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 나눠볼 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.

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

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

```
subset(df, subset = coln1>2 & coln2 <3, select=T)

with(df, df[coln>1 & coln2<3, ])
within(df, coln3 = coln1 + coln2)

attach(df)
df[coln>1 & coln2<3, ]
detach()
```

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

```
> attach(mtcars)
> cyl2 <- 2*cyl
> detach(mtcars)
> mtcars$cyl2
NULL
>
> mtcars <- within(mtcars, cyl2 <- 2*cyl)
> mtcars$cyl2
 [1] 12 12  8 12 16 12 16  8  8 12 12 16 16 16 16 16 16  8  8  8  8 16
    16 16 16  8
[27]  8  8 16 12 16  8
>
> mtcars <- within(mtcars, cyl2 = 2*cyl)
Error in eval(substitute(expr), e) : argument is missing, with no
    default
```

```
>
> mtcars <- within(mtcars, {cyl2 <- 2*cyl; hp2 <- 2*hp})
> mtcars[1:3, c("cyl2", "hp2")]
              cyl2 hp2
Mazda RX4       12 220
Mazda RX4 Wag   12 220
Datsun 710       8 186
```

1 범주형 데이터, 날짜/시간 데이터, 문자열 데이터

1.1 범주형 데이터(factor, ordered)

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

factor형 벡터를 생성하기 위해서는 다음의 함수를 사용한다.

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

문자열, 팩터, 순위형 데이터 타입을 비교해보자.

```
> x <- c("West","East","Middle","East", "Middle")
> y <- factor(x, levels=c("East","Middle","West"))
> z <- ordered(x, levels=c("East","Middle","West"))
> x
[1] "West"   "East"   "Middle" "East"   "Middle"
> y
[1] West   East   Middle East   Middle
Levels: East Middle West
```

```
> z
[1] West   East   Middle East   Middle
Levels: East < Middle < West
```

참고로 levels, nlevels 함수를 사용하여 factor형 벡터의 수준과 수준의 수를 알아낼 수 있다.

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

1.2 날짜, 시간 데이터 (as.Date, as.POSIXct)

날짜, 시간 데이터는 lubridate라는 라이브러리를 사용하면 편리하다. 다음은 R의 기본적인 함수와 lubridate의 함수를 비교한다.

```
library(lubridate)

# Parsing a date data and extracting year/month/day
# Base R
date1 <- as.Date(c("01-06-2018"), format = "%d-%m-%Y")
as.numeric(format(date1, "%m"))

date2 <- as.POSIXct("06-01-2018 11:00:20", format = "%m-%d-%Y %H:%M:%S"
    ) # options for tz?
as.numeric(format(date2, "%m")) #as.POSIXlt(date)$mon + 1

# lubridate : ymd, ydm, mdy, dmy, ymd_hms, ydm_hms, dmy_hms, dmy_hms
date1 <- dmy("01-06-2018")
month(date1)

date2 <- ymd_hms("2018-06-01 11:00:20", tz=Sys.timezone())
```

```
month(date2)
month(date2) <- 7

# Base R vs. lubridate
as.Date("09-05-31")
as.Date("09-05-31", format="%y-%m-%d")
ymd("09-05-31")

as.Date(c("2010-01-01","2011/03/01"))
ymd(c("2010-01-01","2011/03/01"))

date <- as.POSIXct("06-01-2018 11:00:20", format = "%m-%d-%Y %H:%M:%S")
     # options for tz?
date <- ymd_hms("2018-06-01 11:00:20")
date <- ymd_hms("2018-06-01 11:00:20", tz="Asia/Seoul")
OlsonNames() # timezones
Sys.timezone()


# one day earlier
# Base R
date <- as.Date("06-01-2018", format="%m-%d-%Y")
date - 1 # 1 = 1 day

datePOSIX <- as.POSIXct("06-01-2018", format="%m-%d-%Y")
datePOSIX - 1 # 1 = 1 second
datePOSIX - 1*60*60*24 # 1*60*60 seconds = 1 day

# lubridate
date <- mdy("06-01-2018")
day(date) = day(date)-1
day(date)
date

# Today's date and current time
Sys.Date()
```

```
today()

Sys.time()
now()

# accessor functions in lubridate
year()
month()
month(, label=T, abbr=F)
yday()  # day of year
mday()  # day of month
wday()  # day of week
wday(, label=T, abbr=F)

hour()
minute()
second()
tz()

year() <- 2018

update(, year=, month=, day=)


# Difference in time
as.numeric(as.POSIXct("2018/12/31 23:59:59"))-as.numeric(Sys.time())
difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time())
difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="weeks")
difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="hours")
difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="mins")
difftime(as.POSIXct("2018/12/31 23:59:59"), Sys.time(), units="secs")

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

1.3 문자열 데이터(as.character)

```
# paste / sprintf
who = c("Sohn", "Jung")
where = c("San Diego", "Los Angeles")
sprintf("%s told me to go to %s", who, where)

paste(who, "told me to go to", where)

cents = c(50 ,10)
sprintf("%s gave me %d cents", who, cents)
paste(who, "gave me", cents, "cents")

heights = c(170.2522, 180.231)
sprintf("%s is %3.2f centimeters tall", who, heights)
paste(who, "is", format(heights, digits=5),"centimeters tall")

# nchar

names = c("Kim Beazley", "Tom Chaplin", "Tris Mees", "Phillip Kim")
names[nchar(names)<=9]

# substring
names[substring(names,1,3)=="Kim"]

# grep, grepl
grep(pattern = "Kim", names)
grepl(pattern = "Kim", names)

# sub, gsub
sub("i","j",names)
gsub("i","j",names)

# strsplit
nam0 <- unlist(strsplit(names, "i"))
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.