c(1,3,2,4)
2 result = vector("list", length(x))
3 #result = c()
4 for (i in seq_along(x)) {
5   result[[i]] = rep(x,x)
6   #result = c(result, rep(x,x))
7 }
8 result <- unlist(result)

```

Listing 11: 결과가 가변적일 때 반복문의 예시

- 반복의 횟수가 가변적일 때 : while, repeat/break

## 5 반복문의 속도 비교

동일한 반복 작업을 수행하는 여러 가지 다른 방법의 속도를 비교해보자. 행렬의 경우, `system.time()`를 사용하여 측정한 속도의 순서는 다음과 같다.

R의 내장 함수 > `apply` > `for`

```
1 mat <- matrix(1:1000, 1000, 1000, byrow=T)
2
3 # R 내장 함수
4 result <- colSums(mat)
5
6 # apply
7 result <- apply(mat, 2, sum)
8
9 # for
10 result <- rep(NA, 1000) # result <- NA, length(result) <- 1000
11 for (i in 1:ncol(mat)) {
12   result[i] <- sum(mat[,i])
13 }
```

Listing 12: `colSums`, `apply`, `for`의 속도 비교

## 6 연습 문제

- 음이항 분포(Negative Binomial Distribution)는 베르누이 시행의 결과 Success/Fail의 한 결과가 일정 횟수가 나올 때까지 필요한 베르누이 시행의 수를 나타내는 분포이다.
- 동전을 던져서 앞면이 3번 나올 때까지 필요한 횟수를 세어보자.
- `sample(c("S","F"), 1, prob=c(.5, .5))`의 결과는 50%의 확률로 'S', 50%의 확률로 'F'이다. 이 함수를 이용하여 'S'가 5번 나올 때까지 시행을 반복하는 반복문을 작성해 보자. (이를 1000번 반복해보자.)