한빛아카데미
Hanbit Academy, Inc.

R 코딩 & 데이터 분석

Chapter 04. 벡터 다루기

목차

01

자료의 종류를 알아봅니다

01. 자료의 종류를 알아봅니다

I. 자료 분석의 목적과 사례

- 자료 분석(데이터 분석)을 하는 이유는 자료에 담겨있는 어떤 종류의 정보나 지식을 추출하고 이를 통해 현실을 이해하거나 현실의 문제를 해결하는 데 활용하기 위함

01. 자료의 종류를 알아봅니다

I. 자료 분석의 목적과 사례

▪ 자료 분석의 사례 I : 입욕제 사용자 분석

- [그림 4-1] 분석 결과, 입욕제가 어린이용으로 구매될 것이라는 예상과 달리 성인 커플용으로도 많이 구매됨을 확인할 수 있는데, A 기업은 이 분석 결과를 바탕으로 성인 커플용 입욕제 신제품을 출시하여 시장의 좋은 반응을 이끌어냄

No.	구분	언급량	No.	구분	언급량
1	아이	60.737	12	남편	5.014
2	아기	48.821	13	애기	4.688
3	커플	40.986	14	남자친구	3.959
4	데이트	38.903	15	육아	3.259
5	유아	36.393	16	주년	1.381
6	어린이	34.529	17	여자친구	1.336
7	부부	32.434	18	커플여행	646
8	연인	31.615	21	애인	452
9	혼자/홀로	15.316	22	와이프	359
10	아가	10.000	23	육아템	302
11	기념일	9.390	24	위킹맘	134

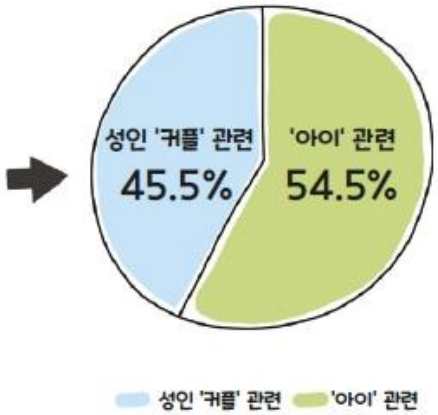


그림 4-1 입욕제 사용자 및 사용 관련 키워드 언급량

I. 자료 분석의 목적과 사례

▪ 자료 분석의 사례 II : 코로나핀

- 코로나핀은 공적 마스크 재고, 마스크 알리미(KF94 방역), 코로나19 상황판, 재난 문자 다시보기 등을 제공하는 모바일 앱으로, 데이터가 실생활에 도움을 줄 수 있는 대표적인 사례.

그림 4-2 코로나핀 앱 화면

I. 자료 분석의 목적과 사례

▪ 자료 분석의 사례 III : 온난화와 기상 이변

- [그림 4-3]은 1980~2008년 사이에 일어난 기상 이변 발생 자료에 대한 그래프로, 지진이나 화산 활동 빈도수는 큰 변화가 없지만 다른 재해들은 빈도수가 증가 중.
- 특히 태풍, 해일, 토네이도의 발생 빈도가 급격히 증가하고 있는데, 이러한 결과는 지구 온난화에 대한 대응이 필요하다는 것을 보여줌.

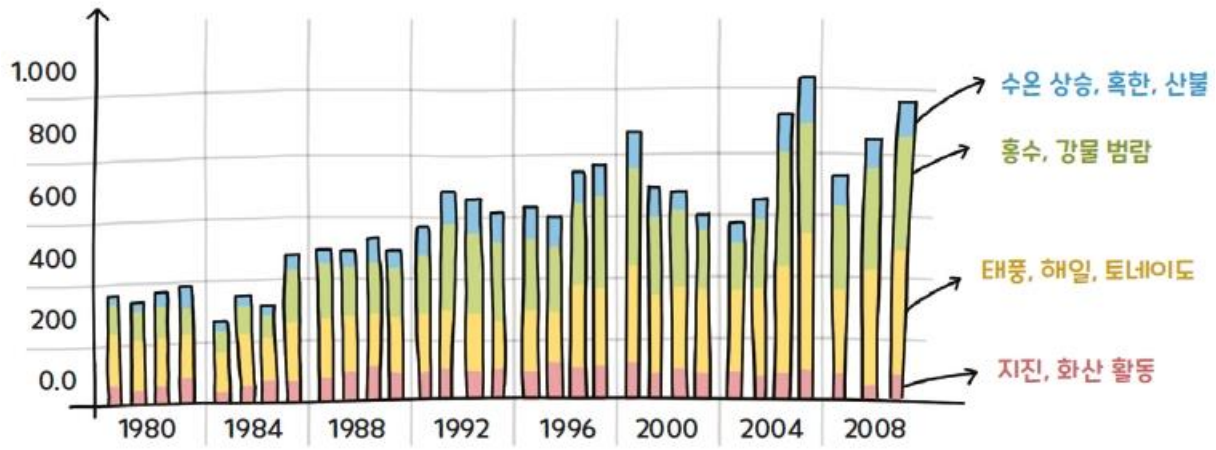


그림 4-3 기상 이변 발생 빈도수 © Extreme Weather and Climate Change

01. 자료의 종류를 알아봅니다

II. 1차원 자료와 2차원 자료

- 1차원 자료는 '전 세계 국가의 GNP'나 '연도별 수출액'과 같이 단일 주제에 대한 값들을 모아 놓은 것을 말함.
- [그림 4-4]는 시도별 인구수 자료의 일부로, 이 자료는 '인구수'라는 단일 주제에 대한 값을 모아 놓은 1차원 자료임.

그림 4-4 1차원 자료의 예 : 시도별 인구수

972, 341, 243과 같은 개별 숫자를 값(value)이라 하고, '인구수'와 같은 특정 주제에 대한 값들을 모아 놓은 것을 자료(data)라고 합니다.

01. 자료의 종류를 알아봅니다

II. 1차원 자료와 2차원 자료

- 2차원 자료는 '4학년 학생의 전 과목 성적'이나 '시도의 월별 실업률'과 같이 복수 주제에 대한 값들을 모아 놓은 자료.
- [그림 4-5]는 광역시별 경제 통계 자료의 일부로, 4개의 주제에 대한 값들을 모아 놓은 2차원 자료임.

도시	전체인구	도시지역면적	농림지역면적	택시업체수
서울	972	606	0	255
부산	341	940	0	97
대구	243	797	37	92
인천	295	580	263	60
광주	145	481	4	76
대전	147	495	28	76
울산	114	755	283	43
세종	34	140	148	5

그림 4-5 2차원 자료의 예 : 광역시별 경제 통계

01. 자료의 종류를 알아봅니다

II. 1차원 자료와 2차원 자료

표 4-1 R에서의 자료구조

자료	자료구조
1차원 자료	벡터(vector), 리스트(list), 팩터(factor)
2차원 자료	매트릭스(matrix), 데이터프레임(data frame)

* **자료구조**: 자료를 효율적으로 이용할 수 있도록 컴퓨터에 저장하는 방법

01. 자료의 종류를 알아봅니다

III. 범주형 자료와 수치형 자료

- **범주형 자료 :** '분류'의 의미를 갖는 값들로 구성된 자료로, 보통 문자로 표현되고 산술연산을 적용할 수 없음.
- **수치형 자료 :** 값들이 크기를 가지며 산술연산이 가능함.

법안 찬성여부	YES, YES, NO, YES, NO, NO, YES, ...
선호하는 색상	blue, red, orange, blue, green, red, ...
음식 맛 평가	good, bad, bad, good, good, bad, ...
성별	M, F, F, M, M, F, F, F, M, ...

(a) 범주형 자료의 예

키	169, 173, 158, 185, 169, 176, 181, ...
시력	1.5, 1.2, 0.7, 0.6, 0.1, 1.3, 2.0 ...
온도	10, 14, 26, −5, −20, 16, 31, −10, ...

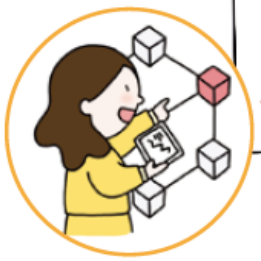
(b) 수치형 자료의 예

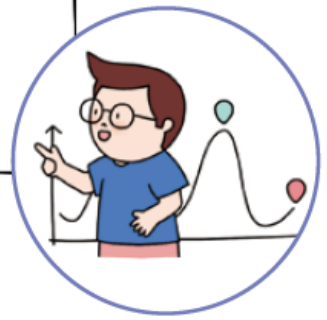
그림 4-6 범주형 자료와 수치형 자료

LAB. 최고의 직업 조사하기

다음은 한 기관에서 조사한 2017년 미국 최고의 직업에 대한 조사 결과입니다. 이를 바탕으로 다음 물음에 답해봅니다.

순위	직업명	채용수	평균 급여	직업 만족도 (5.0 만점)	평가 점수 (5.0 만점)
1	데이터 과학자	4.184	$110.000	4.4	4.8
2	데브옵스 엔지니어	2.725	$110.000	4.0	4.7
3	데이터 엔지니어	2.599	$106.000	4.3	4.7
4	세금 관리자	3.317	$110.000	4.0	4.7
5	분석 관리자	1.958	$112.000	4.1	4.6
6	인사 관리자	4.339	$85.000	3.8	4.6
7	데이터베이스 관리자	2.877	$93.000	3.8	4.5
8	전략 관리자	1.184	$130.000	4.3	4.5
9	UX 디자이너	1.691	$92.500	4.0	4.4
10	솔루션 아키텍처	2.232	$125.000	3.7	4.4





LAB. 최고의 직업 조사하기

1. 조사 결과를 정리한 이 표는 몇 차원 자료인가요?

   ▶ 2차원 자료

2. 이 조사 결과는 몇 개의 주제에 대한 자료를 모아 놓은 것인가요? 그리고 그 주제는 각각 무엇인가요?

   ▶ 6개의 주제에 대한 자료가 정리

3. '직업명' 열의 값이 갖고 있는 특성을 보면 어떤 유형의 자료라 할 수 있나요?

   ▶ '직업명'은 문자로 되어 있고 종류가 정해져 있으므로 범주형 자료

4. 이 조사 결과에서 '평가 점수'가 가장 높은 직업은 무엇인가요?

   ▶ 데이터 과학자

02

벡터 연산을 해볼까요?

02. 벡터 연산을 해볼까요?

I. 벡터에 대한 산술연산

- 벡터에 대한 산술연산은 벡터 안에 포함된 값들 하나하나에 대한 연산으로 바뀌어 실행

[코드 4-1]

```
d <- c(1,4,3,7,8)
2*d
d-5
3*d + 4
```

[코드 4-1] 실행 결과(1)

```
> d <- c(1,4,3,7,8)
```

– 벡터 d에 5개의 값을 저장

02. 벡터 연산을 해볼까요?

I. 벡터에 대한 산술연산

[코드 4-1] 실행 결과(2)

```
> 2*d
[1]  2  8  6 14 16
```

- d에 2를 곱한 결과, d에 포함된 값들 하나하나에 대해 2를 곱한 결과값이 출력
- d에 저장된 모든 값을 실제로 2씩 곱한 값으로 바꾸어 저장하고 싶다면 d <- 2*d와 같이 작성해야 함

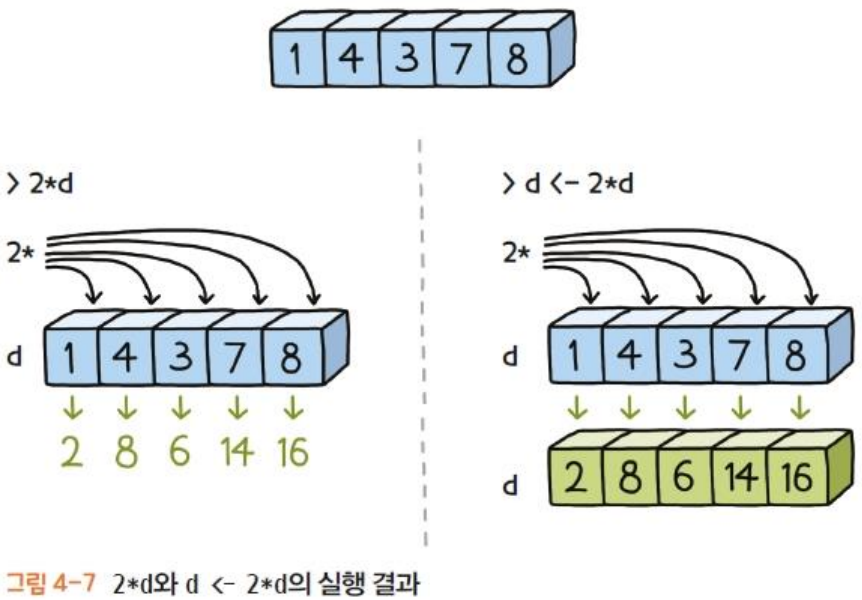


그림 4-7 2*d와 d <- 2*d의 실행 결과

02. 벡터 연산을 해볼까요?

I. 벡터에 대한 산술연산

[코드 4-1] 실행 결과(3)

```
> d-5
[1] -4 -1 -2  2  3
> 3*d + 4
[1]  7 16 13 25 28
```

– 벡터에 대한 산술연산은 벡터에 포함된 값들에 대한 산술연산으로 바뀌어 실행

02. 벡터 연산을 해볼까요?

II. 벡터와 벡터의 연산

- 벡터와 벡터 간의 산술연산은 벡터 간 대응되는 위치에 있는 값들끼리의 연산으로 바뀌어 실행

[코드 4-2]

```
x <- c(1,2,3,4)
y <- c(5,6,7,8)
x + y              # 대응하는 원소끼리 더하여 출력
x * y              # 대응하는 원소끼리 곱하여 출력
z <- x + y         # x, y를 더하여 z에 저장
z
```

02. 벡터 연산을 해볼까요?

II. 벡터와 벡터의 연산

[코드 4-2] 실행 결과(1)

```
> x <- c(1,2,3,4)
> y <- c(5,6,7,8)
```

– 벡터 x와 벡터 y에 각각 4개의 값을 저장

II. 벡터와 벡터의 연산

[코드 4-2] 실행 결과(2)

```
> x + y          # 대응하는 원소끼리 더하여 출력
[1]  6  8 10 12
> x * y          # 대응하는 원소끼리 곱하여 출력
[1]  5 12 21 32
```

– 두 벡터 사이의 합과 곱의 결과를 보면 인덱스가 같은 위치에 있는 값끼리 연산이 이루어진 것을 알 수 있음

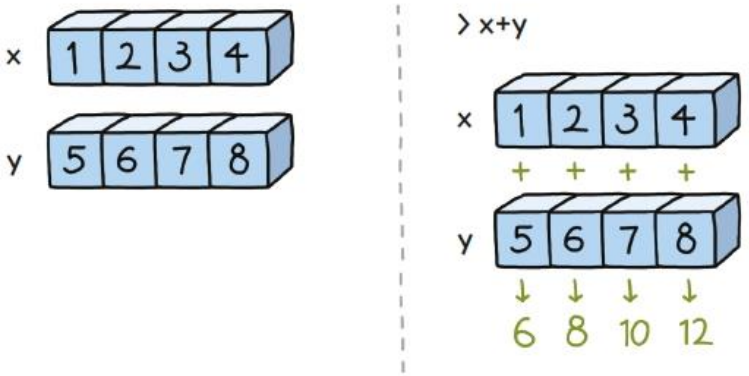


그림 4-8 x+y의 실행

II. 벡터와 벡터의 연산

[코드 4-2] 실행 결과(3)

```
> z <- x + y                # x, y를 더하여 z에 저장
> z
[1] 6 8 10 12
```

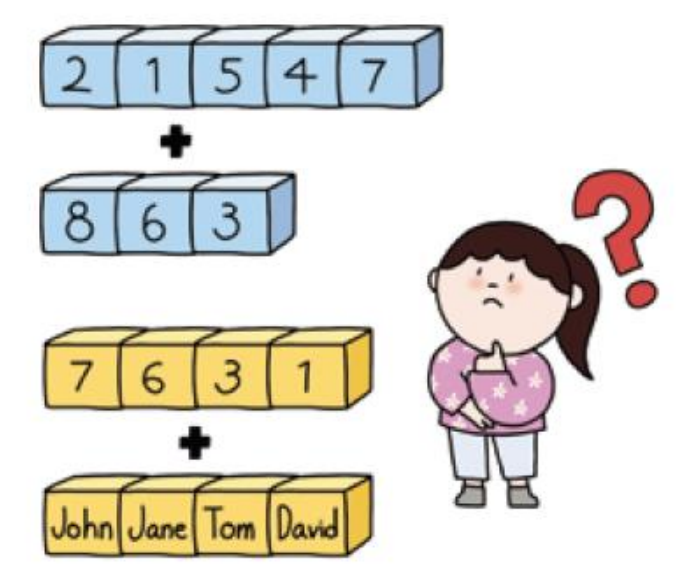
– 벡터 x와 벡터 y를 더하여 z에 저장한 뒤 z의 내용을 출력함

II. 벡터와 벡터의 연산

하나 더 알기 **벡터와 벡터의 연산이 가능하기 위한 조건**

1. 두 벡터의 길이가 같아야 합니다.

2. 두 벡터에 포함된 값의 종류가 같아야 합니다.



02. 벡터 연산을 해볼까요?

II. 벡터와 벡터의 연산

[코드 4-3]

```
# [코드 4-2]에 이어서 실행
m <- c(x, y)                    # x, y의 원소들을 결합하여 m에 저장
m
n <- c(y, x)                    # y, x의 원소들을 결합하여 n에 저장
n
p <- c(x, y, 90, 100)           # x, y의 원소들과 90, 100을 결합하여 p에 저장
p
```

02. 벡터 연산을 해볼까요?

II. 벡터와 벡터의 연산

[코드 4-3] 실행 결과(1)

```
> m <- c(x, y)                    # x, y의 원소들을 결합하여 m에 저장
> m
[1] 1 2 3 4 5 6 7 8
```

– 벡터 x의 원소들과 벡터 y의 원소들을 합쳐서 m에 저장하고 m의 내용을 출력

[코드 4-3] 실행 결과(2)

```
> n <- c(y, x)                    # y, x의 원소들을 결합하여 n에 저장
> n
[1] 5 6 7 8 1 2 3 4
```

– 벡터 y의 원소들과 벡터 x의 원소들을 합쳐서 n에 저장하고 n의 내용을 출력

02. 벡터 연산을 해볼까요?

II. 벡터와 벡터의 연산

[코드 4-3] 실행 결과(3)

```
> p <- c(x, y, 90, 100)        # x, y의 원소들 및 90, 100을 결합하여 p에 저장
> p
 [1]   1  2   3   4   5   6   7   8  90 100
```

– c( ) 함수는 2개 또는 2개 이상의 벡터, 벡터와 단일 값들을 결합 가능

02. 벡터 연산을 해볼까요?

II. 벡터와 벡터의 연산

[코드 4-4]

```
v1 <- c(1, 2, 3, 4)
v2 <- c('John', 'Jane', 'Tom')
v3 <- c(v1, v2)                  # v1, v2의 원소들을 결합하여 v3에 저장
v3
```

[실행결과]

```
> v1 <- c(1, 2, 3, 4)
> v2 <- c('John', 'Jane', 'Tom')
> v3 <- c(v1, v2)                  # v1, v2의 원소들을 결합하여 v3에 저장
> v3
[1] "1"    "2"    "3"    "4"    "John" "Jane" "Tom"
```

– 숫자 벡터와 문자 벡터를 c( )로 결합하면 숫자값이 문자로 변환되어 결합됨

02. 벡터 연산을 해볼까요?

III. 벡터에 적용 가능한 함수들

- 데이터 분석에 많이 사용되는 함수는 [표 4-2]와 같음

표 4-2 벡터를 입력값으로 하는 함수들

함수명	설명
sum()	벡터에 저장된 값들의 합
mean()	벡터에 저장된 값들의 평균
median()	벡터에 저장된 값들의 중앙값
max(), min()	벡터에 저장된 값들의 최댓값, 최솟값
var()	벡터에 저장된 값들의 분산
sd()	벡터에 저장된 값들의 표준편차
sort()	벡터에 저장된 값들을 정렬(오름차순이 기본)
range()	벡터에 저장된 값들의 범위(최솟값, 최댓값)
length()	벡터에 저장된 값들의 개수(벡터의 길이)

02. 벡터 연산을 해볼까요?

III. 벡터에 적용 가능한 함수들

[코드 4-5]

```
d <- c(1,2,3,4,5,6,7,8,9,10)
sum(d)
sum(2*d)
length(d)
mean(d[1:5])
max(d)
min(d)
sort(d)                              # 오름차순 정렬
sort(d, decreasing = FALSE)          # 오름차순 정렬
sort(d, decreasing = TRUE)           # 내림차순 정렬

v1 <- median(d)
v1
v2 <- sum(d)/length(d)
v2
```

III. 벡터에 적용 가능한 함수들

[코드 4-5] 실행 결과(1)

```
> d <- c(1,2,3,4,5,6,7,8,9,10)
> sum(d)
[1] 55
> sum(2*d)
[1] 110
> length(d)
[1] 10
> mean(d[1:5])
[1] 3
> max(d)
[1] 10
> min(d)
[1] 1
> sort(d)                                    # 오름차순 정렬
[1]  1  2  3  4  5  6  7  8  9 10
> sort(d, decreasing = FALSE)                # 오름차순 정렬
[1]  1  2  3  4  5  6  7  8  9 10
> sort(d, decreasing = TRUE)                 # 내림차순 정렬
[1] 10  9  8  7  6  5  4  3  2  1
```

III. 벡터에 적용 가능한 함수들

[코드 4-5] 실행 결과(2)

```
> v1 <- median(d)
> v1
[1] 5.5
> v2 <- sum(d)/length(d)
> v2
[1] 5.5
```

– 이 코드는 벡터 d의 중앙값을 구하여 변수 v1에 저장하는 명령문과,
벡터 d의 합계를 벡터 d의 길이(값의 개수)로 나누어 v2에 저장하는 명령문

02. 벡터 연산을 해볼까요?

IV. 벡터에 논리연산자 적용

- 논리연산자는 연산의 결과값이 TRUE 또는 FALSE인 연산자

표 4-3 논리연산자

연산자	사용 예	설명
<	A < B	B가 A보다 크면 TRUE
<=	A <= B	B가 A보다 크거나 같으면 TRUE
>	A > B	A가 B보다 크면 TRUE
>=	A >= B	A가 B보다 크거나 같으면 TRUE
==	A == B	A와 B가 같으면 TRUE
!=	A != B	A와 B가 같지 않으면 TRUE
¦	A ¦ B	A 또는 B 어느 한 쪽이라도 TRUE이면 TRUE
&	A & B	A와 B 모두 TRUE일 때만 TRUE

02. 벡터 연산을 해볼까요?

IV. 벡터에 논리연산자 적용

[코드 4-6]

```
d <- 1:9
d>=5                          # d 원소 각각이 >=5인지 검사
d[d>5]                        # 6 7 8 9
sum(d>5)                      # 5보다 큰 값의 개수를 출력
sum(d[d>5])                   # 5보다 큰 값의 합계를 출력
d==5

condi <- d > 5 & d < 8        # 조건을 변수에 저장
d[condi]                      # 조건에 맞는 값들을 선택
```

02. 벡터 연산을 해볼까요?

IV. 벡터에 논리연산자 적용

[코드 4-6] 실행 결과(1)

```
> d <- 1:9
```

– 벡터 d에 1부터 9까지의 숫자를 저장

[코드 4-6] 실행 결과(2)

```
> d>=5              # d 원소 각각이 >=5인지 검사
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

– d>=5는 벡터 d에 포함된 값들이 '5보다 크거나 같은지'를 판단하는 논리연산

– 그 결과 벡터 안의 값 1~4에 대해서는 FALSE가, 5~9에 대해서는 TRUE가 도출

IV. 벡터에 논리연산자 적용

[코드 4-6] 실행 결과(3)

```
> d[d>5]
[1] 6 7 8 9
```

– 이 명령문을 보면 벡터의 인덱스를 지정하는 부분에 d>5라는 논리연산이 존재하고 있는데, 이런 경우는 d>5를 먼저 실행한 후 그 결과값으로 인덱스 부분 실행

```
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE        # d>5
```

– d에 저장되어 있는 값 중 TRUE에 대응되는 값들만 추출

```
d      : [1]     1     2     3     4     5     6     7     8     9
d>5    : [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
d[d>5] :                                    6     7     8     9
```

– d[d>5]의 의미는 d에 저장된 값 중 5보다 큰 값들만 추출

IV. 벡터에 논리연산자 적용

[코드 4-6] 실행 결과(4)

```
> sum(d>5)                    # 5보다 큰 값의 개수를 출력
[1] 4
```

– 이 명령문도 앞의 예와 같이 d>5를 먼저 실행한 후 그 결과를 sum( ) 함수에 입력값으로 주어 결과를 도출

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

– 'd에 저장된 값 중 5보다 큰 값들의 개수'를 구하는 것과 동일

IV. 벡터에 논리연산자 적용

[코드 4-6] 실행 결과(5)

```
> sum(d[d>5])              # 5보다 큰 값의 합계를 출력
[1] 30
```

– d>5를 먼저 실행하고, 그 결과값으로 d[d>5]를 실행

```
d>5          : [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
d[d>5]       : [1] 6 7 8 9
sum(d[d>5])  : [1] 30
```

– 'd에 저장된 값 중 5보다 큰 값들의 합계'를 구하는 것

IV. 벡터에 논리연산자 적용

[코드 4-6] 실행 결과(6)

```
> condi <- d > 5 & d < 8          # 조건을 변수에 저장
```

- 이 명령문은 d > 5 & d < 8이라고 하는 조건문을 condi라는 이름의 변수에 저장하는 것.
- 조건문의 의미는 'd에 저장된 값 중 5보다 크고 8보다 작은 값'

[코드 4-6] 실행 결과(7)

```
> d[condi]                  # 조건에 맞는 값들을 선택
[1] 6 7
```

- 이 명령문의 의미는 condi에 저장된 조건문에 부합하는 값들만 추출하는 것.

LAB. R 카페의 매출액 분석하기 I

R 카페의 매니저로부터 매출액 분석을 의뢰받았습니다. 카페 매니저는 최근 일주일간의 요일별 매출액, 총 매출액, 평균 매출액, 평균 매출 이상인 요일에 대해 알기를 원합니다. 우선 매니저에게 카페 메뉴판과 일주일간 판매량에 대한 자료를 요청하여 데이터를 확보했습니다. 카페의 매출을 분석해봅니다

메뉴	월	화	수	목	금	토	일
에스프레소	4	5	3	6	5	4	7
아메리카노	63	68	64	68	72	89	94
카페라떼	61	70	59	71	71	92	88

LAB. R 카페의 매출액 분석하기 I

1. 메뉴별로 벡터를 생성해 판매량을 요일순으로 저장합니다.

```
espresso <- c(4, 5, 3, 6, 5, 4, 7)
americano <- c(63, 68, 64, 68, 72, 89, 94)
latte <- c(61, 70, 59, 71, 71, 92, 88)
```

2. 메뉴별 판매량에 각각의 가격을 곱하여 메뉴마다 요일별 매출액을 구합니다. 그 결과를 새로운 변수에 저장합니다.

```
sale.espresso <- 2 * espresso
sale.americano <- 2.5 * americano
sale.latte <- 3.0 * latte
```

3. 각 메뉴의 매출액을 더하여 요일별 매출액을 구하고 이것을 새로운 변수에 저장합니다. 값의 이름은 요일로 설정합니다.

```
sale.day <- sale.espresso + sale.americano + sale.latte
names(sale.day) <- c('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')
```

LAB. R 카페의 매출액 분석하기 I

4. 요일별 매출액을 합산하여 일주일간 총 매출액을 구합니다.

```
sum(sale.day)
```

5. 요일별 매출액 벡터로 일평균 매출액을 구하고 새로운 변수에 저장합니다.

```
sale.mean <- mean(sale.day)
```

6. 각 요일의 매출액과 평균 매출액을 비교하여 평균 이상인 요일을 구합니다.

```
names(sale.day[sale.day >= sale.mean])
```

03

팩터와 리스트는 무엇인가요?

03. 팩터와 리스트는 무엇인가요?

I. 팩터

- **팩터(factor) :** 문자형 데이터가 저장되는 벡터의 일종으로, 저장되는 문자값들이 어떠한 종류를 나타내는 값일 때 사용

[코드 4-7]

```
bt <- c('A', 'B', 'B', 'O', 'AB', 'A') # 문자형 벡터 bt 정의
bt.new <- factor(bt)                   # 팩터 bt.new 정의
bt                                     # 벡터 bt의 내용 출력
bt.new                                 # 팩터 bt.new의 내용 출력
bt[5]                                  # 벡터 bt의 5번째 값 출력
bt.new[5]                              # 팩터 bt.new의 5번째 값 출력
levels(bt.new)                         # 팩터에 저장된 값의 종류를 출력
as.integer(bt.new)                     # 팩터의 문자값을 숫자로 바꾸어 출력
bt.new[7] <- 'B'                       # 팩터 bt.new의 7번째에 'B' 저장
bt.new[8] <- 'C'                       # 팩터 bt.new의 8번째에 'C' 저장
bt.new                                 # 팩터 bt.new의 내용 출력
```

03. 팩터와 리스트는 무엇인가요?

I. 팩터

[코드 4-7] 실행 결과(1)

```
> bt <- c('A', 'B', 'B', 'O', 'AB', 'A') # 문자형 벡터 bt 정의
> bt.new <- factor(bt)                   # 팩터 bt.new 정의
```

- 6명의 혈액형 데이터가 저장된 문자형 벡터 bt를 생성하고, factor() 함수를 이용하여 팩터 bt.new를 생성

[코드 4-7] 실행 결과(2)

```
> bt                                    # 벡터 bt의 내용 출력
[1] "A"  "B"  "B"  "O"  "AB" "A"
> bt.new                                # 팩터 bt.new의 내용 출력
[1] A  B  B  O  AB A
Levels: A AB B O
```

- 벡터 bt의 내용을 출력하면 따옴표로 묶여 있는 문자값들이 출력
- 그런데 팩터 bt.new를 출력하면 따옴표가 없는 문자값들이 출력

I. 팩터

[코드 4-7] 실행 결과(3)

```
> bt[5]                    # 벡터 bt의 5번째 값 출력
[1] "AB"
> bt.new[5]                # 팩터 bt.new의 5번째 값 출력
[1] AB
Levels: A AB B O
```

– 팩터도 벡터의 일종이기 때문에 값을 추출하는 방법은 벡터와 같이 인덱스를 이용하거나, 값에 이름이 붙어 있다면 이름을 통해서도 값을 추출

[코드 4-7] 실행 결과(4)

```
> levels(bt.new)                    # 팩터에 저장된 값의 종류를 출력
[1] "A" "AB" "B" "O"
```

– levels( ) 함수는 팩터에 저장된 값들의 종류를 알아내는 함수

03. 팩터와 리스트는 무엇인가요?

I. 팩터

[코드 4-7] 실행 결과(5)

```
> as.integer(bt.new)      # 팩터의 문자값을 숫자로 바꾸어 출력
[1] 1 3 3 4 2 1
```

- as.integer( ) 함수를 이용하여 팩터의 문자값을 숫자로 바꿀 수 있음
- 문자값의 알파벳 순서에 따라 숫자값 부여
  - » A : 1
  - » AB : 2
  - » B : 3
  - » O : 4

I. 팩터

[코드 4-7] 실행 결과(6)

```
> bt.new[7] <- 'B'                    # 팩터 bt.new의 7번째에 'B' 저장
> bt.new[8] <- 'C'                    # 팩터 bt.new의 8번째에 'C' 저장
경고메시지(들):
In `[<-.factor`(`*tmp*`, 8, value = "C") :
  invalid factor level, NA generated
> bt.new                              # 팩터 bt.new의 내용 출력
[1] A    B    B    O    AB    A    B    <NA>
Levels: A AB B O
```

- 첫 번째 명령문은 정상 실행되는 이유는 'B'가 Levels에 정해져 있는 값이기 때문
- 두 번째 명령문을 실행하면 경고 메시지가 뜨는 이유는 입력하려는 'C'가 Levels에 없는 값이기 때문
- 세 번째 명령문을 실행하면 8번째 값은 C가 아닌 <NA>로 표시

03. 팩터와 리스트는 무엇인가요?

II. 리스트

- **리스트(list) :** 자료형이 다른 값들을 한곳에 저장하고 다룰 수 있도록 해주는 수단
- 아래 사람의 정보를 보면 저장할 값에 문자, 숫자, 논리형이 섞여 있을 뿐만 아니라 취미의 경우 하나의 값이 아니라 여러 개의 값을 저장해야 하는 상황이므로 벡터에는 저장할 수 없는 유형인데, 여러 자료형이 섞여 있기 때문에 리스트에 저장해야 함

이름 : 'Tom'	나이 : 25
학생 여부 : TRUE	취미 : 'balling', 'tennis', 'ski'

03. 팩터와 리스트는 무엇인가요?

II. 리스트

[코드 4-8]

```
h.list <- c('balling', 'tennis', 'ski')          # 취미를 벡터에 저장
person <- list(name='Tom', age=25, student=TRUE, hobby=h.list) # 리스트 생성
person                              # 리스트에 저장된 내용을 모두 출력
person[[1]]                         # 리스트의 첫 번째 값을 출력
person$name                         # 리스트에서 값의 이름이 name인 값을 출력
person[[4]]                         # 리스트의 네 번째 값을 출력
```

03. 팩터와 리스트는 무엇인가요?

II. 리스트

[코드 4-8] 실행 결과(1)

```
> h.list <- c('balling', 'tennis', 'ski')          # 취미를 벡터에 저장
```

– 3개의 취미가 저장된 벡터 h.list를 생성

[코드 4-8] 실행 결과(2)

```
> person <- list(name='Tom', age=25, student=TRUE, hobby=h.list) # 리스트 생성
```

– person이라는 이름의 리스트를 생성

– 리스트는 1차원 자료 구조이면서 다양한 자료형들의 값을 저장

03. 팩터와 리스트는 무엇인가요?

II. 리스트

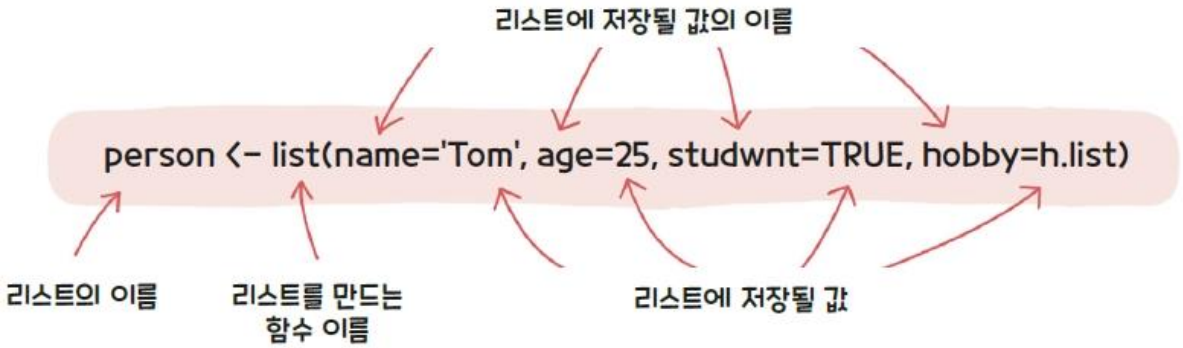


그림 4-9 리스트 person의 구성

하나 더 알기

리스트에 저장할 명령문이 많으면 한 줄에 작성하기가 어려워 여러 줄에 나누어 작성해야 합니다.

```
person <- list(name='Tom',
               age=25,
               student=TRUE,
               hobby=h.list)      # 리스트 생성
```

II. 리스트

[코드 4-8] 실행 결과(3)

```
> person                    # 리스트에 저장된 내용을 모두 출력
$name
[1] "Tom”

$age
[1] 25

$student
[1] TRUE

$hobby
[1] "balling" "tennis" "ski"
```

– 저장된 값들이 값의 이름과 함께 세로 방향으로 하나씩 출력

II. 리스트

[코드 4-8] 실행 결과(4)

```
> person[[1]]    # 리스트의 첫 번째 값을 출력
[1] "Tom"
```

- 인덱스를 이용하는 방법인데 벡터와 다른 점은 인덱스 지정 부분에 [ ]가 아닌 [[ ]]를 사용

[코드 4-8] 실행 결과(5)

```
> person$name    # 리스트에서 값의 이름이 name인 값을 출력
[1] "Tom"
```

- 리스트에 저장된 값을 추출하는 두 번째 방법은 값의 이름을 이용

II. 리스트

[코드 4-8] 실행 결과(6)

```
> person[[4]]   # 리스트의 네 번째 값을 출력
[1] "balling" "tennis" "ski"
```

– person[[4]]의 결과를 보면 하나의 값이 아닌 3개의 값이 들어 있는 벡터

LAB. R 카페의 매출액 분석하기 II

R 카페의 매니저로부터 매출액 분석을 의뢰받았습니다. 카페 매니저는 최근 일주일간의 요일별 매출액, 총 매출액, 평균 매출액, 평균 매출 이상인 요일에 대해 알기 원합니다. 우선 매니저에게 카페 메뉴판과 일주일간 판매량에 대한 자료를 요청하여 데이터를 확보했습니다. 어떻게 카페의 매출을 분석할 수 있을까요? 이번에는 앞서 살펴본 LAB과는 달리 리스트와 팩터를 이용해 분석해보겠습니다.

메뉴	월	화	수	목	금	토	일
에스프레소	4	5	3	6	5	4	7
아메리카노	63	68	64	68	72	89	94
카페라떼	61	70	59	71	71	92	88

LAB. R 카페의 매출액 분석하기 II

1. 리스트를 생성해 메뉴별 판매량, 메뉴별 가격, 메뉴 이름을 저장합니다.

```
cafe <- list(espresso = c(4, 5, 3, 6, 5, 4, 7),
             americano = c(63, 68, 64, 68, 72, 89, 94),
             latte = c(61, 70, 59, 71, 71, 92, 88),
             price = c(2.0, 2.5, 3.0),
             menu = c('espresso','americano','latte'))
```

2. 생성한 리스트에서 메뉴만 추출해 팩터로 변경합니다.

```
cafe$menu <- factor(cafe$menu)
```

3. 리스트 내 가격 벡터를 선택해 값의 이름을 메뉴명으로 설정합니다.

```
names(cafe$price) <- cafe$menu
```

LAB. R 카페의 매출액 분석하기 II

4. 리스트에서 가격 벡터를 선택하고, 그 벡터에서 메뉴명에 해당하는 가격 정보를 선택합니다. 그리고 리스트에서 메뉴명에 해당하는 판매량을 선택하여 곱합니다. 각 메뉴마다 반복하여 메뉴의 요일별 매출액을 구해 각각 새로운 변수에 저장합니다.

```
sale.espresso <- cafe$price['espresso'] * cafe$espresso
sale.americano <- cafe$price['americano'] * cafe$americano
sale.latte <- cafe$price['latte'] * cafe$latteo
```

5. 요일별 매출액, 총 매출액, 평균 매출액, 평균 매출 이상인 요일을 구합니다.

```
sale.day <- sale.espresso + sale.americano + sale.latte # 요일별 매출액
names(sale.day) <- c('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')
sum(sale.day)                                           # 총 매출액
sale.mean <- mean(sale.day)                             # 평균 매출액
names(sale.day[sale.day >= sale.mean] )                 # 평균 매출액 이상인 요일 추출
```

실전분석. 월별 감전사고 통계 분석하기

[문제]

다음은 어느 해의 월별 감전사고 통계입니다. 이를 벡터에 저장하고 분석해보겠습니다.

M1	M2	M3	M4	M5	M6
31	26	42	47	50	54
M7	M8	M9	M10	M11	M12
70	66	43	32	32	22

실전분석. 월별 감전사고 통계 분석하기

[해결]

1. 통계 자료를 accident 벡터에 저장합니다.

```
accident <- c(31,26,42,47,50,54,70,66,43,32,32,22)
names(accident) <- c('M1','M2','M3','M4','M5','M6','M7','M8’,
                     'M9','M10','M11','M12')
accident
```

```
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12
31 26 42 47 50 54 70 66 43  32  32  22
```

실전분석. 월별 감전사고 통계 분석하기

2. 1년간 총 감전사고 건수를 알아봅니다.

```
sum(accident)
```

```
[1] 515
```

3. 가장 사고가 많은 달과 가장 적은 달의 건수를 알아봅니다.

```
max(accident)
min(accident)
```

```
[1] 70
[1] 22
```

4. 만일 사고율이 10% 감소한다면 사고 건수가 어떻게 변화하는지 구해봅니다.

```
accident*0.9
```

```
  M1   M2   M3   M4   M5   M6   M7   M8   M9   M10  M11  M12
27.9 23.4 37.8 42.3 45.0 48.6 63.0 59.4 38.7 28.8 28.8 19.8
```

실전분석. 월별 감전사고 통계 분석하기

5. 사고 건수가 50건을 넘는 달의 통계만 출력해봅니다.

```
accident[accident>=50]
```

```
M5 M6 M7 M8
50 54 70 66
```

6. 사고 건수가 50건을 넘는 달의 이름을 출력해봅니다.

```
month.50 <- accident[accident >= 50] # 사고 건수 50이 넘는 달만 추출
names(month.50)
names(accident[accident >= 50])    # 이렇게 실행해도 됨
```

```
[1] "M5" "M6" "M7" "M8"
```

실전분석. 월별 감전사고 통계 분석하기

7. 사고 건수가 50 미만인 달은 1년 중 몇 개월인지 구해봅니다.

```
length(accident[accident<50])
```

```
[1] 8
```

8. 6월보다 사고가 많은 달과 사고 건수를 구해봅니다.

```
M6.acc <- accident[6]
accident[accident > M6.acc]
# 다음과 같이 한 줄 명령어로도 처리 가능
accident[accident > accident[6]]
```

```
M7 M8
70 66
```

Thank You !

한빛아카데미 Hanbit Academy, Inc.