R 프로그래밍 기초다지기

2강 - 벡터와 친해지기

슬기로운통계생활

Issac Lee

벡터와 친해지기

벡터 (Vector) 만들기

벡터를 만드는 2가지 방법

- 빈 벡터 선언 후 채우기
- 만들면서 채우기; `c()`함수, `:` 연산자, `seq()` 함수

빈 벡터 선언 후 채우기

```
x <- vector(length = 3)
x
```

```
## [1] FALSE FALSE FALSE
```

- 자동으로 `FALSE`를 채워줌.
- `[]`을 사용하여 원소에 접근

```
x[2] <- 3
x
```

```
## [1] 0 3 0
```

벡터를 만들면서 채우는 방법

`c()` 함수와 `:` 연산자 활용

```
c(1:5)
```

```
## [1] 1 2 3 4 5
```

```
c(1:5) * 2
```

```
## [1]  2  4  6  8 10
```

```
c(1:5) * 2 - 1
```

```
## [1] 1 3 5 7 9
```

`:` 의 연산 순서

- `:` 연산자의 연산 순서는 다른 사칙 연산보다 위에 있음

```
1:3 - 2
```

```
## [1] -1  0  1
```

```
1:(3 - 2)
```

```
## [1] 1
```

```
1:3 * 2
```

```
## [1] 2 4 6
```

```
1:(3 * 2)
```

```
## [1] 1 2 3 4 5 6
```

벡터를 만들면서 채우는 방법 2

seq() 함수 이해하기

- `seq(시작값, 끝값, 옵션)`
- `by` 옵션과 `length.out` 옵션 활용

```
seq(2, 10, by = 2)
```

```
## [1]  2  4  6  8 10
```

```
seq(2, 10, length = 3)
```

```
## [1]  2  6 10
```

같은 원소들로 채우는 방법 3

`rep()` 함수 이해하기

- 구문: `rep(반복대상, 반복횟수)`

```
rep(8, 4)
```

```
## [1] 8 8 8 8
```

```
rep(c(1, 2, 4), 2)
```

```
## [1] 1 2 4 1 2 4
```

```
rep(c(1, 2, 4), each = 2)
```

```
## [1] 1 1 2 2 4 4
```

벡터 인덱싱 (Indexing)

원하는 벡터 원소만을 선택하기

- 구문 형식: `vector1[vector2]`

```
x <- 1:10 * 2
x
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
x[3:6]
```

```
## [1]  6  8 10 12
```

벡터 인덱싱 (Indexing)

- 인덱싱 중복 가능

```
x[c(2, 2, 4, 3)]
```

```
## [1] 4 4 8 6
```

- 특정 원소 빼고 선택

```
x[-1]
```

```
## [1]  4  6  8 10 12 14 16 18 2
```

```
x[-c(1:3)]
```

```
## [1]  8 10 12 14 16 18 20
```

벡터에 대한 논리 연산

`all()` 함수, `any()` 함수

```
x
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
all(x < 10)
```

```
## [1] FALSE
```

```
any(x < 10)
```

```
## [1] TRUE
```

벡터 논리 연산과 필터링

- x 벡터의 각 원소가 10보다 작은가?

```
x < 10
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
##  [7] FALSE FALSE FALSE FALSE
```

벡터 필터링

- 구문: `vector[condition]`

```
x[x < 10]
```

```
## [1] 2 4 6 8
```

논리 연산자와 조건문

알아두면 쓸데있는 연산자들

- `==`, `!=`

```
x[x == 4]
```

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

```
x[x != 4]
```

```
## [1]  2  6  8 10 12 14 16 18 2
```

- `%%` (나머지), `%/%` (몫)

```
x[x %% 4 == 0]
```

```
## [1]  4  8 12 16 20
```

```
x[x %/% 4 == 2]
```

```
## [1]  8 10
```

논리 연산자와 조건문 2

```
a <- c(TRUE, TRUE, FALSE)
b <- c(TRUE, FALSE, FALSE)
```

- `&` (AND) 와 `&&`

```
a & b
```

```
## [1]  TRUE FALSE FALSE
```

```
a && b # 첫번째 원소만 비교
```

```
## [1] TRUE
```

- `|` (OR) 와 `||`

```
a | b
```

```
## [1]  TRUE  TRUE FALSE
```

```
a || b # 첫번째 원소만 비교
```

```
## [1] TRUE
```

조건문 혼합하기

- 해석 연습

```
x[x == 4 | x > 15]
```

```
## [1]  4 16 18 20
```

- 왜 안될까?

```
x[x == 4 || x > 15]
```

```
## numeric(0)
```

필터링을 이용한 벡터 변경

```
x
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
x[x >= 10] <- 10
x
```

```
##  [1]  2  4  6  8 10 10 10 10 10 10
```

조건을 만족하는 위치 탐색 `which()`

```
x
```

```
##  [1]  2  4  6  8 10 10 10 10 10 10
```

```
x < 7
```

```
##  [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
##  [7] FALSE FALSE FALSE FALSE
```

```
which(x < 7)
```

```
## [1] 1 2 3
```

데이터가 없는 비어있을 땐 NA

- NA (Not Available): missing data

```
a <- c(20, NA, 13, 24, 309)
a
```

```
## [1]  20  NA  13  24 309
```

- NA 무시 옵션

```
mean(a)
```

```
## [1] NA
```

```
mean(a, na.rm = TRUE)
```

```
## [1] 91.5
```

존재하지 않음을 나타내는 NULL

NA와 NULL의 차이

```
NULL_is_not_blank <- NULL
c(1, NULL_is_not_blank)
```

```
## [1] 1
```

```
NA_is_blank <- NA
c(1, NA_is_blank)
```

```
## [1]  1 NA
```

벡터에 이름 붙여주기

- 경우에 따라서 벡터의 각 원소에 이름을 붙여줄 수 있음

- 보통의 경우

```
my_vector <- c(1, 20, 300)
names(my_vector)
```

```
## NULL
```

```
my_vector
```

```
## [1]   1  20 300
```

- 이름 붙여준 경우

```
names(my_vector) <- c("first", "
my_vector
```

```
##  first second  third
##      1     20    300
```

```
my_vector["second"]
```

```
## second
```

여러 벡터들을 묶어보자

`cbind()`와 `rbind()`

- 세로로 붙어주는 column bind

```
cbind(1:4, 12:15)
```

```
##      [,1] [,2]
## [1,]    1   12
## [2,]    2   13
## [3,]    3   14
## [4,]    4   15
```

- 가로로 쌓아주는 row bind

```
rbind(1:4, 12:15)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]   12   13   14   15
```

다음시간

행렬 다루기

참고자료 및 사용교재

[1] The art of R programming

- R 공부하시는 분이면 꼭 한번 보셔야하는 책입니다.
- 위 교재의 한글 번역본 빅데이터 분석 도구 R 프로그래밍도 있습니다. 도서 제목 클릭하셔서 구매하시면 저의 사리사욕을 충당하는데 도움이 됩니다.