# Chapter 3, 4 Review

2017100057 / 이영노

October 4, 2022

## Chapter 3

Chapter3 Intro: Conditonal Expressions, for-loops, and functions.
`split`,`cut`,`do.call`,`Reduce` **data.table** package

### 3.1 Conditional Expressions

- example1

```
a <- 0
if(a!=0){
  print(1/a)
} else {
  print("No reciprocal for 0.")
}
```

```
## [1] "No reciprocal for 0."
```

- example2

```
library(dslabs)
```

```
data(murders)
murder_rate <- murders$total / murders$population*100000

ind <- which.min(murder_rate)
if(murder_rate[ind] < 0.5){
  print(murders$state[ind])
} else {
  print("No state has murder rate that low")
}
```

```
## [1] "Vermont"
```

**인덱싱 접근:**

**1) which.min index extraction,**

**2) getting access to elements by location(index)**

- example3

```r
a <- 0
ifelse(a>0, 1/a, NA)
```

```
## [1] NA
```

```r
a<- c(0,1,2,-4,5)
result <- ifelse(a > 0, 1/a, NA)
print(result)
```

```
## [1]  NA 1.0 0.5  NA 0.2
```

```r
tmp <- data.frame(a=a, is_a_positive = a>0, answer1 = 1/a, answer2 = NA, result=result)
knitr::kable(tmp) #이거 코드 잘 보고 외우셈
```

| a | is_a_positive | answer1 | answer2 | result |
|---|---|---|---|---|
| 0 | FALSE | Inf | NA | NA |
| 1 | TRUE | 1.00 | NA | 1.0 |
| 2 | TRUE | 0.50 | NA | 0.5 |
| -4 | FALSE | -0.25 | NA | NA |
| 5 | TRUE | 0.20 | NA | 0.2 |

- example4 (working directory)

```r
getwd()
```

```
## [1] "C:/Users/user/Documents"
```

```r
setwd("C:/Users/user/Desktop/SDS_data") #이번 chunk가 아닌 전체 chunk의 wd를
#바꾸려면 `knitr root.dir`에서 바꿔야한다.
list.files()
```

```
## character(0)
```

- example 5

```r
data(na_example)
no_nas <- ifelse(is.na(na_example),0,na_example)
# ifelse구문에 vector를 넣었더니, 자동으로 다음순서로 넘어가서 ifelse를 해줌.
# 이렇게 된 이유는 무엇일까? python class와 같은 기능과 관련이 있지 않을까?

#####

z=c(TRUE,TRUE,FALSE)
any(z) # any true value?
```

```
## [1] TRUE
```

```r
all(z) # all true value?
```

```
## [1] FALSE
```

### 3.2 User-defined functions

- example1 : **매개변수의 형식을 지역변수로 정할 수 있다.**

```
avg <- function(x){
  s <- sum(x) # sum은 vector를 argument로 받는다. 따라서, 위 표현식 function(x)
  # 에서 매개변수(parameter) x의 형식은 vector여야함! 이라고 정한 것에 해당한다.
  n <- length(x)
  s/n
}

x <- 1:100
identical(mean(x), avg(x)) # 똑같냐?
```

지역변수(local variable): 블록 내에서 선언된 변수. outside workspace가 아닌 memory에 저장됨.

**user-defined function 정의에 필요한것들: 1)매개변수, 2)내부변수, 3) 매개변수 형식, 4)default**

- example2

`{r eval=FALSE}`: chunk의 코드를 실행하지 않고, txt자체만 보여줌
`{r result=hide}`: chunk의 코드를 실행하나, result를 숨김.

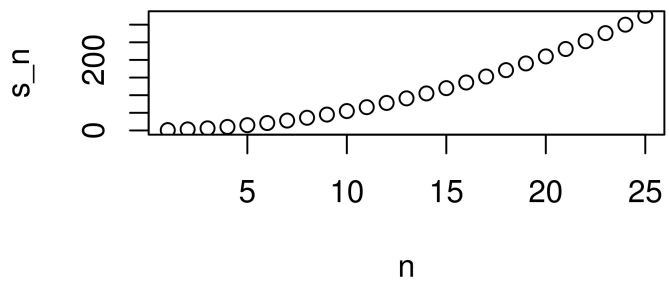
### 3.3 for-loops & vectorization and functionals

- example1

```
compute_s_n <- function(n){
  x <- 1:n # 매개변수 n은 scalar라고 우리가 정할거임!
  sum(x)
}

m <- 25
s_n <- vector(length=m)
for (n in 1:m){
  s_n[n] <- compute_s_n(n) # n: each loop # 1:m loop range
} # loop으로 각각의 vector를 만들어줄게!
```

```
n <- 1:m
plot(n,s_n)
```




**실질적으로 for loop은 방대한 데이터를 축적하기 때문에 시간도 오래걸려서 안씀.**

**대신 vectorization을 사용**

- example1: vectorization

```
x <- 1:10; y <- 1:10
round(sqrt(x),2)
```

```
##  [1] 1.00 1.41 1.73 2.00 2.24 2.45 2.65 2.83 3.00 3.16
```

```
x*y
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

- example2: **함수에 쓸때 매개변수 형태가 달라서 오류남 ~> functionals**

```
n <- 1:25
compute_s_n(n) #scalar인 매개변수에 vector인수(argument) ~ 매개변수 형태 다르기 때문에 오류
```

```
## Warning in 1:n: 수치형 표현식(numerical expression)은 25개의 구성요소들을 가지고
## 있기 때문에 오로지 첫번째 것만을 사용합니다.
```

- example3: **functionals**

functionals: **input 형태에 따라 element-wise function operation을 가능하게 해준다.**

```
x <- 1:10
round(sapply(x,sqrt),2)
```

```
##  [1] 1.00 1.41 1.73 2.00 2.24 2.45 2.65 2.83 3.00 3.16
```

```
# sapply: input on numeric,logical,character vectors
# 여기서 sapply function의 "반환값"형태는 vector가 아닌 scalar다.

n <- 1:25
s_n <- sapply(n,compute_s_n)
```

other functionals: `apply`,`lapply`,`tapply`,`mapply`,`vapply`,`replicate`

## Chapter 4

Chapter4 Intro: `dplyr`for manipulating dataframe, `purrr`for working with functions.

### tidyverse

- Definition of tidy data
- Comparison between p8.(tidy) and p9.(not): **Variables: what datapoint expresses about**

### 4.1 `dplyr`: manipulating dataframe

- 추가:

`mutate()`: 열 추가

- 추출: **인덱싱 접근**적용 가능

`filter()`: 행 추출 **r이 마지막에 있음. row**

`select()`: 열 추출

```
library(tidyverse)
```

#### 4.1.1 추가: `mutate()` 열 추가

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.7      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dslabs)
data("murders")
murders <- mutate(murders, rate = total / population*100000)
# murders data가 mutate내부 local data로 지정됨.
knitr::kable(head(murders))
```

| state | abb | region | population | total | rate |
|---|---|---|---|---|---|
| Alabama | AL | South | 4779736 | 135 | 2.824424 |
| Alaska | AK | West | 710231 | 19 | 2.675186 |
| Arizona | AZ | West | 6392017 | 232 | 3.629527 |
| Arkansas | AR | South | 2915918 | 93 | 3.189390 |
| California | CA | West | 37253956 | 1257 | 3.374138 |
| Colorado | CO | West | 5029196 | 65 | 1.292453 |

#### 4.1.2 추출: filter() 행 추출 **(중요!) 행추출은 filter()내부 조건값이 TRUE/FALSE 반환값을 가짐

**TRUE/FALSE 접근**

```
filter(murders, rate<=0.71)
```

```
##           state abb        region population total      rate
## 1        Hawaii  HI          West    1360301     7 0.5145920
## 2          Iowa  IA North Central    3046355    21 0.6893484
## 3 New Hampshire  NH     Northeast    1316470     5 0.3798036
## 4  North Dakota  ND North Central     672591     4 0.5947151
## 5       Vermont  VT     Northeast     625741     2 0.3196211
```

```
filter(murders,state=="New York")
```

```
##      state abb    region population total    rate
## 1 New York  NY Northeast   19378102   517 2.66796
```

```
new_table <- select(murders,state,region,rate)
filter(new_table, rate<=0.71)
```

#### 4.1.3 추출: select() 열 추출

```
##           state        region      rate
## 1        Hawaii          West 0.5145920
## 2          Iowa North Central 0.6893484
## 3 New Hampshire     Northeast 0.3798036
## 4  North Dakota North Central 0.5947151
## 5       Vermont     Northeast 0.3196211
```

**Sample Question(중요)**

```
library(dslabs)
data(murders)
mutate(murders, rate=total/population*100000, rank=rank(-rate)) %>%
  select(state,rate,rank) %>% filter(rank<=5)
```

```
##                  state      rate rank
## 1 District of Columbia 16.452753    1
## 2            Louisiana  7.742581    2
## 3             Maryland  5.074866    4
## 4             Missouri  5.359892    3
## 5       South Carolina  4.475323    5
```

```
# select(): 새로운 데이터셋을 지정해준 뒤 사용해야함. 데이터 추가 생성 않으려면 %>% 필요.
# rank(): 값이 작은것부터 순위를 나타냄. 큰것부터 순위를 매기려면 rank(-rate)
# rank를 changing the murders dataset하지 않고 order에 따라 순위매기려면 어떻게?(질문)
# a[order(rank(-a),]로 a가 큰값이 맨 위로 오게 전처리 할 수는 있음. (인덱싱 접근이지,
# filter는 TRUE/FALSE 접근임! 따라서 filter못씀)
```

```
s=mutate(murders, rate=total/population*100000, rank=rank(-rate)) %>%
  select(state,rate,rank) %>% filter(rank<=5)
s[order(s$rank),]%>%select(state,rate,rank) # 새로운 데이터프레임 만들어서
```

```
##                  state      rate rank
## 1 District of Columbia 16.452753    1
## 2            Louisiana  7.742581    2
## 4             Missouri  5.359892    3
## 3             Maryland  5.074866    4
## 5       South Carolina  4.475323    5
```

```
# (3) 대괄호 데이터프레임 접근. 사용함.(벡터와이즈)
```

### 4.2 %>% Pipe Operator

sending the results of one function to another

데이터 추가 생성 없이 결과값만 전달 가능. 코드 간단해짐. Sequential code.

%>% 는 다음 함수의 첫번째 argument로 전달받은 값을 사용함

%in% 은 전달받은 값에 다음 값이 있느냐?의 logical value를 반환함.

- example1

```
# Sample Question
mutate(murders,rate=total/population*100000,rank=rank(-rate)) %>%
  filter(region %in% c("Northeast","West") & rate <1) %>% select(state, rate, rank) %>% head()
```

```
##           state      rate rank
## 1        Hawaii 0.5145920   49
## 2         Idaho 0.7655102   46
## 3         Maine 0.8280881   44
## 4 New Hampshire 0.3798036   50
## 5        Oregon 0.9396843   42
## 6          Utah 0.7959810   45
```

#### 4.2.1 summarize()

- example2

summarize(): we can use variables names directly.

```
library(dplyr)
library(dslabs)
data(heights)

s <- heights %>% filter(sex=="Female") %>% summarize(average = mean(height),
                                                     standard_deviation = sd(height))
s
```

```
##    average standard_deviation
## 1 64.93942           3.760656
```

#### 4.2.2 `pull()`

- example3:

**조심할것**: average US murder rate구할땐 단순히 mean(rate)하면 안됨!

`pull()`: tibble객체에서 벡터를 하나 추출할때 사용하는 함수

```
# INCORRECT: population이 다 달라서 다르게 weight해줘야됨.
murders %>% mutate(rate = total/population*100000) %>% summarize(avg_rate = mean(rate))
```

```
##   avg_rate
## 1 2.779125
```

```
# CORRECT: 인구비례 rate
us_murder_rate <- murders %>% summarize(avg_rate = sum(total)/sum(population)*100000)
us_murder_rate
```

```
##   avg_rate
## 1 3.034555
```

```
class(us_murder_rate)
```

```
## [1] "data.frame"
```

```
us_murder_rate %>% pull(avg_rate)
```

```
## [1] 3.034555
```

#### 4.2.3 `group_by()`: will behave differently when acting with `summarize()`, mutate data.frame by **factors**.

```
heights %>% group_by(sex) %>% head()
```

```
## # A tibble: 6 x 2
## # Groups:   sex [2]
##   sex    height
##   <fct>   <dbl>
## 1 Male       75
## 2 Male       70
## 3 Male       68
## 4 Male       74
## 5 Male       61
## 6 Female     65
```

```
heights %>% group_by(sex) %>% summarize(average = mean(height),
                                        standard_deviation = sd(height))
```

```
## # A tibble: 2 x 3
##   sex    average standard_deviation
##   <fct>    <dbl>              <dbl>
## 1 Female    64.9               3.76
## 2 Male      69.3               3.61
```

```
murders %>% arrange(-population) %>% head()
```

#### 4.2.4 arrange()

```
##          state abb        region population total
## 1   California  CA          West   37253956  1257
## 2        Texas  TX         South   25145561   805
## 3      Florida  FL         South   19687653   669
## 4     New York  NY     Northeast   19378102   517
## 5     Illinois  IL North Central   12830632   364
## 6 Pennsylvania  PA     Northeast   12702379   457
```

```
#예전에 했던 Sample Question 질문에서, 새로운 데이터프레임을 할당하고 (3)대괄호
#접근해야 한다고 했지만, arrange()사용하면 그렇게 안해도 됨.

mutate(murders, rate=total/population*100000) %>% arrange(-rate) %>%
  select(state,abb,rate) %>% head()
```

```
##                  state abb      rate
## 1 District of Columbia  DC 16.452753
## 2            Louisiana  LA  7.742581
## 3             Missouri  MO  5.359892
## 4             Maryland  MD  5.074866
## 5       South Carolina  SC  4.475323
## 6             Delaware  DE  4.231937
```

#### 4.2.5 top_n(NUMBER, VARIABLES) = head() sorting이 아니라 결과값을 보여주는것에 불과함 (시험)

```
murders %>% top_n(3,total)
```

```
##        state abb region population total
## 1 California  CA   West   37253956  1257
## 2    Florida  FL  South   19687653   669
## 3      Texas  TX  South   25145561   805
```

### 4.3 tibbles

```
murders %>% class()
```

```
## [1] "data.frame"
```

```
murders %>% group_by(region) %>% class() #special kind of data.frame
```

```
## [1] "grouped_df" "tbl_df"     "tbl"        "data.frame"
```

1. subsets of tibbles are tibbles.

```
head(murders[,4],5)
```

```
## [1]  4779736   710231  6392017  2915918 37253956
```

```r
class(as_tibble(murders)[,4])
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

2. column can be complex objects: lists, functions

```r
tibble(id=c(1,2,3),func=c(mean,median,sd))
```

```
## # A tibble: 3 x 2
##      id func
##   <dbl> <list>
## 1     1 <fn>
## 2     2 <fn>
## 3     3 <fn>
```

3. create tibble using `tibble` instead of `data.frame`, `as_tibble()`

### 4.4 dot operator

```r
rates <- mutate(murders,rate<-total/population*100000) %>%
  filter(region=="South") %>% .$rate #access to column vector without `pull()`
median(rates)
```

```
## [1] 3.398069
```

### 4.5 purrr = saaply(DATA,FUNCTION)

purrrpackage deals working with functions

```r
library(purrr)

compute_s_n <- function(n){
  x <- 1:n
  sum(x)
}
n <- 1:25
s_n1 <- sapply(n, compute_s_n)

s_n2 <- map(n,compute_s_n)

s_n3 <- map_dbl(n, compute_s_n)

#s_n4 <- map_df(n, compute_s_n) #compute_s_n returns a scalar.(vector, name)
c(class(s_n1),class(s_n2),class(s_n3))
```

```
## [1] "integer" "list"    "numeric"
```

so we need to change the function into this form

```r
compute_s_n_tibble <- function(n){
  x <- 1:n
  tibble(sum = sum(x))
```

```
}
s_n5 <- map_df(n,compute_s_n_tibble)
class(s_n5)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

### 4.6 tidyverse conditionals

#### 4.6.1 case_when(): vectorizing conditional statements.

- **반환값을 지정할수 있다.**
- 입력값이 TRUE/FALSE

```
x <- c(-2,-1,0,1,2)
case_when(x<0 ~ "Negative", x>0 ~ "Positive", x==0~"Zero")
```

```
## [1] "Negative" "Negative" "Zero"     "Positive" "Positive"
```

일반적으로 **범주형 변수를 지정할때(중요!)** 쓰인다.

```
murders %>%
  mutate(group=case_when(
    abb %in% c("ME","NH","VT","MA","RI","CT") ~ "New England",
    abb %in% c("WA","OR","CA") ~ "West Coast",
    region == "South" ~ "South",
    TRUE ~ "Other")) %>%
  group_by(group) %>% summarize(rate = sum(total)/sum(population)*10^5) %>%
  arrange(desc(rate))
```

```
## # A tibble: 4 x 2
##   group        rate
##   <chr>       <dbl>
## 1 South        3.63
## 2 West Coast   2.90
## 3 Other        2.71
## 4 New England  1.72
```

```
x <- c(-2,-1,0,1,2)
between(x,-1.5,0.5)
```

#### 4.6.2 between()

```
## [1] FALSE  TRUE  TRUE FALSE FALSE
```