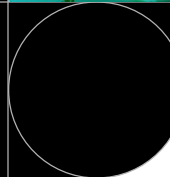
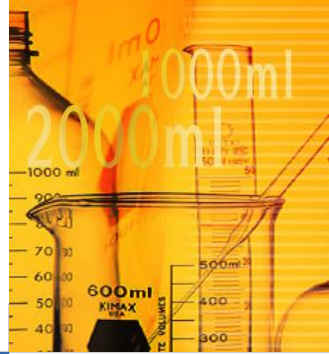


Chapter 3

R 기초 [2] Matrix, Data Frame

Sejong Oh

Bio Information technology Lab.



Content

- 논리값 벡터 다루기
- matrix
- data frame
- 파일에서 데이터 읽기/쓰기

논리값 벡터 다루기

- 논리연산자 : <, <=, >, >=, ==, !=, | (or), & (and)

```
d <- c(1,2,3,4,5,6,7,8,9)
d>=5           # FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
d[d>5]         # 6 7 8 9
sum(d>5)       # 5 보다 큰 값의 개수를 출력
sum(d[d>5])    # 5 보다 큰 값의 합계를 출력
d==5           # FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE

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

주의. 프로그래밍 언어에서 equal 연산자는 = 가 아닌 ==
R 에서 = 는 <- 와 같은 의미

[연습1]

- 다음과 같이 벡터 v1 을 생성하시오

```
v1 <- 51:90
```

- 1) v1 에서 60 보다 작은 수 들을 보이시오
- 2) v1 에서 70 보다 작은 수 들은 몇개인지 보이시오
- 3) v1 에서 65 보다 큰 수들의 합을 보이시오
- 4) v1 에서 60보다 크고 73 보다 작은 수들을 보이시오
- 5) v1 에서 65 보다 작거나 80 보다 큰 수들을 보이시오
- 6) v1 에서 7로 나눈 나머지가 3 인 숫자들만 보이시오
- 7) v1 에서 짝수들의 합계를 보이시오
- 8) v1 에서 홀수이거나 80 보다 큰 수를 보이시오
- 9) v1 에서 3과 5의 공배수를 보이시오

2차원 데이터 다루기

- 분석을 위한 데이터는 2차원 테이블 형태인 경우가 대부분
- Vector : 1차원 데이터를 저장하기 위한 자료구조
- 2차원 데이터를 저장하기 위해서 R에서는 matrix 와 data frame 을 제공
 - ◉ matrix : 모든 저장된 데이터의 데이터 타입이 동일
 - ◉ data.frame : 서로 다른 유형의 데이터 타입을 가진 값들을 저장

	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10

	Petal.Length	Petal.Width	Species
1	1.4	0.2	setosa
2	1.4	0.2	setosa
3	1.3	0.2	setosa
4	1.5	0.2	setosa
5	1.4	0.2	setosa
6	1.7	0.4	setosa

2차원 데이터 다루기

행
(row)

	Petal.Length	Petal.Width	Species
1	1.4	0.2	setosa
2	1.4	0.2	setosa
3	1.3	0.2	setosa
4	1.5	0.2	setosa
5	1.4	0.2	setosa
6	1.7	0.4	setosa

열
(column)

2행3열
[2,3]

행번호
(데이터 아님)

- Matrix 생성

```
z <- matrix(1:20, nrow=4, ncol=5)
```

```
z
```

행의 수

열의 수

```
> 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

```
> |
```

Note. 이렇게 직접 matrix 를 만드는 경우는 거의 없음
대부분 데이터를 파일에서 불러옴

matrix

- 기존의 vector(들) 또는 matrix(들)을 결합해 새로운 행렬을 만들 수도 있다.

```
x <- 1:4
y <- 5:8

m1 <- cbind(x,y) # 열방향 결합
m2 <- rbind(x,y) # 행방향 결합
m1
m2
```

```
> m1
      x y
[1,] 1 5
[2,] 2 6
[3,] 3 7
[4,] 4 8
> m2
      [,1] [,2] [,3] [,4]
x         1     2     3     4
y         5     6     7     8
```


matrix

```
m3 <- rbind(m2,x)
m4 <- cbind(z,x)
m3
m4
```

```
> m3
```

	[,1]	[,2]	[,3]	[,4]
x	1	2	3	4
y	5	6	7	8
x	1	2	3	4

```
> m4
```

						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

- matrix 안에서의 위치 지정

```
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
```

```
> z[2,3] # 2행 3열에 있는 값  
[1] 10  
> z[1,4] # 1행 4열에 있는 값  
[1] 13  
> z[2,]  # 2행에 있는 모든 값  
[1] 2 6 10 14 18  
> z[,4]  # 4열에 있는 모든 값  
[1] 13 14 15 16  
>
```

```
> 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

z[2,]

z[,4]

- 행과 열에 이름 붙이기

```
rownames(z) # 행의 이름 보이기
colnames(z) # 열의 이름 보이기
rownames(z) <- c("row1", "row2", "row3", "row4")
colnames(z) <-
c("col1", "col2", "col3", "col4", "col5")
```

```
> rownames(z) # 행의 이름 보이기
NULL
> colnames(z) # 열의 이름 보이기
NULL
```

```
> z
      col1 col2 col3 col4 col5
row1     1     5     9    13    17
row2     2     6    10    14    18
row3     3     7    11    15    19
row4     4     8    12    16    20
~
```

- 행, 열 이름으로 데이터 접근하기

```
z[, "col3"]  
z["row2", ]
```

```
> z[, "col3"]  
row1 row2 row3 row4  
  9   10   11   12  
> z["row2", ]  
col1 col2 col3 col4 col5  
  2    6   10   14   18
```

[연습2]

1. 다음과 같은 내용의matrix 를 생성하시오 (이름은 score)

```
> score
      m  f
[1,] 10 21
[2,] 40 60
[3,] 60 70
[4,] 20 30
```

2. 컬럼의 이름을 각각 male, female 로 바꾸시오
3. 2행에 있는 모든 값을 보이시오
4. female 의 모든 값을 보이시오
5. 3행 2열의 값을 보이시오

data frame

- Data frame 만들기

```
city <- c("Seoul", "Tokyo", "Washington")  
rank <- c(1, 3, 2)  
city.info <- data.frame(city, rank)
```

```
> city <- c("seoul", "Tokyo", "washington")  
> rank <- c(1, 3, 2)  
> city.info <- data.frame(city, rank)  
>  
> city.info  
  city rank  
1 seoul   1  
2 Tokyo   3  
3 washington 2
```

- 컬럼별로는 데이터 타입이 동일해야 한다
- 만들어진 후에는 matrix 와 동일하게 다룰 수 있다

data frame

- iris : R 에서 제공하는 dataset

iris

```
> 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
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
7         4.6         3.4         1.4         0.3   setosa
8         5.0         3.4         1.5         0.2   setosa
9         4.4         2.9         1.4         0.2   setosa
10        4.9         3.1         1.5         0.1   setosa
11        5.4         3.7         1.5         0.2   setosa
12        4.8         3.4         1.6         0.2   setosa
13        4.8         3.0         1.4         0.1   setosa
14        4.3         3.0         1.1         0.1   setosa
15        5.8         4.0         1.2         0.2   setosa
16        5.7         4.4         1.5         0.4   setosa
17        5.4         3.9         1.3         0.4   setosa
18        5.1         3.5         1.4         0.3   setosa
```



붓꽃에 대한 정보 저장

- Sepal.Length, Sepal.Width : 꽃받침 길이, 폭
- Petal.Length, Petal.Width : 꽃잎 길이, 폭
- Species : 종(種)

data frame

- data.frame 객체인 경우 열별 데이터를 뽑는 방법이 여러가지 있음

```
is.data.frame(iris) # 객체가 data.frame 인지 확인
iris[, "Species"]    # 결과가 vector == iris[,5]
iris["Species"]      # 결과가 nx1 data frame
iris$Species         # 결과가 vector
```

```
> is.data.frame(iris)
[1] TRUE
> iris[, "species"]
 [1] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[13] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[25] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[37] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[49] setosa    setosa    versicolor versicolor versicolor versicolor versicolor
[61] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[73] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
```

```
> iris["species"]
  species
1   setosa
2   setosa
3   setosa
4   setosa
5   setosa
6   setosa
7   setosa
8   setosa
9   setosa
10  setosa
11  setosa
12  setosa
```

```
> iris$Species
 [1] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[13] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[25] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[37] setosa    setosa    setosa    setosa    setosa    setosa    setosa
[49] setosa    setosa    versicolor versicolor versicolor versicolor versicolor
[61] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[73] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
```

dataset이름\$컬럼이름: data frame에만 적용되고
matrix에는 적용이 안된다

data frame

```
iris[,c(1:2)]           # 앞의 2개 컬럼 데이터 보기
iris[,c(1,3,5)]
iris[,c("Sepal.Length", "Species")]
iris[1:50,]
iris[1:50,c(1,3)]
```

```
> iris[1:50,c(1,3)]
  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
6           5.4           1.7
7           4.6           1.4
8           5.0           1.5
9           4.4           1.4
10          4.9           1.5
11          5.4           1.5
```

matrix, data frame 다루기

```
dim(iris)      # 행과 열의 수 보이기
nrow(iris)     # 행의 수 보이기
ncol(iris)     # 열의 수 보이기
names(iris)    # 컬럼이름 보이기
head(iris)     # 데이터셋의 앞부분 일부 보기
tail(iris)     # 데이터셋의 뒷부분 일부 보기
```

```
> dim(iris)      # 행과 열의 수 보이기
[1] 150    5
> nrow(iris)     # 행의 수 보이기
[1] 150
> ncol(iris)     # 열의 수 보이기
[1] 5
> names(iris)    # 컬럼이름 보이기
[1] "Sepal.Length" "Sepal.width"  "Petal.Length" "Petal.width"
[5] "species"
> 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
> 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
```

matrix, data frame 다루기

```
str(iris)           # 데이터셋 요약 보기
unique(iris[,5])     # 종의 종류 보기 (중복 제거)
table(iris[, "Species"]) # 종의 종류별 instance count
```

```
> str(iris)           # 데이터셋 요약 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 1 ...
> unique(iris[,5])     # 종의 종류 보기 (중복 제거)
[1] setosa    versicolor virginica
Levels: setosa versicolor virginica
> table(iris[, "Species"]) # 종의 종류별 instance count

    setosa versicolor  virginica 
      50         50         50
```

matrix, data frame 다루기

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

```
> 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  9.3
[14]  8.5 11.2 12.0 11.0 10.3 11.5 10.7 10.7 10.7  9.4 10.6 10.3  9.8
[27] 10.4 10.4 10.2  9.7  9.7 10.7 10.9 11.3  9.7  9.6 10.5 10.0  8.9
[40] 10.2 10.1  8.4  9.1 10.7 11.2  9.5 10.7  9.4 10.7  9.9 16.3 15.6

> 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 2.700
[12] 2.500 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675 2.675
[23] 2.350 2.650 2.575 2.450 2.600 2.600 2.550 2.425 2.425 2.675 2.725
[34] 2.825 2.425 2.400 2.625 2.500 2.225 2.550 2.525 2.100 2.275 2.675
```

matrix, data frame 다루기

- subset() 함수 : 조건에 맞는 행(row) 추출

```
IR.1 <- subset(iris, Species=="setosa")
IR.1
IR.2 <- subset(iris, Sepal.Length>5.0 &
               Sepal.Width>4.0)
IR.2
```

```
> 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
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
7           4.6         3.4         1.4         0.3  setosa
8           5.0         3.4         1.5         0.2  setosa
```

```
> 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
```

matrix 연산

- matrix 간에도 사칙연산 가능 (행/열의 수가 동일할 때)

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

```
> 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

> 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

> b-a
      [,1] [,2] [,3] [,4] [,5]
[1,]   20   20   20   20   20
[2,]   20   20   20   20   20
[3,]   20   20   20   20   20
[4,]   20   20   20   20   20
```

Note. 수학시간에 배운
행렬 곱셈은 %*% 를 이용

3*a

b-5

2*a + 3*b

```
> 3*a
      [,1] [,2] [,3] [,4] [,5]
[1,]     3    15    27    39    51
[2,]     6    18    30    42    54
[3,]     9    21    33    45    57
[4,]    12    24    36    48    60
> b-5
      [,1] [,2] [,3] [,4] [,5]
[1,]    16    20    24    28    32
[2,]    17    21    25    29    33
[3,]    18    22    26    30    34
[4,]    19    23    27    31    35
> 2*a + 3*b
      [,1] [,2] [,3] [,4] [,5]
[1,]    65    85   105   125   145
[2,]    70    90   110   130   150
[3,]    75    95   115   135   155
[4,]    80   100   120   140   160
```



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

```
> a <- a*3  
> b <- b-5  
> a  
      [,1] [,2] [,3] [,4] [,5]  
[1,]    3   15   27   39   51  
[2,]    6   18   30   42   54  
[3,]    9   21   33   45   57  
[4,]   12   24   36   48   60  
> b  
      [,1] [,2] [,3] [,4] [,5]  
[1,]   16   20   24   28   32  
[2,]   17   21   25   29   33  
[3,]   18   22   26   30   34  
[4,]   19   23   27   31   35  
.
```

[연습3]

- R 에서 제공하는 state.x77 데이터셋을 이용하여 실습하시오
미국 50개 주에 대한 통계데이터

1. state.x77 를 st 에 저장하시오

```
st <- data.frame(state.x77) # matrix 를 data.frame 으로 변환
```

2. st 의 내용을 보이시오

3. st 의 열 이름을 보이시오

4. st 의 행 이름을 보이시오

5. st 의 행의 개수와 열의 개수를 보이시오

6. st 의 요약정보를 보이시오

7. st 의 행별 합계와 평균을 보이시오

8. st 의 열별 합계와 평균을 보이시오

9. Florida 주의 모든 정보를 보이시오

```
> is.data.frame(state.x77)
[1] FALSE
> is.matrix(state.x77)
[1] TRUE
> st <- data.frame(state.x77)
> is.data.frame(st)
[1] TRUE
```

[연습3]

10. 50개 주의 Income 정보만 보이시오
11. texas 주의 면적(area) 을 보이시오
12. ohio 주의 인구(population) 와 수입(income)을 보이시오
13. 인구가 5000 이상인 주의 데이터만 보이시오
14. 수입이 4500 이상인 주의 인구, 수입, 면적을 보이시오
15. 수입이 4500 이상인 주는 몇 개인지 보이시오
16. 전체면적(area)이 100000 이상이고 결빙일수(frost) 가 120 이상인 주의 정보를 보이시오
17. 전체면적(area)이 100000 이상이고 결빙일수(frost) 가 120 이상인 주의 정보를 보이시오
18. 문맹률(illiteracy)이 2.0 이상인 주의 평균 수입은 얼마인가
19. 문맹률(illiteracy)이 2.0 미만인 주와 2.0 이상인 주의 평균 수입의 차이를 보이시오
20. 기대수명(life.exp)이 가장 높은 주는 어디인가
21. Pennsylvania 보다 수입이 높은 주들을 보이시오

파일에 데이터 읽기/쓰기

- 파일에서 데이터 읽어오기
 - Excel 에서 .csv 포맷으로 저장
 - read.csv() 함수 이용
 - 주의: 디렉토리 구분자는 "\\" 가 아닌 "/" 를 사용해야 함

```
setwd("C:/Rwork") # 파일이 있는 폴더 지정
mydata <- read.csv("test.csv", header = TRUE)
mydata
```

파일의 첫줄은
데이터가 아닌
헤더부분

	A	B	C	D	E	F	G	H	I
1	Gene sym	survival cc	BT005.INV	BT008.INV	BT009.INV	BT010.INV	BT016.INV	BT024.INV	BT029.INV
2	S100A9	0.520578	8.07	8.01	11.23	8.3	7.98	12.24	9.3
3	S100A8	0.351361	8.59	9.83	11.68	8.42	8.06	15.56	11.3
4	IL1B	0.401457	7.42	7.39	7.35	7.85	7.31	11.63	7.3
5	EGFR	1.202672	8.51	8.02	9.05	7.53	7.62	9.3	8.3

- 파일에서 데이터 읽어오기

(앞에서 이어서)

```
mydata          # 전체 데이터 출력
head(mydata)     # 앞의 몇줄 데이터만 출력
tail(mydata)     # 뒤의 몇줄 데이터만 출력

mydata[2,3]      # 2행 3열의 원소값 출력
nrow(mydata)     # 행의 개수 출력
ncol(mydata)     # 열의 개수 출력
dim(mydata)      # 행,열의 개수 출력

myRow1 <- mydata[2,]  # 2행의 값들을 추출하여 벡터생성
myRow2 <- mydata[,3]  # 3열의 값들을 추출하여 벡터생성
```

- 파일에 데이터 저장

(앞에서 이어서)

```
mynew <- mydata[,c(2,3)]           # 2,3 열만 추출  
write.csv(mynew, "kid_new.csv",  
           row.names=F, quote=F)
```

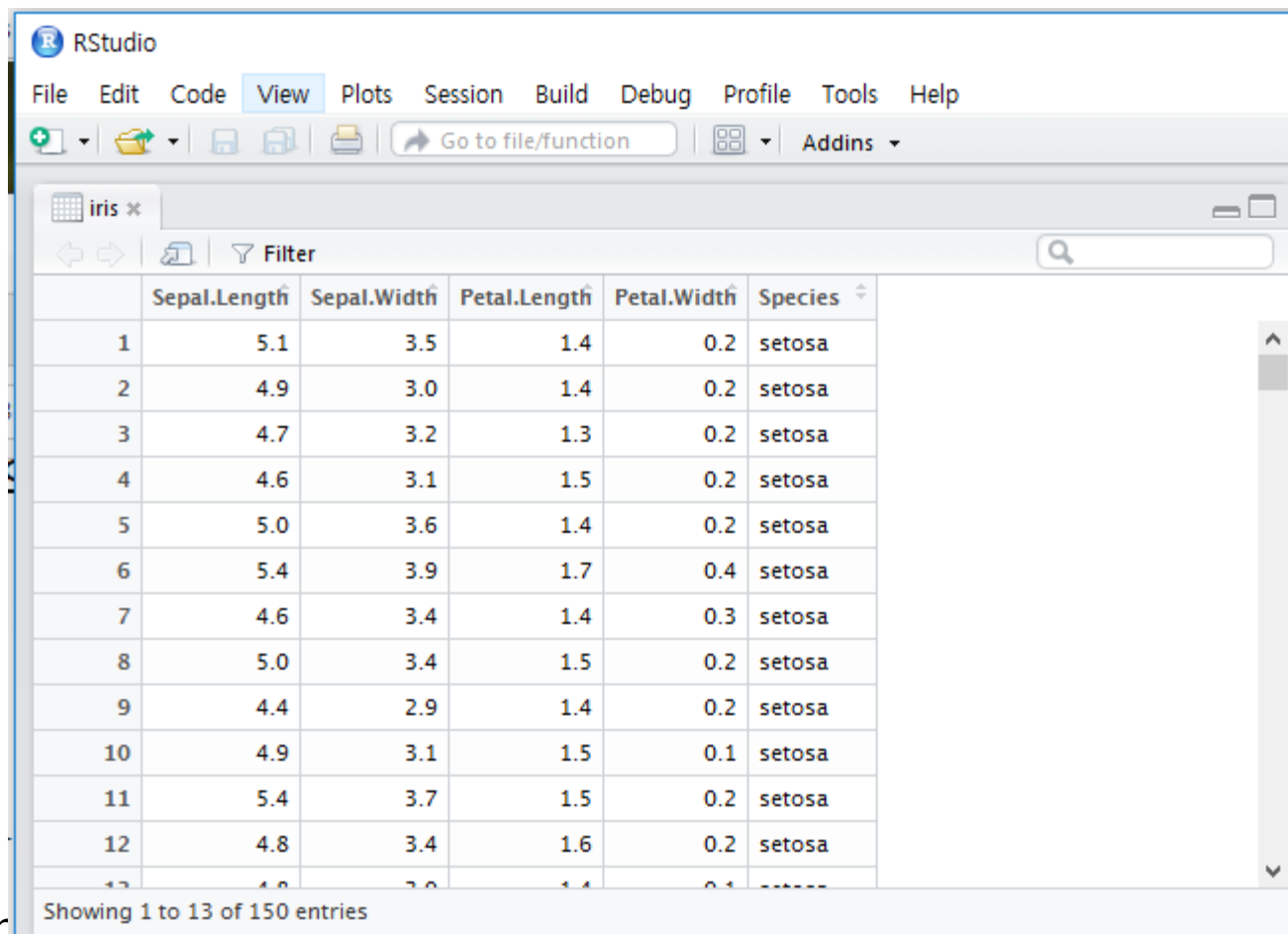
[연습4]

1. R 에서 제공하는 state.x77 데이터셋에서 수입이 5000 이상인 주의 데이터만 추출하여 rich_state.csv 에 저장하시오
2. rich_state.csv 파일을 읽어서 ds 변수에 저장후 ds 의 내용을 보이시오

[R Tip]

- R studio 에서 matrix, data frame 편리하게 보기

View(iris) # V 는 대문자



The screenshot shows the RStudio interface with the 'View' menu selected. The 'iris' dataset is open in the View window, displaying a table of 150 entries. The table has columns for Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. The first 13 rows are visible, showing various measurements for the 'setosa' species.

	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
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa

Showing 1 to 13 of 150 entries

- 데이터 타입 factor

- 문자형 변수로서 특정 종류의 값만을 가질 수 있는 데이터 타입
- 예를 들어 ABO식 혈액형을 나타내는 변수를 문자형 변수를 정의할 때 이 변수가 취할 수 있는 값은 A, B, AB, O의 네 가지 값만을 가져야 할 것이고, 다른 종류의 문자가 들어오면 에러가 나와야 할 것이다.
- 이러한 종류의 데이터 타입을 factor라고 한다. factor() 함수를 이용해 생성할 수 있다.
- iris 의 "Species" 가 factor 타입

```
blood.type <- factor(c("A", "A", "AB", "O",  
"B"))  
blood.type  
is.factor(blood.type)
```

```
> blood.type  
[1] A  A  AB O  B  
Levels: A AB B O  
> is.factor(blood.type)  
[1] TRUE
```