R을 이용한 통계 기초와 데이터 분석

Lecture 2

남현진

한성대학교

2020

Section 1

R의 기본 문법

R의 연산자

- +, -, *, / : 사칙연산
- <, >, <=, >=: 작다, 크다, 작거나 같다, 크거나 같다

```
x <- 1:5
x
## [1] 1 2 3 4 5
x + 1
## [1] 2 3 4 5 6
x + x
## [1]  2  4  6  8 10
x >= 3
## [1] FALSE FALSE  TRUE  TRUE  TRUE
```

R 문법

- == : 같다
- != : 다르다
- %in% : 포함되다

```
x
## [1] 1 2 3 4 5
x == c(8,7,3,5,5)
## [1] FALSE FALSE  TRUE FALSE  TRUE
x != c(8,7,3,5,5)
## [1]  TRUE  TRUE FALSE  TRUE FALSE
x %in% c(1,4)
## [1]  TRUE FALSE FALSE  TRUE FALSE
```

R의 연산자

- sum : 합계
- mean : 평균
- median : 중앙값

```
x
## [1] 1 2 3 4 5
sum(x)
## [1] 15
mean(x)
## [1] 3
median(x)
## [1] 3
```

R의 연산자

- na.rm : 결측치가 있을 때 해당 값을 연산에서 제외할지 지정한다.

```
x <- c(1:10, NA)
x
##  [1]  1  2  3  4  5  6  7  8  9 10 NA
sum(x)
## [1] NA
sum(x, na.rm = TRUE)
## [1] 55
```

R의 연산자

- %%: 나머지
- %/%: 몫
- n^m: n의 m승

```
9 %% 2
## [1] 1
9 %/% 2
## [1] 4
2^5
## [1] 32
```

R 문법

- 조건문: if, ifelse
- 반복문: for, while
- 지정문: function

If 문

- 문법 : if(cond){cond가 참일 때 실행할 문장} else {cond가 거짓일 때 실행할 문장}
- 의미 : 조건 cond가 참, 거짓인 경우에 따라 {} 블록을 실행한다. 필요한 경우 else블록을 지정할 수 있다.

```
x <- 3
  if( x >= 10 ){
  print('x is over 10') } else{
    print('x is not over 10')}
## [1] "x is not over 10"
```

Ifelse 문

- 문법: ifelse(cond, cond가 참일 때 실행할 문장, cond가 거짓일 때 실행할 문장)

```
x <- 1:5

x

## [1] 1 2 3 4 5

ifelse(x %% 2 == 0, 'even', 'odd')

## [1] "odd"  "even" "odd"  "even" "odd"
```

For 문

- 문법 : for(i in data){ i를 사용한 문장 }
- 의미 : data에 들어 있는 각각의 값을 변수 i에 할당하면서 각각에 대한 블록 안의 문장을 수행한다.

```
for( i in 1:5){
  print(i*2)
}
## [1] 2
## [1] 4
## [1] 6
## [1] 8
## [1] 10
```

For 문

```r
fruit <- c('Apple', 'Orange', 'Passion fruit', 'Banana')
for ( i in fruit){
 print(i)
}
## [1] "Apple"
## [1] "Orange"
## [1] "Passion fruit"
## [1] "Banana"
```

While 문

- 문법 : while(cond){ 조건이 참일 때 수행할 문장 }
- 의미 : 조건 cond가 참일 때 블록 안의 문장을 수행한다.

```
x <- 1
while( x < 5){
  print(x)
  x = x+1
}
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

Function 문

- 문법 : function(인자1, 인자2,…){ 함수 본문 return(반환 값) }
- 의미 : 조건 cond가 참일 때 블록 안의 문장을 수행한다.

```
x <- 4
square <- function(x){
  result <- x * x
  return(result)
}

square(x)
## [1] 16
```

R 라이브러리

R의 가장 큰 장점 중 하나는 전세계 사용자들이 구축해놓은 오픈소스 패키지이다.
패키지들을 이용하면 R의 내장 함수 이외에 다양한 함수들을 사용할 수 있다.

- 패키지 설치: install.pakages(패키지명)
- 패키지 사용:library(패키지명)

```
install.packages("dplyr")
library("dplyr")
```

- dplyr 패키지는 데이터 처리에 특화된 R 패키지이다.
- Dplyr의 다양한 함수들을은 Help > Cheetsheets > “Data transformation with dply”에서 확인할 수 있다.

Wage data

- Wage 데이터는 Mid-Atlantic 지역에 거주하는 3000명의 남성 근로자에 대한 서베이 데이터로 Steve Miller, Open BI (www.openbi.com)에 의해 수집되었다.
- help 함수를 사용해서 데이터에 대한 자세한 정보를 찾아볼 수 있다.

```
library(ISLR)
help(Wage)
```

Head 함수

- 문법: head(데이터, 열의 개수)
- 의미: 상위에 있는 열의 데이터를 추출하여 보여준다. 개수를 지정하지 않으면 10 개가 기본 값으로 출력된다.

```
head(Wage,3)
```

```
##        year age           maritl     race       education                re
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad 2. Middle Atla
## 86582  2004  24 1. Never Married 1. White 4. College Grad 2. Middle Atla
## 161300 2003  45       2. Married 1. White 3. Some College 2. Middle Atla
##               jobclass          health health_ins  logwage      wage
## 231655  1. Industrial      1. <=Good      2. No 4.318063  75.04315
## 86582  2. Information 2. >=Very Good      2. No 4.255273  70.47602
## 161300  1. Industrial      1. <=Good     1. Yes 4.875061 130.98218
```

Tail 함수

- 문법: tail(데이터, 열의 개수)
- 의미: 하위에 있는 열의 데이터를 추출하여 보여준다. 개수를 지정하지 않으면 10 개가 기본 값으로 출력된다.

```
head(Wage,3)
```

```
##        year age            maritl     race       education                re
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad 2. Middle Atla
## 86582  2004  24 1. Never Married 1. White 4. College Grad 2. Middle Atla
## 161300 2003  45       2. Married 1. White 3. Some College 2. Middle Atla
##              jobclass          health health_ins  logwage      wage
## 231655  1. Industrial      1. <=Good      2. No 4.318063  75.04315
## 86582  2. Information 2. >=Very Good      2. No 4.255273  70.47602
## 161300  1. Industrial      1. <=Good     1. Yes 4.875061 130.98218
```

Str 함수

- 문법: str(데이터)
- 의미: 데이터의 구조를 보여준다.

```
str(Wage)
```

```
## 'data.frame':    3000 obs. of  11 variables:
##  $ year       : int  2006 2004 2003 2003 2005 2008 2009 2008 2006 2004 ..
##  $ age        : int  18 24 45 43 50 54 44 30 41 52 ...
##  $ maritl     : Factor w/ 5 levels "1. Never Married",..: 1 1 2 2 4 2 2 1
##  $ race       : Factor w/ 4 levels "1. White","2. Black",..: 1 1 1 3 1 1 
##  $ education  : Factor w/ 5 levels "1. < HS Grad",..: 1 4 3 4 2 4 3 3 3 2
##  $ region     : Factor w/ 9 levels "1. New England",..: 2 2 2 2 2 2 2 2 2
##  $ jobclass   : Factor w/ 2 levels "1. Industrial",..: 1 2 1 2 2 2 1 2 2 
##  $ health     : Factor w/ 2 levels "1. <=Good","2. >=Very Good": 1 2 1 2
##  $ health_ins: Factor w/ 2 levels "1. Yes","2. No": 2 2 1 1 1 1 1 1 1 1
##  $ logwage    : num  4.32 4.26 4.88 5.04 4.32 ...
```

Unique 함수

- 문법: unique(데이터)
- 의미: 데이터가 가지고 있는 유니크한 값들을 보여준다.

```
unique(Wage$education)
## [1] 1. < HS Grad      4. College Grad    3. Some College    2. HS Grad
## [5] 5. Advanced Degree
## 5 Levels: 1. < HS Grad 2. HS Grad 3. Some College ... 5. Advanced Degree
```

Summary 함수

- 문법: summary(데이터)
- 의미: 데이터의 간단한 요약 통계를 보여준다.

```
summary(Wage)
```

```
##       year           age                      maritl           race
##  Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480
##  1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293
##  Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190
##  Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37
##  3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##
##                education                  region               jobclas
##  1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :15
##  2. HS Grad        :971   1. New England       :   0   2. Information:14
```

Table 함수

- 문법: table(데이터)
- 의미: 데이터의 해당 값 별 카운트 수를 보여준다.

```
table(Wage$race)
##
## 1. White 2. Black 3. Asian 4. Other
##     2480      293      190       37
table(Wage$race, Wage$education)
##
##                1. < HS Grad 2. HS Grad 3. Some College 4. College Grad
##    1. White             211        822             532              576
##    2. Black              31        105              92               40
##    3. Asian              15         31              18               66
##    4. Other              11         13               8                3
##
```

Section 2

Dplyr 패키지

Select 함수

- 문법: select(추출하고자 하는 열)
- 의미: 사용하고자 하는 열만 뽑아서 데이터를 만들 수 있다.

```
wage1 <- Wage %>%
  select(year, age, wage)
str(wage1)
## 'data.frame':    3000 obs. of  3 variables:
##  $ year: int  2006 2004 2003 2003 2005 2008 2009 2008 2006 2004 ...
##  $ age : int  18 24 45 43 50 54 44 30 41 52 ...
##  $ wage: num  75 70.5 131 154.7 75 ...

#Alternative codes
#1: select(Wage, c(year, age, wage))
#2: Wage[,c('year', 'age', 'wage')]
```

Filter 함수

- 문법: filter(지정하고자 하는 조건식)
- 의미: 데이터에서 필요한 조건에 맞는 데이터만 추출할 수 있다.

```
wage2 <- Wage %>%
  filter(jobclass == "1. Industrial")


table(Wage$jobclass)
##
##  1. Industrial 2. Information
##           1544            1456
table(wage2$jobclass)
##
##  1. Industrial 2. Information
##           1544               0
```

Filter 함수

- &: and
- |: or

& 과 | 를 사용하면 여러가지 조건들을 한 함수안에 적을 수 있다.

```
wage3 <- Wage %>%
  filter(education == "1. < HS Grad" & jobclass == "1. Industrial" )
unique(wage3$education)
## [1] 1. < HS Grad
## 5 Levels: 1. < HS Grad 2. HS Grad 3. Some College ... 5. Advanced Degree
unique(wage3$jobclass)
## [1] 1. Industrial
## Levels: 1. Industrial 2. Information
```

Mutate 함수

- 문법: mutate(열 이름 = 추가하고자 하는 데이터)
- 의미: 데이터에 새로운 열을 추가하고 싶다면 열 이름과 데이터를 지정해줄 수 있다. 만약 이미 존재하는 열 이름에 mutate함수를 사용하면 현재 가지고 있는 데이터에서 덮어쓰기가 된다.

```
wage4 <- Wage %>%
  select(education, jobclass) %>%
  mutate(new_variable = 1:nrow(Wage))


head(wage4$new_variable, 5)
## [1] 1 2 3 4 5
#Alternative codes
#Wage$new_variable <- 1:nrow(Wage)
```

Summarise 함수

- 문법: summarise(열 이름 = 지정함수)
- 의미: mean(), sd(), var(), median() 등의 함수를 지정하여 기초 통계량을 구할 수 있다.

```
wage5 <- Wage %>%
  summarize(median_wage = median(wage))
wage5
##   median_wage
## 1    104.9215
```

Summarise 함수

group_by를 이용하면 해당 열을 그룹으로 묶어서 summarise 함수를 사용할 수 있다.

```
wage6 <- Wage %>%
  group_by(education) %>%
  summarize(median_wage = median(wage))
## `summarise()` ungrouping output (override with `.groups` argument)
wage6
## # A tibble: 5 x 2
##   education           median_wage
##   <fct>                     <dbl>
## 1 1. < HS Grad               81.3
## 2 2. HS Grad                 94.1
## 3 3. Some College           105.
## 4 4. College Grad           119.
## 5 5. Advanced Degree        142.
```

Summarise 함수

summarize 는 여러가지 함수를 한번에 넣어서 사용할 수 있다.

```
wage7 <- Wage %>%
  group_by(education) %>%
  summarize(median_wage = median(wage), average_wage = mean(wage))
## `summarise()` ungrouping output (override with `.groups` argument)
wage7
## # A tibble: 5 x 3
##   education           median_wage average_wage
##   <fct>                     <dbl>        <dbl>
## 1 1. < HS Grad               81.3         84.1
## 2 2. HS Grad                 94.1         95.8
## 3 3. Some College           105.         108.
## 4 4. College Grad           119.         124.
## 5 5. Advanced Degree        142.         151.
```

Join 함수 데이터 만들기

- Join 함수를 사용하기 위한 예제 데이터를 만든다.

```
library(dplyr)
table1 <- data.frame(ID = c('A', 'B', 'C', 'D', 'F'),
                     y = c(5,5,8,0,9))
table2 <- data.frame(ID = c('A', 'B', 'C', 'D', 'E'),
                     z = c(30,21,22,25,29))
```

Left Join 함수

- 문법: left_join(병합할 데이터, by = 키)
- 의미: 왼쪽에 있는 테이블의 키를 기준으로 두 테이블을 병합한다.

```
left_table <- table1 %>%
  left_join(table2, by = 'ID')
left_table
##   ID y  z
## 1  A 5 30
## 2  B 5 21
## 3  C 8 22
## 4  D 0 25
## 5  F 9 NA
```

Right Join 함수

- 문법: right_join(병합할 데이터, by = 키)
- 의미: 오른쪽에 있는 테이블의 키를 기준으로 두 테이블을 병합한다.

```
right_table <- table1 %>%
  right_join(table2, by = 'ID')
right_table
##   ID  y  z
## 1  A  5 30
## 2  B  5 21
## 3  C  8 22
## 4  D  0 25
## 5  E NA 29
```

Full Join 함수

- 문법: full_join(병합할 데이터, by = 키)
- 의미: 키를 기준으로 두 테이블에 존재하는 모든 데이터를 뽑아내어 병합한다.

```
full_table <- table1 %>%
  full_join(table2, by = 'ID')
full_table
##   ID  y  z
## 1  A  5 30
## 2  B  5 21
## 3  C  8 22
## 4  D  0 25
## 5  F  9 NA
## 6  E NA 29
```

Inner Join 함수

- 문법: inner_join(병합할 데이터, by = 키)
- 의미: 키를 기준으로 두 테이블에 같이 존재하는 데이터를 추출한다.

```
inner_table <- table1 %>%
  inner_join(table2, by = 'ID')
inner_table
##   ID y  z
## 1  A 5 30
## 2  B 5 21
## 3  C 8 22
## 4  D 0 25
```

Section 3

R 실습

평균을 구하는 함수 작성하기

1 부터 100까지의 수 중 랜덤한 5개의 숫자를 생성하고 벡터 x에 저장한다.

```
set.seed(1)
x <- sample(1:100,5)
x
## [1] 68 39  1 34 87
```

벡터의 합을 구한 후 벡터의 길이를 나눈다.

```
sum(x)
## [1] 229
length(x)
## [1] 5
sum(x)/length(x)
## [1] 45.8
```

평균을 구하는 함수 작성하기

평균을 구하는 함수를 직접 만들어 보면 다음과 같다.

```
my.mean <- function(x) {
  return(sum(x) / length(x))
}


mean(x)
## [1] 45.8
my.mean(x)
## [1] 45.8
```

분산을 구하는 함수 작성하기

- step 1: 편차를 구한다.
- step 2: 편차를 제곱한다.
- step 3: 편차의 평균을 구한다.
- step 4: 자유도를 맞춰준다.

```
step1 <-  x - mean(x)
step1
## [1]  22.2  -6.8 -44.8 -11.8  41.2
step2 <- step1^2
step2
## [1]  492.84   46.24 2007.04  139.24 1697.44
step3 <- mean(step2)
step4 <- step3 * length(x) / (length(x)-1)
step4
## [1] 1095.7
```

분산을 구하는 함수 작성하기

분산을 구하는 함수를 직접 만들어 보면 다음과 같다.

```
my.var <- function(x) {          # Create function for population variance
  return( mean((x - mean(x))^2)* length(x) / (length(x)-1) )
}


var(x)
## [1] 1095.7
my.var(x)
## [1] 1095.7
```