제3장 데이터 구조의 이해와 코딩의 시작

1. R의 데이터 구조

- 벡터(vector) : 순서화된 동일한 데이터 유형(숫자 또는 문자)의 값으로 구성
- 배열(array) : 1차원 이상의 동일한 데이터 유형의 데이터 구조.
  - 행렬(matrix) : 행과 열로 구성되는 2차원 배열
- 리스트(list) : 각 원소들이 서로 다른 데이터 유형(벡터, 배열, 또는 리스트)으로 구성
- 데이터 프레임(data frame) : 2차원 테이블 형태, 각 열은 동일한 데이터 유형

2. 벡터와 연산자

1) 벡터와 대입 연산자

-. 벡터 만들기 : `c()` 이용

```
x <- c(80, 85, 70)   # 학생 x의 성적 : 중간고사, 기말고사, 출석
x

c(80, 85, 70) -> x   # 동일한 결과를 가져옴. '->' 보다는 '<-' 사용.
x
```

- **식별자**

  -. 변수 또는 함수 등을 다른 것들과 구별하기 위해 사용하는 '이름' 등을 지칭

  -. 식별자 이름은 일반적으로 문자, 숫자, '_', 그리고 '.' 등으로 구성된다.

  -. 식별자 이름은 숫자로 시작해서 '.'으로 끝나면 안됨. (예, '5a', '_a', '3.' 등은 안됨)

  -. 예약어를 식별자로 사용하면 안됨. (예, if, else, repeat, while, function, for, in, next, break, TRUE, FALSE, NULL, inf, NaN, NA 등)

- **벡터 생성 함수와 대입 연산 함수**
  - 벡터 생성 함수 `c()` : c는 concatenate로 여러 숫자 (또는 문자)들을 하나로 연결시키는 것을 의미
  - 대입 연산 함수 `assign()` : `<-` `assign("x", c(80, 85, 70))` 는 `x <- c(80, 85, 70)` 와 같은 결과.

-. 벡터 원소가 하나일 때

```r
x <- c(80)
x

x <- 80     # 위와 같은 결과. 단일 값의 경우 c()를 생략함.
x
```

- R의 **데이터 유형(data type)**
  - R에서는 사용되는 변수에 대하여 데이터 유형을 선언하지 않고 사용한다.
  - 데이터 유형 간 변환이 가능하다.
    - 논리값(TRUE, FALSE)
    - 숫자
    - 복소수
    - 문자

2) 연산자

- 대입(할당) 연산자 : `<-`
- 산술연산자 : 사칙연산
- 비교연산자 : 값들의 크기 비교
- 논리연산자 : "참"과 "거짓" 판단

2-1. 산술연산자

기호	설명	예	결과
+	더하기	2 + 3	5
-	빼기	10 - 3	7
*	곱하기	3 * 4	12
/	나누기	8 / 2	4
^ 또는 **	거듭 제곱	2 ^ 3 (또는 2 ** 3)	8
%%	나머지	10 %% 3	1
%/%	몫	10 %/%3	3

-. 단일 값에 대한 산술연산

```r
x <- 7 + 2
x
```

```r
x <- 7 - 3
x
```

```r
x <- 7 * 3
x
```

```
x <- 7 / 3
x
```

```
x <- 7 ^ 3
x

x <- 7 ** 3
x
```

```
x <- 7 %% 3
x
```

```
x <- 7 %/% 3
x
```

-. 벡터의 산술연산

- 벡터의 정의

```
mid <- c(2, 4, 6, 8)        # 4개 원소
final <- c(1, 3, 5, 7)      # 4개 원소
att <- c(1, 2)              # 2개 원소
```

- 벡터의 각 원소에 같은 값을 더하기 (, 빼기, 곱하기, 나누기)

```
mid <- mid + 8
mid

# 또는
mid <- mid + 8 ; mid

# 또는
(mid <- mid + 8)   # (  )로 실행결과를 console에 표시
```

- 벡터와 벡터의 덧셈(, 뺄셈, 곱셈, 나눗셈)

```
(total <- mid + final)    # 새로운 total 변수 생성, mid + final 값을 가짐
```

```
(increase <- final - mid)
```

```
(prod <- mid * final)
```

```
(ratio <- final / mid)
```

- 서로 다른 데이터 유형과 연산

```
x <- c(2, 4, 6) ; x
y <- c("A", "B", "C"); y
# x + y   # 오류 발생
```

- 문자와 숫자가 혼합된 벡터의 경우, 문자 벡터로 변환됨.

```
y <- c("A", 1, 2); y   # 1, 2는 "1"과 "2" 등과 같이 문자로 변환됨
```

2-2. 비교연산자

기호	설명	예	결과
<	작다	**x <- 3** x < 10	TRUE
<=	작거나 같다 (이하)	**x <- 3** x <= 10	TRUE
>	크다	**x <- 3** x > 10	FALSE
>=	크거나 같다(이상)	**x <- 3** x >= 10	FALSE
==	같다	**x <- 3** x == 10	FALSE
!=	같지 않다	**x <- 3** x != 10	TRUE

- 숫자 간 비교

```
x <- 5 < 3
```

- 벡터와 숫자의 비교

```
y <- c(5, 10, 15); y
z <- y <= 10   # y의 각 요소를 10과 비교한 결과를 z에 TRUE 또는 FALSE로 반환
```

- 벡터와 벡터의 비교

```
x <- c(10, 13, 15); x
y <- c(5, 12, 19); y
z <- x >= y            # z <- ( x >= y )
```

2-3. 논리연산자

TRUE와 FALSE 등의 진위값을 다룰 때 사용한다.

기호	설명	예	결과
|	or (논리합)	x <- TRUE y <- FALSE x | y	TRUE
&	and (논리곱)	x <- TRUE y <- FALSE x & y	FALSE
!	not (논리부정)	x <- TRUE !x x <- 3 !x	FALSE 숫자의 경우 0이면 FALSE, 그외는 TRUE
isTRUE(x)	진위여부	x <- TRUE isTRUE(x)	TRUE

- 논리합

```
x <- c(TRUE, TRUE, FALSE, FALSE); x
y <- c(TRUE, FALSE, TRUE, FALSE); y
x | y
```

- 논리곱

```
x <- c(TRUE, TRUE, FALSE, FALSE); x
y <- c(TRUE, FALSE, TRUE, FALSE); y
x & y
```

- 논리부정

```
x <- c(TRUE, FALSE); x
!x
```

- 진위여부 파악

```
x <- c(TRUE, FALSE); x
isTRUE(x)
```

3) 벡터 생성 함수

`seq()` : 규칙적인 숫자 벡터(예, 수열 등)를 생성

`rep()` : 반복되는 숫자 벡터 생성

3-1. `seq()` 함수

- seq() 함수

```
seq(from, to, by)
seq(from, to, length.out)
- from : 시작 값
- to : 종료 값
- by : 증가 값
- length.out : 벡터 원소의 갯수
```

- seq()의 사용 예

```
x <- seq(1, 10); x
```

또는

```
x <- 1:10; x
```

```
x <- seq(10, 1)
```

또는

```
x <- 10:1 ; x
```

- 증가하는 숫자 벡터 생성

```
x <- seq(1, 10, by=3) ; x
```

```
y <- seq(1, 10, length.out = 5); y  # 1로 시작하고 10으로 끝나는 5개의 숫자
```

3-2. `rep()` 함수

- rep() 함수

```
rep(x, time|each)      # x에 있는 값들을 복사
- x : 반복할 벡터 자료
- times : 자료 x의 반복 횟수
- each : 자료 x를 구성하는 개별 원소들의 반복 횟수
```

예)

```
x <- c(1, 2, 3)
rep(x, times=2)        # x 벡터를 2번 반복
```

```
rep(x, each=2)         # x 벡터의 각 원소를 2번씩 반복
```

4) 벡터의 원소

벡터를 구성하는 특정 원소 값들을 출력하거나, 수정하는 과정

4-1. 원소 위치에 따른 출력

```
x <- c(1, 2, 3, 4, 5)
x[2]                  # x의 2번째 요소

x[c(1, 3, 5)]         # x의 1, 3, 5번째 요소

x[-c(2, 4)]           # x의 2, 4번째를 제외한 나머지 요소
```

4-2. 원소 값의 조건에 따른 출력

```
x[x>2]                # x의 원소 값이 2보다 큰 값들만 출력

x[x>=2 & x<=4]        # x의 원소 값이 2와 4 사이의 값들만 출력
```

4-3. 원소 값의 수정

```
x[2] <- 20; x         # x의 2번째 요소 값을 20으로 수정

x[c(3, 4)] <- 15; x   # x의 3번째, 4번째 요소의 값을 15로 수정

x[x<=15] <- -10; x    # x의 요소 값이 15이하이면 -10으로 수정
```

5) 함수를 이용한 벡터의 연산

5-1. 벡터에 대한 함수의 사용