# 데이터 연산 및 조작

## Extract과 Replace를 중심으로

---

유충현

Updated: 2017/02/07

# Overview

1. 벡터의 조작

2. 행렬의 조작

# 벡터의 조작

# extract: index

"추출 및 변경 연산자 [] 안에 인덱스인 수치벡터를 넣어 추출"

- 인덱스를 이용한 추출
  - 추출할 원소의 위치를 연산자에 기술
  - 인덱스 벡터에 음수를 지정하면 제거로 동작
- examples
  - letters[1:3]
    - 'a', 'b', 'c'
  - letters[c(1, 3, 5)]
    - 'a', 'c', 'e'
  - letters[-c(1:23)]
    - 'x', 'y', 'z'

# execirses: extract-index

scores = {88, 79, 92, 85, 95, 83} 일 경우, index extract로 다음을 수행하라.

1. 1번부터 3번 학생의 성적을 추출하라.
2. 90점 이상의 성적을 추출하라.
    - 힌트) 90 이상의 값은 3, 5번째 원소임
3. 80점 이하의 성적을 제거하라.
    - 힌트) 80 이하의 값은 2번째 원소임

# execirses: extract-index

```
> scores <- c(88, 79, 92, 85, 95, 83)
>
> scores[1:3]

[1] 88 79 92

> scores[c(3, 5)]

[1] 92 95

> scores[-2]

[1] 88 92 85 95 83
```

# extract: names

"추출 및 변경 연산자 [] 안에 원소 이름인 문자벡터를 넣어 추출"

- 원소 이름을 이용한 추출
  - 추출할 원소 이름을 연산자에 기술
  - 이름에 음수를 지정하면 에러 발생
- examples
  - vec <- c(23, 43, 19)
  - names(vec) <- c("1st", "2nd", "3rd")
  - vec["3rd"]
    - 19
  - vec[c("3rd")]
    - 19
  - vec[c("1st", "2nd")]
    - 23, 43

# execirses: extract-names

scores 이름이 "Lee", "Kim", "Park", "Ryu", "Choi", "Kim2" 일 경우, 다음을 수행하라.

1. "Lee", "Kim", "Park" 학생의 성적을 추출하라.
2. "Park", "Choi" 학생의 성적을 추출하라.

# execirses: extract-names

```
> nm <- c("Lee", "Kim", "Park", "Ryu", "Choi", "Kim2")
> names(scores) <- nm
>
> scores[c("Lee", "Kim", "Park")]

 Lee  Kim Park
  88   79   92

> scores[c("Park", "Choi")]

Park Choi
  92   95
```

# extract: logical

"추출 및 변경 연산자 [] 안에 논리벡터를 넣어 추출"

- 벡터와 같은 길이의 논리벡터를 사용하여,
  - TRUE에 해당하는 위치의 원소를 추출
  - 논리벡터는 추출하려는 벡터의 길이와 같아야 함
- **가장 많이 응용되는 유형**
- examples
  - x = 1:10
  - x[x%%2 == 0]
    - 2, 4, 6, 8, 10
  - x[x%%2 == 1]
    - 1, 3, 5, 7, 9
  - x[x >= 8]
    - 8, 9, 10

# execirses: extract-logical

scores로 logical extract 이용해 다음을 수행하라.

1. 90점 이상의 성적을 추출하라.
2. 최고의 성적을 추출하라.
    - 힌트) max() 함수 이용
3. 평균 이하의 성적을 추출하라.
    - 힌트) mean() 함수 이용

# execirses: extract-logical

```
> (flag <- scores >= 90)

  Lee   Kim  Park   Ryu  Choi  Kim2
FALSE FALSE  TRUE FALSE  TRUE FALSE

> scores[flag]

Park Choi
  92   95

> scores[scores == max(scores)]

Choi
  95
```

# execirses: extract-logical

```
> scores[scores <= mean(scores)]


 Kim  Ryu Kim2
  79   85   83
```

# replace

"추출 및 변경 연산자 []와 할당 연산자를 이용하여 변경"

- 할당 연산자를 취해 값 변경
- x <- 1:26; names(x) <- letters 일 경우,
- examples
  - x[1:3] <- c(3, 5, 7)
    - 앞 세 개를 3, 5, 7로 변경
  - x[c("x", "y")] <- 3
    - "x", "y" 이름의 원소를 3으로 변경
  - x[x <= 5] <- 5
    - 5 이하의 값을 5로 변경

# execirses: replace

scores = {88, 79, 92, 85, 95, 83} 일 경우, 다음을 수행하라.

1. 1번부터 3번 학생의 성적을 1점 올려주어라.
    - 주문 : 인덱스를 이용한 변경
2. "Kim" 학생의 성적을 85점으로 변경하라.
    - 주문 : 이름을 이용한 변경
3. 평균 이하의 성적을 평균으로 변경하라.
    - 주문 : 논리값을 이용한 변경

# execirses: replace

```
> scores

 Lee  Kim Park  Ryu Choi Kim2
  88   79   92   85   95   83

> scores[1:3] <- scores[1:3] + 1
> scores[c("Kim")] <- 85
> scores[scores <= mean(scores)] <- mean(scores)
> scores

     Lee      Kim     Park      Ryu     Choi     Kim2
89.00000 88.33333 93.00000 88.33333 95.00000 88.33333
```

# replace, append function

## replace

- replace(x, list, values)
- 벡터 x의 list 위치에 있는 원소를 values로 변경
- y <- 1:5; replace(y, 1:2, 5)
  - 5, 5, 3, 4, 5

## append

- append(x, values, after = length(x))
- 벡터 x의 after 다음에 values를 추가함
- append(1:5, 0:1, after = 3)
  - 1, 2, 3, 0, 1, 4, 5

# 기본 벡터 연산

- 사칙연산
  - 이항 연산자 + , - *, /를 용한 가감승제 연산
  - 해당 위치의 원소끼리 개별 연산을 수행
  - x <- 1:3; y <- 2:4일 경우,
  - x + y
    - 3, 5, 7
  - 벡터의 내적 및 외적
  - %*%를 이용한 벡터의 내적 계산
  - %o%를 이용한 벡터의 외적 계산

# execirses: 기본 벡터 연산

"x <- 1:3; y <- 2:4일 경우, 벡터 x와 y의 내적 및 외적은?"

```
> x <- 1:3; y <- 2:4
> x %*% y

     [,1]
[1,]   20

> x %o% y

     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    4    6    8
[3,]    6    9   12
```

# 재사용성 (recycling rules) ★★★★★

## 재사용성 (recycling rules)

- 벡터 연산에서 두 벡터의 길이가 다를 경우,
  - 짧은 쪽의 벡터를 긴 쪽의 벡터의 길이에 맞춘 후 연산
- 긴 벡터의 길이가 짧은 벡터 길이의 배가 되지 않을 경우,
  - 연산 후 warning 메시지 출력

# execirses: recycling rules

```
> x <- 1:2
> y <- 1:4
> z <- 1:3
> x + y

[1] 2 4 4 6

> x + z

Warning in x + z: 두 객체의 길이가 서로 배수관계에 있지
않습니다

[1] 2 4 4
```

# 행렬의 조작

# extract: index

"추출 및 변경 연산자 [] 안에 인덱스인 수치벡터를 넣어 추출"

- 벡터의 인덱스를 이용한 추출의 개념을 확장
  - 행의 추출 : x[idex, ]
  - 열의 추출 : x[, idex]
  - 인덱스 벡터에 음수를 지정하면 제거로 동작
- examples
  - x[1:3, ]
    - 행렬 x의 1, 2, 3 행 추출
  - x[, c(1, 3, 5)]
    - 행렬 x의 1, 3, 5 열 추출
  - x[1:3, -5]
    - 행렬 x의 1, 2, 3 행 중 5열 제거하여 추출

# execirses: extract-index

| 구분 | 국어 | 영어 | 수학 | 사회 | 과학 |
|---|---|---|---|---|---|
| 김홍도 | 88 | 79 | 92 | 85 | 96 |
| 이순신 | 85 | 83 | 89 | 88 | 84 |
| 박지원 | 90 | 89 | 92 | 95 | 87 |

성적이 상기 테이블과 같을 경우, index를 이용하여 다음을 수행하라.

1. 1번부터 3번 열의 성적을 추출하라.
2. 90점 이상의 성적을 많이 받은 행을 추출하라.
   - 힌트) 90 이상이 많은 행은 3번째 행임
3. 80점 이하의 성적을 받은 열을 제거하라.
   - 힌트) 80 이하의 값은 2번째 열임

# execirses: extract-index

```
> scores <- c(88, 79, 92, 85, 96,
+             85, 82, 89, 88, 84,
+             90, 89, 92, 95, 87)
> scores <- matrix(scores, nrow = 3, byrow = TRUE)
>
> scores[, 1:3]     # 1번부터 3번 열의 성적

     [,1] [,2] [,3]
[1,]   88   79   92
[2,]   85   82   89
[3,]   90   89   92
```

# execirses: extract-index

```
> scores[3, ]        # 3번째 행의 성적

[1] 90 89 92 95 87

> scores[, -2]       # 2번째 열의 성적 제거

     [,1] [,2] [,3] [,4]
[1,]   88   92   85   96
[2,]   85   89   88   84
[3,]   90   92   95   87
```

# extract: names

"추출 및 변경 연산자 [] 안에 원소 이름인 문자벡터를 넣어 추출"

- 벡터의 원소 이름을 이용한 추출의 개념을 확장
  - 추출할 원소 이름을 연산자에 기술
  - 이름에 음수를 지정하면 에러 발생
- examples
  - 행의 이름: "1st", "2nd", "3rd"
  - 열의 이름: "col1", "col2", "col3"
  - mat[, c("col2", "col3")]
    - 열 이름이 "col2", "col3"인 모든 행 추출
  - mat["3rd", ]
    - 행 이름이 "3rd"인 모든 열 추출
  - vec[c("1st", "2nd"), "col1"]
    - 행 이름이 "1st", "2nd"인 "col1" 열 추출

# execirses: extract-names

행(학생) 이름이 "Kim", "Lee", "Park"
열(과목) 이름이 "Kor", "Eng", "Mat", "Soc", "Sci" 일 경우, 다음을 수행하라.

1. 행렬에 행의 이름과 열의 이름을 정의하라.
2. "Kim" 학생의 성적을 추출하라.
3. "Kim", "Park" 학생의 국어, 영어 성적을 추출하라.

# execirses: extract-names (행렬 이름 생성-방법1)

```
> student <- c("Kim", "Lee", "Park")
> subject <- c("Kor", "Eng", "Mat", "Soc", "Sci")
> dimnames(scores) <- list(student, subject)
> scores

     Kor Eng Mat Soc Sci
Kim   88  79  92  85  96
Lee   85  82  89  88  84
Park  90  89  92  95  87
```

# execirses: extract-names (행렬 이름 생성-방법2)

```
> student <- c("Kim", "Lee", "Park")
> subject <- c("Kor", "Eng", "Mat", "Soc", "Sci")
> rownames(scores) <- student
> colnames(scores) <- subject
> scores

     Kor Eng Mat Soc Sci
Kim   88  79  92  85  96
Lee   85  82  89  88  84
Park  90  89  92  95  87
```

# execirses: extract-names

```
> scores["Kim", ]      # Kim의 성적을 추출

Kor Eng Mat Soc Sci
 88  79  92  85  96

> # Kim, Park의 국어, 영어 성적
> scores[c("Kim", "Park"), c("Kor", "Eng")]

     Kor Eng
Kim   88  79
Park  90  89
```

# extract: logical

"추출 및 변경 연산자 [] 안에 논리벡터를 넣어 추출"

- 행이나 열과 같은 길이의 논리벡터를 사용하여,
  - TRUE에 해당하는 위치의 원소를 추출
  - 논리벡터는 추출하려는 행/열의 길이와 같아야 함
  - 추출 및 변경 연산자로 구현하기가 매우 까다로움
- examples
  - x[seq(NROW(x))%%2 == 0, ]
    - 짝수 행 추출
  - x[, seq(NCOL(x))%%2 == 1]
    - 홀수 열 추출
  - x[seq(NROW(x))%%2 == 0, seq(NCOL(x))%%2 == 1]
    - 짝수 행의 홀수 열 추출

# execirses: extract-logical

scores로 다음을 수행하라.

1. 한 과목이라도 90점 이상을 받은 학생의 국어점수를 추출하라.
2. 최고의 성적을 받은 학생과 과목을 추출하라.
    - 힌트) max() 함수 이용

# execirses: extract-logical

```
> scores >= 90

       Kor   Eng   Mat   Soc   Sci
Kim  FALSE FALSE  TRUE FALSE  TRUE
Lee  FALSE FALSE FALSE FALSE FALSE
Park  TRUE FALSE  TRUE  TRUE FALSE

> ge90 <- apply(scores >= 90, 1, sum)
> ge90

 Kim  Lee Park
   2    0    3
```

# execirses: extract-logical

```
> scores[ge90 > 0, "Kor"]   # 90점 이상 학생의 국어점수

 Kim Park
  88   90
```

# execirses: extract-logical

```
> highest <- scores == max(scores)
> highest

       Kor   Eng   Mat   Soc   Sci
Kim  FALSE FALSE FALSE FALSE  TRUE
Lee  FALSE FALSE FALSE FALSE FALSE
Park FALSE FALSE FALSE FALSE FALSE

> idx <- which(highest)
> idx

[1] 13
```

# execirses: extract-logical

```
> stu_name <- rownames(scores)[idx %% 3]          # 이름
> stu_name

[1] "Kim"

> sub_name <- colnames(scores)[idx %/% 3 + 1]     # 과목
> sub_name

[1] "Sci"
```

# execirses: extract-logical

```
> matrix(scores[stu_name , sub_name],
+        dimnames = list(stu_name, sub_name))


    Sci
Kim  96
```

# 잠깐만, apply() 함수를 아시나요?

"반목문 없이 행렬의 marginal 집계 수행"

- apply(X, MARGIN, FUN, ...)
- X
  - 집계하려는 행렬이나 배열 객체
- MARGIN
  - 집계하려는 차원
  - 1: 행의 차원, 2: 열의 차원
- FUN
  - 집계 작업을 수행하는 함수
- ...
  - 집계함수의 선택 인수

# execirses: apply

학생의 성적표 행렬에서, 다음을 수행하라.

1. 학생별 성적의 평균을 구하라.
2. 과목별 성적의 평균을 구하라.
3. 학생별로 종합 성적의 석차를 구하라.
    - 힌트 : rank() 함수 사용

# execirses: apply

```
> apply(scores, 1, mean)      # 학생별 평균

 Kim  Lee Park
88.0 85.6 90.6

> apply(scores, 2, mean)      # 과목별 평균

     Kor      Eng      Mat      Soc      Sci
87.66667 83.33333 91.00000 89.33333 89.00000

> rank(100 - apply(scores, 1, mean), ties.method = "min")

 Kim  Lee Park
   2    3    1
```

# execirses: apply

Kim 학생이 영어시험을 결시하였다. 다음을 수행하라.

1. 결시한 성적에 NA를 담은 scores2 행렬을 만들어라.
2. scores2 행렬로 학생별 성적의 평균을 구하라.
3. scores2 행렬로 과목별 성적의 평균을 구하라.

# execirses: apply

```
> scores2 <- scores
> scores2["Kim", "Eng"] <- NA
> apply(scores2, 1, mean, na.rm = TRUE) # 학생별 평균

  Kim   Lee  Park
90.25 85.60 90.60

> apply(scores2, 2, mean, na.rm = TRUE) # 과목별 평균

     Kor       Eng       Mat       Soc       Sci
87.66667 85.50000 91.00000 89.33333 89.00000
```

# 내친 김에 달려보자!!! colSums 함수군

"행렬의 marginal 집계를 수행하는 간편한 함수들"

- colSums(x, na.rm = FALSE, dims = 1)
  - 행렬 x의 열의 합을 구함
- rowSums(x, na.rm = FALSE, dims = 1)
  - 행렬 x의 행의 합을 구함
- colMeans(x, na.rm = FALSE, dims = 1)
  - 행렬 x의 열의 평균을 구함
- rowMeans(x, na.rm = FALSE, dims = 1)
  - 행렬 x의 행의 평균을 구함

# execirses: colSums 함수군

scores2 행렬에서, colSums 함수군으로 다음을 수행하라.

1. 학생별 성적의 평균을 구하라.
2. 과목별 성적의 총점을 구하라.
3. 학생별로 종합 성적의 석차를 구하라.
    - 힌트 : rank() 함수 사용

# execirses: colSums 함수군

```
> rowMeans(scores2, na.rm = TRUE)      # 학생별 평균

  Kim   Lee  Park
90.25 85.60 90.60

> colSums(scores2, na.rm = TRUE)       # 과목별 총점

Kor Eng Mat Soc Sci
263 171 273 268 267

> rank(100-rowMeans(scores2, na.rm = TRUE), ties.method = "min"

 Kim  Lee Park
   2    3    1
```

# replace

"추출 및 변경 연산자 []와 할당 연산자를 이용하여 변경"

- 할당 연산자를 취해 값 변경
- examples
  - 행의 이름: "1st", "2nd", "3rd"
  - 열의 이름: "col1", "col2", "col3"
  - x[1, ] <- c(3, 5, 7)
    - 1행의 값을 3, 5, 7로 변경
  - x["1st", "col3"] <- 3
    - "1st" 행 "col3"열의 원소를 3으로 변경
  - x[x <= 5] <- 5
    - 모든 원소에 대해서 5 이하의 값을 5로 변경

# execirses: replace

성적을 담은 행렬에 대해서, 다음을 수행하라.

1. 1번부터 2번 학생의 모든 과목 성적을 1점 올려주어라.
    - 주문 : 인덱스를 이용한 변경
2. "Kim" 학생의 영어 성적을 85점으로 변경하라.
    - 주문 : 이름을 이용한 변경
3. 반 평균 이하의 성적을 평균으로 변경하라.
    - 주문 : 논리값을 이용한 변경

# execirses: replace

```
> scores

     Kor Eng Mat Soc Sci
Kim   88  79  92  85  96
Lee   85  82  89  88  84
Park  90  89  92  95  87

> scores[1:2, ] <- scores[1:2, ] + 1
> scores

     Kor Eng Mat Soc Sci
Kim   89  80  93  86  97
Lee   86  83  90  89  85
Park  90  89  92  95  87
```

# execirses: replace

```
> scores

      Kor Eng Mat Soc Sci
Kim    89  80  93  86  97
Lee    86  83  90  89  85
Park   90  89  92  95  87

> scores["Kim", "Eng"] <- 85
> scores

      Kor Eng Mat Soc Sci
Kim    89  85  93  86  97
Lee    86  83  90  89  85
Park   90  89  92  95  87
```

# execirses: replace

```
> mscore <- mean(scores)
> mscore

[1] 89.06667

> mscore <- round(mscore)
> mscore

[1] 89
```

# execirses: replace

```
> scores

      Kor Eng Mat Soc Sci
Kim    89  85  93  86  97
Lee    86  83  90  89  85
Park   90  89  92  95  87

> scores[scores <= mscore] <- mscore
> scores

      Kor Eng Mat Soc Sci
Kim    89  89  93  89  97
Lee    89  89  90  89  89
Park   90  89  92  95  89
```

# execirses: replace

scores2 행렬에서 Kim 학생의 미응시 영어점수에 대해 말들이 많다. 그래서 영어의 평균 점수로 Choi의 영어점수를 대체하기로 합의하였다.

1. Kim 학생의 영어점수를 영어의 평균으로 대체하라.
2. 학생별 성적의 평균을 구하라.
    - 주문 : apply() 함수 사용
3. 과목별 성적의 총점을 구하라.
    - 주문 : colSums 함수군 사용
4. 학생별로 종합 성적의 석차를 구하라.
    - 주문 : rowSums 함수군 사용

# execirses: replace

```
> newScore <- colMeans(scores2, na.rm = TRUE)["Eng"]
> newScore

 Eng
85.5

> scores2["Kim", "Eng"] <- newScore
> scores2

     Kor  Eng Mat Soc Sci
Kim   88 85.5  92  85  96
Lee   85 82.0  89  88  84
Park  90 89.0  92  95  87
```

# execirses: replace

```
> apply(scores2, 1, mean)   # 학생별 평균

 Kim  Lee Park
89.3 85.6 90.6

> colSums(scores2)          # 과목별 총점

  Kor   Eng   Mat   Soc   Sci
263.0 256.5 273.0 268.0 267.0

> rank(300-rowSums(scores2), ties.method = "min")

 Kim  Lee Park
   2    3    1
```

# binding function

## rbind

- rbind(..., deparse.level = 1)
- 임의 개수의 행렬/벡터를 행 방향으로 묶음
- 행의 개수가 같지 않으면, recycling rule 발생
- rbind(x, y) : 행렬(벡터) x의 행에 y 행렬(벡터)를 추가

## cbind

- cbind(..., deparse.level = 1)
- 임의 개수의 행렬/벡터를 열 방향으로 묶음
- 열의 개수가 같지 않으면, recycling rule 발생
- cbind(x, y) :행렬(벡터) x의 열에 y 행렬(벡터)를 추가

# execirses: rbind

"길이가 5인 문자벡터와 2행 5열 수치 행렬을 행병합하라."

```
> x <- letters[1:5]
> y <- matrix(1:10, ncol = 5)
> rbind(x, y)

  [,1] [,2] [,3] [,4] [,5]
x "a"  "b"  "c"  "d"  "e"
  "1"  "3"  "5"  "7"  "9"
  "2"  "4"  "6"  "8"  "10"
```

# execirses: cbind

"2행 3열 행렬과 2행 2열 행렬을 열병합하라."

```
> x <- matrix(1:6, nrow = 2)
> y <- matrix(1:4, nrow = 2)
> cbind(x, y)

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

# 기본 행렬 연산

- 사칙연산
  - 이항 연산자 + , - *, /를 용한 가감승제 연산
  - 해당 위치의 원소끼리 개별 연산을 수행
- 행렬의 곱
  - %*% 연산자를 이용해서 행렬의 곱 계산
- 역행렬의 계산
  - solve() 함수 이용

# execirses: 기본 행렬 연산

"2행 2열의 두 행렬에 대해서 덧셈과 곱셈을 수행하라."

```
> x <- matrix(1:4, ncol = 2)
> y <- matrix(2,  nrow = 2, ncol = 2)
> x + y

     [,1] [,2]
[1,]    3    5
[2,]    4    6

> x * y

     [,1] [,2]
[1,]    2    6
[2,]    4    8
```

# execirses: 기본 행렬 연산

"2행 2열의 행렬의 곱을 계산하고, 역행렬을 구하라."

```
> x %*% y

     [,1] [,2]
[1,]    8    8
[2,]   12   12

> x.invers <- solve(x)
> x.invers

     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

# execirses: 기본 행렬 연산

"2행 2열의 행렬의 곱을 계산하고, 역행렬을 구하라."

```
> x.invers

     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5

> x %*% x.invers

     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

THE END