

함수와 함수의 작성 방법

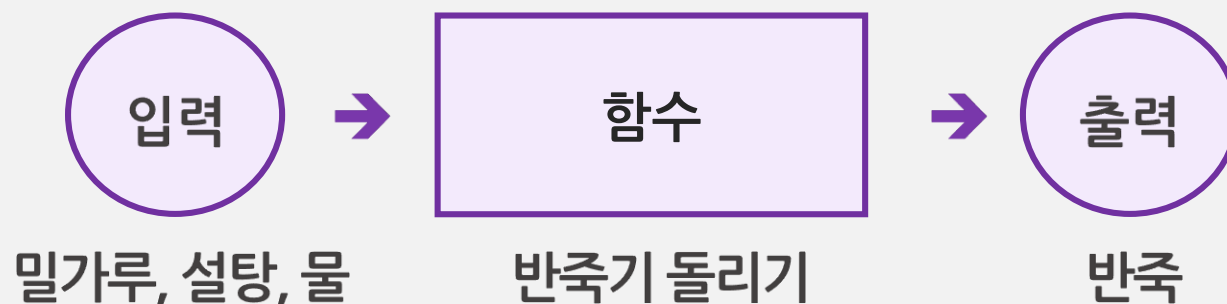
- 📖 함수와 R-언어의 함수 작성 문법
- 📖 매개변수에 따른 함수 사용
- 📖 계산 보기

함수와 R-언어의 함수 작성 문법



함수의 생성

 입력(생략 가능)을 받아서 적절한 작업을 한 후 출력을 생성함(생략 가능)







- 매개변수(parameter) : 밀가루, 설탕, 물
- 출력 : 내부의 값만 바꾸는 경우도 있음



함수와 R-언어의 함수 작성 문법



R에 내장된 함수

-  평균이나 표준편차의 계산에 사용한 mean, sd 등
-  변수값에 이름을 지정하거나 이름을 확인할 수 있는 names 등
-  R이 사용자의 모든 필요성에 대해서 함수를 제공할 수 없음
-  사용자가 필요한 함수는 사용자가 작성할 수 있도록 함수 작성 및 그 함수를 사용하는 기능을 제공



함수와 R-언어의 함수 작성 문법



R-언어의 함수 작성 문법

 하나의 작업을 '모듈'로 할 수 있는 함수를 작성할 수 있는 기능을 R언어에서 제공함

 사용법

```
fcn_name <- function (x) {  
  함수의 내용  
  ⋮  
}
```

- fcn_name : 사용자가 설정하는 함수의 이름으로 벡터 등의 이름과 마찬가지로 숫자로 시작하거나 특수문자 등은 사용 불가
- x : 함수에서 사용할 변수로 필요한 경우에 전달 받으며, 전달 받을 값이 없으면 생략하며 두 개 이상인 경우 콤마(,)로 구분함
→ 매개변수



함수와 R-언어의 함수 작성 문법



R-언어의 함수 작성 문법

함수의 호출

```
> fcn_name()
```

```
> fcn_name(x)
```

- 함수의 이름과 괄호 안의 매개변수 값을 설정한 함수의 이름을 사용함
- 매개변수가 없더라도 괄호는 반드시 사용함





매개변수에 따른 함수 사용



매개변수가 없는 경우

- + 1부터 10까지의 합을 계산하고 이를 인쇄하는 함수

```
myfcn <- function() {  
  x <- seq(1,10)  
  print(sum(x))  
}
```

- + 위의 함수 호출 결과

```
> myfcn()  
[1] 55
```



매개변수에 따른 함수 사용



매개변수가 없는 경우

- + 주의** : 함수로 전달할 값이 없더라도 반드시 괄호()를 주어야 하며 이를 생략하면 함수의 정의가 출력됨

```
> myfcn  
function() {  
  x <- seq(1,10)  
  print(sum(x))  
}
```





매개변수에 따른 함수 사용



매개변수가 있는 경우

- + 매개변수 nn을 사용하여 1부터 nn까지의 자연수의 합을 계산하고 이를 인쇄하는 함수

```
myfcn2 <- function(nn) {  
  x <- seq(1,nn)  
  print(sum(x))  
}
```

- 사용자가 끝점 nn을 함수를 호출할 때마다 다른 값으로 사용할 수 있게 매개변수를 전달함
- nn이라는 매개변수를 함수 호출 시 사용하여 x에 1부터 nn까지의 자연수를 저장한 벡터를 설정하고
이의 합을 인쇄함



매개변수에 따른 함수 사용



매개변수가 있는 경우

+ nn을 10으로 설정하여 호출한 결과

```
> myfcn2(10)  
[1] 55
```

+ nn을 100으로 매개변수를 설정한 후 이 함수를 호출한 결과

```
> myfcn2(100)  
[1] 5050
```



매개변수에 따른 함수 사용



전달할 변수로 두 개 이상 사용한 경우



자연수 start부터 end 사이의 합을 계산하는 함수

```
myfcn3 <- function(start, end) {  
  x <- seq(start, end)  
  print(sum(x))  
}
```

- start와 end를 매개변수로 전달받아 start부터 end 사이의 자연수(양 끝점 포함)를 x에 저장한 후 합을 인쇄함
- 매개변수 11과 100을 사용하면 11부터 100사이의 모든 자연수를 합한 결과를 얻음(양 끝점 포함)



매개변수 11과 100을 사용하여 호출한 결과

```
> myfcn3 (11,100)  
[1] 4995
```



매개변수에 따른 함수 사용



전달할 변수로 두 개 이상 사용한 경우

- + 매개변수의 값 뿐 아니라 어느 값이 어느 값인지 명확하게 하기 위해 **등호를 사용**하여 매개변수의 이름과 값을 함께 설정 가능

```
> myfcn3(start=11, end=100)
[1] 4995
```

- + 만일 전달할 변수가 함수의 정의의 매개변수 순서와 다를 경우 이름을 정확하게 설정해야 함

```
> myfcn3(end=100, start=11)
[1] 4995
```

* 호출 결과 : myfcn3(11, 100) 및 myfcn3(start=11, end=100)과 같음



계산 보기



한 번에 평균과 표준편차를 계산하는 함수 작성하기



자료

많은 경우 자료에 대한 기술통계를 작성할 때 평균과 표준편차를 함께 계산한다. 따라서 벡터 x에 대해서 매번 mean 함수와 sd 함수를 두 번씩 호출하지 않고 한 번에 평균과 표준편차를 계산하는 함수를 작성해보자.

사용할 자료

```
> BMI <- read.table(url("http://jupiter.hallym.ac.kr/ftpdata/data/bmi.txt"),  
  col.names=c("height", "weight", "year", "religion", "gender", "marriage"))
```



내용 : 2000년, 177명에 대한 조사 결과

- 키, 몸무게, 출생년도
- 종교(Bu=불교, C1=개신교, C2=가톨릭, No=없음)
- 성별(F=여자, M=남자)
- 결혼여부(N=미혼, Y=기혼)



계산 보기



한 번에 평균과 표준편차를 계산하는 함수 작성하기



평균과 표준편차를 한 번에 계산하기 위해 meansd라는 함수 작성하기

```
meansd <- function(x) {  
  mm <- mean(x)  
  stdev <- sd(x)  
  result <- c(mm, stdev)  
  names(result) <- c("평균", "표준편차")  
  print(result)  
}
```

- 이 함수는 수치자료 x를 매개변수를 받고, mm에 평균을 stdev에 표준편차를 저장하여 이를 result라고 하고 가독성을 위해 이 두 값에 이름을 설정한 후 결과를 인쇄함



호출 결과

```
> meansd(BMI$height)  
      평균      표준편차  
162.096045 5.  5.472746
```



계산 보기



변동계수의 계산을 위한 함수 작성하기



자료

CV의 계산은 내장함수가 없으므로 패키지를 사용하거나 매번 평균, 표준편차 계산을 해야 하므로 이를 함수로 작성해 두면 패키지를 불러 필요도 매번 긴 명령을 사용할 필요가 없다. 이 함수를 작성해보자.

```
CV <- function(x) {  
  cv <- sd(x)/mean(x) * 100  
  return(cv)  
}
```



이 함수를 1:100의 자료로 호출한 결과

```
> CV(seq(1,100))  
[1] 57.4485
```



계산 보기



매개변수가 정해지지 않은 경우 함수 작성하기



... (빈 칸 없이 점 세 개)를 사용하여 설정 가능

➤ ...의 주 사용목적은 정해지지 않은 개수의 매개변수를 다른 함수에 전달하는 역할



매개변수 ... 사용예

➤ NA가 없는 자료

```
x <- 1:100
```

➤ NA가 있는 자료

```
xna <- c(x, NA)
```




계산 보기



매개변수가 정해지지 않은 경우 함수 작성하기



자료 1

함수를 다음과 같이 작성하고 매개변수에 ... 이 있는 상태에서 몇 가지 방법으로 호출해보자.

```
my.mean1 <- function(x, ...){  
  mx <- mean(x,...)  
  return(mx)  
}
```



매개변수 하나만으로 호출한 결과

```
> my.mean1(x)  
[1] 50.5
```




계산 보기



매개변수가 정해지지 않은 경우 함수 작성하기



NA가 있는 매개변수로 호출한 결과

```
my.mean1(xna)
```

```
[1] NA
```

* NA가 결과로 얻어짐



정해지지 않은 매개변수 ...에 추가 옵션을 설정한 결과

```
> my.mean1(xna, na.rm=T)
```

```
[1] 50.5
```

* NA 제외 가능



계산 보기



매개변수가 정해지지 않은 경우 함수 작성하기



자료 2

비슷한 사용 예로 호출해보자.

```
my.mean2 <- function(..., narm = T) {  
  mx <- mean(..., na.rm=narm)  
  return(mx)  
}
```



함수 my.mean2를 설정한 결과

```
> my.mean2(x)  
[1] 50.5  
> my.mean2(xna)  
[1] 50.5
```

➤ 함수의 호출에서 자료 x만 설정하면 x가 ...에 해당되어 함수 내에서 na.rm=T가 추가됨

함수의 매개변수 확인

📖 args 함수

📖 attributes 함수

📖 is.function 함수

Loren ipsum dolor sit amet, ius an molestie facilisi erroribus, mutat nalerum delectus ei vis. Has ornatus conclusionemque id, an videri molestatis sit. In etqui praesent sit. An vel agan porro comprehensan, ad ludus constituto nea, et ius utroque scaevola assuaverit.

Vix cu nodus nulla feugait, oratio facilisi ex usu, eili vitae sea te. Ea fabulas accusamus dissentias sea, facete tacinales definitiones et per. Nihil dicant mediocrem pro eu, no mei nostro sensibus platonem. Qui id sunno perpetus neglegentur. Vel ipsum novum copiosae ut. Quo et liber detracto probatus. Nam augue scribentur an. Sea oporteat percipitur incidereit at. Qui viris nemore an.



args 함수



함수의 매개변수의 값을 확인하기

```
> args(meansd)
```

```
function (x)
```

```
NULL
```

+ R-언어의 내장함수인 log 함수의 매개변수 확인

```
> args(log)
```

```
function (x, base = exp(1))
```

* 밑이 $\exp(1)=e$ 이므로 R-언어에서 log 함수는 밑이 e인 자연 log임을 알 수 있음



attributes 함수



사용자가 직접 작성한 함수의 소스 코드를 확인하기

> attributes(meansd)

```
$srcref  
function(x) {  
  mm <- mean(x)  
  stdev <- sd(x)  
  result <- c(mm, stdev)  
  names(result) <- c("평균", "표준편차")  
  print(result)  
}
```

- 함수의 소스 코드는 매개변수를 생략하고 함수를 호출하여도 얻을 수 있음



is.function 함수



함수인지 확인하기

```
> is.function(meansd)
[1] TRUE
> xx <- rep(0, 10)
> is.function(xx)
[1] FALSE
```



source 함수를 사용한 함수 파일 불러오기

 source 함수 사용

 함수의 사용법



source 함수 사용



함수로 작성한 R-언어 소스 파일

- + 대개 다른 이름으로 컴퓨터 내의 어딘가 저장된 경우가 많음
- + 편집기 등으로 열어 '복사 → 붙여넣기'를 하는 것도 한 방법이나 source 함수를 사용하는 것을 권장함



source 함수를 사용하는 경우

- + 함수의 작성과 같이 긴 프로그램을 R 콘솔에서 작성하는 것은 **단 한 번의 오타로 처음부터 입력**해야 하는 문제 발생
- + 함수의 일부 수정의 경우도 처음부터 다시 입력해야 하는 문제 발생



R-언어의 편집은 여러 가지 기능이 제공되는 문서편집기를 사용하고 필요한 만큼 작성 또는 수정한 후 작성·수정한 파일을 한 줄의 명령으로 불러 오는 것이 편리함



함수의 사용법



source 함수

```
source(file, encoding, ...)
```

- file : 읽어 들일 파일의 이름 또는 URL
- encoding : 읽을 파일의 인코딩의 종류로 한글이 포함된 경우 UTF-8이 권장됨

* source 함수로 읽어 들이는 것은 꼭 함수일 필요는 없음



함수의 사용법



만일 앞에서 사용한 meansd 함수가 이전에 한 번도 사용되지 않았거나 사용했더라도 저장하지 않은 상태인 경우

> meansd

에러: 객체 'meansd'를 찾을 수 없습니다

* meansd 함수 사용 불가

+ R 콘솔 창에서 meansd.r 파일 내의 모든 명령을 직접 입력한 것과 같은 효력을 발생하기 위한 방법

C 드라이브의 hwp 디렉토리 아래의 STI 디렉토리에 meansd 함수를 meansd.r이라는 파일로 저장



`source("c:/hwp/STI/meansd.r", encoding="UTF-8")` 라고 명령



함수의 사용법



만일 앞에서 사용한 meansd 함수가 이전에 한 번도 사용되지 않았거나 사용했더라도 저장하지 않은 상태인 경우



meansd 함수가 정의되었음을 확인하기

```
> meansd
```

```
function(x) {  
  mm <- mean(x)  
  stdev <- sd(x)  
  print(c(mm, stdev))  
}
```



실제로 자료를 사용하여 meansd 함수를 호출한 결과

```
> meansd(seq(1,100))
```

```
    평균    표준편차  
50.50000 29.01149
```



함수의 사용법



R-언어에서 파일 경로구분

- + 경로구분은 유닉스 계열과 같이 /를 사용함
- + 윈도우의 경로 구분 표시인 \는 R-언어에서 특수문자이므로 \는 \\로 표현 가능

```
source("C:\\hwp\\STI\\meansd.r")
```

- + 윈도우의 경우 대소문자를 구분하지 않으므로 파일이나 드라이브 이름에 대소문자를 구분하지 않아도 되지만 Unix 계열 등의 운영체제에서는 대소문자를 구분함
- + 경로 구분을 하지 않고 특정한 폴더에서 작업하려면 setwd 명령을 먼저 사용 가능 (wd=working directory)

```
setwd("C:/hwp/STI")  
source("meansd.r")
```