4. R 기본 문법

Data Frame

강의 목표

- Data Frame에 대해 이해한다.
- 타입 식별 함수와 타입 변환 함수에 대해 이해한다.

Data Frame

데이터 프레임(Data Frame)

- 데이터 프레임은 2개의 차원으로 이뤄진 table 형태의 자료
- 처리할 데이터를 엑셀의 스프레드 시트와 같이 표 형태로 정리한 모습을 하고 있음.
- 각 변수는 서로 다른 데이터를 가질 수 있음.
- 단, 데이터 프레임은 모든 관측값의 길이가 같아야 함(원소의 개수)
- 데이터 프레임의 각 열에는 관측값의 이름이 저장되고, 각 행에는 매 관측 단위마다 실제 얻어진 값이 저장됨.

성명	국어	영어
홍길동	80	94
김길동	97	100
박길동	85	97

Datasets

- Observations
- Variabls
- Example: people
  - each person = observation
  - properties (name, age ⋯) = variables
- Matrix? Need different types
- List? Not very practical

name	age	child
Anne	28	FALSE
Pete	30	TRUE
Frank	21	TRUE
Julia	39	FALSE
Cath	35	TRUE

Data Frame

- Specifically for datasets
- Rows = observations (persons)
- Columns = variables (age, name, ⋯)
- Contain elements of different type
- Elements in same column: same type

name	age	child
Anne	28	FALSE
Pete	30	TRUE
Frank	21	TRUE
Julia	39	FALSE
Cath	35	TRUE

Data Frame vs. List vs. Matrix

- Data frames are used to store tabular data
  - A data.frame object in R has similar dimensional properties to a matrix but it may contain categorical data, as well as numeric.
  - They are represented as a special type of list where every element of the list has to have the **same length**
  - Each element of the list can be thought of as a column and the length of each element of the list is the number of rows
  - Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class
  - Data frames are usually created by calling read.table() or read.csv()

Create Data Frame

- Import from data source
- CSV file
- Relational Database (e.g. SQL)
- Software packages (Excel, SPSS …)

Create Data Frame – data.frame()

```
> name <- c("Anne", "Pete", "Frank", "Julia", "Cath")
> age <- c(28, 30, 21, 39, 35)
> child <- c(FALSE, TRUE, TRUE, FALSE, TRUE)
> people <- data.frame(name, age, child)
> people
```

column names match variable names

```
  name age  child
1 Anne  28  FALSE
2 Pete  30   TRUE
3 Frank 21   TRUE
4 Julia 39  FALSE
5 Cath  35   TRUE
```

Name Data Frame

```
> names(people) <- c("Name", "Age", "Child")
> people
    Name Age Child
  1 Anne  28 FALSE
  2 Pete  30 TRUE
  ...
  5 Cath  35 TRUE
> people <- data.frame(Name = name, Age = age, Child = child)
> people
    Name Age Child
  1 Anne  28 FALSE
  2 Pete  30 TRUE
  ...
  5 Cath  35 TRUE
```

```
> people
   Name Age Child
1  Anne  28 FALSE
2  Pete  30  TRUE
3 Frank  21  TRUE
4 Julia  39 FALSE
5  Cath  35  TRUE
```

```
> people
   Name Age Child
1  Anne  28 FALSE
2  Pete  30  TRUE
3 Frank  21  TRUE
4 Julia  39 FALSE
5  Cath  35  TRUE
```

Data Frame Attributes

- Both List and Matrix

```
> names(people)
[1] "Name"  "Age"   "Child"
> rownames(people)
[1] "1" "2" "3" "4" "5"
> people$Age
[1] 28 30 21 39 35
> people$age
  Error in `[.data.frame`(people, , "age") : undefined columns selected
> rownames(people) <- c("s1", "s2", "s3", "s4", "s5"); people
> colnames(people)<-c("name","age","child")
```

```
> people
   Name Age Child
1  Anne  28 FALSE
2  Pete  30  TRUE
3 Frank  21  TRUE
4 Julia  39 FALSE
5  Cath  35  TRUE
```

```
    Name Age Child
s1  Anne  28 FALSE
s2  Pete  30  TRUE
s3 Frank  21  TRUE
s4 Julia  39 FALSE
s5  Cath  35  TRUE
```

```
> people
    name age child
s1  Anne  28 FALSE
s2  Pete  30  TRUE
s3 Frank  21  TRUE
s4 Julia  39 FALSE
s5  Cath  35  TRUE
```

Data Frame Structure

```
> str(people) Factor instead of character
'data.frame': 5 obs. of 3 variables:
    $ Name : Factor w/ 5 levels "Anne","Cath",..: 1 5 3 4 2
    $ Age : num 28 30 21 39 35
    $ Child: logi FALSE TRUE TRUE FALSE TRUE
> data.frame(name[-1], age, child)
Error : arguments imply differing number of rows: 4, 5
> df <- data.frame(name, age, child, stringsAsFactors = FALSE)
> str(people)
'data.frame': 5 obs. of 3 variables:
    $ name : chr "Anne" "Pete" "Frank" "Julia" ...
    $ age : num 28 30 21 39 35
    $ child: logi FALSE TRUE TRUE FALSE TRUE
```

Factor **instead of** character

Subset Data Frame

- Subsetting syntax from matrices and lists
- [from matrices
- [[and $ from lists

Subset Data Frame

```
> people[3,2]
[1] 21
> people[3,"age"]
[1] 21
> people[3,]
      name age child
   s3 Frank 21 TRUE
> people[,“age"]
[1] 28 30 21 39 35
```

```
> people
    name age child
s1  Anne  28 FALSE
s2  Pete  30  TRUE
s3 Frank  21  TRUE
s4 Julia  39 FALSE
s5  Cath  35  TRUE
```

Subset Data Frame

> people[c(3, 5), c("age", "child")]

```
   age child
s3  21  TRUE
s5  35  TRUE
```

> people[2]

```
   age
s1  28
s2  30
s3  21
s4  39
s5  35
```

```
> people
    name age child
s1  Anne  28 FALSE
s2  Pete  30  TRUE
s3 Frank  21  TRUE
s4 Julia  39 FALSE
s5  Cath  35  TRUE
```

Data Frame ~ List

```
> people$age
[1] 28 30 21 39 35
> people[["age"]]
[1] 28 30 21 39 35
> people[[2]]
[1] 28 30 21 39 35
```

```
> people
    name age child
s1  Anne  28 FALSE
s2  Pete  30  TRUE
s3 Frank  21  TRUE
s4 Julia  39 FALSE
s5  Cath  35  TRUE
```

Expanding Data Frames

- Add Columns = add variables
- Add rows = add observations

1. Components can be added to a data frame in the natural way using an assignment
2. Or row binding or column binding
   - If you expand the experiment to add data, use row binding to expand.
   - If other data are kept on the same samples in another data frame, it can be combined with the original using cbind

Add column

```
>people$height <- c(163, 177, 163, 162, 157); people
     name age child height
s1  Anne  28 FALSE    163
s2  Pete  30  TRUE    177
s3 Frank  21  TRUE    163
s4 Julia  39 FALSE    162
s5  Cath  35  TRUE    157
> people1<-people
> people1$birthyear <- paste0("198", 1:5); people1
     name age child birthyear
s1  Anne  28 FALSE      1981
s2  Pete  30  TRUE      1982
s3 Frank  21  TRUE      1983
s4 Julia  39 FALSE      1984
s5  Cath  35  TRUE      1985
```

Add column

```
> weight <- c(74, 63, 68, 55, 56)
> cbind(people, weight)
    name age child height weight
  s1 Anne 28 FALSE 163 74
  s2 Pete 30 TRUE 177 63
  s3 Frank 21 TRUE 163 68
  s4 Julia 39 FALSE 162 55
  s5 Cath 35 TRUE 157 56
> rownames(people)<-1:5
```

Add row

```
> tom <- data.frame("Tom", 37, FALSE, 183)
> rbind(people, tom)
Error : names do not match previous names
> tom <- data.frame(name = "Tom", age = 37,
child = FALSE, height = 183)
> rbind(people, tom)
```

```
     name age child height
s1 Anne 28 FALSE 163
s2 Pete 30 TRUE 177
3 Frank 21 TRUE 163
4 Julia 39 FALSE 162
5 Cath 35 TRUE 157
6 Tom 37 FALSE 183
```

```
> people
   name age child height
1  Anne  28 FALSE    163
2  Pete  30  TRUE    177
3 Frank  21  TRUE    163
4 Julia  39 FALSE    162
5  Cath  35  TRUE    157
```

Sorting

- Often data are better viewed when sorted. The function order sorts a column and gives output that can sort the rows of a data.frame. The following sorts people by age.

```
> sort(people$age)
[1] 21 28 30 35 39
> ranks <- order(people$age)
> ranks
[1] 3 1 2 5 4
> people$age
[1] 28 30 21 39 35
```

```
> people
   name age child height
1  Anne  28 FALSE    163
2  Pete  30  TRUE    177
3 Frank  21  TRUE    163
4 Julia  39 FALSE    162
5  Cath  35  TRUE    157
```

21 is lowest: its index, 3, comes first in ranks

28 is second lowest: its index, 1, comes second in ranks

39 is highest: its index, 4, comes last in ranks

Sorting

```
> sort(people$age)
[1] 21 28 30 35 39
> ranks <- order(people$age)
> ranks
[1] 3 1 2 5 4
> people[ranks, ]
     name age child height
3   Frank  21  TRUE    163
1    Anne  28 FALSE    163
2    Pete  30  TRUE    177
5    Cath  35  TRUE    157
4   Julia  39 FALSE    162
```

```
> people
   name age child height
1  Anne  28 FALSE    163
2  Pete  30  TRUE    177
3 Frank  21  TRUE    163
4 Julia  39 FALSE    162
5  Cath  35  TRUE    157
```

Sorting – Decreasing

```
> sort(people$age)
[1] 21 28 30 35 39
> ranks <- order(people$age)
> ranks
[1] 3 1 2 5 4
> people[order(people$age, decreasing = TRUE), ]
     name age child height
4   Julia  39 FALSE    162
5    Cath  35  TRUE    157
2    Pete  30  TRUE    177
1    Anne  28 FALSE    163
3   Frank  21  TRUE    163
```

```
> people
   name age child height
1  Anne  28 FALSE    163
2  Pete  30  TRUE    177
3 Frank  21  TRUE    163
4 Julia  39 FALSE    162
5  Cath  35  TRUE    157
```

Extracting ALL Components

- All components in a data frame can be extracted as vectors with the corresponding name:

```
> mtcars                    # mtcars dataset
> mean(mtcars$mpg)     # mpg's mean of mtcars
> cor(mtcars$mpg, mtcars$wt ) # correlation between mpg and wt


> attach(mtcars)  # mtcars dataset attach
> mean(mpg)
> cor(mpg, wt )
> detach(mtcars) # mtcars dataset detach
```

Data Frame – Utility Function

- Obtain the first several rows of a matrix or data frame using head()
  - head(x, n=6)
  - x – A matrix, data frame, or vector.
  - n – The first n rows (or values if x is a vector) will be returned.
- And use tail() to obtain the last several rows
  - tail(x, n=6)
  - x – A matrix, data frame, or vector.
  - n – The last n rows (or values if x is a vector) will be returned.
- Use View() to invoke a spreadsheet-style data viewer on a matrix-like R object
  - View(x, title)
  - x- a data frame
  - title – title for viewer window, Defaults to name of x prefixed by Data

Data Frame – Utility Function

```
> mtcars
                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla     33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona      21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger   15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin        15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28         13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird   19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9          27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2      26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa       30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L     15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino       19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E         21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

View(mtcars)

Untitled1* × mtcars ×

Filter

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1

Showing 1 to 22 of 32 entries

```
> tail(mtcars)
               mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

Data Frame 관련 함수들

함 수	기 능
ncol(dataframe)	data frame 의 열의 개수를 구합니다
nrow(dataframe)	data frame 의 행의 개수를 구합니다
names(dataframe)	data frame 의 열 이름을 출력합니다
rownames(dataframe) / row.names(dataframe)	data frame 의 행 이름을 출력합니다.
colnames(dataframe) / col.names(dataframe)	data frame 의 열 이름을 출력합니다.

타입 판별 및 타입 변환

데이터 타입 판별 함수

함수	의미
class(x)	객체 x 클래스
str(x)	객체 x의 내부구조
is.factor(x)	주어진 객체 x가 팩터인가
is.numeric(x)	주어진 객체 x가 숫자를 저장한 벡터인가
is.character(x)	주어진 객체 x가 문자를 저장한 벡터인가
is.matrix(x)	주어진 객체 x가 행렬인가
is.array(x)	주어진 객체 x가 배열인가
is.data.frame(x)	주어진 객체 x가 데이터 프레임인가

데이터 타입 판별 함수

```
> class (c(1, 2))
[1] " numeric "
> class ( matrix (c(1, 2)))
[1] " matrix "
> class ( list (c(1 ,2)))
[1] " list "
> class ( data.frame (x=c(1 ,2)))
[1] " data.frame "
> str(c(1, 2))
num [1:2] 1 2
> str( matrix (c(1 ,2)))
num [1:2 , 1] 1 2
> str( list (c(1 ,2)))
List of 1
$ : num [1:2] 1 2
> str( data.frame (x=c(1 ,2)))
'data.frame ': 2 obs. of 1 variable :
$ x: num 1 2
```

```
> is.numeric (c(1, 2, 3))
[1] TRUE
> is.numeric (c('a', 'b', 'c'))
[1] FALSE
> is.matrix ( matrix (c(1, 2)))
[1] TRUE
```

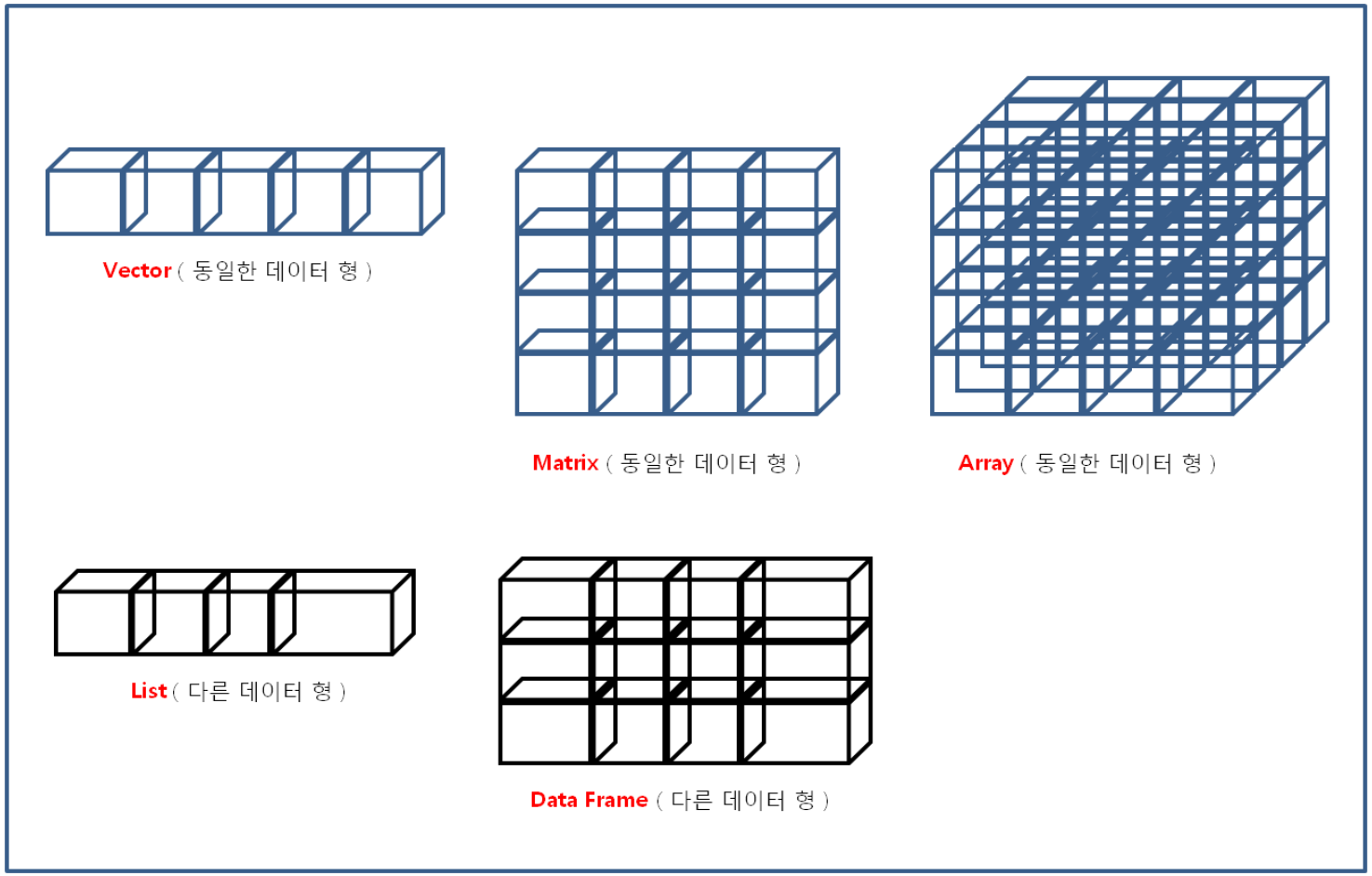
데이터 타입 변환 함수

함수	의미
as.factor(x)	주어진 객체 x를 팩터로 변환
as.numeric(x)	주어진 객체 x를 숫자를 저장한 벡터로 변환
as.character(x)	주어진 객체 x를 문자열을 저장한 벡터로 변환
as.matrix(x)	주어진 객체 x를 행렬로 변환
as.array(x)	주어진 객체 x를 배열로 변환
as.data.frame(x)	주어진 객체 x를 데이터 프레임으로 변환

데이터 타입 변환 함수

```
> x <- data.frame ( matrix (c(1, 2, 3, 4) , ncol =2) )
> x
  X1 X2
1 1 3
2 2 4
> colnames (x) <- c("X", "Y")
> x
   X Y
1 1 3
2 2 4
> data.frame ( list (x=c(1, 2) , y=c(3, 4)))
   x y
1 1 3
2 2 4
> x <- c("m", "f")
> as.factor (x)
[1] m f
Levels : f m
> as.numeric ( as.factor (x))
[1] 2 1
> factor (c("m", "f"), levels =c("m", "f"))
[1] m f
Levels : m f
```

데이터 타입 한눈에 파악하기



참고문헌

- R을 활용한 데이터 분석 – 김성근
- R 라뷰 – 서진수
- Coursera R lecture materials
- Edx R lecture materials
- Steven Buechler, R course Materials

END

Thank you for your attention

경청해주셔서 감사합니다