한빛아카데미 Hanbit Academy, Inc.

R 코딩 & 데이터 분석

Chapter 05. 매트릭스와 데이터프레임 알아보기

목차

01

이것이 매트릭스입니다

01. 이것이 매트릭스입니다

I. 2차원 자료의 저장

- 매트릭스(matrix)와 데이터프레임(data frame)은 2차원 자료를 저장하기 위한 대표적인 자료구조
  - **1차원 자료 :** '학생들의 몸무게'와 같이 단일 주제의 값들을 모아 놓은 것
  - **2차원 자료 :** 키 · 몸무게 · 나이와 같이 한 사람에 대한 여러 주제로 데이터를 수집한 형태
- 매트릭스와 데이터프레임의 차이점은 매트릭스에 저장되는 모든 자료의 종류(data type)가 동일한 반면 데이터프레임에는 서로 다른 종류의 데이터가 저장

01. 이것이 매트릭스입니다

I. 2차원 자료의 저장

몸무게
62.4
65.3
59.8
46.5
49.8
58.7

(a) 1차원 자료 : 학생들의 몸무게

키	몸무게	나이
168.4	62.4	29
169.5	65.3	27
172.1	59.8	26
185.2	46.5	25
173.7	49.8	26
175.2	58.7	28

(b) 2차원 자료 : 학생들의 키·몸무게·나이

그림 5-1 1차원 자료와 2차원 자료

- 여러 개의 벡터를 모아 놓은 것이 매트릭스 또는 데이터프레임

01. 이것이 매트릭스입니다

I. 2차원 자료의 저장

키	몸무게	나이
168.4	62.4	29
169.5	65.3	27
172.1	59.8	26
185.2	46.5	25
173.7	49.8	26
175.2	58.7	28

벡터 벡터 벡터

1차원 자료

키	몸무게	나이
168.4	62.4	29
169.5	65.3	27
172.1	59.8	26
185.2	46.5	25
173.7	49.8	26
175.2	58.7	28

매트릭스, 데이터프레임

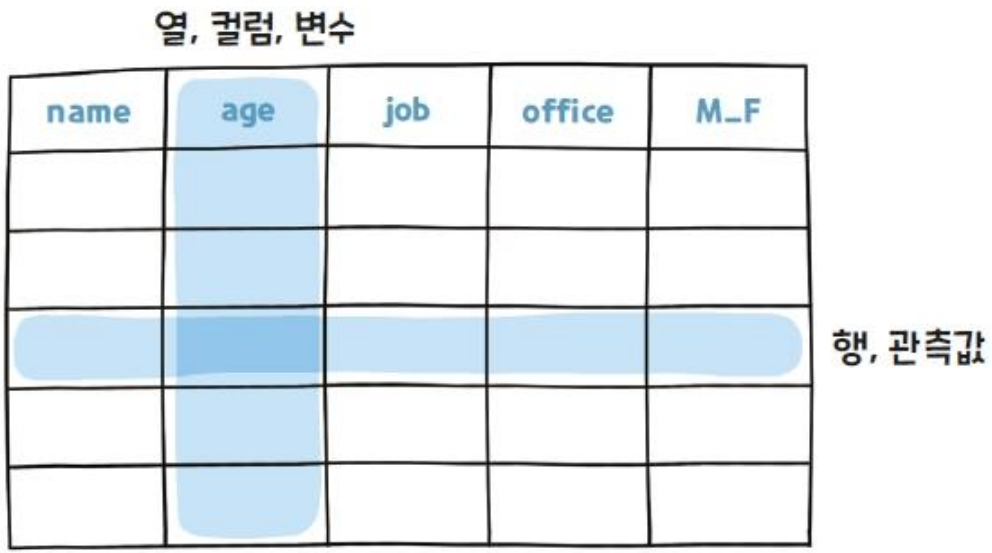
2차원 자료

그림 5-2 2차원 자료의 형태는 1차원 자료들의 모음

01. 이것이 매트릭스입니다

I. 2차원 자료의 저장

- 테이블에서 가로줄 방향은 행(row) 또는 관측값(observation)
- 세로줄 방향은 열(column), 컬럼, 변수(variable)



01. 이것이 매트릭스입니다

II. 매트릭스 만들기

- **매트릭스(Matrix) :** 2차원 테이블 형태의 자료구조
- 매트릭스의 모든 셀(cell)에 저장되는 값은 동일한 종류(data type)이어야 함
- 매트릭스는 보통 숫자로만 구성된 2차원 자료를 저장하고 처리하는 데 이용

01. 이것이 매트릭스입니다

II. 매트릭스 만들기

[코드 5-1]

```
z <- matrix(1:20, nrow=4, ncol=5)
z                                    # 매트릭스 z의 내용을 출력
```

[실행결과]

```
> z <- matrix(1:20, nrow=4, ncol=5)
> z                                  # 매트릭스 z의 내용을 출력

>
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

01. 이것이 매트릭스입니다

II. 매트릭스 만들기

– matrix( ) 함수 매개변수의 의미

- **1:20 :** 매트릭스에 저장될 값을 지정한다.
- **nrow=4 :** 매트릭스 행(row)의 개수를 의미한다.
- **ncol=5 :** 매트릭스 열(column)의 개수를 의미한다.

– 첫 번째 명령문은 1~20까지의 숫자값으로 4행 5열의 매트릭스를 만들어 z에 저장한다는 뜻

– 행 방향으로 값을 채우려면 다음과 같이 매개변수에 byrow =T를 추가함

```
> z2 <- matrix(1:20, nrow=4, ncol=5, byrow=T)
> z2
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
```

01. 이것이 매트릭스입니다

II. 매트릭스 만들기

[코드 5-2]

```
x <- 1:4                                          # 벡터 x 생성
y <- 5:8                                          # 벡터 y 생성
z <- matrix(1:20, nrow=4, ncol=5)                 # 매트릭스 z 생성

m1 <- cbind(x,y)      # x와 y를 열 방향으로 결합하여 매트릭스 생성
m1                     # 매트릭스 m1의 내용을 출력
m2 <- rbind(x,y)      # x와 y를 행 방향으로 결합하여 매트릭스 생성
m2                     # 매트릭스 m2의 내용을 출력
m3 <- rbind(m2,x)     # 매트릭스 m2와 벡터 x를 행 방향으로 결합
m3                     # 매트릭스 m3의 내용을 출력
m4 <- cbind(z,x)      # 매트릭스 z와 벡터 x를 열 방향으로 결합
m4                     # 매트릭스 m4의 내용을 출력
```

II. 매트릭스 만들기

[코드 5-2] 실행 결과 (1)

```
> x <- 1:4                                  # 벡터 x 생성
> y <- 5:8                                  # 벡터 y 생성
> z <- matrix(1:20, nrow=4, ncol=5)         # 매트릭스 z 생성
```

– 벡터 x, 벡터 y, 매트릭스 z를 생성

[코드 5-2] 실행 결과(2)

```
> m1 <- cbind(x,y)                          # x와 y를 열 방향으로 결합하여 매트릭스 생성
> m1                                        # 매트릭스 m1의 내용을 출력
     x  y
[1,] 1  5
[2,] 2  6
[3,] 3  7
[4,] 4  8
```

– m1에는 벡터 x와 벡터 y가 열 방향으로 묶인 2차원 매트릭스가 만들어짐

II. 매트릭스 만들기

[코드 5-2] 실행 결과(3)

```
> m2 <- rbind(x,y)    # x와 y를 행 방향으로 결합하여 매트릭스 생성
> m2                   # 매트릭스 m2의 내용을 출력
  [,1] [,2] [,3] [,4]
x    1    2    3    4
y    5    6    7    8
```

– m2에는 벡터 x와 벡터 y가 행 방향으로 묶인 2차원 매트릭스가 만들어짐

II. 매트릭스 만들기

[코드 5-2] 실행 결과(4)

```
> m3 <- rbind(m2,x)          # 매트릭스 m2와 벡터 x를 행 방향으로 결합
> m3                         # 매트릭스 m3의 내용을 출력
  [,1] [,2] [,3] [,4]
x    1    2    3    4
y    5    6    7    8
x    1    2    3    4
> m4 <- cbind(z,x)           # 매트릭스 z와 벡터 x를 열 방향으로 결합
> m4                         # 매트릭스 m4의 내용을 출력
                   x
[1,] 1 5  9 13 17 1
[2,] 2 6 10 14 18 2
[3,] 3 7 11 15 19 3
[4,] 4 8 12 16 20 4
```

01. 이것이 매트릭스입니다

III. 매트릭스에서 값의 추출

- 2차원 상에서 위치를 지정하려면 인덱스 2개가 필요
- '몇 번째 행, 몇 번째 열'을 지정하는 2개의 인덱스값이 필요

[코드 5-3]

```
z <- matrix(1:20, nrow=4, ncol=5)  # 매트릭스 z 생성
z                                  # 매트릭스 z의 내용 출력

z[2,3]                             # 2행 3열에 있는 값
z[1,4]                             # 1행 4열에 있는 값
z[2,]                              # 2행에 있는 모든 값
z[,4]                              # 4열에 있는 모든 값
```

III. 매트릭스에서 값의 추출

[코드 5-3] 실행 결과(1)

```
> z <- matrix(1:20, nrow=4, ncol=5)   # 매트릭스 z 생성
> z                                   # 매트릭스 z의 내용 출력
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

– 매트릭스 z를 생성하고 내용을 출력

III. 매트릭스에서 값의 추출

[코드 5-3] 실행 결과(2)

```
> z[2,3]                    # 2행 3열에 있는 값
[1] 10
```

– 여기서는 2가 행의 위치, 3이 열의 위치를 나타내는 인덱스값

[코드 5-3] 실행 결과(3)

```
> z[1,4]                    # 1행 4열에 있는 값
[1] 13
```

– z에서 1행 4열에 위치한 값을 지정하는 명령어

III. 매트릭스에서 값의 추출

[코드 5-3] 실행 결과(4)

```
> z[2,]          # 2행에 있는 모든 값
[1] 2 6 10 14 18
```

- 열의 위치를 지정하지 않으면 '모든 열의 값'을 의미
- 이 명령문은 2행에 있는 모든 열의 값을 뜻하게 됨

[코드 5-3] 실행 결과(5)

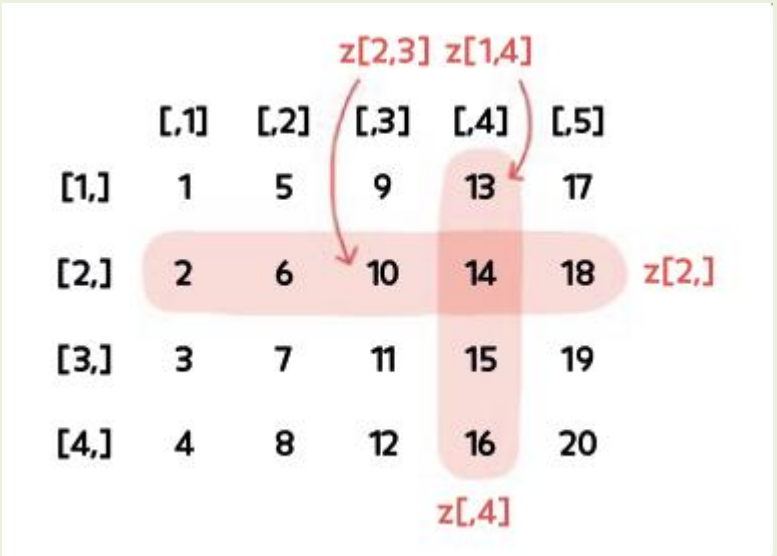
```
> z[,4]          # 4열에 있는 모든 값
[1] 13 14 15 16
```

- 이 명령문은 4열에 있는 모든 행의 값을 의미

III. 매트릭스에서 값의 추출

하나 더 알기

- [코드 5-3]에 있는 매트릭스 z에서 인덱스를 지정한 예를 정리



01. 이것이 매트릭스입니다

III. 매트릭스에서 값의 추출

[코드 5-4]

```
z <- matrix(1:20, nrow=4, ncol=5)  # 매트릭스 z 생성
z                                  # 매트릭스 z의 내용 출력

z[2,1:3]                        # 2행의 값 중 1~3열에 있는 값
z[1,c(1,2,4)]                   # 1행의 값 중 1, 2, 4열에 있는 값
z[1:2,]                         # 1~2행에 있는 모든 값
z[,c(1,4)]                      # 1, 4열에 있는 모든 값
```

III. 매트릭스에서 값의 추출

[코드 5-4] 실행 결과(1)

```
> z <- matrix(1:20, nrow=4, ncol=5)      # 매트릭스 z 생성
> z                                      # 매트릭스 z의 내용 출력
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
>
> z[2,1:3]                        # 2행의 값 중 1~3열에 있는 값
[1]  2  6 10
> z[1,c(1,2,4)]                   # 1행의 값 중 1, 2, 4열에 있는 값
[1]  1  5 13
```

III. 매트릭스에서 값의 추출

[코드 5-4] 실행 결과(2)

```
> z[1:2,]                                  # 1~2행에 있는 모든 값
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18

> z[,c(1,4)]                               # 1, 4열에 있는 모든 값
     [,1] [,2]
[1,]    1   13
[2,]    2   14
[3,]    3   15
[4,]    4   16
```

01. 이것이 매트릭스입니다

IV. 행과 열에 이름 붙이기

- 매트릭스 데이터에 행과 열에 이름을 붙이면 데이터를 이해하는 데에 도움이 되고, 이름을 이용해 값을 추출 수 있음

[코드 5-5]

```
score <- matrix(c(90,85,69,78,85,96,49,95,90,80,70,60), nrow=4)
score
rownames(score) <- c('John','Tom','Mark','Jane')
colnames(score) <- c('English','Math','Science')
score
```

IV. 행과 열에 이름 붙이기

[코드 5-5] 실행 결과(1)

```
> score <- matrix(c(90,85,69,78,85,96,49,95,90,80,70,60),
+ nrow=4)
> score
     [,1] [,2]  [,3]
[1,]   90   85    90
[2,]   85   96    80
[3,]   69   49    70
[4,]   78   95    60
```

– score라는 매트릭스를 생성한 후에 score의 내용을 출력

IV. 행과 열에 이름 붙이기

[코드 5-5] 실행 결과(2)

```
> rownames(score) <- c('John','Tom','Mark','Jane')
> colnames(score) <- c('English','Math','Science')
```

- 행과 열에 이름을 부여함
- rownames() : 행의 이름을 붙이거나 행의 이름을 출력할 때 이용
- colnames() : 열의 이름을 붙이거나 열의 이름을 출력할 때 이용

IV. 행과 열에 이름 붙이기

[코드 5-5] 실행 결과(3)

```
> score
       English  Math  Science
John        90    85       90
Tom         85    96       80
Mark        69    49       70
Jane        78    95       60
```

– 행과 열에 이름을 붙인 후 score를 출력

01. 이것이 매트릭스입니다

IV. 행과 열에 이름 붙이기

[코드 5-6]

```
score['John','Math']                  # John의 수학 성적
score['Tom',c('Math','Science')]      # Tom의 수학, 과학 성적
score['Mark',]                        # Mark의 모든 과목 성적
score[,'English']                     # 모든 학생의 영어 성적
rownames(score)                       # score의 행의 이름
colnames(score)                       # score의 열의 이름
colnames(score)[2]                    # score의 열의 이름 중 두 번째 값
```

IV. 행과 열에 이름 붙이기

[코드 5-6] 실행 결과

```
> score['John','Math']                  # John의 수학 성적
[1] 85
> score['Tom',c('Math','Science')]       # Tom의 수학, 과학 성적
  Math Science
    96      80
> score['Mark',]                        # Mark의 모든 과목 성적
English Math Science
     69   49      70
> score[,'English']                     # 모든 학생의 영어 성적
John Tom Mark Jane
  90  85   69   78
> rownames(score)                       # score의 행의 이름
[1] "John" "Tom" "Mark" "Jane"
> colnames(score)                       # score의 열의 이름
[1] "English" "Math" "Science"
> colnames(score)[2]                    # score의 열의 이름 중 두 번째 값
[1] "Math"
```

LAB. 햄버거 영양 성분 정보 제공하기

A사는 브랜드별 햄버거 영양 성분 정보를 제공하는 서비스 개발을 의뢰받았습니다. 각 브랜드에서 공개한 영양 성분이 다음 표와 같을 때 이 자료를 R에서 다룰 수 있도록 적절한 자료구조를 이용하여 저장해 보도록 하겠습니다.

브랜드	열량(kcal)	나트륨(na)	포화지방(fat)
M	514	917	11
L	533	853	13
B	566	888	10

1. 먼저 표 안에 있는 각 값들의 유형을 살펴봅니다. 모두 숫자형 자료로 동일한 유형임을 확인합니다.

LAB. 햄버거 영양 성분 정보 제공하기

2. 자료는 3개의 관측값과 3개의 변수로 구성되어 있으므로 3행 3열의 매트릭스를 생성합니다. 매트릭스를 저장할 변수명은 burger로 합니다. 자료를 입력하는 순서에 따라 매개변수에 적절한 값을 선택하는 것에 주의합니다.

```
burger <- matrix(c (514, 917, 11,
                    533, 853, 13,
                    566, 888, 10),
                    nrow = 3,
                    byrow = T)
```

LAB. 햄버거 영양 성분 정보 제공하기

3. 생성한 매트릭스를 출력해 확인합니다.

```
burger            # 매트릭스의 내용 확인
rownames(burger) <- c('M', 'L', 'B')
colnames(burger) <- c('kcal', 'na', 'fat')
burger            # 매트릭스의 내용 확인
```

4. 행과 열의 이름을 사용해 M사의 나트륨 함량, M사의 모든 영양 정보, 모든 브랜드의 칼로리 정보를 추출해봅니다.

```
burger['M', 'na’]          # M사의 나트륨 함량
burger['M', ]              # M사의 모든 영양 정보
burger[, 'kcal’ ]          # 모든 브랜드의 칼로리 정보
```

02

데이터프레임은 무엇인가요?

02. 데이터프레임은 무엇인가요?

- **데이터프레임(data frame) :** 매트릭스와 마찬가지로 2차원 형태의 데이터를 저장하고 분석하는데 사용되는 자료구조
- [그림 5-4](a)를 보면 저장된 모든 값들의 자료형이 숫자로 동일 (매트릭스)
- [그림 5-4](b)를 보면 키와 몸무게 열의 자료형은 숫자이고 성별 열에는 문자가 저장되어 있는데, 이렇게 숫자형 자료와 문자형 자료가 결합되어 있는 형태가 데이터프레임임

키	몸무게
168.4	62.4
169.5	65.3
172.1	59.8
185.2	46.5
173.7	49.8
175.2	58.7

(a) 매트릭스의 예

키	몸무게	성별
168.4	62.4	M
169.5	65.3	F
172.1	59.8	F
185.2	46.5	M
173.7	49.8	M
175.2	58.7	F

(b) 데이터프레임의 예

그림 5-4 매트릭스와 데이터프레임의 예

데이터프레임은 2개 이상의 자료형으로 구성된 2차원 자료를 저장하고 처리하는 데 사용됩니다.

02. 데이터프레임은 무엇인가요?

I. 데이터프레임 만들기

[코드 5-7]

```
city <- c("Seoul","Tokyo","Washington")          # 문자로 이루어진 벡터
rank <- c(1,3,2)                                 # 숫자로 이루어진 벡터
city.info <- data.frame(city, rank)              # 데이터프레임 생성
city.info                                        # city.info의 내용 출력
```

02. 데이터프레임은 무엇인가요?

I. 데이터프레임 만들기

[코드 5-7] 실행 결과

```
> city <- c("Seoul","Tokyo","Washington")          # 문자로 이루어진 벡터
> rank <- c(1,3,2)                                 # 숫자로 이루어진 벡터
> city.info <- data.frame(city, rank)              # 데이터프레임 생성
> city.info                                        # city.info의 내용 출력
            city rank
1          Seoul    1
2          Tokyo    3
3     Washington    2
```

- 문자로 이루어진 city라는 벡터와 숫자로 이루어진 rank라는 벡터 생성
- 두 개의 벡터를 data.frame( ) 함수로 묶어 city.info라는 데이터프레임을 생성
- 벡터들이 열 방향으로 결합됨

02. 데이터프레임은 무엇인가요?

II. iris 데이터셋

- iris는 150그루의 붓꽃에 대해 4개 분야의 측정 데이터와 품종 정보를 결합하여 만든 데이터셋
- iris의 내용을 출력하는 방법은 다음과 같이 iris라고 입력

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
...(생략)
150          5.9         3.0          5.1         1.8  virginica
```

02. 데이터프레임은 무엇인가요?

II. iris 데이터셋

표 5-1 iris 데이터셋

열 이름	의미	자료형(data type)
Sepal.Length	꽃받침의 길이	숫자형
Sepal.Width	꽃받침의 폭	숫자형
Petal.Length	꽃잎의 길이	숫자형
Petal.Width	꽃잎의 폭	숫자형
Species	붓꽃의 품종	문자형(팩터)

- 4개의 숫자형 열과 1개의 문자형 열이 결합 → 데이터프레임

02. 데이터프레임은 무엇인가요?

II. iris 데이터

[코드 5-8]

```
iris[,c(1:2)]                                  # 1~2열의 모든 데이터
iris[,c(1,3,5)]                                # 1, 3, 5열의 모든 데이터
iris[,c("Sepal.Length","Species")]             # 1, 5열의 모든 데이터
iris[1:5,]                                     # 1~5행의 모든 데이터
iris[1:5,c(1,3)]                  # 1~5행의 데이터 중 1, 3열의 데이터
```

02. 데이터프레임은 무엇인가요?

II. iris 데이터

[코드 5-8] 실행 결과(1)

```
> iris[,c(1:2)]                                    # 1~2열의 모든 데이터
   Sepal.Length Sepal.Width
1           5.1         3.5
2           4.9         3.0
...(생략)
150         5.9         3.0
> iris[,c(1,3,5)]                                   # 1, 3, 5열의 모든 데이터
   Sepal.Length Petal.Length   Species
1           5.1          1.4    setosa
2           4.9          1.4    setosa
...(생략)
150         5.9          5.1 virginica
> iris[,c("Sepal.Length","Species")]      # 1, 5열의 모든 데이터
    Sepal.Length  Species
1           5.1    setosa
```

II. iris 데이터

[코드 5-8] 실행 결과(2)

```
...(생략)
> iris[1:5,]                          # 1~5행의 모든 데이터
   Sepal.Length Sepal.Width Petal.Length Petal.Width  Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
...(생략)
> iris[1:5,c(1,3)]                    # 1~5행의 데이터 중 1, 3열의 데이터
   Sepal.Length Petal.Length
1           5.1          1.4
2           4.9          1.4
3           4.7          1.3
4           4.6          1.5
5           5.0          1.4
```

LAB. 햄버거 영양 성분 정보 추가하기

이전 LAB에서 브랜드별 햄버거 영양 성분 정보를 제공하는 서비스에 대한 자료를 매트릭스에 저장하였습니다. 이번에는 메뉴명 정보를 추가하여 기능을 확대하려고 합니다. 데이터를 어떻게 저장하는 것이 좋은지 프로그램을 작성하며 알아봅니다.

브랜드	열량(kcal)	나트륨(na)	포화지방(fat)	메뉴명
M	514	917	11	새우
L	533	853	13	불고기
B	566	888	10	치킨

1. 숫자형과 문자형 자료가 같이 있습니다. 따라서 이 경우 매트릭스는 사용할 수 없기 때문에 데이터프레임을 이용합니다.

LAB. 햄버거 영양 성분 정보 추가하기

2. 데이터프레임을 만들기 위해 각 열의 내용을 저장할 벡터를 생성합니다.

```
kcal <- c(514, 533, 566)
na <- c(917, 853, 888)
fat <- c(11, 13, 10)
menu <- c('새우', '불고기', '치킨')
```

3. 각 벡터를 묶어 burger라는 데이터프레임을 만들고 내용을 확인합니다.

```
burger <- data.frame(kcal, na, fat, menu)
burger
```

LAB. 햄버거 영양 성분 정보 추가하기

4. 데이터프레임의 행 이름을 각 브랜드명으로 설정합니다.

```
rownames(burger) <- c('M', 'L', 'B')
```

5. 행과 열의 이름을 사용해 M사의 나트륨 함량, M사의 모든 영양 정보, 모든 브랜드의 칼로리 정보, M과 B사의 메뉴명을 추출합니다.

```
burger['M', 'na’]                    # M사의 나트륨 함량
burger['M', ]                        # M사의 모든 영양 정보
burger[, 'kcal’ ]                    # 모든 브랜드의 칼로리 정보
burger[c('M','B'),'menu’]            # M과 B사의 menu 정보
```

03
매트릭스와 데이터프레임을 다루어볼까요?

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-9]

```
dim(iris)          # 행과 열의 개수 보이기
nrow(iris)         # 행의 개수 보이기
ncol(iris)         # 열의 개수 보이기
colnames(iris)     # 열 이름 보이기, names() 함수와 결과 동일
head(iris)         # 데이터셋의 앞부분 일부 보기
tail(iris)         # 데이터셋의 뒷부분 일부 보기
```

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-9] 실행 결과(1)

```
> dim(iris)                 # 행과 열의 개수 보이기
[1] 150 5
```

– iris 데이터셋은 150행 5열의 크기를 갖는 데이터프레임

[코드 5-9] 실행 결과(2)

```
> nrow(iris)                # 행의 개수 보이기
[1] 150
> ncol(iris)                # 열의 개수 보이기
[1] 5
```

– nrow( ) 함수는 행의 개수를, ncol( ) 함수는 열의 개수를 알려주는 함수

I. 데이터셋의 기본 정보 알아보기

[코드 5-9] 실행 결과(3)

```
> colnames(iris)          # 열 이름 보이기, names() 함수와 결과 동일
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

- colnames( ) 함수는 데이터셋에 있는 열의 이름을 출력하는 함수

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-9] 실행 결과(4)

```
> head(iris)     # 데이터셋의 앞부분 일부 보기
  Sepal.Length Sepal.Width Petal.Length Petal.Width  Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

- head( ) 함수는 자주 이용하는 함수
- 데이터셋에서 시작 부분에 있는 일부 자료(보통 1~6행)의 내용을 보여줌

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-9] 실행 결과(5)

```
> tail(iris)     # 데이터셋의 뒷부분 일부 보기
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
145          6.7         3.3          5.7         2.5 virginica
146          6.7         3.0          5.2         2.3 virginica
147          6.3         2.5          5.0         1.9 virginica
148          6.5         3.0          5.2         2.0 virginica
149          6.2         3.4          5.4         2.3 virginica
150          5.9         3.0          5.1         1.8 virginica
```

– tail( ) 함수는 데이터셋의 끝부분에 있는 자료 중 일부를 보여주는 함수

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-10]

```
str(iris)                    # 데이터셋 요약 정보 보기
iris[,5]                     # 품종 데이터 보기
levels(iris[,5])             # 품종의 종류 보기(중복 제거)
table(iris[,"Species"])      # 품종의 종류별 행의 개수 세기
```

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-10] 실행 결과(1)

```
> str(iris)                    # 데이터셋 요약 정보 보기
'data.frame':150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1..
```

– str( ) 함수는 데이터셋의 전반적인 정보를 알아냄

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-10] 실행 결과(1)

– str( ) 함수 매개변수 설명

- **'data.frame' : 150 obs. of 5 variables**
  data.frame은 iris의 형태가 데이터프레임인 것을 나타냅니다. 150 obs는 150개의 행을 포함하고 있음을 나타냅니다. 5 variables는 열이 5개 있음을 의미합니다.
- **$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...**
  5개의 열 중 1번째 열의 이름이 Sepal.Length이고, 저장된 자료형은 num입니다. 5.1 4.9 4.7 등은 Sepal.Length에 저장된 값을 나타냅니다.
- **$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ..**
  5개의 열 중 5번째 열 이름이 Species이고 자료형은 Factor로 문자형을 의미합니다. w/ 3 levels는 'with 3 levels'의 약자로 3가지 종류의 품종이 있다는 것을 나타내며, 각 품종의 이름으로 "setosa", "versicolor" 등이 있다는 것을 알려줍니다. 1 1 1은 품종의 이름을 숫자로 표현한 것입니다.

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-10] 실행 결과(2)

```
> iris[,5]                                  # 품종 데이터 보기
 [1] setosa setosa setosa setosa setosa setosa
...(생략)
 [97] versicolor versicolor versicolor versicolor virginica virginica
...(생략)
[145] virginica virginica virginica virginica virginica virginica
Levels: setosa versicolor virginica
> levels(iris[,5])                          # 품종의 종류 보기(중복 제거)
[1] "setosa" "versicolor" "virginica"
```

- iris 데이터셋의 5번째 열의 정보인 붓꽃의 품종(Species) 정보를 출력
- levels( ) 함수는 팩터 자료에 중복된 값은 제거하고 값 종류를 알 수 있는 함수

03. 매트릭스와 데이터프레임을 다루어볼까요?

I. 데이터셋의 기본 정보 알아보기

[코드 5-10] 실행 결과(3)

```
> table(iris[,"Species"])          # 품종의 종류별 행의 개수 세기
    setosa versicolor  virginica
        50         50         50
```

– table( ) 함수는 그룹을 나타내는 값이 포함된 자료에서 각 그룹별로 몇 개의 관측값이 존재하는지 알려주는 기능

03. 매트릭스와 데이터프레임을 다루어볼까요?

II. 매트릭스와 데이터프레임에 함수 적용

- **행별, 열별로 합계와 평균 계산하기**

[코드 5-11]

```
colSums(iris[,-5])                  # 열별 합계
colMeans(iris[,-5])                 # 열별 평균
rowSums(iris[,-5])                  # 행별 합계
rowMeans(iris[,-5])                 # 행별 평균
```

II. 매트릭스와 데이터프레임에 함수 적용

- 행별, 열별로 합계와 평균 계산하기

[코드 5-11] 실행 결과(1)

```
> colSums(iris[,-5])                # 열별 합계
Sepal.Length Sepal.Width Petal.Length Petal.Width
       876.5       458.6        563.7       179.9
> colMeans(iris[,-5])               # 열별 평균
Sepal.Length Sepal.Width Petal.Length Petal.Width
    5.843333    3.057333     3.758000    1.199333
> rowSums(iris[,-5])                # 행별 합계
  [1] 10.2 9.5  9.4   9.4 10.2 11.4  9.7 10.1  8.9  9.6 10.8 10.0
 [13]  9.3 8.5 11.2  12.0 11.0 10.3 11.5 10.7 10.7 10.7  9.4 10.6
...(생략)
[133] 17.0 15.7 15.7 19.1 17.7 16.8 15.6 17.5 17.8 17.4 15.5 18.2
[145] 18.2 17.2 15.7 16.7 17.3 15.8
```

II. 매트릭스와 데이터프레임에 함수 적용

- **행별, 열별로 합계와 평균 계산하기**

[코드 5-11] 실행 결과(2)

```
> rowMeans(iris[,-5]) # 행별 평균
  [1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400
 [11] 2.700 2.500 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675
...(생략)
[131] 4.550 5.025 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375
[141] 4.450 4.350 3.875 4.550 4.550 4.300 3.925 4.175 4.325 3.950
```

II. 매트릭스와 데이터프레임에 함수 적용

▪ 행과 열의 방향 변환하기

- 4행 5열의 매트릭스를 5행 4열의 매트릭스로 변환(transpose)하는 경우에 사용하는 함수가 t( ) 함수

[코드 5-12]

```
z <- matrix(1:20, nrow=4, ncol=5)
z
t(z)                # 행과 열 방향 변환
```

II. 매트릭스와 데이터프레임에 함수 적용

- 행과 열의 방향 변환하기

[실행 결과]

```
> z <- matrix(1:20, nrow=4, ncol=5)
> z
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> t(z)            # 행과 열 방향 변환
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20
```

II. 매트릭스와 데이터프레임에 함수 적용

- 조건에 맞는 행과 열의 값 추출하기

[코드 5-13]

```
IR.1 <- subset(iris, Species=='setosa')
IR.1
IR.2 <- subset(iris, Sepal.Length>5.0 &
                  Sepal.Width>4.0)
IR.2
IR.2[, c(2,4)]            # 2열과 4열의 값만 추출
```

II. 매트릭스와 데이터프레임에 함수 적용

- 조건에 맞는 행과 열의 값 추출하기

[코드 5-13] 실행 결과(1)

```
> IR.1 <- subset(iris, Species=='setosa')
> IR.1
  Sepal.Length Sepal.Width Petal.Length   Petal.Width   Species
1          5.1         3.5          1.4           0.2    setosa
2          4.9         3.0          1.4           0.2    setosa
3          4.7         3.2          1.3           0.2    setosa
 ...(생략)
49         5.3         3.7          1.5           0.2    setosa
50         5.0         3.3          1.4           0.2    setosa
```

– subset() 함수는 전체 데이터에서 조건에 맞는 행들만 추출

II. 매트릭스와 데이터프레임에 함수 적용

▪ 조건에 맞는 행과 열의 값 추출하기

[코드 5-13] 실행 결과(1)

– subset() 함수 매개변수

- **iris :** 데이터를 추출하는 대상이 iris 데이터셋
- **Species=='setosa' :** 데이터를 추출할 조건을 지정하는 부분으로, 품종 열의 값이 'setosa'인 행만 추출하라는 의미.

– IR.1에는 iris의 전체 150개의 행 중 조건에 맞는 50개의 행만 저장

II. 매트릭스와 데이터프레임에 함수 적용

- 조건에 맞는 행과 열의 값 추출하기

[코드 5-13] 실행 결과(2)

```
> IR.2 <- subset(iris, Sepal.Length>5.0 &
+ Sepal.Width>4.0)
> IR.2
   Sepal.Length Sepal.Width Petal.Length  Petal.Width  Species
16          5.7         4.4          1.5          0.4   setosa
33          5.2         4.1          1.5          0.1   setosa
34          5.5         4.2          1.4          0.2   setosa
```

– 'Sepal.Length 값이 5.0보다 크고 Sepal.Width 값이 4.0보다 큰 행들을 추출

II. 매트릭스와 데이터프레임에 함수 적용

- 조건에 맞는 행과 열의 값 추출하기

하나 더 알기 **명령문의 줄바꿈**

일반적으로 하나의 명령문은 한 줄에 작성하지만, 명령문이 길어지면 여러 줄에 걸쳐서 작성할 수 있습니다. 그렇다면 어느 부분에서 줄바꿈을 하면 좋을지의 문제가 생기는데, 콤마(,) and(&), or(|) 등과 같이 다음에 어떤 내용이 따라온다는 것을 암시하는 기호 뒤에서 줄바꿈을 하는 것이 좋습니다.

II. 매트릭스와 데이터프레임에 함수 적용

- 조건에 맞는 행과 열의 값 추출하기

[코드 5-13] 실행 결과(3)

```
> IR.2[, c(2,4)]                    # 2열과 4열의 값만 추출
   Sepal.Width Petal.Width
16         4.4         0.4
33         4.1         0.1
34         4.2         0.2
```

– IR.2에서 2열과 4열의 값들만 추출하는 작업을 수행

II. 매트릭스와 데이터프레임에 함수 적용

▪ 매트릭스와 데이터프레임에 산술연산 적용하기

- 숫자로 구성된 매트릭스나 데이터프레임에 대해서도 같은 원리로 산술연산을 적용할 수 있음

[코드 5-14]

```
a <- matrix(1:20,4,5)
b <- matrix(21:40,4,5)
a
b

2*a                 # 매트릭스 a에 저장된 값들에 2를 곱하기

a+b

a <- a*3
b <- b-5
```

II. 매트릭스와 데이터프레임에 함수 적용

- 매트릭스와 데이터프레임에 산술연산 적용하기

[코드 5-14] 실행 결과(1)

```
> a <- matrix(1:20,4,5)
> b <- matrix(21:40,4,5)
> a
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> b
     [,1] [,2] [,3] [,4] [,5]
[1,]   21   25   29   33   37
[2,]   22   26   30   34   38
[3,]   23   27   31   35   39
[4,]   24   28   32   36   40
```

– 매트릭스 a와 b를 생성한 뒤 a와 b의 내용을 출력하는 명령문

II. 매트릭스와 데이터프레임에 함수 적용

▪ 매트릭스와 데이터프레임에 산술연산 적용하기

[코드 5-14] 실행 결과(2)

```
> 2*a             # 매트릭스 a에 저장된 값들에 2를 곱하기
     [,1] [,2] [,3] [,4] [,5]
[1,]    2   10   18   26   34
[2,]    4   12   20   28   36
[3,]    6   14   22   30   38
[4,]    8   16   24   32   40
```

- 매트릭스 a에 2를 곱하는 명령문으로 a에 저장된 모든 값들에 2가 곱해짐
- 매트릭스에 대한 산술연산은 매트릭스 안에 저장된 값들에 대한 연산으로 바꾸어 실행

II. 매트릭스와 데이터프레임에 함수 적용

▪ 매트릭스와 데이터프레임에 산술연산 적용하기

[코드 5-14] 실행 결과(3)

```
> a+b
    [,1] [,2] [,3] [,4] [,5]
[1,]   22   30   38   46   54
[2,]   24   32   40   48   56
[3,]   26   34   42   50   58
[4,]   28   36   44   52   60
```

- 매트릭스 a와 매트릭스 b를 더하는 연산은 두 개의 매트릭스 상에서 동일 위치에 있는 값들 간에 더하는 연산으로 바뀌어 실행
- 매트릭스 간 산술연산을 하려면 두 개의 매트릭스 크기(행과 열의 개수)가 같아야 함

II. 매트릭스와 데이터프레임에 함수 적용

▪ 매트릭스와 데이터프레임에 산술연산 적용하기

[코드 5-14] 실행 결과(4)

```
> a <- a*3
> b <- b-5
```

– 이 명령을 실행한 후 a, b를 출력하면 a, b에 저장된 값이 변경된 것을 확인할 수 있음

III. 매트릭스와 데이터프레임의 자료구조 확인과 변환

- 매트릭스와 데이터프레임의 자료구조 확인하기

[코드 5-15]

```
class(iris)                 # iris 데이터셋의 자료구조 확인
class(state.x77)            # state.x77 데이터셋의 자료구조 확인
is.matrix(iris)             # 데이터셋이 매트릭스인지 확인하는 함수
is.data.frame(iris)         # 데이터셋이 데이터프레임인지 확인하는 함수
is.matrix(state.x77)
is.data.frame(state.x77)
```

III. 매트릭스와 데이터프레임의 자료구조 확인과 변환

▪ 매트릭스와 데이터프레임의 자료구조 확인하기

[코드 5-15] 실행 결과(1)

```
> class(iris)              # iris 데이터셋의 자료구조 확인
[1] "data.frame"
> class(state.x77)         # state.x77 데이터셋의 자료구조 확인
[1] "matrix" "array"
```

– iris는 데이터프레임이고, state.x77은 기본적으로는 매트릭스이면서 상위 개념인 배열(array)에도 속함

III. 매트릭스와 데이터프레임의 자료구조 확인과 변환

▪ 매트릭스와 데이터프레임의 자료구조 확인하기

[코드 5-15] 실행 결과(2)

```
> is.matrix(iris)          # 데이터셋이 매트릭스인지 확인하는 함수
[1] FALSE
> is.data.frame(iris)      # 데이터셋이 데이터프레임인지 확인하는 함수
[1] TRUE
> is.matrix(state.x77)
[1] TRUE
> is.data.frame(state.x77)
[1] FALSE
```

– is.matrix() 함수와 is.data.frame() 함수를 이용하면 매트릭스인지, 또는 데이터프레임인지의 여부를 각각 확인할 수도 있음

III. 매트릭스와 데이터프레임의 자료구조 확인과 변환

- 매트릭스와 데이터프레임의 자료구조 변환하기

[코드 5-16]

```
# 매트릭스를 데이터프레임으로 변환
is.matrix(state.x77)
st <- data.frame(state.x77)
head(st)
class(st)

# 데이터프레임을 매트릭스로 변환
is.data.frame(iris[,1:4])
iris.m <- as.matrix(iris[,1:4])
head(iris.m)
class(iris.m)
```

III. 매트릭스와 데이터프레임의 자료구조 확인과 변환

- 매트릭스와 데이터프레임의 자료구조 변환하기

[코드 5-16] 실행 결과(1)

```
> # 매트릭스를 데이터프레임으로 변환
> is.matrix(state.x77)
[1] TRUE
> st <- data.frame(state.x77)
> head(st)
           Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
> class(st)
[1] "data.frame"
```

III. 매트릭스와 데이터프레임의 자료구조 확인과 변환

- 매트릭스와 데이터프레임의 자료구조 변환하기

[코드 5-16] 실행 결과(2)

```
> # 데이터프레임을 매트릭스로 변환
> is.data.frame(iris[,1:4])
[1] TRUE
> iris.m <- as.matrix(iris[,1:4])
> head(iris.m)
     Sepal.Length  Sepal.Width   Petal.Length Petal.Width
[1,]          5.1          3.5          1.4         0.2
[2,]          4.9          3.0          1.4         0.2
[3,]          4.7          3.2          1.3         0.2
[4,]          4.6          3.1          1.5         0.2
[5,]          5.0          3.6          1.4         0.2
[6,]          5.4          3.9          1.7         0.4
> class(iris.m)
[1] "matrix" "array
```

데이터프레임인 iris에서 문자로 된 열은 제외하고 숫자로 된 부분만 떼어서 매트릭스로 변환

03. 매트릭스와 데이터프레임을 다루어볼까요?

IV. 데이터프레임에만 적용되는 열 추출 방법

[코드 5-17]

```
iris[,"Species"]     # 결과가 벡터-매트릭스, 데이터프레임 모두 가능
iris[,5]             # 결과가 벡터-매트릭스, 데이터프레임 모두 가능
iris["Species"]      # 결과가 데이터프레임-데이터프레임만 가능
iris[5]              # 결과가 데이터프레임-데이터프레임만 가능
iris$Species         # 결과가 벡터-데이터프레임만 가능
```

IV. 데이터프레임에만 적용되는 열 추출 방법

[코드 5-17] 실행 결과(1)

```
> iris[,"Species"]   # 결과가 벡터-매트릭스, 데이터프레임 모두 가능
  [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa
  [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa
 ...(생략)
[141] virginica virginica virginica virginica virginica virginica virginica
[148] virginica virginica virginica
Levels: setosa versicolor virginica
> iris[,5]                    # 결과가 벡터-매트릭스, 데이터프레임 모두 가능
 [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa
 ...(생략)
[141] virginica virginica virginica virginica virginica virginica virginica
[148] virginica virginica virginica
Levels: setosa versicolor virginica
```

– 특정 열 데이터 추출 방법은 열의 이름을 지정하거나 인덱스 번호를 지정하는 것

IV. 데이터프레임에만 적용되는 열 추출 방법

[코드 5-17] 실행 결과(2)

```
> iris["Species"]          # 결과가 데이터프레임-데이터프레임만 가능
      Species
1      setosa
2      setosa
 ...(생략)
> iris[5]                  # 결과가 데이터프레임-데이터프레임만 가능
      Species
1      setosa
2      setosa
 ...(생략)
```

- 인덱스 부분에 2개의 값이 아닌 1개의 값만 지정함.
- iris[,5]의 결과는 값의 개수가 150개인 벡터, iris[5]의 결과는 자료의 크기가 150×1(150행 1열)인 데이터프레임

IV. 데이터프레임에만 적용되는 열 추출 방법

- **[코드 5-17] 실행 (3)**

```
> iris$Species        # 결과가 벡터-데이터프레임만 가능
  [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa
  [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa
 ...(생략)
[148] virginica virginica virginica
Levels: setosa versicolor virginica
```

– 데이터셋 이름 다음에 $를 붙이고 이어서 열의 이름을 따옴표 없이 붙이는 방법은 데이터프레임에만 적용되며, 실행 결과는 벡터

LAB. 벚나무 판매하기

R 화원에서는 벚나무를 판매하고 있습니다. 어느 날 다음 [조건]을 모두 만족하는 벚나무가 있다면 벚나무를 구매하겠다는 고객이 나타났습니다. R 화원은 고객에게 모두 몇 그루의 벚나무를 판매할 수 있는지 알려주는 프로그램을 작성해보겠습니다.

[조건]

① 직경(Girth)은 화원에서 보유한 벚나무의 평균보다 커야 한다.

② 높이(Height)는 80보다 커야 한다.

③ 부피(Volume)는 50보다 커야 한다.

LAB. 벚나무 판매하기

1. 우선 R 화원에서 보유하고 있는 데이터셋을 확인합니다.

```
class(trees)
str(trees)
```

2. 조건 ①을 만족시키기 위해 먼저 벚나무 직경의 평균값을 구하여 girth.mean에 저장합니다.

```
girth.mean <- mean(trees$Girth)
```

3. 3가지 조건에 해당하는 벚나무를 찾아 변수 candidate에 저장하고 자료의 개수를 구합니다.

```
candidate <- subset(trees, Girth >= girth.mean & Height > 80 &
                    Volume > 50)
candidate
nrow(candidate)
```

실전분석. 종업원의 팁 계산하기

[문제]

reshape2 패키지 안에 들어있는 tips 데이터셋은 한 종업원이 식당에서 일하면서 몇 달 동안 받은 각각의 팁에 대한 자료입니다. 이에 대해 분석해보겠습니다.

[해결]

1. **tips 데이터셋을 불러옵니다. tips 데이터셋은 reshape2 패키지 안에 들어있기 때문에 패키지를 설치하고 로드하는 과정이 필요합니다.**

```
install.packages('reshape2')
library(reshape2)
tips
```

실전분석. 종업원의 팁 계산하기

2. tips의 자료구조가 매트릭스인지 확인합니다. 확인 결과 tips의 자료구조는 데이터 프레임입니다.

```
> is.matrix(tips)
[1] FALSE
> class(tips)
[1] "data.frame"
```

3. 처음 6개의 관측값을 출력합니다.

```
> head(tips)
  total_bill   tip    sex smoker day   time size
1      16.99  1.01 Female     No Sun Dinner    2
2      10.34  1.66   Male     No Sun Dinner    3
3      21.01  3.50   Male     No Sun Dinner    3
4      23.68  3.31   Male     No Sun Dinner    2
5      24.59  3.61 Female     No Sun Dinner    4
6      25.29  4.71   Male     No Sun Dinner    4
```

실전분석. 종업원의 팁 계산하기

4. **str() 함수를 사용하여 데이터 구조를 알아봅니다.**

```
> str(tips)
'data.frame':244 obs. of 7 variables:
 $ total_bill: num 17 10.3 21 23.7 24.6 ...
 $ tip       : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...
 $ sex       : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...
 $ smoker    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ day       : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ time      : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...
 $ size      : int 2 3 3 2 4 4 2 4 2 2 ...
```

- 데이터셋은 244개의 관측값과 7개의 열로 구성되었음
- sex, smoker, day, time 열은 팩터형

실전분석. 종업원의 팁 계산하기

5. tips 데이터셋에서 요일(day)별 팁을 받는 빈도를 구합니다.

```
> table(tips$day)
Fri Sat Sun Thur
 19 87  76    62
```

6. 요일별로 시간대(time)가 'Dinner'인 경우와 'Lunch'인 경우로 나누어 팁의 빈도에 차이가 있는지 알아봅니다.

```
> dinner <- subset(tips, time == 'Dinner')
> lunch <- subset(tips, time == 'Lunch')
>
> table(dinner$day)
Fri Sat Sun Thur
 12 87  76     1
> table(lunch$day)
Fri Sat Sun Thur
  7   0   0   61
```

실전분석. 종업원의 팁 계산하기

7. 앞서 만든 dinner와 lunch의 데이터셋을 이용하여 결제 금액(total_bill), 팁(tip), 일행 수(size)에 대한 평균을 구합니다.

```
> colMeans(dinner[c('total_bill', 'tip', 'size')])
total_bill      tip     size
 20.797159 3.102670 2.630682
> colMeans(lunch[c('total_bill', 'tip', 'size')])
total_bill      tip     size
 17.168676 2.728088 2.411765
```

8. 손님들은 결제 금액 대비 평균적으로 몇 퍼센트(%)를 팁으로 주었는지 알아봅니다.

```
> tip.rate <- tips$tip/tips$total_bill    # 손님별 팁의 비율
> mean(tip.rate)                          # 평균 팁의 비율
[1] 0.1608026
```

Thank You !

한빛아카데미 Hanbit Academy, Inc.