

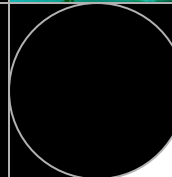
# R 데이터 분석 입문

## Chapter 4

# R 프로그래밍

오 세 종

 DANKOOK UNIVERSITY



# Contents



- if
- for, while
- 사용자정의 함수

- R은 데이터 분석 도구인 동시에 프로그래밍 언어의 성격도 가지고 있다
- R 프로그래밍 기본 문법과 활용에 대해 학습한다.
- 프로그래밍 : 주어진 문제(problem)를 컴퓨터가 해결(solution)하도록 하기 위한 절차(procedure)를 문법에 맞추어 서술하는 과정
- R 에서 제공하는 함수만으로는 문제를 해결할 수 없는 경우도 있는데, 이때 프로그래밍으로 문제를 해결한다

# If 문

```
if (logical expression) {  
  statements  
} else {  
  alternative statements  
}
```

```
a <- 10  
if (a>5) {  
  print (a)  
} else {  
  print (a*10)  
  print (a/10)  
}
```

```
> a = 10  
> if (a>5){  
+   print (a)  
+ } else {  
+   print (a*10)  
+   print (a/10)  
+ }  
[1] 10
```


\* **else** 는 앞의 **}** 와 같은 줄에 있어야함

```
a <- 10
b <- 20
if (a>5 & b>5) {                # and
  print (a+b)
}
if (a>5 | b>30) {               # or
  print (a*b)
}
```

```
> a <- 10
> b <- 20
> if (a>5 & b>5) {              # and
+   print (a+b)
+ }
[1] 30
> if (a>5 | b>30) {             # or
+   print (a*b)
+ }
[1] 200
```

# Ifelse 문

```
a <- 10
b <- 20
ifelse (a>b, c<-a, c<-b)
c
```



조건이 참일때  
실행

조건이 거짓일때  
실행

```
a <- 10
b <- 20
if (a > b) {
  print(a)
} else {
  print(b)
}
c
```

## 반복문: for

```
for(i in 1:10) {  
    print(i)  
}
```

```
> for(i in 1:10) {  
+   print(i)  
+ }  
[1] 1  
[1] 2  
[1] 3  
[1] 4  
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

## 반복문: for

```
for(i in 1:10) {  
  cat("2*",i,"=",2*i,"\n")  
}
```

```
> for(i in 1:10) {  
+   cat("2*",i,"=",2*i,"\n")  
+ }  
2* 1 = 2  
2* 2 = 4  
2* 3 = 6  
2* 4 = 8  
2* 5 = 10  
2* 6 = 12  
2* 7 = 14  
2* 8 = 16  
2* 9 = 18  
2* 10 = 20
```



## 반복문: for

```
for(i in 1:20) {  
  if(i%%2==0) {  
    print(i)  
  }  
}
```

# 짝수인지 확인

```
> for(i in 1:20) {  
+   if(i%%2==0) {  
+     print(i)  
+   }  
+ }  
[1] 2  
[1] 4  
[1] 6  
[1] 8  
[1] 10  
[1] 12  
[1] 14  
[1] 16  
[1] 18  
[1] 20
```

## 반복문: for

```
v1 <- 101:200
for(i in 1:length(v1)) {
  if(v1[i]%%2==0) {
    print(v1[i]*2)
  } else {
    print(v1[i]+2)
  }
}
```

```
[1] 103
[1] 204
[1] 105
[1] 208
[1] 107
[1] 212
[1] 109
[1] 216
[1] 111
[1] 220
[1] 113
[1] 224
[1] 115
```

## 반복문: for

```
sum <- 0
for(i in 1:100) {
  sum <- sum + i
}
print(sum)
```

```
> sum <- 0
> for(i in 1:100) {
+   sum <- sum + i
+ }
> print(sum)
[1] 5050
```

## 반복문: while

```
i<-1
while(i <= 10) {
  print(i)
  i <- i+1
}
```

```
> i<-1
>
> while(i <= 10) {
+   print(i)
+   i <- i+1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

## Note

- 반복 횟수가 정해진 반복문은 for 를 이용하고, 조건에 따라 반복 횟수가 달라질 수 있는 반복문에는 while 을 사용한다

```
for(i in 1:100) {  
  ss <- ss + i      # error 발생  
}  
print(ss)
```

이 값이 무엇인지 알 수 없음

# 사용자정의 함수 만들기

```
mymax <- function(x,y) {  
  num.max <- x  
  if (y > x) {  
    num.max <- y  
  }  
  return(num.max)  
}
```

```
mymax(10,15)  
mymax(20,15)
```

```
> mymax(10,15)  
[1] 15  
> mymax(20,15)  
[1] 20
```

# 사용자정의 함수 만들기

- 자주 반복적으로 하는 작업은 함수로 정의해 놓고 불러서 사용하는 것이 편리하다.
- R 은 다양한 함수를 패키지 형태로 제공하는데, 사용자도 함수를 정의하여 사용할 수 있다.

# 사용자정의 함수 만들기

함수의 이름

함수의 매개변수(parameter)

```
mymax <- function(x,y) {  
  num.max <- x  
  if (y > x) {  
    num.max <- y  
  }  
  return(num.max)  
}
```

함수의 실행결과 값(return value)



# 사용자정의 함수 만들기

- Default value

```
mydiv <- function(x, y=2) {  
  result <- x/y  
  return(result)  
}
```

default value 선언

```
mydiv(x=10, y=3)  
mydiv(10, 3)  
mydiv(10)
```

```
> mydiv(x=10, y=3)  
[1] 3.333333  
> mydiv(10, 3)  
[1] 3.333333  
> mydiv(10)  
[1] 5
```

`sum {base}`

R Documentation

## Sum of Vector Elements

### Description

`sum` returns the sum of all the values present in its arguments.

### Usage

`sum(..., na.rm = FALSE)`

default value 선언

### Arguments

`...` numeric or complex or logical vectors.

`na.rm` logical. Should missing values (including `NaN`) be removed?

# 사용자정의 함수 만들기

- 내가 정의한 함수가 저장된 파일 사용하기

```
mydiv <- function(x,y=2) {  
  result <- x/y  
  return(result)  
}
```

➡ myfunc.R

```
setwd("c:/works")      # myfunc.R 이 저장된 폴더  
source("myfunc.R")     # myfunc.R 의 명령어 실행  
# 함수 사용  
a <- mydiv(20,4)  
b <- mydiv(30,4)  
a+b  
mydiv(mydiv(20,2),5)
```

# 화면에서 사용자 입력값 받기

```
n <- readline(prompt="숫자를 입력하세요: ")  
cat("입력한 숫자는", n, "입니다. \n")
```

```
> n <- readline(prompt="숫자를 입력하세요: ")  
숫자를 입력하세요: 7  
> cat("입력한 숫자는", n, "입니다. \n")  
입력한 숫자는 7 입니다.
```

# 숫자 맞추기 게임

```
num <- round(runif(1) * 100, digits = 0)
guess <- -1
cat("Guess a number between 0 and 100.\n")

while(guess != num){
  guess <- readline(prompt="Guess number :")
  guess <- as.integer(guess)
  if (guess == num) {
    cat("Congratulations,", num, "is right.\n")
  } else if (guess < num){
    cat("It's smaller!\n")
  } else if(guess > num) {
    cat("It's bigger!\n")
  }
}
```

## [연습 1]

1. 1~100 사이의 숫자중 3의 배수를 출력하는 프로그램을 작성하시오
2. 101~200 사이의 숫자중 3과 4의 공배수를 출력하는 프로그램을 작성하시오
3. 24의 약수를 출력하는 프로그램을 작성하시오
4. 10! (팩토리얼) 을 출력하는 프로그램을 작성하시오
5. 세개의 숫자를 매개변수로 입력하면 그중에 가장 큰 수를 돌려주는 함수를 작성하고 테스트 하시오
6. 화면에서 숫자 2개를 입력 받아 두숫자의 합과 곱을 출력하는 프로그램을 작성하시오 (이작업을 계속 반복하되 두 숫자가 모두 0 이면 프로그램을 중지한다)