통계학 실습

7. R 벡터, 행렬, 리스트

R 자료구조

- R에는 다른 프로그래밍 언어와 같이 다양한 자료구조가 존재한다.

- 기본적인 자료구조로 벡터, 행렬, 리스트가 존재한다.
  - 벡터는 해당 변수에 묶음 형태의 변수가 저장되는 것으로 매우 가변적인 형태의 자료구조다.
  - 행렬은 우리가 흔히 아는 N by M의 행렬로써 두 변수에 의한 관계를 저장하기 좋다.
  - 리스트는 일련의 데이터들을 정렬하는데 유리하다.

벡터의 개념

- R에서의 벡터는 흔히 알고 있는 (I, J, K)와 같은 물리적인 벡터가 아니라 여러 입력 값들을 하나의 변수로 처리하는 자료구조 변수이다.

- 다른 프로그래밍 언어에서 사용하는 배열과 유사한 역할을 하는데, R에서는 하나의 숫자나 배열의 하나의 원소 조차도 벡터로 취급한다. 특히, 이 벡터에 속한 각각의 값들을 ‘요소 (element)’라고 부른다.

벡터의 사용법 (1)

```
> x <- c(1, 2, 3, 4, 5)
> x
[1] 1 2 3 4 5
> x[0]
numeric(0)
> x[1]
[1] 1
> x[3]
[1] 3
> x <- c(1, TRUE, "String", 4, 5)
> x
[1] "1"      "TRUE"   "String" "4"      "5"
> 
```

- c() 함수를 이용하여 저장한다.
- 벡터의 인덱스는 다른 프로그래밍 언어와 다르게 1부터 시작한다.
  - 접근 방법은 다른 프로그래밍 언어의 배열과 같이 []를 이용한다.
- 하나의 벡터에는 하나의 자료형만 존재할 수 있다.
  - 숫자들 틈에 문자열이 들어갈 경우, 모두 문자열이 된다.
  - 숫자들 틈에 boolean이 들어갈 경우, TRUE는 1로, FALSE는 0이 된다.

벡터의 사용법 (2)

```
> x <- c(NA, 60, 50, 30, 20)
> x
[1] NA 60 50 30 20
> mean(x)
[1] NA
> x <- c(NULL, 60, 50, 30, 20)
> x
[1] 60 50 30 20
> mean(x)
[1] 40
>
```

- ‘NA’는 결측값으로써 에러 값으로 판단된다. 하나의 자리를 차지하지만 에러인 값이므로 평균이 계산될 수 없어서 평균 계산 함수인 MEAN함수의 결과가 출력되지 않는다.

- ‘NULL’은 기존 프로그래밍 언어와 같이 아예 존재하지 않는 특별한 객체를 뜻하므로, 벡터에 저장하여도 자리를 차지 하지 않기 때문에 평균값이 계산된다.

벡터와 관련된 다양한 함수

x <– c(1, 2, 3, 4, 5)

함수	결과	의미
cumsum(x)	1 3 6 10 15	누적합
length(x)	5	길이
max(x), min(x)	5 1	최대값, 최소값
prod(x)	120	총 요소의 곱
range(x)	1 5	범위
sev(x)	5 4 3 2 1	역순
sum(x)	15	총합
unique(x)	1 2 3 4 5	중복 제거
summary(x)	다양한 요약	요약된 통계량

벡터끼리의 연산

```
> x <- c(1, 2, 3)
> y <- c(3, 4, 5)
> x + y
[1] 4 6 8
> x * y
[1]  3  8 15
> y <- c(1, 2, 3, 4)
> x + y
[1] 2 4 6 5
경고메시지(들):
In x + y : 두 객체의 길이가 서로 배수관계에 있지 않습니다
> x * y
[1] 1 4 9 4
경고메시지(들):
In x * y : 두 객체의 길이가 서로 배수관계에 있지 않습니다
>
```

- 벡터끼리 연산할 경우 자동으로 각 단위에 맞추어 계산하여 준다.
- 두 벡터의 길이가 서로 배수 관계인 경우 더 작은 벡터가 반복되며 큰 벡터에게 맞추어진다.
  - 배수 관계가 아닐 경우, 맞춰지는 크기까지 계속하여 연산을 한다.

행렬의 개념

- 행렬은 다양한 행과 다양한 열에 속하는 각 값들이 모인 자료구조이다.

- 행렬의 각 행과 열은 하나의 벡터이며, 전체가 또 하나의 벡터가 된다. 여러 변수 간의 관계에 따른 수치나 결과를 표현하고 싶을 때 이용하기에 유리하다.

행렬의 사용법

```
> x <- matrix(c(1, 2, 3, 4), nrow=2, ncol=2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> x[1]
[1] 1
> x[1,]
[1] 1 3
> x[,1]
[1] 1 2
> y <- matrix(c(1, 2, 3, 4, 5, 6), nrow=3)
> y
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
>
```

- 행렬은 'matrix' 함수를 이용한다.
- 저장되는 순서는 열부터 채워지며, nrow는 행의 개수, ncol은 열의 개수가 된다. 여기서 행렬 안에 들어가는 원소들의 개수가 nrow 혹은 ncol의 배수여야만 한다.
- 각 값에 접근하기 위해서는 기존의 인덱싱 방법을 이용하고, 한 행에 접근하고 싶을 때는 [행의 번호,]로 접근하고, 한 열에 접근하고 싶다면 [,열의 번호]로 접근한다.

행렬 연산

```
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> x*3
     [,1] [,2]
[1,]    3    9
[2,]    6   12
> x+x
     [,1] [,2]
[1,]    2    6
[2,]    4    8
> colnames(x) <- c("col1", "col2")
> x
     col1 col2
[1,]    1    3
[2,]    2    4
>
```

- 일반적으로 행렬에서 가능한 모든 연산이 가능하다. 사칙연산을 포함하여, 역함수 혹은 행렬 곱도 이용할 수 있다.
- 각 행과 열에 이름을 지정하여 줄 수 있다.
  - 행은 rownames(x) 함수를 이용한다.
  - 열은 colnames(x) 함수를 이용한다.

행렬의 행과 열 추가 및 제거

```
> x <- matrix(c(1, 2, 3, 4), nrow = 2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> x <- cbind(c(5, 6), x)
> x
     [,1] [,2] [,3]
[1,]    5    1    3
[2,]    6    2    4
> x <- x[,1:2]
> x
     [,1] [,2]
[1,]    5    1
[2,]    6    2
>
```

- 행렬은 생성 시에 크기가 고정되어 변경할 수 없지만, 함수를 통하여 행이나 열을 추가할 수 있다. 또한 인덱싱을 통해 제거할 수 있다.
- cbind()함수는 열을 추가하여 주는 함수이다. 하나의 벡터가 열이 되어 원래 행렬에 추가되는 방식이다. 행을 추가하려면 rbind() 함수를 사용하면 된다.
- 제거를 원하는 경우에 해당 행이나 열의 범위를 설정하여 인덱싱으로 자르고 새롭게 저장한다.

리스트의 사용 (1)

```
> x <- list(apple=1, banana=2, kiwi="good")
> x
$apple
[1] 1

$banana
[1] 2

$kiwi
[1] "good"

> x[apple]
에러: 객체 'apple'를 찾을 수 없습니다
> x["apple"]
$apple
[1] 1

> x <- list(1, 2, 3)
> x
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

>
```

- 리스트는 벡터의 종류 중 하나이다.
- 다른 프로그래밍 언어에서 '사전'에 해당하는 기능을 해주면서, 이중 배열의 역할을 하기도 한다.
- 각 변수에 대해 이름을 지정하여 사전처럼 사용하거나, 각 변수를 이중 인덱스로 접근할 수 있다.

리스트의 사용 (2)

```
> x <- list(1, 2, 3)
> x
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

> x[[3]] <- 4
> x
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 4
```

- 각 인덱스에 해당되는 값을 직접 바꾸거나, 새로운 키에 대한 값을 추가할 수 있다.
- 각 자리에 하나의 값이 아니라 여러 값들을 가지고 싶다면, 벡터를 추가하면 된다.

실습 (1)

- 라이브러리 패키지에서 원하는 데이터를 추출 후 다루어보자.
  - MASS 패키지에서 Cars93 데이터셋을 추출.
  - summary(), str() 함수를 통해 데이터셋 내부를 확인한다.
  - 벡터, 행렬, 리스트 중 원하는 자료구조에 Cars93의 'Price' 항목을 저장하자.
  - mean(), sum() 함수를 통해 평균과 총합을 확인한다.
  - sort() 함수를 이용하면 값들을 정렬시켜 확인할 수 있다.

```
> Cars93$Price
 [1] 15.9 33.9 29.1 37.7 30.0 15.7 20.8 23.7 26.3 34.7 40.1 13.4 11.4 15.1 15.9 16.3 16.6 18.8 38.0 18.4
[34] 15.9 14.0 19.9 20.2 20.9  8.4 12.5 19.8 12.1 17.5  8.0 10.0 10.0 13.9 47.9 28.0 35.2 34.3 36.1  8.3
[67] 21.5 13.5 16.3 19.5 20.7 14.4  9.0 11.1 17.7 18.5 24.4 28.7 11.1  8.4 10.9 19.5  8.6  9.8 18.4 18.2
> PriceList <- Cars93$Price
> PriceList
 [1] 15.9 33.9 29.1 37.7 30.0 15.7 20.8 23.7 26.3 34.7 40.1 13.4 11.4 15.1 15.9 16.3 16.6 18.8 38.0 18.4
[34] 15.9 14.0 19.9 20.2 20.9  8.4 12.5 19.8 12.1 17.5  8.0 10.0 10.0 13.9 47.9 28.0 35.2 34.3 36.1  8.3
[67] 21.5 13.5 16.3 19.5 20.7 14.4  9.0 11.1 17.7 18.5 24.4 28.7 11.1  8.4 10.9 19.5  8.6  9.8 18.4 18.2
> mean(PriceList)
[1] 19.50968
> sort(PriceList)
 [1]  7.4  8.0  8.3  8.4  8.4  8.6  9.0  9.1  9.2  9.8 10.0 10.0 10.1 10.3 10.9 11.1 11.1 11.3 11.3 11.4
[34] 15.1 15.6 15.7 15.7 15.8 15.9 15.9 15.9 16.3 16.3 16.5 16.6 17.5 17.7 18.2 18.4 18.4 18.5 18.8 19.0
[67] 21.5 22.7 22.7 23.3 23.7 24.4 25.8 26.1 26.3 26.7 28.0 28.7 29.1 29.5 30.0 31.9 32.5 33.9 34.3 34.7
> PriceList
 [1] 15.9 33.9 29.1 37.7 30.0 15.7 20.8 23.7 26.3 34.7 40.1 13.4 11.4 15.1 15.9 16.3 16.6 18.8 38.0 18.4
[34] 15.9 14.0 19.9 20.2 20.9  8.4 12.5 19.8 12.1 17.5  8.0 10.0 10.0 13.9 47.9 28.0 35.2 34.3 36.1  8.3
[67] 21.5 13.5 16.3 19.5 20.7 14.4  9.0 11.1 17.7 18.5 24.4 28.7 11.1  8.4 10.9 19.5  8.6  9.8 18.4 18.2
```

실습 (2)

```
> class
            smoke non-smoke
drink          15         7
non-drink       5        13
```

- 대학교의 한 반에는 40명의 학생들이 있다. 위는 학생들의 음주 여부와 흡연 여부에 대하여 조사한 표이다.
  - 위의 자료를 저장하기 위한 행렬을 만들어 자료를 저장하고, 행과 열의 이름을 등록하자.
  - 해당 행과 열의 총합을 나타내어 주는 sum이라는 이름을 가진 행과 열을 추가하여라.
    1. [3, 3]에 해당되는 수치는 40이 된다.

문제

- 철수의 이번 년도 중간고사, 기말고사의 점수는 아래 표와 같다.

	Korean	English	Math	Social	Science
Mid	95	75	85	90	77
Final	97	67	99	88	82

- 자료구조에 알맞게 저장한 뒤, 각 고사 별 총점과 평균을 구해보자.
  - 중간고사 : 84.4점, 422점
  - 기말고사 : 86.6점, 433점