Exploratory Data Analysis

Chapter 2

Instructor: Seokho Lee

Hankuk University of Foreign Studies

2. Objects

2.1. Mode

Mode is a type of the minimal element. R objects consists of a single or multiple elements, which are elements in R.

- numeric: numeric values, including integer and double
- logical: logical value of TRUE and FALSE
- character: a character or a string
- complex: complex numbers consisting of real and imaginary parts

Example (Mode of vectors)

```
> 3 + 4
[1] 7
> mode(3+4)
[1] "numeric"
> pi
[1] 3.141593
> mode(pi)
[1] "numeric"
> 3 < 4
[1] TRUE
> mode(3<4)
[1] "logical"
> mode(T)        # T and TRUE are the same
[1] "logical"
> mode(FALSE)        # F and FALSE are the same
[1] "logical"
> mode(True)
다음에 오류mode(True) : 개체 'True'이 없습니다
> mode(f)
다음에 오류mode(f) : 개체 'f'이 없습니다
> 'Hi'
[1] "Hi"
> "Hi"
[1] "Hi"
> mode("Hi")
[1] "character"
> 1+4i
[1] 1+4i
> mode(1+4i)
[1] "complex"
```

Example

```
> "I am Seokho Lee" == "I  am   Seokho         Lee"
[1] FALSE
> T == TRUE
[1] TRUE
> 3 == 3.0
[1] TRUE
```

Example (Special values)

```
> NULL
NULL
> c(1,2,NA)
[1]  1  2 NA
> log(-5)
[1] NaN
경고 메시지가 손실되었습니다
In log(-5) : 계산결과가 NaN가 생성되었습니다
> 1/0
[1] Inf
> log(0)
[1] -Inf
```

- `NULL`: empty. size is 0
- `NA`: Not Available / missing value
- `NaN`: Not a Number
- `Inf`, `-Inf`: positive and negative infinity

Example (Special values)

```
> 5 + NULL
numeric(0)
> paste('sample',NULL)
[1] "sample "
> 5 + NA
[1] NA
> 5 + NaN
[1] NaN
> 5 + Inf
[1] Inf
> 5 + -Inf
[1] -Inf
```

When different types of variables are involved in the arithmetic operation, R changes them into a single type and use them in calculation. This is called the coercion of value. Priority order is

$$\text{logical} < \text{numeric} < \text{complex} < \text{character}$$

For example, (numeric) + (logical) is computed in (numeric).

Example (Coercion)

```
> 3 + TRUE
[1] 4
> (3+4i) + 4
[1] 7+4i
> c('34',4)
[1] "34" "4"
> c('34', 3+4i)
[1] "34"   "3+4i"
```

- `T`(`TRUE`) is coerced to 1, and `F`(`FALSE`) is coerced to 0.

Example (Coercion)

```
> T + T
[1] 2
> T * F
[1] 0
> 5i
[1] 0+5i
> 2 + 3i * 3 + 5i
[1] 2+14i
> (2+3i) * (3+5i)
[1] -9+19i
```

Example (Coercion)

```
> as.numeric('3.141592')     # character to numeric
[1] 3.141592
> as.double('3.141592')     # character to numeric(double)
[1] 3.141592
> as.integer(pi)     # numeric(double) to numeric(integer)
[1] 3
> as.logical(1)     # numeric to logical
[1] TRUE
> as.logical(-4)     # numeric to logical
[1] TRUE
> as.complex(4)     # numeric to complex
[1] 4+0i
> as.character(pi)     # numeric to character
[1] "3.14159265358979"
> as.null(1)     # numeric to NULL
NULL
```

Example (Mode checking)

```
> is.numeric(pi)     # is numeric?
[1] TRUE
> is.double(pi)     # is double?
[1] TRUE
> is.integer(pi)     # is integer?
[1] FALSE
> is.logical(T)     # is logical?
[1] TRUE
> is.complex(4i)     # is complex?
[1] TRUE
> is.character('abc')     # is character?
[1] TRUE
> is.na(NA)     # is NA?
[1] TRUE
> is.null(25)     # is NULL?
[1] FALSE
> is.nan(log(-5))     # is NaN?
[1] TRUE
경고 메시지가 손실되었습니다
In log(-5) : 계산결과가 NaN가 생성되었습니다
> is.finite(25)     # is finite?
[1] TRUE
> is.infinite(1/0)     # is infinite?
[1] TRUE
```

2.2. Data Object

R has several types of data objects to handle specific data, including *vector*, *matrix*, *data frame*, *array*, *list*, *factor* (for categorical data), *ts* (for time series data), etc..

2.3. Vector

- Vector is the very basic data type.
- A vector consists of one or more elements whose modes should be the same.

Example (Create vectors)

```
> c(1,3,5)
[1] 1 3 5
> c(1, 0.1, 0.02)
[1] 1.00 0.10 0.02
> c('A', 'B', 'C')
[1] "A" "B" "C"
> c(2+4i, 3, 1i)
[1] 2+4i 3+0i 0+1i
> c(T,T,F)
[1]  TRUE  TRUE FALSE
> c(T,0,F)
[1] 1 0 0
> c(a=1,b=2)     # assign names to elements
a b
1 2
```

Example (Create vectors)

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> -5:4
 [1] -5 -4 -3 -2 -1  0  1  2  3  4
> 4:-5
 [1]  4  3  2  1  0 -1 -2 -3 -4 -5
> 1.5:5
[1] 1.5 2.5 3.5 4.5
> 0:pi
[1] 0 1 2 3
> options(digits=2)     # 2 digits after period
> -pi:pi
[1] -3.14 -2.14 -1.14 -0.14  0.86  1.86  2.86
> options(digits=7)
```

- ':' creates a sequence from the left argument up to the right argument by adding or subtracting 1.

Example (Create vectors)

```
> seq(from=3, to=5, by=0.2)
 [1] 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0
> seq(from=3, to=5, length=11)
 [1] 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0
> seq(from=0, to=10, by=2)
[1]  0  2  4  6  8 10
> seq(0,10,2)
[1]  0  2  4  6  8 10
> seq(10,0,2)
다음에 오류seq.default(10, 0, 2) : 'by' 인수에 잘못한 부호가 있습니다
> seq(by=2, to=10, from=0)
[1]  0  2  4  6  8 10
> seq(0,10,1.7)
[1] 0.0 1.7 3.4 5.1 6.8 8.5
> seq(10,0,-1.7)
[1] 10.0  8.3  6.6  4.9  3.2  1.5
> seq(5)
[1] 1 2 3 4 5
```

Example (Create vectors)

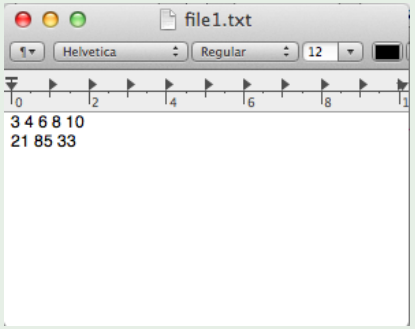
```
> rep(c('A','B','C'), 2)
[1] "A" "B" "C" "A" "B" "C"
> rep(1:3,c(3,2,3))
[1] 1 1 1 2 2 3 3 3
> rep(c('A','B','C'),each=2)
[1] "A" "A" "B" "B" "C" "C"
```

Example (Create vectors)

```
> data1 <- scan()
1: 2 5 7 NA
5: 4 8 NA 9
9:
Read 8 items
> data1
[1]  2  5  7 NA  4  8 NA  9
> scan(what='')     # create a character sequence
1: Hello World !
4:
Read 3 items
[1] "Hello" "World" "!"
> scan(what=complex())     # create a complex sequence
1: 3 4 5
4:
Read 3 items
[1] 3+0i 4+0i 5+0i
```

Example (scan() function)

```
> data2 <- scan(file='file1.txt')
Read 8 items
> data2
[1]  3  4  6  8 10 21 85 33
> data3 <- scan(file='file1.txt',what=character())
Read 8 items
> data3
[1] "3"  "4"  "6"  "8"  "10" "21" "85" "33"
```



Compute the below with R:

(1) $1^2 + 2^2 + \cdots + 10^2$

(2) $\cos(\frac{1}{20}\pi) + \cos(\frac{2}{20}\pi) + \cdots + \cos(\frac{20}{20}\pi)$

Example (Vector calculation)

```
> sum(1:10^2); sum((1:10)^2)      # (1)
[1] 5050
[1] 385
> sum(cos((1:20)*pi/20)); sum(cos(1:20*pi/20))      # (2)
[1] -1
[1] -1
```

Example (Recycle rule)

```
> c(1,2,3) + rep(3,3)
[1] 4 5 6
> c(1,2,3) * rep(3,3)
[1] 3 6 9
> c(1,2,3) + 3
[1] 4 5 6
> c(1,2,3) * 3
[1] 3 6 9
> 1:6 + 1:3
[1] 2 4 6 5 7 9
> 1:6 + 1:5
[1]  2  4  6  8 10  7
경고 메시지가 손실되었습니다
In 1:6 + 1:5 :
  긴 개체의 길이가 더 짧은 개체의 길이의 배수가 되어 있지 않습니다
```

Example (Vector attributes)

```
> age1 <- c(22,25,20)
> age1
[1] 22 25 20
> names(age1)
NULL
> names(age1) <- c('Kim', 'Lee', 'Park')
> age1
 Kim  Lee Park
  22   25   20
> names(age1)
[1] "Kim"  "Lee"  "Park"
> age2 <- c(Kim=22, Lee=25, Park=20)
> names(age2)
[1] "Kim"  "Lee"  "Park"
> mode(age1)
[1] "numeric"
> length(age1)
[1] 3
```

- Vector attributes: mode, names, length

Example (Extraction and manipulation of elements from a vector)

```
> data1 <- (1:5)^2
> data1
[1]  1  4  9 16 25
> data1[3]     # extract the 3rd element
[1] 9
> data1[-4]     # extract all elements except for the 4th element
[1]  1  4  9 25
> data1[c(1,2,4)]    # extract the 1st, 2nd, and 4th elements
[1]  1  4 16
> data1[-c(1,2)]     # extract all elements except for the 1st and 2nd elements
[1]  9 16 25
```

Example (Extraction and manipulation of elements from a vector)

```
> data1
[1]  1  4  9 16 25
> data1[data1>5]     # extract elements greater than 5
[1]  9 16 25
> data1>5
[1] FALSE FALSE  TRUE  TRUE  TRUE
> data1[c(F,F,T,T,T)]     # extract 3th, 4th, 5th elements
[1]  9 16 25
> data1[F]     # c(F,F,F,F,F) from recycle rule
numeric(0)
> data1[T]     # c(T,T,T,T,T) from recycle rule
[1]  1  4  9 16 25
> data1[c(T,F)]     # c(T,F,T,F,T) from recycle rule
[1]  1  9 25
> age2
 Kim  Lee Park
  22   25   20
> age2[names(age2)=='Lee']
Lee
 25
```

Example (Extraction and manipulation of elements from a vector)

```
> data1
[1]  1  4  9 16 25
> data1 <- data1 + 3
> data1
[1]  4  7 12 19 28
> data1[3] <- 10
> data1
[1]  4  7 10 19 28
> data1 <- c(data1, 9)
> data1
[1]  4  7 10 19 28  9
> data1 <- c(1,data1,data1)
> data1
 [1]  1  4  7 10 19 28  9  4  7 10 19 28  9
```

Example (Applying function to vectors)

```
> paste('page', 1:10)
 [1] "page 1"  "page 2"  "page 3"  "page 4"  "page 5"  "page 6"  "page 7"
 [8] "page 8"  "page 9"  "page 10"
```

Example (Applying function to vectors)

```
> month.name
 [1] "January"   "February"  "March"     "April"     "May"       "June"
 [7] "July"      "August"    "September" "October"   "November"  "December"
> month.abb
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"
[19] "s" "t" "u" "v" "w" "x" "y" "z"
> LETTERS
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R"
[19] "S" "T" "U" "V" "W" "X" "Y" "Z"
```

Example (Applying function to vectors)

```
> month.kor<-paste(1:12,'월',sep='')
> month.kor
 [1] "1월"  "2월"  "3월"  "4월"  "5월"  "6월"  "7월"  "8월"  "9월"  "10월"
[11] "11월" "12월"
```

Example (Applying function to vectors)

```
> data2 <- c(1,3,5,NA,7,8,NA)
> sum(data2)
[1] NA
> sum(data2,na.rm=T)     # sum after removing NA
[1] 24
> data3<-rnorm(7)     # generate 7 random numbers
                      # from the standard normal distribution
> data3
[1]  1.0642870 -1.8406357  1.1250090 -0.8578492  1.2329671  2.4614328
[7] -0.4590729
> round(data3)     # round
[1]  1 -2  1 -1  1  2  0
> round(data3,digits=2)     # round unto 2 digits from the period
[1]  1.06 -1.84  1.13 -0.86  1.23  2.46 -0.46
> round(data3,digits=c(2,3))
[1]  1.060 -1.841  1.130 -0.858  1.230  2.461 -0.460
> mean(data3)
[1] 0.3894483
> sd(data3)
[1] 1.486053
```

Note that your random numbers are different from those in the example.

Example (Checking mode of vectors)

```
> (vec<-3:6)
[1] 3 4 5 6
> is.vector(vec)
[1] TRUE
> is.numeric(vec)
[1] TRUE
> is.integer(vec)
[1] TRUE
> is.complex(vec)
[1] FALSE
```

2.4. Matrix

- Matrix is the 2-dimensional data structure where all elements are in the same mode.
- Attributes of matrix: `mode`, `length`, `dim` (dimension), `dimnames` (dimension names)
- Examples
  - $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$: a numeric matrix of 2 rows, 3 columns and 6 length.
  - $\begin{bmatrix} A & D \\ B & E \\ C & F \end{bmatrix}$: a character matrix of 3 rows, 2 columns and 6 length.

To make a matrix $\begin{bmatrix} 7 & 1 & 8 \\ 4 & 6 & 7 \\ 3 & 4 & 5 \end{bmatrix}$, there are several ways to do it:

Example (Creating matrix 1: `rbind()` function)

```
> row1 <- c(7,1,8)
> row2 <- c(4,6,7)
> row3 <- c(3,4,5)
> A1 <- rbind(row1,row2,row3)
> A1
     [,1] [,2] [,3]
row1    7    1    8
row2    4    6    7
row3    3    4    5
```

Example (Creating matrix 2: `cbind()` function)

```
> col1 <- c(7,4,3)
> col2 <- c(1,6,4)
> col3 <- c(8,7,5)
> A1 <- cbind(col1,col2,col3)
> A1
     col1 col2 col3
[1,]    7    1    8
[2,]    4    6    7
[3,]    3    4    5
```

Example (Creating matrix 3: `matrix()` function)

```
> vec1 <- c(7,4,3,1,6,4,8,7,5)
> matrix(vec1,nrow=3)
     [,1] [,2] [,3]
[1,]    7    1    8
[2,]    4    6    7
[3,]    3    4    5
> vec2 <- c(7,1,8,4,6,7,3,4,5)
> matrix(vec2, ncol=3, byrow=T)
     [,1] [,2] [,3]
[1,]    7    1    8
[2,]    4    6    7
[3,]    3    4    5
```

Example (Creating matrix 4: `dim()` function)

```
> A1 <- c(7,4,3,1,6,4,8,7,5)
> dim(A1) <- c(3,3)
> A1
     [,1] [,2] [,3]
[1,]    7    1    8
[2,]    4    6    7
[3,]    3    4    5
```

Example (row vector, column vector)

```
> matrix(1:4, nrow=1)    # row vector
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
> matrix(1:4, ncol=1)    # column vector
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

Example (diagonal matrix, unit matrix)

```
> diag(1:3)    # diagonal matrix of 1,2,3
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
> diag(rep(1,3))    # 3-dim. diagonal matrix of 1
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> A1
[1] 7 4 3 1 6 4 8 7 5
> diag(A1)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    7    0    0    0    0    0    0    0    0
 [2,]    0    4    0    0    0    0    0    0    0
 [3,]    0    0    3    0    0    0    0    0    0
 [4,]    0    0    0    1    0    0    0    0    0
 [5,]    0    0    0    0    6    0    0    0    0
 [6,]    0    0    0    0    0    4    0    0    0
 [7,]    0    0    0    0    0    0    8    0    0
 [8,]    0    0    0    0    0    0    0    7    0
 [9,]    0    0    0    0    0    0    0    0    5
> diag(diag(A1))    # extract the diagonal elements from the diagonal matrix
[1] 7 4 3 1 6 4 8 7 5
> diag(3)    # 3-dim. unit matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Example (matrix algebra)

```
> mat1 <- matrix(1:4,nrow=2)
> mat2 <- diag(c(5,10))
> mat1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> mat2
     [,1] [,2]
[1,]    5    0
[2,]    0   10
> mat1 + mat2
     [,1] [,2]
[1,]    6    3
[2,]    2   14
> mat1 * mat2
     [,1] [,2]
[1,]    5    0
[2,]    0   40
```

```
> mat1 %*% mat2
     [,1] [,2]
[1,]    5   30
[2,]   10   40
> solve(mat1)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
> solve(mat1) %*% mat1
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> t(mat1)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Example (matrix and scalar)

```
> mat1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> mat1 * 3
     [,1] [,2]
[1,]    3    9
[2,]    6   12
> mat1 / 3
          [,1]     [,2]
[1,] 0.3333333 1.000000
[2,] 0.6666667 1.333333
> mat1 + 3
     [,1] [,2]
[1,]    4    6
[2,]    5    7
> 3 - mat1
     [,1] [,2]
[1,]    2    0
[2,]    1   -1
> rbind(1,mat1)
     [,1] [,2]
[1,]    1    1
[2,]    1    3
[3,]    2    4
```

Example (matrix and vector)

```
> mat3 <- matrix(1:16, nrow=4)
> mat3
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> mat3 + c(10,20)
     [,1] [,2] [,3] [,4]
[1,]   11   15   19   23
[2,]   22   26   30   34
[3,]   13   17   21   25
[4,]   24   28   32   36
> mat3 / c(10,20)
     [,1] [,2] [,3] [,4]
[1,]  0.1  0.5  0.9  1.3
[2,]  0.1  0.3  0.5  0.7
[3,]  0.3  0.7  1.1  1.5
[4,]  0.2  0.4  0.6  0.8
> mat3 + c(10,20,30)
     [,1] [,2] [,3] [,4]
[1,]   11   25   39   23
[2,]   22   36   20   34
[3,]   33   17   31   45
[4,]   14   28   42   26
경고 메시지가 손실되었습니다
In mat3 + c(10, 20, 30) :
  긴 개체의 길이가 더 짧은 개체의 길이의 배수가 되어 있지 않습니다
```

Example (attributes of matrix)

```
> logical.mat1 <- matrix(c(T,T,F,T), nrow=2)
> logical.mat1
     [,1]  [,2]
[1,] TRUE FALSE
[2,] TRUE  TRUE
> mode(logical.mat1)
[1] "logical"
> dim(logical.mat1)
[1] 2 2
> nrow(logical.mat1)
[1] 2
> ncol(logical.mat1)
[1] 2
> length(logical.mat1)
[1] 4
> dimnames(logical.mat1)
NULL
> dimnames(logical.mat1) <- list(c('row1','row2'),c('col1','col2'))
> dimnames(logical.mat1)
[[1]]
[1] "row1" "row2"

[[2]]
[1] "col1" "col2"

> rownames(logical.mat1)
[1] "row1" "row2"
> colnames(logical.mat1)
[1] "col1" "col2"
```

Example (extracting and manipulating elements of matrix)

```
> mat4 <- matrix(1:9,nrow=3)^2
> mat4
     [,1] [,2] [,3]
[1,]    1   16   49
[2,]    4   25   64
[3,]    9   36   81
> mat4[2,3]     # extract the (2,3) element
[1] 64
> mat4[c(2,3),-2]    # extract 2nd and 3rd rows without the 2nd column
     [,1] [,2]
[1,]    4   64
[2,]    9   81
> mat4[T,c(T,F,T)]    # extract the 1st and 3rd columns
     [,1] [,2]
[1,]    1   49
[2,]    4   64
[3,]    9   81
> mat4 <- mat4 + 100     # add 100 to all elements
> mat4
     [,1] [,2] [,3]
[1,]  101  116  149
[2,]  104  125  164
[3,]  109  136  181
> mat4 <- cbind(mat4[,-3],0)     # replace the 3rd column by 0
> mat4
     [,1] [,2] [,3]
[1,]  101  116    0
[2,]  104  125    0
[3,]  109  136    0
```

Example (function to matrix)

```
> (mat2 <- matrix(rnorm(9),ncol=3))
           [,1]       [,2]       [,3]
[1,] -2.8984756  1.8173616 -2.0199824
[2,]  0.3211569 -0.3933528  0.3294169
[3,] -0.6340881 -1.4164413  0.2060090
> round(mat2,2)
      [,1]  [,2]  [,3]
[1,] -2.90  1.82 -2.02
[2,]  0.32 -0.39  0.33
[3,] -0.63 -1.42  0.21
> sum(mat2)
[1] -4.688396
> mean(mat2)
[1] -0.5209329
> cos(mat2[1:2,])
           [,1]       [,2]       [,3]
[1,] -0.9705923 -0.2440746 -0.4342325
[2,]  0.9488709  0.9236292  0.9462311
```

Example

```
> (mat1<-matrix(1:4,ncol=2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> is.matrix(mat1)
[1] TRUE
> is.vector(mat1)
[1] FALSE
> (vec1<-as.vector(mat1))    # convert matrix to vector
[1] 1 2 3 4
> is.matrix(vec1)
[1] FALSE
> as.matrix(vec1)    # convert vector to matrix of size 4*1
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
> is.matrix(as.matrix(vec1))
[1] TRUE
> dim(as.matrix(vec1))
[1] 4 1
> (vec2=c(mat1[,1],mat1[,2]))
[1] 1 2 3 4
```

The below table shows the total sales of a convenient stores in a high school during a week.

구 분	월요일	화요일	수요일	목요일	금요일
음료수	210	234	310	340	210
라 면	300	340	420	320	230
문구류	210	220	230	190	180

1. Make a matrix in R
2. Total sales by day
3. Total sales by 구분

Example

```
> drink <-c(210,234,310,340,210)
> ramen <-c(300,340,420,320,230)
> stationery <-c(210,220,230,190,180)
> (school.store<-rbind(drink,ramen,stationery))
           [,1] [,2] [,3] [,4] [,5]
drink       210  234  310  340  210
ramen       300  340  420  320  230
stationery  210  220  230  190  180
> dimnames(school.store)
[[1]]
[1] "drink"      "ramen"      "stationery"

[[2]]
NULL

> dimnames(school.store)[[2]]<-c('Mon','Tue','Wed','Thu','Fri')
> school.store
           Mon Tue Wed Thu Fri
drink      210 234 310 340 210
ramen      300 340 420 320 230
stationery 210 220 230 190 180
> apply(school.store,2,sum)
Mon Tue Wed Thu Fri
720 794 960 850 620
> apply(school.store,1,sum)
     drink      ramen stationery
      1304       1610       1030
```

Example (apply() function)

```
> std1 <- function(x) sqrt(var(x))     # user-defined function (standard deviation)
> std1(1:5)
[1] 1.581139
> apply(school.store,1,std1)
     drink      ramen stationery
  60.35893   68.70226   20.73644
```

2.5. Array

- Array is the 2 or more dimensional data structure where all elements are in the same mode.
- Extraction and manipulation of elements in array is very similar to those for matrix.
- We will not go over the detail for array in this class.

Example

```
> array(1:24,dim=c(3,4,2),dimnames=NULL)
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

2.6. Factor

- Factor is for the categorical variables.
- There are two kinds of factor: factor for nominal variables and ordered factor for ordinal variables
- Factor consists of levels.

Example

```
> age1 <- c(3,2,1,2,3,2)
> fac1 <- factor(age1, labels=c('TEENS','TWENTIES','THIRTIES'))
> fac1
[1] THIRTIES TWENTIES TEENS    TWENTIES THIRTIES TWENTIES
Levels: TEENS TWENTIES THIRTIES
> ord1 <- ordered(age1, labels=c('TEENS','TWENTIES','THIRTIES'))
> ord1
[1] THIRTIES TWENTIES TEENS    TWENTIES THIRTIES TWENTIES
Levels: TEENS < TWENTIES < THIRTIES
> ord2 <- ordered(fac1)
> ord2
[1] THIRTIES TWENTIES TEENS    TWENTIES THIRTIES TWENTIES
Levels: TEENS < TWENTIES < THIRTIES
```

Example (ordered factor)

```
> fives<-c('one','two','three','four','five')
> ordered(fives)
[1] one   two   three four  five
Levels: five < four < one < three < two
> ordered(fives,levels=fives)
[1] one   two   three four  five
Levels: one < two < three < four < five
```

- Factor levels are alphabetical order in default setting.
- `levels=` option can be used for the specific ordering.

Example (attributes and manipulation of factor)

```
> mode(fac1)
[1] "numeric"
> length(fac1)
[1] 6
> levels(fac1)
[1] "TEENS"    "TWENTIES" "THIRTIES"
> fac1[2]
[1] TWENTIES
Levels: TEENS TWENTIES THIRTIES
> fac1[2]<-'TEENS'
> fac1
[1] THIRTIES TEENS    TEENS    TWENTIES THIRTIES TWENTIES
Levels: TEENS TWENTIES THIRTIES
```

Example (2-way cross table using factor)

```
> name<-c(1,2,3,2,3,1)
> surname<-factor(name,labels=c('Kim','Lee','Park'))
> surname
[1] Kim  Lee  Park Lee  Park Kim
Levels: Kim Lee Park
> age<-c(3,2,1,2,3,2)
> ages<-factor(age,labels=c('TEENS','TWENTIES','THIRTIES'))
> ages
[1] THIRTIES TWENTIES TEENS    TWENTIES THIRTIES TWENTIES
Levels: TEENS TWENTIES THIRTIES
> tbl<-table(surname,ages)
> tbl
       ages
surname TEENS TWENTIES THIRTIES
   Kim      0        1        1
   Lee      0        2        0
   Park     1        0        1
> apply(tbl,1,sum)
 Kim  Lee Park
   2    2    2
> apply(tbl,2,sum)
   TEENS TWENTIES THIRTIES
       1        3        2
```

Example (2-way cross table using factor)

```
> prop.table(tbl)     # proportion
       ages
surname     TEENS  TWENTIES  THIRTIES
   Kim  0.0000000 0.1666667 0.1666667
   Lee  0.0000000 0.3333333 0.0000000
   Park 0.1666667 0.0000000 0.1666667
> prop.table(tbl,1)    # row proportion
       ages
surname TEENS TWENTIES THIRTIES
   Kim    0.0      0.5      0.5
   Lee    0.0      1.0      0.0
   Park   0.5      0.0      0.5
> prop.table(tbl,2)    # column proportion
       ages
surname     TEENS  TWENTIES  THIRTIES
   Kim  0.0000000 0.3333333 0.5000000
   Lee  0.0000000 0.6666667 0.0000000
   Park 1.0000000 0.0000000 0.5000000
```

Example (tapply() function)

```
> tapply(age,surname,length)
 Kim  Lee Park
   2    2    2
> tapply(name,ages,length)
   TEENS TWENTIES THIRTIES
       1        3        2
> tapply(age,surname,function(age) length(age)/length(surname))
      Kim       Lee      Park
0.3333333 0.3333333 0.3333333
> tapply(name,ages,function(name) length(name)/length(ages))
    TEENS  TWENTIES  THIRTIES
0.1666667 0.5000000 0.3333333
```

Example (tapply() function)

```
> year<-c(1,2,3,2,3,3,2,1,2,1)
> pencil<-c(5,2,3,4,5,2,5,3,5,6)
> tapply(pencil,year,mean)
       1        2        3
4.666667 4.000000 3.333333
```

2.7. List

- Different from objects that we saw, list object can have different modes and lengths of elements
- List is the object that consists of heterogeneous types of variables.
- List consists of components.

Example (Creating list)

```
> mat1 <- matrix(1:4,nrow=2)
> list1<-list('A',1:8,mat1)
> list1
[[1]]
[1] "A"
[[2]]
[1] 1 2 3 4 5 6 7 8
[[3]]
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> son<-list(son.name=c('sangmin','sangwon'),son.cnt=2,son.age=c(2,6))
> son
$son.name
[1] "sangmin" "sangwon"
$son.cnt
[1] 2
$son.age
[1] 2 6
```

Example (attributes of list)

```
> mode(list1)
[1] "list"
> length(list1)     # number of components
[1] 3
> names(list1)     # component name is not defined
NULL
> mode(son)     # mode of list is always list
[1] "list"
> length(son)
[1] 3
> names(son)     # names of components
[1] "son.name" "son.cnt"  "son.age"
```

Example (extracting components from list)

```
> list1[[2]]
[1] 1 2 3 4 5 6 7 8
> list1[2]
[[1]]
[1] 1 2 3 4 5 6 7 8
> list1[[3]][1,2]
[1] 3
> son$son.name
[1] "sangmin" "sangwon"
> son[[1]]
[1] "sangmin" "sangwon"
> son$son.name[2]
[1] "sangwon"
```

Example (naming components)

```
> names(list1)=c('character','vector','matrix')
> list1
$character
[1] "A"

$vector
[1] 1 2 3 4 5 6 7 8

$matrix
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Example (adding components)

```
> list1[[4]]<-letters[1:4]    # adding the 4th component
> list1[5]<-list(c(T,F))     # adding the 5th component
> list1[[2]][9]<-9     # a element is added to the 2nd component
> list1
$character
[1] "A"
$vector
[1] 1 2 3 4 5 6 7 8 9
$matrix
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[[4]]
[1] "a" "b" "c" "d"
[[5]]
[1]  TRUE FALSE
> son$son.birthday<-c('2000/01/13','2004/04/11')# way 1 of adding components to a list
> son<-c(son,list(son.height=c(156,150)))     # way 2 of adding components to a list
> son
$son.name
[1] "sangmin" "sangwon"
$son.cnt
[1] 2
$son.age
[1] 2 6
$son.birthday
[1] "2000/01/13" "2004/04/11"
$son.height
[1] 156 150
```

Example (deleting components)

```
> list1[5]<-NULL     # deleting the 5th component
> list1[[4]]<-NULL    # deleting the 4th component
> list1[[2]]<-list1[[2]][-9]     # deleting the 9th element of the 2nd component
> list1
$character
[1] "A"

$vector
[1] 1 2 3 4 5 6 7 8

$matrix
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> son$son.birthday<-NULL     # deleting son.birthday compoent
> son[['son.height']]<-NULL     # deleting son.height component
> son
$son.name
[1] "sangmin" "sangwon"

$son.cnt
[1] 2

$son.age
[1] 2 6
```

Example (list as a component of a list)

```
> sangmin<-list(name='sangmin',age=6,sex='M')
> sangwon<-list(name='sangwon',age=2,sex='M')
> child<-list(cnt=2,child=list(sangmin,sangwon))
> child$cnt
[1] 2
> child
$cnt
[1] 2

$child
$child[[1]]
$child[[1]]$name
[1] "sangmin"
$child[[1]]$age
[1] 6
$child[[1]]$sex
[1] "M"
$child[[2]]
$child[[2]]$name
[1] "sangwon"
$child[[2]]$age
[1] 2
$child[[2]]$sex
[1] "M"

> mode(child$child)
[1] "list"
```

Example (unlist)

```
> son
$son.name
[1] "sangmin" "sangwon"

$son.cnt
[1] 2

$son.age
[1] 2 6

> unlist(son)
son.name1 son.name2   son.cnt  son.age1  son.age2
"sangmin" "sangwon"       "2"       "2"       "6"
```

Example

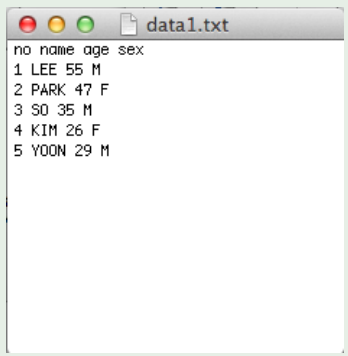
```
> is.list(son)
[1] TRUE
> mat1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> is.list(mat1)
[1] FALSE
> as.list(mat1)
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

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

2.8. Data frame

- Data frame is a data object for data analysis.
- Data frames have features similar with matrix and list: (1) Its shape is a rectangular as a matrix and (2) columns can have different modes as a list.
- Rows of data frame are observations and columns of data frame are variables.

Example (creating data frame: `read.table()`)

```
> data1<-read.table('data1.txt')
> data1
  V1   V2  V3  V4
1 no name age sex
2  1  LEE  55   M
3  2 PARK  47   F
4  3   SO  35   M
5  4  KIM  26   F
6  5 YOON  29   M
```



Example (creating data frame: `read.table()`)

```
> data2<-read.table('data1.txt',row.names='no',header=T)
> data2
  name age sex
1  LEE  55   M
2 PARK  47   F
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
> col1<-data2[,1]
> col2<-data2[,2]
> col3<-data2[,3]
> col1
[1] LEE  PARK SO   KIM  YOON
Levels: KIM LEE PARK SO YOON
> class(col1)     # the first column is a factor
[1] "factor"
> class(col2)     # the second column is numeric(integer)
[1] "integer"
> class(col3)     # the third column is a factor
[1] "factor"
```

- 'header' option says the first row is the names of variables.
- `row.names='no'` denotes that the variable 'no' is the row names.

Example (creating data frame: data.frame())

```
> data3<-scan('data1.txt',list(no=0,name='',age=0,sex=''),skip=1)
Read 5 records
> data3
$no
[1] 1 2 3 4 5
$name
[1] "LEE"  "PARK" "SO"   "KIM"  "YOON"
$age
[1] 55 47 35 26 29
$sex
[1] "M" "F" "M" "F" "M"
> no<-data3$no
> name<-data3$name
> age<-data3$age
> sex<-data3$sex
> data4<-data.frame(no,name,age,sex)
> data4
  no name age sex
1  1  LEE  55   M
2  2 PARK  47   F
3  3   SO  35   M
4  4  KIM  26   F
5  5 YOON  29   M
```

- In the option 'list', the first and third columns are numeric and the second and fourth columns are characters: '0' stands for numeric and ''' stands for character.
- The option skip=1 means the first row is skipped.

Example (creating data frame: data.frame())

```
> data5<-data.frame(no=1:5,
+                   name=c('LEE','PARK','SO','KIM','YOON'),
+                   age=c(55,47,35,26,29),
+                   sex=c('M','F','M','F','M'))
> data5
  no name age sex
1  1  LEE  55   M
2  2 PARK  47   F
3  3   SO  35   M
4  4  KIM  26   F
5  5 YOON  29   M
```

Example (attributes of data frame – list & matrix)

```
> data2
  name age sex
1  LEE  55   M
2 PARK  47   F
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
> mode(data2)
[1] "list"
> length(data2)
[1] 3
> names(data2)
[1] "name" "age"  "sex"
> row.names(data2)
[1] "1" "2" "3" "4" "5"
```

```
> dim(data2)
[1] 5 3
> dimnames(data2)
[[1]]
[1] "1" "2" "3" "4" "5"
[[2]]
[1] "name" "age"  "sex"
> colnames(data2)
[1] "name" "age"  "sex"
> rownames(data2)
[1] "1" "2" "3" "4" "5"
> ncol(data2)
[1] 3
> nrow(data2)
[1] 5
```

Example (manipulating components of data frame)

```
> data2
  name age sex
1  LEE  55   M
2 PARK  47   F
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
> data2[2,1]     # the first variable of the second observation
[1] PARK
Levels: KIM LEE PARK SO YOON
> data2[[1]][2]     # the first variable of the second observation
[1] PARK
Levels: KIM LEE PARK SO YOON
> data2[data2$name=='LEE',]     # observation of name='LEE'
  name age sex
1  LEE  55   M
> data2[,2]     # the second component (or variable)
[1] 55 47 35 26 29
> data2$age     # the second (age) component -- same as a list
[1] 55 47 35 26 29
> data2[[2]]     # the second (age) component -- same as a list
[1] 55 47 35 26 29
> is.list(data2)
[1] TRUE
> is.data.frame(data2)
[1] TRUE
```

Example (adding observations and variables)

```
> levels(data2$name)<-c(levels(data2$name),'RYU')     # add new level
> data2<-rbind(data2,c('RYU',36,'M'))     # add new row (observation)
> data2[7,]<-c('CHOI',41,'F')     # 'CHOI' is not in the set of levels
경고 메시지가 손실되었습니다
In `[<-.factor`(`*tmp*`, iseq, value = "CHOI") :
  invalid factor level, NAs generated
> data2
  name age sex
1  LEE  55   M
2 PARK  47   F
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
6  RYU  36   M
7 <NA>  41   F
> class(data2$age)
[1] "character"
> data2$age
[1] "55" "47" "35" "26" "29" "36" "41"
> data2$age<-as.numeric(data2$age)     # convert age into numeric
> data2<-cbind(data2,married=c(T,T,T,F,F,T,T))    # add new column (variable)
> data2
  name age sex married
1  LEE  55   M    TRUE
2 PARK  47   F    TRUE
3   SO  35   M    TRUE
4  KIM  26   F   FALSE
5 YOON  29   M   FALSE
6  RYU  36   M    TRUE
7 <NA>  41   F    TRUE
```

Example (merge data frames)

```
> df1<-data.frame(name=c('LEE','PARK','SO','KIM','YOON'),
+                 age=c(55,47,35,26,29),
+                 sex=c('M','F','M','F','M'))
> df2<-data.frame(name=c('KIM','SO','LEE','YOON','PARK'),
+                 married=c(F,T,T,F,T))
> df1
  name age sex
1  LEE  55   M
2 PARK  47   F
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
> df2
  name married
1  KIM   FALSE
2   SO    TRUE
3  LEE    TRUE
4 YOON   FALSE
5 PARK    TRUE
> merge(df1,df2)     # merge two data frames with a key 'name'
  name age sex married
1  KIM  26   F   FALSE
2  LEE  55   M    TRUE
3 PARK  47   F    TRUE
4   SO  35   M    TRUE
5 YOON  29   M   FALSE
```

Example (merge data frames)

```
> df3<-data.frame(surname=c('KIM','SO','LEE','YOON','PARK'),
+                 married=c(F,T,T,F,T))
> merge(df1,df3,by.x='name',by.y='surname')
  name age sex married
1  KIM  26   F   FALSE
2  LEE  55   M    TRUE
3 PARK  47   F    TRUE
4   SO  35   M    TRUE
5 YOON  29   M   FALSE
```

- The key variable can be defined using the options '`by.x`', and '`by.y`'.

Example (merge data frames)

```
> df4<-data.frame(name=c('LEE','PARK','SO','KIM','YOON'), sex=c('M','M','F','F','M'),
+                 married=c(T,T,F,T,F))
> df1
  name age sex
1  LEE  55   M
2 PARK  47   F
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
> df4
  name sex married
1  LEE   M    TRUE
2 PARK   M    TRUE
3   SO   F   FALSE
4  KIM   F    TRUE
5 YOON   M   FALSE
> merge(df1,df4)     # 2 keys (name and sex): intersection
  name sex age married
1  KIM   F  26    TRUE
2  LEE   M  55    TRUE
3 YOON   M  29   FALSE
> merge(df1,df4,all=T)     # 2 keys (name and sex): union
  name sex age married
1  KIM   F  26    TRUE
2  LEE   M  55    TRUE
3 PARK   F  47      NA
4 PARK   M  NA    TRUE
5   SO   F  NA   FALSE
6   SO   M  35      NA
7 YOON   M  29   FALSE
```

Example (delete rows and columns)

```
> data2
  name age sex married
1  LEE  55   M    TRUE
2 PARK  47   F    TRUE
3   SO  35   M    TRUE
4  KIM  26   F   FALSE
5 YOON  29   M   FALSE
6  RYU  36   M    TRUE
7 <NA>  41   F    TRUE
> (data2<-data2[-2,])     # delete the second row
  name age sex married
1  LEE  55   M    TRUE
3   SO  35   M    TRUE
4  KIM  26   F   FALSE
5 YOON  29   M   FALSE
6  RYU  36   M    TRUE
7 <NA>  41   F    TRUE
> (data2<-data2[,-4])     # delete the fourth column
  name age sex
1  LEE  55   M
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
6  RYU  36   M
7 <NA>  41   F
```

Example (edit elements)

```
> data2
  name age sex
1  LEE  55   M
3   SO  35   M
4  KIM  26   F
5 YOON  29   M
6  RYU  36   M
7 <NA>  41   F
> data2[4,2]<-28
> data2
  name age sex
1  LEE  55   M
3   SO  35   M
4  KIM  26   F
5 YOON  28   M
6  RYU  36   M
7 <NA>  41   F
```

Example (access a component)

```
> data2$age
[1] 55 35 26 28 36 41
> mean(data2$age)
[1] 36.83333
```

Example (attach())

```
> attach(data2)
> age
[1] 55 35 26 28 36 41
> detach(data2)
> age
에러: 개체 'age'이 없습니다
> data2$age
[1] 55 35 26 28 36 41
```

- 'attach()' is useful when a data frame has many components (variables).
- It is a good practice to use 'detach()' after you finish using a data frame that is attached in the workspace.