제3주 : 벡터와 팩터

3.1. R의 자료구조

앞선 R의 기본 자료형은 단일값(scalar)의 자료형을 말한다면 R의 자료구조는 이런 단일값들로 구성된 자료 모음을 말합니다. 본 강의에서 설명할 자료구조는 vector, factor, matrix, array, list, data.frame 여섯가지입니다.

3.2. 벡터(Vector)

- 구성: vector는 동일한 자료형을 갖는 값들의 집합을 말한다.
- 만드는 법

① Vector 생성 연산자를 이용한 방법

▷ 시작값:종료값

```
> 1:5       # 1, 2, 3, 4, 5의 다섯 개 실수로 구성된 벡터
> 5:1       # 5, 4, 3, 2, 1의 다섯 개 실수로 구성된 벡터
```

② Vector 생성 함수를 이용한 방법

▷ c(원소1, 원소2, ...) : 기존 벡터를 결합하여 새로운 벡터를 반환한다.

```
> c(1,2,3)          # 1, 2, 3  세 개 실수로 구성된 벡터
> c(4,5,c(6,7,8))   # 4,5,6,7,8의 다섯 개 실수로 구성된 벡터
> x <- c(1,2,3)     # x는 1,2,3 세 개의 원소를 갖는 벡터
```

▷ seq(from=시작값, to=종료값, by=증가분, length.out=원소개수)

```
> seq(1, 3)
> seq(1, 5, by=2)            # 1부터 5까지 2씩 증가하는 벡터
> seq(from=1, to=5, by=2)    # 1부터 5까지 2씩 증가하는 벡터
> seq(0, 1, by=0.01)         # 0부터 1까지 0.01씩 증가
> seq(0, 1, length.out=101) # 0부터 1까지 길이가 101인 벡터
```

▷ rep(자료벡터, times=자료벡터의 전체 반복횟수,
each=자료벡터의 개별 원소들의 반복횟수)

```
> rep(c(1,2,3), times=2)  # vector 1,2,3 전체를 두 번 반복
                           # 결과: 1,2,3,1,2,3
> rep(c(1,2,3), each=2)   # 개별 원소를 각각 두 번씩 반복
                           # 결과: 1,1,2,2,3,3
```

● 데이터 접근

```
> x <- 5:1
> ########## (1) 인덱스(색인) 벡터를 이용한 접근 ##########
> x[1]           # 1번째 원소에 접근
[1] 5
> x[c(1, 2, 3)]   # 여러 위치의 원소를 가져올 때
[1] 5 4 3
> x[-c(1, 2, 3)]  # 음수 벡터를 전달하면 해당 위치의
[1] 2 1          # 값을 제외하고 가져온다.
> ########## (2) 논리형 벡터를 이용한 접근 ##########
> x > 3
[1] TRUE TRUE FALSE FALSE FALSE
> x[x > 3]
[1] 5 4
> ########## (3) 이름을 이용한 접근: x["이름"] ##########
> names(x)
NULL
> names(x) <- c("James", "John", "Robert", "Michael",
"William")
> x
  James    John  Robert Michael William
      5       4       3       2       1
> x["John"]
John
   4
> x[c("Robert", "John")]
Robert   John
     3      4
```

- 벡터 관련 함수

```
> x <- c(3, 4, 5)
> length(x)      # 벡터 x의 원소의 개수를 반환
[1] 3
> is.vector(x)   # 주어진 자료가 벡터이면 TRUE를 반환
[1] TRUE
> names(x)       # 주어진 벡터의 원소들의 이름을 반환
[1] NULL
> names(x) <- c("X1", "X2", "X3")  # 원소들의 이름을 설정
> x
X1 X2 X3
 3  4  5
```

3.3. Vector의 유용한 함수들

- 집합관련 함수들: `%in%`, `union(x,y)`, `intersect(x,y)`, `setdiff(x,y)`, `setequal(x,y)`

```
> 3 %in% x       # 값 3이 벡터 x의 원소인지
[1] TRUE
> c(2,3) %in% x
[1] FALSE TRUE
```

- any(x), all(x)

  - 논리형 벡터 x의 원소들 중 하나라도 TRUE가 있는지(any), 또는 모두 TRUE인지(all)을 판별하여 반환하는 함수

```
> x <- 1:5
> x > 3            # 3보다 큰 원소는TRUE, 그렇지않으면 FALSE
[1] FALSE FALSE FALSE TRUE TRUE
> all(x > 3)       # 모든 원소가 3보다 크지는 않으므로 FALSE
[1] FALSE
> any(x > 3)       # 3보다 큰 원소가 있으므로 TRUE
[1] TRUE
```

- 자료의 앞과 뒤에서 일부를 추출하는 head(x, n), tail(x, n)
  - 자료(x)로부터 n개(기본값은 6)의 자료를 앞과(head()) 뒤(tail()) 에서 가져옵니다.

```
> x < - 1:100
> head(x)                 # x 의 처음 6개를 가져옵니다.
[1] 1 2 3 4 5 6
> head(x, n=7)            # x 의 처음 7개를 가져옵니다.
[1] 1 2 3 4 5 6 7
> tail(x)                  # x의 마지막 6개를 가져옵니다.
[1] 95 96 97 98 99 100
> tail(x, n=7)            # x의 마지막 7개를 가져옵니다.
[1] 94 95 96 97 98 99 100
```

- 임의의 자료를 선택 및 추출하는 sample()

  - 사용법: sample(x, size, replace=FALSE, prob=NULL)

    - x: 양의 스칼라 또는 벡터
    - size: 추출할 개수
    - replace: 복원추출 여부. 기본값=FALSE
    - prob: x의 각 원소별 추출 확률. 기본값=NULL

```
> sample(10)       # 1:10의 원소들 중 10개를 임의로 선택
                   # 기본 replace=FALSE이므로 중복하지 않음
[1]  4  1  9 10  3  6  8  5  2  7

> sample(45, 6)   # 1:45의 원소들 중 6개를 임으로 선택
                   # 기본 replace=FALSE이므로 중복하지 않음
                   # 로또?
[1] 33 11 30  1 27 26

> sample(10, 3, replace=TRUE)
                   # 1:10의 원소들 중 3개를 임의로 선택
                   # replace=TRUE이므로 중복 선택 가능
[1] 3 3 3
```

```
> sample(10, 3, prob=(1:10) / 55)
                  # 1:10의 원소들 각각의 추출 확률을 부여함
[1]  9 10  4

> x <-seq(0, 1, by=0.1)
> x
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> x[sample(length(x), 3)]
                # 대괄호 안에 쓰여 무작위로 자료를 선택합니다.
[1] 0.6 0.5 0.8

> sample(x, 3)   # x의 원소 중 3개를 무작위로 선택합니다.
[1] 0.4 0.1 0.3
```

- 선택함수 which()

  - 사용법: **which**(x, arr.index=FALSE)

    - 논리형 벡터x를 받아 원소들 중 TRUE값을 가진 원소의 인덱스(위치, 색인)을 반환

```
> x <- c(2, 4, -1, 3)
> x > 2
[1] FALSE  TRUE FALSE  TRUE
> which(x > 2)      # x의 원소 중 2보다 큰 원소의 인덱스출력
[1] 2 4
> names(x) <-c("1st", "2nd", "3rd", "4th")
> which(x > 2)
2nd 4th
2   4
```

3.3. 팩터(Factor)

- 벡터 중에서 질적(범주형) 벡터를 저장하기 위한 자료구조
- 만드는 법:

f <- factor(벡터, levels=범주값들)

```
factor(x = character(), levels, labels = levels,
       exclude = NA, ordered = is.ordered(x), nmax = NA)
```

- x: 팩터로 만들 벡터
- levels: 범주값(수준)들. 여기 없는 값은 NA로 처리
- labels : 실제 값 외에 사용할 범주 이름 (벡터). 예를 들어 데이터에 1이 남자를 가리킬 경우 labels를 통해 "남자" 혹은 "M" 등으로 변경
- exclude : 범주수준으로 사용하지 않을 값 지정 (벡터)
- ordered : 순서 여부 지정 (TRUE/FALSE). 순서 있는 범주 수준값들의 경우 사용하며, 순서는 levels에 의해 명시적으로 지정하는 것을 추천
- nmax: 최대 level의 수

```
> x <- c("Man", "Male", "Man", "Lady", "Female")
> x
[1] "Man"    "Male"   "Man"    "Lady"   "Female"
# 길이가 5인 팩터를 정의함:
> xf <- factor(x)
> xf              # 범주 수준(levels)이 자동으로 설정됨
[1] Man    Male   Man    Lady   Female
Levels: Female Lady Male Man
> levels(xf)      # 유용한 함수 (1) 범주 수준들을 보여줌
[1] "Female" "Lady"   "Male"   "Man"
> nlevels(xf)     # 유용한 함수 (2) 범주 수준들의 개수
[2] 4
> levels(xf) <- c("F", "F", "M", "M") # 수준들 바꾸기
> xf
[1] M M M F F
Levels: F M     # 수준의 개수가 2개로 줄었음
```

```
> x <- c(1, 2, 3, 4, 5)
> factor(x, levels=c(1, 2, 3, 4))
[1] 1    2    3    4    <NA>
Levels: 1 2 3 4
```

➔ levels를 통해 자료 중 1, 2, 3, 4 네 개의 값만 범주값으로 사용

```
> factor(x, levels=c(1, 2, 3, 4), exclude=c(1, 2))
[1] <NA> <NA> 3    4    <NA>
Levels: 3 4
```

➔exclude를 사용해서 1, 2를 범주값에서 제거

```
>  factor(x,  levels=c(1,  2,  3,  4),  exclude=c(1,  2),
ordered=TRUE)
[1] <NA> <NA> 34    <NA>
Levels: 3 < 4
```

➔ordered에 TRUE를 주어 levels에 나열된1, 2, 3, 4를 순서대로 지정하지만 exclude를 통해1, 2 가 제거되어3, 4가 순서를 갖게 함

- 같이 사용할 수 있는 함수

  is.factor(): 주어진 자료가 팩터이면 TRUE를, 그렇지 않으면 FALSE를 반환

  as.factor(): 주어진 자료를 팩터로 변환하는 함수

  tapply(x=자료벡터, f, Fun, ..., simplify=TRUE):

  - x : 집계할 자료. 일반적으로 벡터
  - f : X를 집계할 팩터나 팩터의 리스트로서, 팩터가 아닐 경우 as.factor() 가 호출되어 팩터로 만든다. 또한 사용되는 자료의 길이가 X와 길이가 같아야 한다.
  - FUN : 집계에 집계에 사용할 사용할 함수
  - … : FUN에 사용되는 함수의 추가적인 전달인자.
  - simplify : TRUE이면 결과를 스칼라로 FALSE이면 리스트형의 배열로 반환, 기본값 TRUE

```
> score <- c(92, 90, 82, 88, 78, 64, 82, 90)
```
➔ 학생들의 성적벡터
```
> subject <- c("English", "English", "Math", "Math",
"Math", "Math", "English", "English")
```
➔ 각 학생들의 과목 벡터
```
> tapply(score, subject, mean)
English    Math
   88.5    78.0
```
➔ 과목별로 학생들 성적의 평균을 구한다.