프로그래밍 언어의 기초

Vector, Matrix

강 범 일
(kangbeomil@gmail.com)

여러 값을 저장하는 데이터 타입

- 100명의 성적을 계산하기 위해 100개의 변수를 일일이 생성해야 할까?

```
stu001 <- 30
stu002 <- 50
stu003 <- 70
stu004 <- 40
…
stu100 <- 70
```

→ 여러 데이터를 한꺼번에 저장하는 데이터 타입 필요

여러 값을 저장하는 데이터 타입

- 여러 값을 저장할 수 있는 대표적인 4가지 데이터 타입

	동일한 타입 저장	다양한 타입 저장
1-dimension	Vector	List
2-dimension	Matrix	Data frame

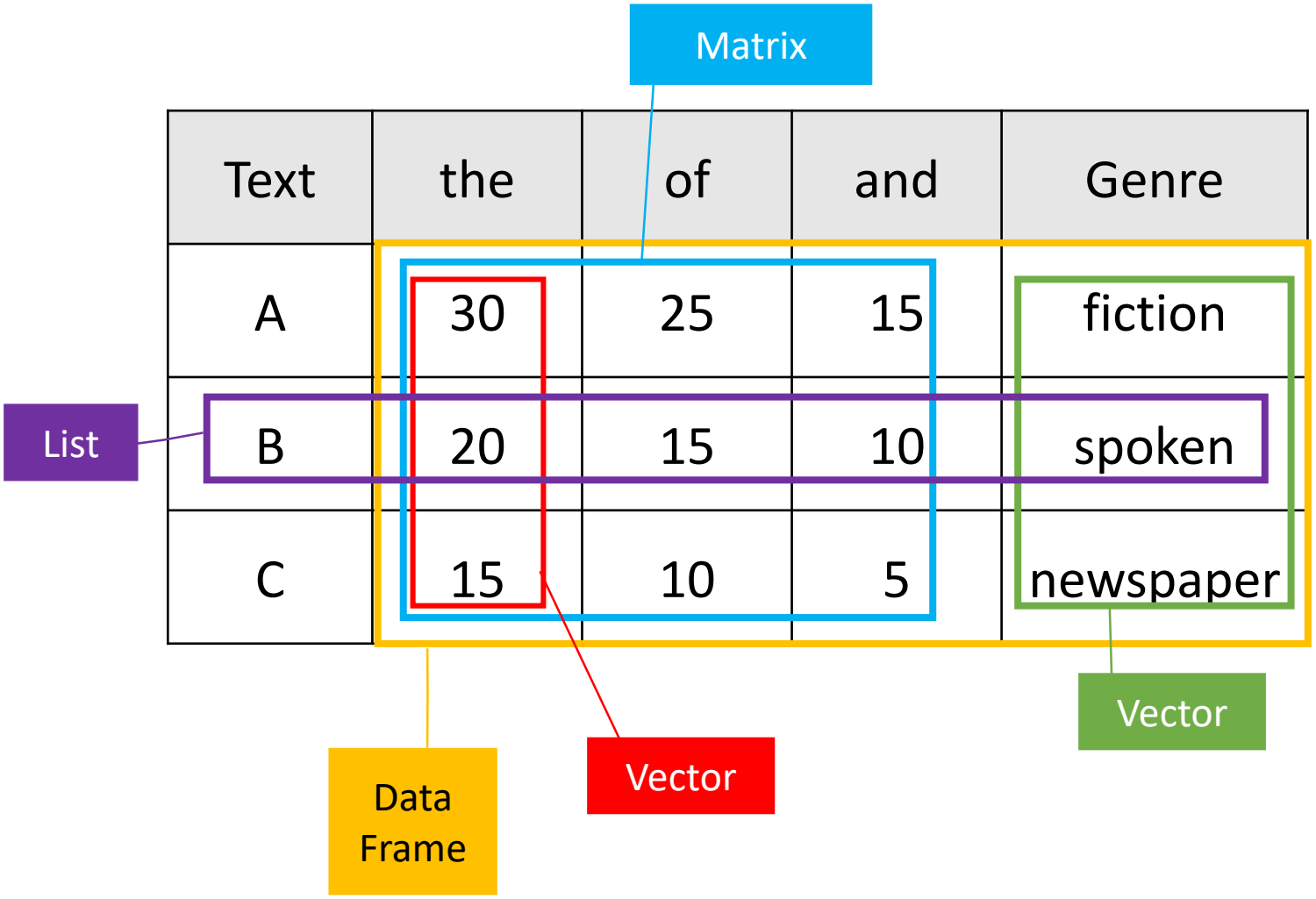
  - Vector와 List는 1차원의 자료 저장 가능
  - Matrix와 Data Frame은 2차원의 자료 저장 가능
  - Vector와 Matrix는 동일한 타입의 데이터만 저장 가능
  - List와 Data Frame은 다양한 타입의 데이터 저장 가능

여러 값을 저장하는 데이터 타입

Text	the	of	and	Genre
A	30	25	15	fiction
B	20	15	10	spoken
C	15	10	5	newspaper

- 표의 해석
  - 텍스트 A에서 ‘the’의 출현 빈도는 30
  - 텍스트 B의 장르는 구어

여러 값을 저장하는 데이터 타입



Vector

- 1차원의 행으로 묶인 동일한 유형의 데이터 저장
- c( ) 함수를 이용하여 여러 값을 하나의 벡터로 결합(combine)할 수 있음

```
the <- c(30, 20, 15)
print(the)

of <- c(25, 15, 10)
print(of)

theOf <- c('the', 'of')
print(theOf)

tot = sum(the)
print(tot)
```

```
# 숫자형과 문자형이 섞여 있으면 전체를 문자형으로 변환
the <- c(30, "20", 15)
print(the)

print(sum(the))
Error in sum(the) : invalid 'type' (character) of
argument

# 숫자형과 논리형 섞여 있으면 전체를 숫자형으로 변환
# TRUE = 1, FALSE = 0
of <- c(25, T, FALSE)
print(of)
```

* sum( ) 함수: 넘긴 값들의 합을 반환

Vector

- 수열(sequence)로 이루어진 벡터 생성

  - 콜론(:) 연산자를 이용

```
x <- 1:5   # 5개 원소(1,2,3,4,5)를 가지는 벡터 생성
print(x)

y <- 10:5  # 6개의 원소(10,9,8,7,6,5)를 가지는 벡터 생성
print(y)
```

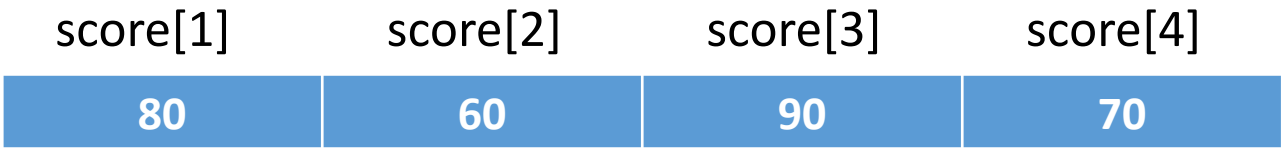
  - 벡터 생성 함수 seq( )

```
x = seq(from=1, to=20, by=2)  # 1부터 20까지 2씩 증가하는 수열 벡터
                              # parameter(매개변수)인 from, to 등 생략 가능
x = seq(1, 5, 0.5) # 1부터 5까지 0.5씩 증가하는 수열 벡터

x = seq(1, 7, length.out=4) # 1부터 7까지 4개의 자료 생성, length.out 생략 불가
```

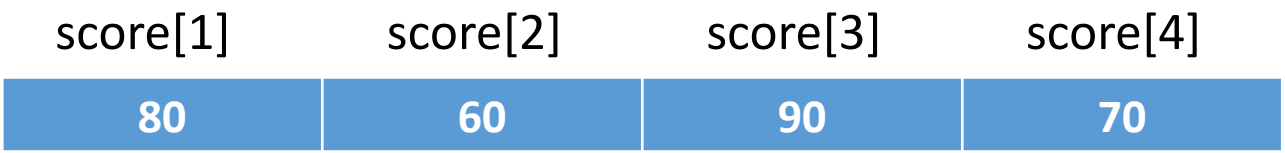
Vector의 접근

- “score 〈- c(80, 60, 90, 70)”를 통해 생성된 score의 각 원소는 다음과 같이 인덱스 번호를 이용해 참조 가능




```
score <- c(80, 60, 90, 70)
res = score[1] + score[3]
print(res)
-----------------------------------
[1] 170
```

Vector의 접근



- 콜론 기호를 이용해 연속된 원소에 접근

```
score[2:4]  # 2~4번째 원소 접근
```

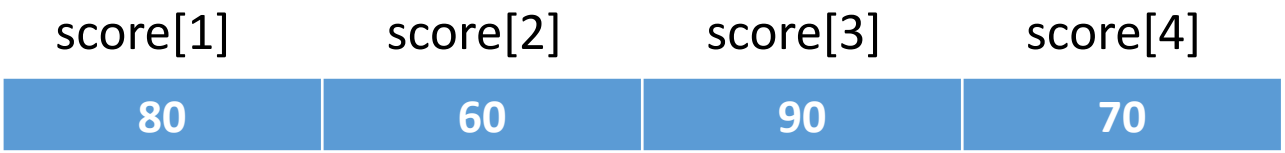
- 인덱스값으로 구성된 벡터를 이용하여 여러 원소에 접근

```
score[c(1,2,4)]  # 1, 2, 4번째 원소 접근
```

- 마이너스 기호를 사용하여 특정 원소를 제외한 나머지에 접근

```
score[-3]  # 3번째 원소 제외한 나머지
score[-c(1,2,4)]  # 1, 2, 4번째 원소 제외한 나머지
```

Vector의 수정




```
score[2:4]  = c(20,30,40) # 2~4번째 원소 수정
```

```
score[c(1,2,4)] = c(20,30,40)  # 1, 2, 4번째 원소 수정
```

```
score[-2]  = 100  # 2번째 원소 제외한 나머지를 수정
```

Vector의 연산

```
a1 <- c(2,3,4,5)          a3 <- c(7,8,9)
a2 <- c(4,5,6,7)          a4 <- c(2,3)
```

- 두 벡터의 길이가 같을 때의 연산 → 동일한 위치의 원소끼리 연산

```
a1 + a2   # 2+4, 3+5, 4+6, 5+7
```

- 두 벡터의 길이가 다를 때의 연산 → recycling rule 적용

```
a1 + a4   # 2+2, 3+3, 4+2, 5+3
a1 + a3   # 2+7, 3+8, 4+9, 5+7
```

- 벡터와 단일 값의 연산 → 모든 원소와 단일 값의 연산을 각각 수행

```
a1 + 10  # 2+10, 3+10, 4+10, 5+10
```

Vector의 연산

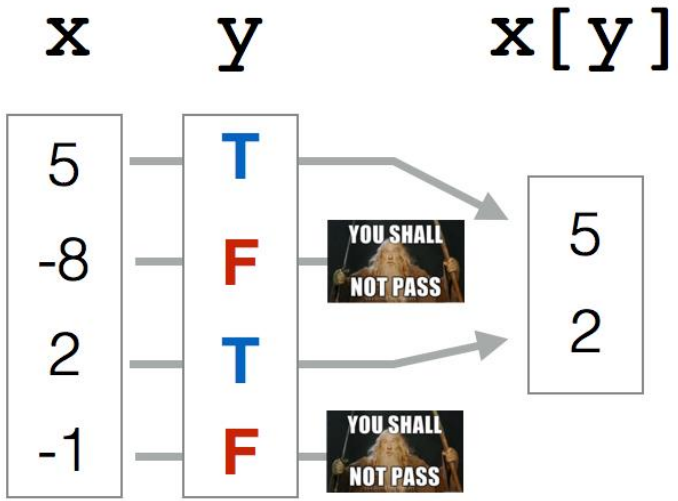
- 함수를 이용한 연산
  - min( ) : 넘겨진 값들 중 가장 작은 값 반환
  - max( ) : 넘겨진 값들 중 가장 큰 값 반환
  - mean( ) : 넘겨진 값들의 평균 반환
  - length( ) : 넘겨진 값들의 개수 반환

Exercise

- 1부터 100까지의 합 구하기: seq( ), sum( ) 이용
- 1부터 100 사이의 짝수로 구성된 벡터 생성하기: seq() 이용
- 55부터 100 사이의 수 중 23번째 값 구하기
- 33부터 103 사이의 수 중 3, 17, 29번째 값을 제외한 나머지를 출력하기
- 1부터 100까지의 값 중 1의 자리의 수에 300을 더한 벡터 출력
  - 출력 결과: 301,302,…,309, 10, 11, …100

Logical Indexing

- Logical Vector: True와 False로만 이루어진 벡터
- Logical Vector로 벡터에 접근할 때 True에 해당하는 값만 반환(return)




```
x <- c(5, -8, 2, -1)
y <- c(TRUE, FALSE, TRUE, FALSE)
x[y]
----------------------
[1] 5 2
```

Logical Indexing

```
x <- c(5, -8, 2, -1)
print(x>0)
---------------------
[1]  TRUE FALSE  TRUE FALSE
```

```
x <- c(5, -8, 2, -1)
y <- x>0
print(x[y])
----------------------
[1] 5 2
```

```
x <- c(5, -8, 2, -1)
print(x[x>0])
---------------------
[1] 5 2
```

함수: cat( )

- Concatenate and Print
- 여러 개의 argument를 concatenate한 결과를 출력
- sep이라는 parameter를 이용해 separator를 지정(default: white space)
- print( )와 달리 출력 후 줄바꿈이 되지 않는다.
- 줄바꿈을 하려면 \n 문자를 사용

```
> cat(1, '더하기', 2, '는', 3, '입니다.')
1 더하기 2 는 3 입니다.
> cat(1, '더하기', 2, '는', 3, '입니다.', sep='')
1더하기2는3입니다.
> cat(1, '더하기', 2, '는', 3, '입니다.\n')
```

Exercise

- 최근 일주일 동안의 최고 기온은 다음과 같다.
    - 31, 29, 30, 32, 35, 36, 34
- 다음과 같이 출력되도록 프로그래밍 하기
    - 최고 기온의 평균은 32.42857 이다.
    - 가장 더운 날의 온도는 36 도이다.
    - 최고 기온이 35도 이상인 날은 2 일이다.

Exercise

- 7명의 키와 몸무게가 다음과 같다.
  - 키: 185, 166, 172, 180, 163, 170, 177
  - 몸무게: 80, 73, 72, 100, 72, 67, 75
- 이 학생들의 BMI를 계산하고 BMI가 26보다 큰 학생의 몸무게 구하기
  - BMI = 몸무게*10000 / 키^2

Matrix

- 2차원의 표 형식으로 묶인 동일한 유형의 데이터 저장
- 행(row)은 표에서 가로에 해당
- 열(column)은 표에서 세로에 해당

Text	the	of	and
A	30	25	15
B	20	15	10
C	15	10	5

- 같은 열에 있는 데이터는 동일한 의미를 가진 값이어야 함
- 같은 행에 있는 데이터는 같은 대상에 해당하는 값이어야 함

Matrix

- 행렬 생성 방법

```
matrix(data,                    # 벡터 자료
       nrow = 1,                # 행의 개수
       ncol = 1                 # 열의 개수
       byrow = FALSE,           # data를 행 우선으로 채울지 여부
      )
```

```
data = c(1, 2, 3, 4, 5, 6, 7, 8)
score <- matrix(data, nrow=4)
print(score)
-------------------------------------
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

```
data = c(1, 2, 3, 4, 5, 6, 7, 8)
score <- matrix(data, nrow=4, byrow=TRUE)
print(score)
-------------------------------------
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```

Matrix

```
matrix(data,                    # 벡터 자료
       nrow = 1,                # 행의 개수
       ncol = 1                 # 열의 개수
       byrow = FALSE,           # data를 행 우선으로 채울지 여부
      )
```

```
data = c(1, 2, 3, 4, 5, 6, 7, 8)
score <- matrix(data, ncol=4)
print(score)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

```
data = c(1, 2, 3, 4, 5, 6, 7, 8)
score <- matrix(data, ncol=4, byrow=TRUE)
print(score)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Exercise

- 다음과 같은 내용의 Matrix를 생성하는 코드 작성

```
     [,1] [,2]
[1,]    5    8
[2,]    6    9
[3,]    7   10
```

```
     [,1] [,2] [,3]
[1,]    5    7    9
[2,]    6    8   10
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    3    4    5
[2,]    6    7    8    9
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]   12   14   16   18   20
[2,]   22   24   26   28   30
[3,]   32   34   36   38   40
[4,]   42   44   46   48   50
```

Matrix

```
matrix(data,                    # 벡터
       nrow = 1,                # 행의 개수
       ncol = 1                 # 열의 개수
       byrow = FALSE,           # data를 행 우선으로 채울지 여부
      )
```

```
data = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
# 주어진 데이터로 4행을 채우지 못할 경우
# 벡터의 앞쪽 값을 재사용 → Recycling Rule
score <- matrix(data, nrow=4)
print(score)
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7    1
[4,]    4    8    2
```

```
data = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
score <- matrix(data, ncol=4)
print(score)
```

?

Error가 발생하면 프로그램은 종료되지만 Warning이 발생하면 프로그램은 실행되고 결과에 문제가 있음을 사용자에게 알려준다.

Matrix

```
#행과 열의 인덱스 대신 행과 열의 이름 지정하기
score <- matrix(c(80,60,90,70,30,20), 2)
colnames(score) <- c('국어', '영어', '수학')
rownames(score) <- c('서진용', '김태훈')
print(score)
```

```
        국어 영어 수학
서진용   80   90   30
김태훈   60   70   20
```

Matrix

cbind(): 여러 벡터를 열 기준으로 결합하여 Matrix 생성
rbind(): 여러 벡터를 행 기준으로 결합하여 Matrix 생성

```
the <- c(30, 20, 15)
of <- c(25, 15, 10)
and <- c(15, 10, 5)
freq <- cbind(the, of, and)
print(freq)
-------------------------------------
     the of and
[1,]  30 25  15
[2,]  20 15  10
[3,]  15 10   5
freq <- cbind(the, of, and, to=c(10, 12, 14))
print(freq)
-------------------------------------
     the of and to
[1,]  30 25  15 10
[2,]  20 15  10 12
[3,]  15 10   5 14
```

```
the <- c(30, 20, 15)
of <- c(25, 15, 10)
and <- c(15, 10, 5)
freq <- rbind(the, of, and)
print(freq)
-------------------------------------
     [,1] [,2] [,3]
the   30   20   15
of    25   15   10
and   15   10    5
```

Exercise

- 다음의 표를 Matrix로 나타내기

	stu1	stu2	stu3
sci	A	B	A+
math	B	B	C
eng	C	B	A

Matrix의 접근

score <- matrix(c(80, 60, 90, 70, 30, 20), 2)

Matrix에 저장된 결과

80	90	30
60	70	20

개별 값에 접근하기 위한 인덱스

[1, 1]	[1, 2]	[1,3]
[2, 1]	[2, 2]	[2,3]

- 개별 원소에 대한 접근

```
score[2,2]  # 70
```

- 행 전체에 대한 접근

```
score[1,]  # 80, 90, 30
```

- 일부분에 대한 접근

```
score[2, c(1,2)]  # 60, 70
```

- 열 전체에 대한 접근

```
score[,2]  # 90,70
```

Matrix의 수정

score <- matrix(c(80, 60, 90, 70, 30, 20), 2)

Matrix에 저장된 결과

80	90	30
60	70	20

개별 값에 접근하기 위한 인덱스

[1, 1]	[1, 2]	[1,3]
[2, 1]	[2, 2]	[2,3]

- 개별 원소 수정

```
score[2,2] = 3
```

- 행 전체 수정

```
score[1,] = c(85, 95, 70)
```

- 일부분 수정

```
score[2, c(1,2)]  = 23
```

- 열 전체 수정

```
score[,2] = c(66,88)
```

Exercise

- 앞서 작성했던 다음의 Matrix에서

	stu1	stu2	stu3
sci	A	B	A+
math	B	B	C
eng	C	B	A

  - stu2의 math 학점을 'F'로 수정하기
  - stu1의 eng 학점을 'A+'로 수정하기
  - sci의 학점을 B B A로 수정하기

Matrix의 연산

```
a1 <- matrix(c(1:6), nrow=3)
a2 <- matrix(c(-6:-1), nrow=3)
a3 <- matrix(c(-2:1), nrow=2)
```

- 두 행렬의 길이가 같을 때의 연산 → 동일한 위치의 원소끼리 연산

```
a1 + a2
```

- 두 행렬의 길이가 다를 때의 연산 → 오류 메시지 출력

```
a1 + a3
Error in a1 + a3 : non-conformable arrays
```

- 행렬과 단일 값의 연산 → 모든 원소와 단일 값의 연산을 각각 수행

```
a3 + 10
```