

## 분석프로그래밍 I.04

### 범주형, 날짜/시간, 문자열 데이터, 그리고 R로 데이터 읽어 오기

국민대학교 경영학부 빅데이터경영통계학과

2018. 3. 26(월)

지난 시간에 R의 여러 데이터 구조를 살펴보았다. 이들은 크게 1차원 구조와 2차원 구조로 나뉘볼 수 있다.

- 1차원 구조 : `vector`, `list`
- 2차원 구조 : `matrix`, `data.frame`

구조에 따라서 참조하는 방법이 달라진다.

- 1차원 구조 : `var[i]`, `var[[i]]`, `var[["element-name"]]`, `var$element-name`
- 2차원 구조 : `var[i, j]`

데이터 분석에서는 주로 데이터 프레임(`data.frame`)을 사용한다. 데이터 프레임에서 자주 사용하는 방법은 다음과 같다.

- 컬럼(열) 하나 선택하기 : `df$coln`, `df$"coln"`, `df[["coln"]]`
- 로우(행) 하나 선택하기 : `df[i, ]`
- 컬럼(열) 하나 제거하기 : `df$coln= NULL`, `df$"coln" = NULL`

특정 조건을 만족하는 열을 선택하기 위해서는 다음과 같은 구문을 자주 사용한다.

```
1 df[df$coln > 2, ]  
2 df[df$coln1 > 2 & df$coln2 < 3, ]
```

하지만 조건이 복잡하게 되면 df가 반복된다. 편의를 위해 subset, with, within, attach 함수를 사용할 수 있다.

```
1 subset(df, subset = coln1>2 & coln2 <3, select=T)  
2  
3 with(df, df[coln>1 & coln2<3, ])  
4 within(df, coln3 = coln1 + coln2)  
5  
6 attach(df)  
7 df[coln>1 & coln2<3, ]  
8 detach()
```

이때 몇 가지 주의할 점이 있다. attach의 경우 열을 수정해도 원래 데이터 프레임에 반영이 되지 않는다. 수정을 하고자 한다면 within을 활용하자. 다음의 예를 보자.

```
1 > attach(mtcars)  
2 > cyl12 <- 2*cyl  
3 > detach(mtcars)  
4 > mtcars$cyl12  
5 NULL  
6 >  
7 > mtcars <- within(mtcars, cyl12 <- 2*cyl)  
8 > mtcars$cyl12  
9 [1] 12 12 8 12 16 12 16 8 8 12 12 16 16 16 16 16 16 8 8 8 8 16  
10      16 16 16 8  
11 [27] 8 8 16 12 16 8  
12 >
```

```

12 > mtcars <- within(mtcars, cyl2 = 2*cyl)
13 Error in eval(substitute(expr), e) : argument is missing, with no
      default
14 >
15 > mtcars <- within(mtcars, {cyl2 <- 2*cyl; hp2 <- 2*hp})
16 > mtcars[1:3, c("cyl2", "hp2")]
17           cyl2 hp2
18 Mazda RX4      12 220
19 Mazda RX4 Wag  12 220
20 Datsun 710       8 186

```

## 1 범주형 데이터, 날짜/시간 데이터, 문자열 데이터

### 1.1 범주형 데이터 (factor, ordered)

데이터 타입 **factor**는 범주형 데이터를 처리하기 위해 쓰인다. 하지만 데이터 추가 등이 까다롭기 때문에 분석 전에는 문자열 (**character**) 데이터 타입으로 유지하다가 분석 바로 전에 **factor**형으로 변환하는 방법을 권장한다.

**factor**형 벡터를 생성하기 위해서는 다음의 함수를 사용한다.

```

1 factor(c("West", "East", "Middle", "East"), levels=c("East", "Middle", "West"))
2 ordered(c("West", "East", "Middle", "East"), levels=c("East", "Middle", "West"))

```

문자열, 팩터, 순위형 데이터 타입을 비교해보자.

```

1 > x <- c("West", "East", "Middle", "East", "Middle")
2 > y <- factor(x, levels=c("East", "Middle", "West"))
3 > z <- ordered(x, levels=c("East", "Middle", "West"))

```

```

4 > x
5 [1] "West" "East" "Middle" "East" "Middle"
6 > y
7 [1] West East Middle East Middle
8 Levels: East Middle West
9 > z
10 [1] West East Middle East Middle
11 Levels: East < Middle < West

```

참고로 `levels`, `nlevels` 함수를 사용하여 `factor`형 벡터의 수준과 수준의 수를 알아낼 수 있다.

```

1 > levels(y)
2 [1] "East" "Middle" "West"
3 > nlevels(y)
4 [1] 3

```

## 1.2 날짜, 시간 데이터(`as.Date`, `as.POSIXct`)

날짜, 시간 데이터는 `lubridate`라는 라이브러리를 사용하면 편리하다. 다음은 R의 기본적인 함수와 `lubridate`의 함수를 비교한다.

```

1 library(lubridate)
2
3 # Parsing a date data and extracting year/month/day
4 # Base R
5 date1 <- as.Date(c("01-06-2018"), format = "%d-%m-%Y")
6 as.numeric(format(date1, "%m"))
7
8 date2 <- as.POSIXct("06-01-2018 11:00:20", format = "%m-%d-%Y %H:%M:%S"
9 ) # options for tz?
10 as.numeric(format(date2, "%m")) #as.POSIXlt(date)$mon + 1

```

```
11 # lubridate : ymd, ydm, mdy, dmy, ymd_hms, ydm_hms, dmy_hms, dmy_hms
12 date1 <- dmy("01-06-2018")
13 month(date1)
14
15 date2 <- ymd_hms("2018-06-01 11:00:20", tz=Sys.timezone())
16 month(date2)
17 month(date2) <- 7
18
19 # Base R vs. lubridate
20 as.Date("09-05-31")
21 as.Date("09-05-31", format="%y-%m-%d")
22 ymd("09-05-31")
23
24 as.Date(c("2010-01-01", "2011/03/01"))
25 ymd(c("2010-01-01", "2011/03/01"))
26
27 date <- as.POSIXct("06-01-2018 11:00:20", format = "%m-%d-%Y %H:%M:%S")
28 # options for tz?
29 date <- ymd_hms("2018-06-01 11:00:20")
30 date <- ymd_hms("2018-06-01 11:00:20", tz="Asia/Seoul")
31 OlsonNames() # timezones
32 Sys.timezone()
33
34 # one day earlier
35 # Base R
36 date <- as.Date("06-01-2018", format="%m-%d-%Y")
37 date - 1 # 1 = 1 day
38
39 datePOSIX <- as.POSIXct("06-01-2018", format="%m-%d-%Y")
40 datePOSIX - 1 # 1 = 1 second
41 datePOSIX - 1*60*60*24 # 1*60*60 seconds = 1 day
42
43 # lubridate
44 date <- mdy("06-01-2018")
45 day(date) = day(date)-1
```

```
46 day(date)
47 date
48
49 # Today's date and current time
50 Sys.Date()
51 today()
52
53 Sys.time()
54 now()
55
56 # accessor functions in lubridate
57 year()
58 month()
59 month(, label=T, abbr=F)
60 yday() # day of year
61 mday() # day of month
62 wday() # day of week
63 wday(, label=T, abbr=F)
64
65 hour()
66 minute()
67 second()
68 tz()
69
70 year() <- 2018
71
72 update(, year=, month=, day=)
73
74
75 # Difference in time
76 as.numeric(as.POSIXct("2018/12/31 23:59:59"))-as.numeric(Sys.time())
77 difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time())
78 difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="weeks")
79 difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="hours")
80 difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="mins")
81 difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="secs")
```

```
82
83 dif = ymd_hms("2018/12/31 23:59:59") - now()
84 dif = difftime(ymd_hms("2018/12/31 23:59:59"), now())
85 now() + dif
```

### 1.3 문자열 데이터 (as.character)

```
1 # paste / sprintf
2 who = c("Sohn", "Jung")
3 where = c("San Diego", "Los Angeles")
4 sprintf("%s told me to go to %s", who, where)
5
6 paste(who, "told me to go to", where)
7
8 cents = c(50 ,10)
9 sprintf("%s gave me %d cents", who, cents)
10 paste(who, "gave me", cents, "cents")
11
12 heights = c(170.2522, 180.231)
13 sprintf("%s is %3.2f centimeters tall", who, heights)
14 paste(who, "is", format(heights, digits=5),"centimeters tall")
15
16 # nchar
17
18 names = c("Kim Beazley", "Tom Chaplin", "Tris Mees", "Phillip Kim")
19 names[nchar(names)<=9]
20
21 # substring
22 names[substring(names,1,3)=="Kim"]
23
24 # grep, grepl
25 grep(pattern = "Kim", names)
26 grepl(pattern = "Kim", names)
27
```

```
28 # sub, gsub
29 sub("i","j",names)
30 gsub("i","j",names)
31
32 # strsplit
33 nam0 <- unlist(strsplit(names, "i"))
34 paste(nam0, collapse="+")
35
36 nam1 <- strsplit(names, "i")
37 nam1l <- lapply(nam1, paste, collapse="+")
38 unlist(nam1l)
```

## 2 R로 데이터 읽어오기

교재의 6장 ‘R로 데이터 읽어오기’를 참조하길 바란다.