03 벡터

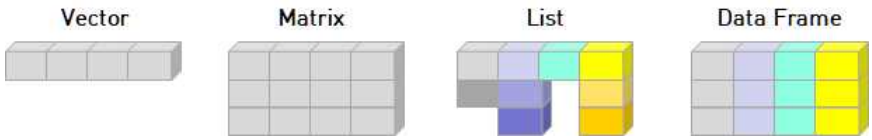
학습목표

1. R의 4가지 자료구조와 벡터의 특징을 이해한다.
2. 다양한 방법으로 벡터를 생성하고, 각 원소에 이름을 부여할 수 있다.
3. 인덱싱을 통해 벡터 원소에 접근할 수 있다.
4. 벡터 연산 및 벡터에 적용 가능한 연산을 적용할 수 있다. .

강의내용

R 자료구조 – 단일값들로 구성된 자료의 모음
– 벡터(vector), 행렬(matrix), 리스트(list), 데이터프레임(dataframe) 등

벡터(vector)	◦ 하나 이상의 원소로 이루어진 1차원 구조, R에서 가장 기본이 되는 자료구조 ◦ 동일한 자료형으로 구성되어야 함
행렬(matrix)	◦ 행과 열로 구성된 2차원 벡터
리스트(list)	◦ 다양한 자료형을 가질 수 있는 자료구조, 벡터의 확장형
데이터프레임 (dataframe)	◦ 데이터 분석에서 가장 많이 사용하는 테이블 형태의 2차원 자료구조 ◦ 각 열마다 다른 자료형을 가질 수 있으나 하나의 열은 동일한 자료형으로 구성



백터(vector) 생성

c() 함수 이용	– 원하는 값을 입력하여 벡터 생성 – 벡터 내부에 자료형이 뒤섞여 있을 경우 논리형<순자형<문자형 순으로 강제 형변환됨
시퀀스 연산자(:) 이용	– start:end 구조, 1씩 증가 또는 감소하는 경우 사용
seq(), req() 이용	– seq(start, end, by) # start 시작값, end 종료값, by 증감값 – rep(x, times, each) # x 벡터, times 전체 반복횟수, each 개별 반복횟수

```
# c() 함수 이용
a <- c(1, 3, 5)                    # 숫자형 벡터 a
b <- c("R","한림", '1', 'TRUE')    # 문자형 벡터 b
c <- c(T, F, TRUE, FALSE)          # 논리형 벡터 c

# 시퀀스 연산자( : )이용
1:10
50:-30
```

```
# seq(), req() 이용
seq(1, 100, 3);
seq(10, -10, -0.5)
rep(1, 3)
rep(1:4, times=3)
rep(c("a","b","c"), each=3)

# 우선순위가 높은 문자형으로 강제 형 변환
tc <- c("R", T, FALSE, 20, -1)
```

자료형 확인 함수

– 자료형 확인 : class() 함수, is.xxxxx() 함수 – is.numeric(), is.character(), is.factor() 등
– 자료의 구조 정보 확인 : str()

강제형 변환 함수

– as.xxxxx() 함수
– as.numeric(), as.logical(), as.character(), as.factor(), as.matrix(), as.data.frame() 등

03 벡터

벡터 원소에 접근

- 특정 조건을 만족하는 원소에 접근하는 것
- 대괄호([ ]) 안에 색인이나 이름을 입력한다. (예) vec[2], vec['b']

이름	a	b	c
	한림	R	코딩
색인	1	2	3

벡터 원소에 이름 붙이기

- 벡터는 색인을 가지고 있다. 색인은 1부터 시작
- c() 이용하여 벡터 생성 시 이름을 부여 하거나 names()를 이용하여 이름을 부여할 수 있음

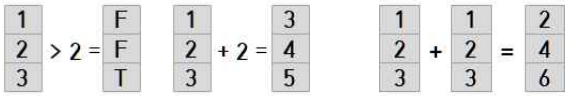
```
# 벡터 원소에 접근
vn <- c("한림대", "심리학")                 # vn 벡터의 1st 원소는 "한림대" 2nd 원소는 "심리학"
vn[1]                                       # 색인으로 벡터 원소에 접근

vn1 <- c(대학="한림대", 전공="심리학")      # 이름을 가진 벡터 vn1
vn["대학"]                                  # 이름으로 벡터 원소에 접근

vn2 <-  c("하나", "둘", "셋", "넷", "다섯")
names(vn2) <- c( "a", "b", "c", "d", "e")    # 이름을 가진 벡터 vn2

vn2[c(1,3,5)]             # 1,3,5번째 원소
vn2[-3]                   # 3번째를 제외한 모든 원소
vn2[2:4]                  # 2,3,4번째 원소
vn2[-c(2,4)]              # 2,4번째를 제외한 모든 원소
vn2[c(3,3,3)]             # 3번째 원소에 3번 접근
vn2[c("a","c","d")]       # 이름이 a,c,d인 원소
vn2[6]; vn2["f"]          # 해당되는 원소가 없으므로 NA 출력
```

벡터의 연산

(1) 벡터의 각 원소에 대해 연산

(2) 벡터 간 연산 : 대응하는 원소끼리 연산

(3) a %in% vec : a가 벡터에 포함되었는지 여부 확인



벡터에 사용가능한 함수

```
length(v)          # 자료의 개수
sum(v); mean(v)    # 자료의 합, 평균 mean(v)
max(v); min(v)     # 자료의 최대값, 최소값
sort(v, FALSE)     # 자료를 정렬하여 출력 (FALSE: 오름차순, TRUE: 내림차순)
```

```
# 벡터 연산
(v1 <- 1:5)
(v2 <- seq(10, 50, 10))
(v3 <- 1:3)
(v4 <- 1:10)

# 벡터 각 원소에 대해 연산
v1 + 10; v2 * 2
v1 > 2; v1 == 2

# 벡터 간 연산
v1 + v2     # 동일한 길이의 벡터
v1 + v3     # warning 발생
v1 + v4     # 벡터 길이가 배수일 때 벡터 재사용
```

```
# 어떤 값이 벡터에 포함되어있는지를 여부 알려줌
20 %in% v2
c(20, 60) %in% v2

# 벡터에 사용가능 함수
length(v2)          # 자료의 개수
sum(v2)             # 자료의 합
mean(v2)            # 자료의 평균
median(v2)          # 자료의 중앙값
max(v2); min(v2)    # 자료의 최대값, 최소값
var(v2); sd(v2)     # 자료의 분산, 표준편차
sort(v2)            # 자료를 정렬하여 출력
sort(v2, TRUE)      #       내림차순 출력
```