



# 02. R 프로그래밍 기초

2주차

2019년 2학기

성현곤



충북대학교 도시공학과  
Dept. of URBAN ENGINEERING

# 목차

## R 기초

- 연산
- 객체
- 객체 연산
- 객체의 유형(종류)와 속성

## R 함수

- 함수
- 내장 함수
- 함수 만들기(사용자 정의 함수)

## R 프로그래밍 기초

- 프로그래밍 개요
- 조건문
- 반복문

## R 패키지와 스크립트

- R 패키지
- R 스크립트

# R 기초

## R 연산

- 사용자 인터페이스(user interface, UI)
  - UI: 사람과 사물 또는 컴퓨터 프로그래밍 등에서 의사소통을 하기 위하여 만든 물리적 또는 가상적 매개체
    - 여기서는 R 사용자 인터페이스, RStudio 사용자 인터페이스
  - UI에서의 연산은 결과를 바로 볼 수 있음
- 벡터(vector) 연산
  - 벡터: 1차원 자료구조
    - 숫자형: 1, c(1, 3, 5)
    - 문자형: "a", c("a", "b", "c")
    - 논리형: TRUE, c(TRUE, FALSE, TRUE)
- 콜론(:) 연산
  - 콜론(:) = 하나의 벡터, 즉 숫자들의 일차원 집합
- 오류와 경고 메시지의 차이
  - 오류(error): 치명적인 문제로 코드 실행 정지
  - 경고(warning): 오류가 있으나 치명적이지 않아 코드 실행 지속

참조: c ( )  
- 값들을 하나의 벡터 또는 리스트로 결합하는 생성함수

```
> ### 연산
> 1+1
[1] 2
> 1/3
[1] 0.3333333
> 3*2.5
[1] 7.5
> 1 + (2-5)
[1] -2
> 1 % 5
Error: unexpected input in "1 % 5"
> ?c() # Combine values into a vector or List
> c(1, 3) - c(1, 3)
[1] 0 0
> c(1) - c(1, 3)
[1] 0 -2
> c(1, 2) - c(1, 3, 1)
[1] 0 -1 0
Warning message:
In c(1, 2) - c(1, 3, 1) :
  longer object length is not a multiple of shorter object length
> c(1, 2) - c(1, 3, 1, 4)
[1] 0 -1 0 -2
> c(1, 3) / c(1, 3)
[1] 1 1
> c(1) / c(1, 3)
[1] 1.0000000 0.3333333
> c(1, 2) / c(1, 3, 1)
[1] 1.0000000 0.6666667 1.0000000
Warning message:
In c(1, 2)/c(1, 3, 1) :
  longer object length is not a multiple of shorter object length
> c(1, 2) / c(1, 3, 1, 4)
[1] 1.0000000 0.6666667 1.0000000 0.5000000
> c(1, 3) * c(3)
[1] 3 9
> c(1, 2, 3) * c(2, 3)
[1] 2 6 6
Warning message:
In c(1, 2, 3) * c(2, 3) :
  longer object length is not a multiple of shorter object length
> c(1, 2, 3) * c(1, 2, 1, 2, 1, 2)
[1] 1 4 3 2 2 6
> 1:5 * 2 # 콜론 연산
[1] 2 4 6 8 10
> 1:5 * 2:6
[1] 2 6 12 20 30
> 1:3 * 2:6
[1] 2 6 12 5 12
Warning message:
In 1:3 * 2:6 :
  longer object length is not a multiple of shorter object length
> 1:3 * 2:7
[1] 2 6 12 5 12 21
> c(1, 3, 5) * 2:7 # 콜론과 c()의 연산
[1] 2 9 20 5 18 35
> 1:3 * 2:6
[1] 2 6 12 5 12
Warning message:
In 1:3 * 2:6 :
  longer object length is not a multiple of shorter object length
```

## R 연산

- 연산자(operator)
  - 산술연산자(arithmetic operators)

Operator	Description
+	덧셈(Addition)
-	뺄셈(Subtraction)
*	곱셈(Multiplication)
/	나눗셈(Division)
^ or **	누승(거듭제곱, Exponentiation)
x %% y	나눗셈의 나머지 5%%2 → 1
x %/% y	나눗셈의 몫 5%/2 → 2

출처: <http://www.statmethods.net/>

- 논리연산자(logical operators)

Operator	Description
<	미만(Less than)
<=	이하(Less than or equal to)
>	초과(Greater than)
>=	이상(Greater than or equal to)
==	정확히 같음(Exactly equal to)
!=	같지 않음(Not equal to)
!x	x가 아님(Not x)
x   y	논리합(x OR y)
x & y	논리곱(x and y)
isTRUE(x)	X가 참인지 진단(test if x is true)

출처: <http://www.statmethods.net/>

## R 연산

- 연산자(operator)
  - 기타 연산자

Operator	Description
:: ::::	접근(Access variables in a namespace)
\$ @	Component / slot extraction
[ [[	색인(Indexing)
^	누승(Exponentiation)
:	연속연산자(Sequence operator) 1:5 → c(1, 2, 3, 4, 5)
%in%	특정 값 포함여부 확인 연산자 x %in% y = x 의 값들이 y에도 포함되었는 지 확인 c(1, 2) %in c(1, 3, 5)
* /	곱, 나누기(Multiply, divide)
-> ->>	오른 쪽으로 할당(Rightwards assignment)
<- <<-	왼쪽으로 할당(Assignment (right to left))
=	할당(Assignment (right to left))
?	도움말(Help) 기능
??	찾아보기(search) 기능

- 연산자(operator) 우선순위

- ① { }
- ② ( )
- ③ \$
- ④ ^
- ⑤ -(음수)
- ⑥ %%
- ⑦ 사칙연산(\*, >, -, +)
- ⑧ 논리비교
- ⑨ 부정(!)
- ⑩ 논리(&, &&, |, ||)
- ⑪ 할당(<-, =, ->)

## R 객체

- 객체(Objects)
  - 저장된 데이터 또는 결과를 담고 있는 그릇의 이름
- R 객체 생성방법
  - (1) 데이터 혹은 결과 등의 내용으로 부터,
  - (2) 할당(assignment): 화살표 방향(<-, =, ->)으로,
  - (3) 객체의 명명
- 객체 내용물 확인
  - 객체 이름으로 실행
- 객체의 명명시 주의사항
  - 모든 대소문자, 숫자, 특수기호 등 사용가능
    - 객체 이름은 대소문자를 구분함에 주의 필요
  - 단, 숫자가 맨 처음 사용되거나 일부 특수기호는 사용할 수 없음
    - 객체 명명시 사용할 수 없는 특수기호: ^, !, \$, @, +, -, /, or, \*

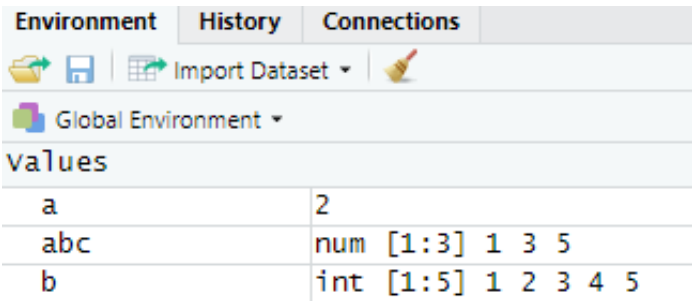
## R 객체

- 객체 생성
  - `a <- 1`
  - `abc <- c(1, 3, 5)`
  - `1:5 -> b`

- 객체 확인

- RStudio의 환경창
- 명령문창
  - `ls()` # 객체 리스트 확인
  - 직접 객체 확인

```
> ##### 객체 생성
> a <- 2 # 2를 a 라는 객체에 할당
> abc = c(1, 3, 5) # c(1, 3, 5)의 벡터를 abc라는 객체에 할당
> 1:5 -> b # 1부터 5까지 연속된 숫자를 b라는 객체에 할당
```



Environment	History	Connections
Global Environment		
values		
a		2
abc		num [1:3] 1 3 5
b		int [1:5] 1 2 3 4 5

```
> ##### 객체 확인
> ls() # 객체 리스트 확인
[1] "a" "abc" "b"
> a # 직접 a의 객체 내용 확인
[1] 2
> abc
[1] 1 3 5
> b
[1] 1 2 3 4 5
```

## R 객체 연산

- 객체 연산
  - 객체와 벡터 연산

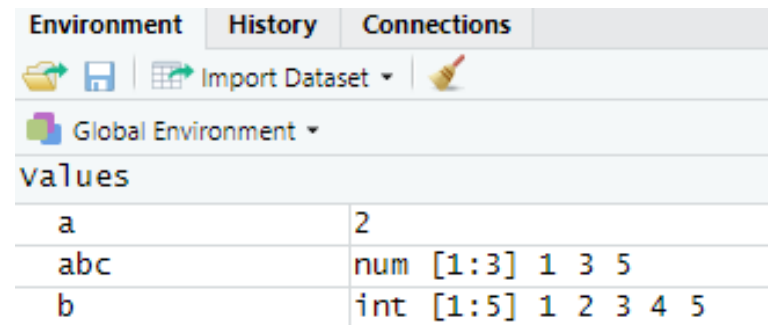
```
> ##### 객체와 벡터의 연산
> a + 3
[1] 5
> 3 + abc
[1] 4 6 8
> c(2, 3, 5) + abc
[1] 3 6 10
```

- 객체와 객체 연산

```
> ##### 객체와 객체의 연산
> a * b
[1] 2 4 6 8 10
> a / b
[1] 2.000000 1.000000 0.6666667 0.500000 0.400000
```

- 객체 연산에서의 오류와 경고

```
> ##### 객체 연산에서의 오류와 경고
> b + abc
[1] 2 5 8 5 8
warning message:
In b + abc :
  longer object length is not a multiple of shorter object length
> b + c(2, 3, 5)
[1] 3 5 8 6 8
warning message:
In b + c(2, 3, 5) :
  longer object length is not a multiple of shorter object length
> a % b
Error: unexpected input in "a % b"
```



Global Environment	
a	2
abc	num [1:3] 1 3 5
b	int [1:5] 1 2 3 4 5



## 연습문제 01

- 다음 중 객체 명명이 잘못된 것을 모두 고르시오.

- ① A
- ② abC
- ③ \_var
- ④ 3Car
- ⑤ car\_var
- ⑥ car-var
- ⑦ Var\*
- ⑧ no\_car
- ⑨ own.car
- ⑩ OWN
- ⑪ ^car
- ⑫ 자동차
- ⑬ .자동차
- ⑭ !소유
- ⑮ 수치\_해석

## 연습문제 02

- $x$ 가  $c(1:5)$ 이고,  $y$ 가 3:7일 때, 다음 연산의 결과를 적으시오.
  - $x > 7$
  - $x \leq 3$
  - $x[(x > 2) \ \& \ (x \leq 5)]$
  - $x - y$

## R 객체의 유형과 속성

- R 객체의 종류(type)
  - 벡터(vector)
    - 상수(atomic) 벡터: 값이 1개
    - 차원 배열 혹은 값들
  - 리스트(list)
    - 벡터를 일차원 집합으로 그룹화
  - 행렬(행렬)
    - 이차원 배열의 값들
  - 데이터프레임(dataframe)
    - 이차원 자료, 각 벡터는 열임
  - 배열(array)
    - N차 행렬

	단일 자료	다중 자료
1차원 (1Dimension)	벡터 	리스트 
2차원 (2Dimension)	행렬 	데이터 프레임 
n차원 (nDimension)	배열 	

## R 객체의 유형과 속성

- 벡터(vector)
  - 차원 배열 혹은 값들
  - 벡터의 종류
    - 실수형(numeric, 동의어: double)
    - 정수형(integer)
    - 문자형(character)
    - 논리형(logical)
    - 복소수형(complex)
    - 원시형(raw)

- 벡터(길이)의 확인

```
x <- c(1, 3, 5)
is.vector(x) # 벡터 여부 진단
length(x) # 벡터 길이 산정
```

- 벡터로의 변환

```
> y <- c("1", "3", "5") # 문자형 벡터
> y
[1] "1" "3" "5"
> ?mode
> ?as.vector()
> z <- as.vector(y, mode="numeric") # 실수형 벡터로 변환
> z
[1] 1 3 5
> mode(x)
[1] "numeric"
> mode(y)
[1] "character"
> mode(z)
[1] "numeric"
> typeof(x)
[1] "double"
> typeof(y)
[1] "character"
> typeof(z)
[1] "double"
```

## R 객체의 유형과 속성

- 벡터

- 정수형 벡터

- 숫자 뒤에 대문자 L을 붙여 정수형 벡터로 정의

- 정수형과 실수형의 차이

- 메모리 용량

- 정밀도

- 실수형의 16개의 소숫점, 오차

- 부동소숫점 오차는 정수형을 사용하여 회피 가능

```
> x <- c(1, 3, 5)
> mode(x)
[1] "numeric"
> typeof(xL)
[1] "integer"
> xL <- c(1L, 3L, 5L)
> ?mode
> ?typeof
> ##### 정수형 벡터
> x <- c(1, 3, 5)
> mode(x)
[1] "numeric"
> typeof(xL)
[1] "integer"
> xL <- c(1L, 3L, 5L)
> mode(xL)
[1] "numeric"
> typeof(xL)
[1] "integer"

> sqrt(2)
[1] 1.414214
> sqrt(2)^2
[1] 2
> sqrt(2)^2 - 2
[1] 4.440892e-16
```

## R 객체의 유형과 속성

- 벡터

- 문자형 벡터

- 텍스트, 문자나 문자열은 " "로 생성

- 논리형 벡터

- TRUE 또는 FALSE
    - 대문자로 따옴표 없이 입력
    - T, F로 사용가능
    - 사칙연산 가능(TRUE=1, FALSE=0)

```
> text <- c("hello", "my students", " ", "to my class!")
> text
[1] "hello"          "my students"    " "              "to my class!"
> is.character(text)
[1] TRUE
> typeof(text)
[1] "character"
> mode(text)
[1] "character"

> ##### 논리형 벡터
> logic <- c(TRUE, FALSE, TRUE, FALSE)
> logic
[1] TRUE FALSE TRUE FALSE
> typeof(logic)
[1] "logical"
> length(logic)
[1] 4
> 3 - logic
[1] 2 3 2 3
```

## R 객체의 유형과 속성

- 속성(attributes)
  - 원자 벡터(또는 모든 객체)에 부여하는 일종의 정보
    - 객체 값에 영향을 주지 않으며, 표시되지 않음
    - 객체의 관련 정보를 담아두는 장소
  - 종류
    - 이름(names), 차원(dimension: dim), 클래스(class)
- 속성의 부여
  - names()
  - dim()
- 속성의 확인
  - attributes()

```
> x <- 1:9
> attributes(x) # 속성확인
NULL
> dim(x)
NULL
> names(x) <- c("v1", "v2", "v3", "v4", "v5", "v6", "v7", "v8", "v9")
> attributes(x) # 속성확인
$names
[1] "v1" "v2" "v3" "v4" "v5" "v6" "v7" "v8" "v9"

> names(x) <- NULL # 속성 이름 제거
> attributes(x) # 속성 제거
NULL
> x + 1 # 실제 객체 값에 영향을 주지 않음
[1] 2 3 4 5 6 7 8 9 10
> dim(x) <- c(3, 3) # 차원을 3*3으로 부여
> x
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> attributes(x) # 속성확인
$dim
[1] 3 3

> dim(x) <- NULL # 속성 제거
> x
[1] 1 2 3 4 5 6 7 8 9
```

## R 객체의 유형과 속성

- 행렬(matrix)
  - 이차원 배열의 값
    - matrix() 함수로 생성
      - 인수
        - 행의 개수: nrow
        - 열의 개수: ncol
        - 배열순서: TRUE = 행방향, FALSE = 열방향
  - 속성 및 열 및 행의 개수 확인
    - attributes()
    - ncol()
    - nrow()
    -

```
> m <- matrix(1:10, nrow=2)
> m
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> m2 <- matrix(m, ncol=2, byrow=TRUE)
> m2
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10
> attributes(m2)
$dim
[1] 5 2

> ncol(m2)
[1] 2
> nrow(m2)
[1] 5
```



## R 객체의 유형과 속성

- 배열(array)
  - N 차원 배열 혹은 값들
    - array() 함수로 생성
  - 속성 확인
    - class()
    - attributes()

```
> aa <- array(1:30, dim=c(2, 3, 5)) # 2*3 행렬이 5자원 배열
> aa
, , 1
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
, , 2
      [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
, , 3
      [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18
, , 4
      [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
, , 5
      [,1] [,2] [,3]
[1,]   25   27   29
[2,]   26   28   30
> class(aa)
[1] "array"
> attributes(aa)
$dim
[1] 2 3 5
```

## R 객체의 유형과 속성

- 리스트(list)
  - 데이터를 일차원 집합으로 그룹화함
    - list() 함수로 생성
  - 개별적인 값들의 그룹화가 아니라 원자 벡터나 또 다른 리스트의 그룹화 하는 것임
  - 리스트에서는 객체의 속성이 다름을 허용함

```
> x <- list(1:10, c("R", "N"), c(TRUE, FALSE))
> x
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
 [1] "R" "N"

[[3]]
 [1] TRUE FALSE

> x2 <- list(1:10, c("R", "N"), list(TRUE, FALSE)) # 리스트의 리스트
> x2
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
 [1] "R" "N"

[[3]]
[[3]][[1]]
 [1] TRUE

[[3]][[2]]
 [1] FALSE
```

## R 객체의 유형과 속성

- 데이터프레임
  - 2차원의 자료형으로 각 벡터는 열이 됨
    - 리스트의 특별한 클래스(이차원 리스트)
  - 데이터 프레임과 행렬의 차이
    - 각각의 열에 다른 유형의 속성 (문자, 숫자, 논리형) 을 가질 수 있음
    - 행렬은 단일 속성만 허용
      - 데이터 프레임을 행렬로 변환
        - `as.matrix()`

```
> ##### 데이터 프레임
> x <- data.frame(id=1:3,
+                 name= c("R", "N", "S"),
+                 logic= c(TRUE, FALSE, TRUE))
> x
  id name logic
1  1   R  TRUE
2  2   N FALSE
3  3   S  TRUE
> class(x)
[1] "data.frame"
> attributes(x)
$names
[1] "id"    "name"  "logic"

$class
[1] "data.frame"

$row.names
[1] 1 2 3

> y <- as.matrix(x) # 행렬로 강제 변환
> y # 단일 속성만 허용되기 때문에 문자로 모두 강제 변환
  id name logic
[1,] "1" "R"  " TRUE"
[2,] "2" "N"  "FALSE"
[3,] "3" "S"  " TRUE"
> class(y)
[1] "matrix"
```

## R 객체의 유형과 속성

- 강제변환의 규칙
  - 벡터와 행렬 등과 같은 자료형인 경우 오직 한 개의 속성만 허용
    - 그러므로 2개 이상의 자료들이 변환되거나 생성되는 경우에 강제 변환의 규칙이 적용됨
  - 문자형 > 숫자형 > 논리형

```
> ##### 강제변환
> x <- c(1, "T")
> x
[1] "1" "T"
> x2 <- c(1, FALSE)
> x2
[1] 1 0
> x3 <- c(1, "F", FALSE)
> x3
[1] "1"      "F"      "FALSE"
```

## R 객체의 유형과 속성

- 요인(factor)
  - 요인은 문자로 된 범주형(categorical) 정보로 저장하여 분석하는 데 사용됨
    - 요인은 범부형 변수를 통계모델에 적용하기 용이하도록 숫자로 부호화 함
  - 범주형 정보
    - 성별(female, male), 음식종류(백반, 도시락, 라면, 짜장, 짬뽕) 등
    - factor() 함수로 생성
  - 요인을 표시할 때 라벨속성을 사용
  - unclass() 함수 적용
    - Level 표현 없이 class 특징을 없애고, 단순한 상수만 출력
    - Attribute에 따라 상수에 매칭되는 데이터 값을 같이 출력

```
> ##### 요인: 범주형 정보
> x <- c("male", "female", "female", "male", "female", "bi-gender")
> x
[1] "male"      "female"    "female"    "male"      "female"
[6] "bi-gender"
> typeof(x)
[1] "character"
> attributes(x)
NULL
> x2 <- factor(x)
> x2
[1] male      female    female    male      female    bi-gender
Levels: bi-gender female male
> typeof(x2)
[1] "integer"
> attributes(x2)
$levels
[1] "bi-gender" "female"    "male"

$class
[1] "factor"

> ?unclass
> unclass(x2) # Level 표현 없이 class 특징을 없애고, 단순한 상수만 출력
[1] 3 2 2 3 2 1
attr(,"levels")
[1] "bi-gender" "female"    "male"
> x2
[1] male      female    female    male      female    bi-gender
Levels: bi-gender female male
```

## R 객체의 유형과 속성

- NA, NaN
  - 있지만 정의되지 않은 값
    - NA: Not Available
    - NaN: not a Number
  - NA, NaN의 확인
    - is.na()
    - is.nan()
- NULL
  - 없는 값

```
> ##### NA, NaN, NULL
> x <- c(NaN, NA)
> x
[1] NaN NA
> is.na(x)
[1] TRUE TRUE
> is.nan(x)
[1] TRUE FALSE
> x + 2
[1] NaN NA
> x <- NULL
> x
NULL
```

## 연습문제 03

- 다음의 문자형 벡터를 숫자형 벡터로 전환하고, 그 속성값을 확인하는 명령문을 작성하시오.
  - `z <- c("1", "5", "10")`

## 연습문제 04

- 하나의 벡터에서 문자형, 숫자형, 요인형, 논리형 값들이 모두 존재할 때, 어떠한 자료형으로 강제 변환되어 나타나는가?
  - ① 문자형
  - ② 숫자형
  - ③ 요인형
  - ④ 논리형



## R 함수

- R 함수(function)
  - 특정한 작업을 수행하기 위하여 일련의 구문들을 체계적으로 작성한 것
  - R에서의 함수
    - 내장함수: 이미 R 프로그램에서 필요하다고 판단하여 작성해 놓은 것
      - 예: round(), mean(), abs()
    - 사용자 정의 함수(또는 나만의 함수): 사용자가 필요하다고 하여 직접 작성한 것
      - 함수의 구성요소
        - 함수명: function\_name
        - 인수: 함수가 호출될 때의 어떤 값 (arg\_1, arg\_2)
        - 함수 본문(내용): 함수가 수행하여야 하는 내용의 정의

```
> ##### 내장함수의 예
> round(2.5456)
[1] 3
> mean(c(1,3,5))
[1] 3
> x <- 1:20
> mean(x)
[1] 10.5
> round(mean(x))
[1] 10
> abs(-3)
[1] 3
```

```
function_name <- function(arg_1, arg_2, ...) {
    함수 본문(내용)
}
```

# R 함수

## R 함수

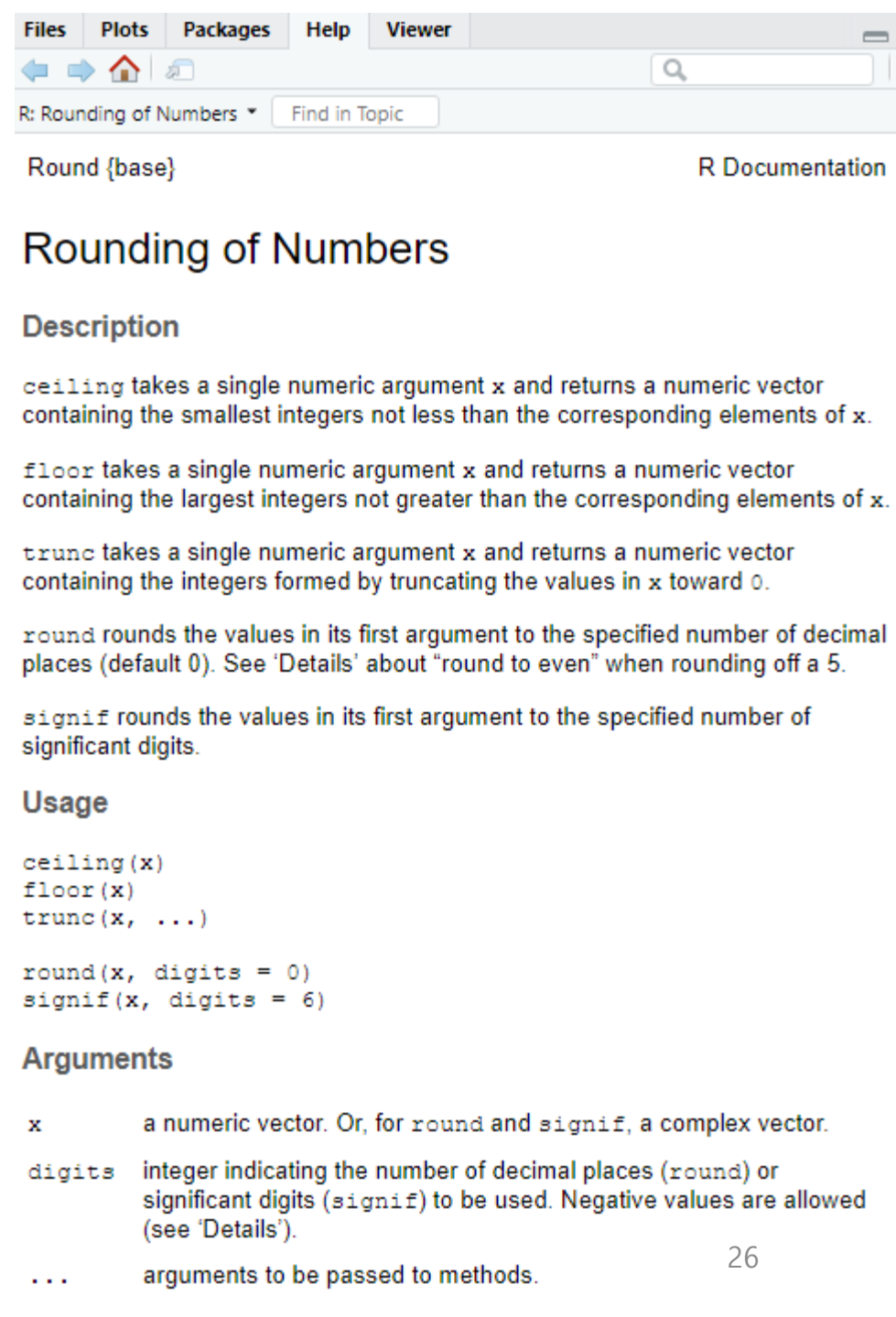
- R 함수의 정의와 내용(인수)
  - 도움말(?)로 내장함수의 정의와 내용 확인하기
    - ?round
    - ?mean
    - ?abs
- 도움말(Help)의 내용
  - 정의(Definition), 설명(Description), 용법(Usage), 인수(Arguments), 상세(Details), 예제(Examples) 등으로 구성

### Examples

```
round(.5 + -2:4) # IEEE / IEC rounding: -2 0 0 2 2 4 4  
## (this is *good* behaviour -- do *NOT* report it as bug !)
```

```
( x1 <- seq(-2, 4, by = .5) )  
round(x1) #-- IEEE / IEC rounding !  
x1[trunc(x1) != floor(x1)]  
x1[round(x1) != floor(x1 + .5)]  
(non.int <- ceiling(x1) != floor(x1))
```

```
x2 <- pi * 100^(-1:3)  
round(x2, 3)  
signif(x2, 3)
```



The screenshot shows the R Documentation page for the 'Rounding of Numbers' topic. The browser tabs include 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. The page title is 'R: Rounding of Numbers' and it includes a search bar and a 'Find in Topic' button. The main heading is 'Rounding of Numbers' with a sub-heading 'Description'. The description explains the behavior of `ceiling`, `floor`, `trunc`, `round`, and `signif` functions. The 'Usage' section lists the function signatures: `ceiling(x)`, `floor(x)`, `trunc(x, ...)`, `round(x, digits = 0)`, and `signif(x, digits = 6)`. The 'Arguments' section defines `x` as a numeric vector (or complex vector for `round` and `signif`), `digits` as an integer for decimal places or significant digits, and `...` as arguments to be passed to methods. The page is identified as 'R Documentation'.

## R 함수

- R 함수의 정의와 내용(인수)
  - 찾아보기(??)
    - 정확한 명령어가 기억나지 않을 경우는?
      - ??me
      - ??va
      - ??log
- 도움 받기 또 다른 명령어
  - help(print)
  - ?var
  - help.search( ) # 괄호 안에 찾고자 하는 문자 입력
  - example( ) # 함수 사용예
  - help.start( ) # R 시스템 전반에 대한 도움말을 담고 있는 html페이지를 보여줌

## R 함수

- 함수와 인수
  - 함수에서의 인수(arguments)를 정의하지 않을 경우, 기본값으로 실행
  - 함수의 오류와 설명
    - 소수점 셋째 자리로 반올림
    - `round(2.47245, degit = 3)`
- 함수 인수 확인 방법(함수의 인수를 모를 경우)
  - `?round`
  - `args(round)`
  - `round(2.47245, digits = 3)`

```
> #### 내장 함수에서의 인수 활용방법
> round(2.47245, degit = 3)
Error in round(2.47245, degit = 3) : unused argument (degit = 3)
```

```
> ?round
```

### Arguments

`x` a numeric vector. Or, for `round` and `signif`, a complex vector.

`digits` integer indicating the number of decimal places (`round`) or significant digits (`signif`) to be used. Negative values are allowed (see 'Details').

... arguments to be passed to methods.

```
> args(round)
function (x, digits = 0)
NULL
> round(2.47245, digits = 3)
[1] 2.472
```

# R 내장 함수(Built-in Functions)

- 숫자 함수(numeric functions)

Function	Description
abs(x)	절댓값(absolute value)
sqrt(x)	제곱근(square root)
ceiling(x)	올림 ceiling(3.475) is 4
floor(x)	내림 floor(3.475) is 3
trunc(x)	자름 trunc(5.99) is 5
round(x, digits=n)	반올림 round(3.475, digits=2) is 3.48
cos(x), sin(x), tan(x), acos(x), cosh(x), acosh(x)	
log(x)	자연로그(natural logarithm)
Log10(x)	상용로그(common logarithm)
exp(x)	지수( $e^x$ )

출처: <http://www.statmethods.net/>

```
> ##### 숫자 함수
> sqrt(10)
[1] 3.162278
> round(2.587, digits=2)
[1] 2.59
> trunc(2.387, digits=2)
[1] 2
> ceiling(2.387)
[1] 3
> log(20)
[1] 2.995732
> exp(3)
[1] 20.08554
```

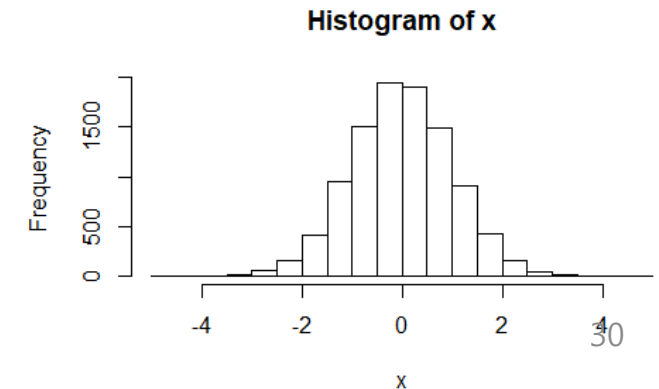
## R 내장 함수(Built-in Functions)

- 통계확률 함수(statistical probability functions)(1)

Function	Description
<code>dnorm(x)</code>	정규밀도함수 normal density function (by default $m=0$ $sd=1$ )
<code>pnorm(q)</code>	누적정규확률 cumulative normal probability for $q$ (area under the normal curve to the right of $q$ ) <code>pnorm(1.96)</code> is 0.975
<code>qnorm(p)</code>	정규분위값 normal quantile. value at the $p$ percentile of normal distribution <code>qnorm(.9)</code> is 1.28 # 90th percentile
<code>rnorm(n, m=0, sd=1)</code>	정규분포 무작위 표본추출 $n$ random normal deviates with mean $m$ and standard deviation $sd$ . #50 random normal variates with mean=50, sd=10 <code>x &lt;- rnorm(50, m=50, sd=10)</code>

출처: <http://www.statmethods.net/>

```
> #### 통계확률 함수
> ?dnorm # 정규분포 밀도(density) 함수: dnorm(x, mean = 0, sd = 1,
> dnorm(0.5, mean = 0, sd = 1) # returns the value of the probab
[1] 0.3520653
> dnorm(0.5)
[1] 0.3520653
> dnorm(4, mean = 5, sd = 3)
[1] 0.1257944
> ?pnorm # 정규분포 확률(density) 함수
> pnorm(2) # pnorm(q, mean = 0, sd = 1)
[1] 0.9772499
> pnorm(2, mean = 5, sd = 3)
[1] 0.1586553
> ?qnorm # 정규분포 분위(density) 함수: qnorm(p, mean = 0, sd = 1)
> qnorm(.5) # 평균=0, 표준편차=1인 정규분포에서의 50번째 분위의 값은?
[1] 0
> qnorm(.96)
[1] 1.750686
> ?rnorm # 무작위 정규분포 생성함수
> rnorm(5) # rnorm(n, mean = 0, sd = 1)
[1] 0.9750579 0.5959888 1.2009551 -0.2114565 -1.6105626
> rnorm(5, mean = 2, sd = 3)
[1] 6.0931999 -1.1986051 3.1345478 0.8239682 -0.7371257
> x <- rnorm(10000) # rnorm(n, mean = 0, sd = 1)
> plot(x)
> hist(x)
```



# R 내장 함수(Built-in Functions)

- 통계확률 함수(statistical probability functions)(2)

```
dbinom(x, size, prob)
pbinom(q, size, prob)
qbinom(p, size, prob)
rbinom(n, size, prob)
```

이항분포

Binomial distribution where size is the sample size and prob is the probability of a heads ( $\pi$ )

# prob of 0 to 5 heads of fair coin out of 10 flips

```
dbinom(0:5, 10, .5)
```

# prob of 5 or less heads of fair coin out of 10 flips

```
pbniom(5, 10, .5)
```

```
dpois(x, lamda)
ppois (q, lamda)
qpois(p, lamda)
rpois (n, lamda)
```

포아송분포

Poisson distribution with  $m=std=lamda$

# probability of 0,1, or 2 events with  $lamda=4$  `dpois(0:2, 4)`

# probability of at least 3 events with  $lamda=4$  `1-ppois(2,4)`

```
dunif(x, min=0, max=1)
punif(q, min=0, max=1)
qunif (p, min=0, max=1)
runif (n, min=0, max=1)
```

균일분포

Uniform distribution, follows the same pattern as the normal distribution above.

#10 uniform random variates

```
x<- runif(10)
```

# R 내장 함수(Built-in Functions)

- 기타통계 함수(other statistical functions)

Function	Description
mean(x, trim=0, na.rm=FALSE)	(절사)평균 mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x, trim=.05, na.rm = TRUE)
sd(x)	표준편차 standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(x)	중위값(Median)
quantile(x, probs)	분위값 quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in [0,1]. # 30th and 84th percentiles of x y <- quantile(x, c(.3,.84))
Range(x)	범위(range)
Sum	합계(sum)
Min	최솟값(minimum)
Max	최댓값(maximum)

```
> ##### 기타 통계함수
> mean(1:20)
[1] 10.5
> mean(1:20, trim=0.1) # 10% 상하위 절사(trim) 평균
[1] 10.5
> ##### 기타 통계함수
> mean(1:20)
[1] 10.5
> mean(c(5, 1:20), trim=0.1) # 10% 상하위 절사(trim) 평균
[1] 10.17647
> sd(1:20)
[1] 5.91608
> median(1:20)
[1] 10.5
> quantile(1:20, c(0.3, 0.5, 0.7))
 30% 50% 70%
 6.7 10.5 14.3
> range(1:20)
[1] 1 20
> sum(1:20)
[1] 210
> min(1:20)
[1] 1
> max(1:20)
[1] 20
```




## R 함수 만들기

- R 함수 정의(function construction)

- 왜 만들까?

- 동일한 작업으로 반복적으로 여러 번 작업을 해야 하는 경우

- 구성요소 = 이름 + 인수 + 코드

`function_name () { }`         $\text{function\_name} \xleftarrow{\text{이름}} (\arg1, \arg2, \dots) \xrightarrow{\text{인수}} \{ \text{코드(실행작업)} \text{return(반환값)} \}$

- 예:

- 평균이 m이고, 표준편차가 s인 정규분포로 n개의 숫자를 무작위로 추출하여 히스토그램을 그리고 또한 이의 중위값을 함수 작성

```
norm_hist <- function(n, m, s) {  
  x <- rnorm(n, mean=m, sd=s)  
  hist(x)  
  median(x)  
}
```

## R 함수 만들기

- R 사용자 정의 함수 호출

- `norm_hist(m=0, s=1, n=1000)`

- 실행결과

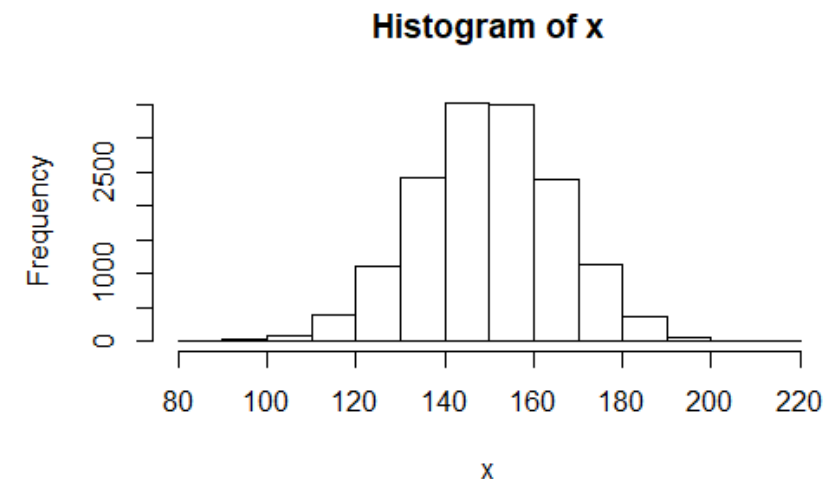
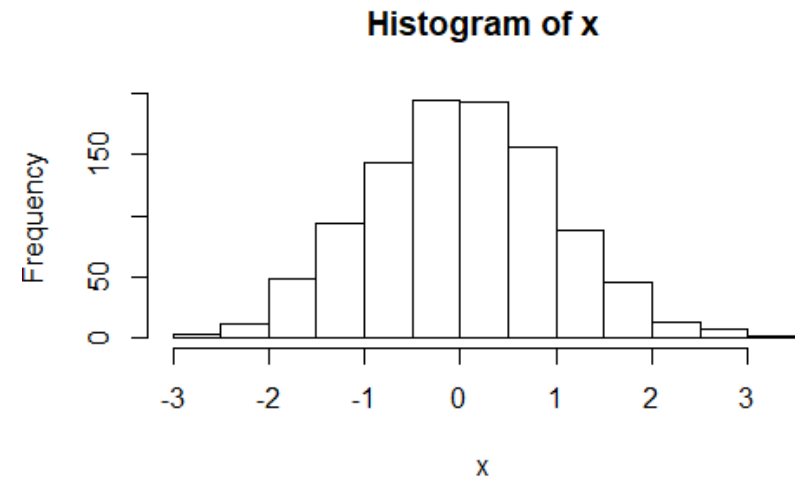
```
> norm_hist(m=0, s=1, n=1000) # 함수 호출  
[1] 0.008107451
```

- `norm_hist(m=150, s=16, n=15000)`

- 실행결과

```
> norm_hist(m=150, s=16, n=15000) # 함수 호출  
[1] 149.942
```

```
norm_hist <- function(n, m, s) {  
  x <- rmnorm(n, mean=m, sd=s)  
  hist(x)  
  median(x)  
}
```



## R 함수 만들기

- R 사용자 정의 함수 정의 및 호출 추가 예제(1)
  - $x$ 와  $y$ 의 객체를 제공하여, 그 합을 구하는 함수 정의

```
f_xy <- function(x, y) {  
  y <- x^2 + y^2  
  return(y)  
}
```

```
f_xy(x=2, y=3)  
f_xy(x=100, y=25)  
f_xy(x=1, y=3)  
f_xy(x=3, y=7)
```

```
> #####  $x$ 와  $y$ 의 객체를 제공하여, 그 합을 구하는 함수 정의  
> f_xy <- function(x, y) {  
+   y <- x^2 + y^2  
+   return(y)  
+ }  
> f_xy(x=2, y=3)  
[1] 13  
> f_xy(x=100, y=25)  
[1] 10625  
> f_xy(x=1, y=3)  
[1] 10  
> f_xy(x=3, y=7)  
[1] 58
```

# R 함수 만들기

- R 사용자 정의 함수 정의 및 호출 추가 예제(2)
  - 객체  $x$ 의 평균, 표준편차, 최솟값, 최댓값을 구하는 사용자 정의 함수

```
> ##### 객체 x의 평균, 표준편차, 최솟값, 최댓값을 구하는 사용자 정의 함수
> x_stat <- function(x) {
+   x_mean <- mean(x)
+   x_sd <- sd(x)
+   x_max <- max(x)
+   x_min <- min(x)
+   return(c(x_mean, x_sd, x_min, x_max))
+ }
> x_stat(x=1:20)
[1] 10.50000  5.91608  1.00000 20.00000
> x_stat(x=5:10)
[1]  7.500000  1.870829  5.000000 10.000000
> x_stat(x=c(1,3,5, 7:15))
[1]  9.000000  4.390071  1.000000 15.000000
```

## 연습문제 05-07

- Q05: 0.1968의 숫자에 대하여 소숫점 두째자리까지 반올림하고자 한다. 이 때 사용하게 되는 함수와 명령문은? 그 결과값은?
- Q06: 평균이 1이고, 표준편차가 5인 정규분포로 10개의 숫자를 무작위로 추출하여 생성하라는 명령문을 작성하시오.
- Q07: 1부터 50까지 숫자로 된 벡터에 대하여 표준편차와 중위값, 그리고 합계를 구하시오.

## 연습문제 08

- 2개의 주사위(1:6)를 던져 나오는 숫자의 합을 구하는 함수를 정의하시오. 이때의 함수는 'r\_dice' 명명하고, 사용하는 무작위 무작위 샘플링 함수는 sample()이다.
  - ?sample

```
r_dice <- function() {  
  x <- 1:6  
  y <- sample(x, size=2, replace = TRUE)  
  sum(y)  
}
```

```
r_dice # 함수호출1  
r_dice # 함수호출2  
r_dice # 함수호출3
```

## R 프로그래밍 개요

- 프로그래밍 전략
  - 프로그램 작성내용이 복잡할 경우에는 모듈화하여 진행
  - 가능한 한 방법을 자연으로 정리하고, 이를 R 코드로 작업
- 프로그래밍 설계원칙
  - 효율성, 일반성, 간결성, 일관성 등
- 프로그래밍의 필수요소: 조건문과 반복문
  - if , ifelse, for , while , repeat

## R 조건문

- 조건문
  - 조건에 따라 코드 실행 여부 결정
  - 문법

```
if (조건) {  
  조건이 참일 때 실행할 코드  
}  
else { 조건이 거짓일 때 실행할 코드  
}
```

```
ifelse (  
  조건 객체 검증(test),  
  참(yes),  
  거짓(no),  
)
```

- if문과 ifelse문의 차이
  - 후자는 다수의 데이터를 한 번에 적용하는 연산에서 유리함



## R 조건문

- 조건문

- 한 개의 숫자가 짝수인 지 홀수인 지에 따라 실행내용이 다른 경우의 조건문 작성

```
if (조건) {  
  조건이 참일 때 실행할 코드  
}  
else { 조건이 거짓일 때 실행할 코드  
}
```

- 다수의 숫자들에 대하여 짝수인 지 홀수인 지에 따라 실행내용이 다른 경우의 조건문 작성

ifelse (조건 객체 검증(test), 참(yes), 거짓(no) )

```
x <- 5  
if (x %% 2 == 0) {  
  y <- "짝수"  
  print(y)  
} else {  
  y <- "홀수"  
  print(y)  
}
```

```
> x <- c(-3:3, 7)  
> if (x %% 2 == 0) {  
+   y <- "짝수"  
+   print(y)  
+ } else {  
+   y <- "홀수"  
+   print(y)  
+ }
```

```
[1] "홀수"  
warning message:  
In if (x%%2 == 0) { :  
  the condition has length > 1 and only the first element will be
```

```
> x <- c(-3:3)  
> ifelse ( x %% 2 == 0, "짝수", "홀수" )  
[1] "홀수" "짝수" "홀수" "짝수" "홀수" "짝수" "홀수"  
> z <- ifelse (x %% 2 == 0, "짝수", "홀수")  
> xz <- data.frame(x, z)  
> xz  
  x    z  
1 -3 홀수  
2 -2 짝수  
3 -1 홀수  
4  0 짝수  
5  1 홀수  
6  2 짝수  
7  3 홀수
```

## R 조건문

- 조건문
  - 두 개 이상의 논리값 진단에 따른 조건문

ifelse (조건 1, 실행 코드 1,  
ifelse (조건 2, 실행코드 2,  
ifelse (조건 3, 실행코드 3, 실행코드 4 )

```
> x <- -5:5
> z <- ifelse (x %% 5 == 0, "zero",
+           ifelse (x %% 5 == 1, "one",
+           ifelse (x %% 5 == 2, y <- "two",
+           ifelse (x %% 5 == 3, y <- "three", y <- "four"))))
> xz <- data.frame(x, z)
> xz
  x     z
1 -5 zero
2 -4  one
3 -3  two
4 -2 three
5 -1  four
6  0 zero
7  1  one
8  2  two
9  3 three
10 4  four
11 5 zero
```

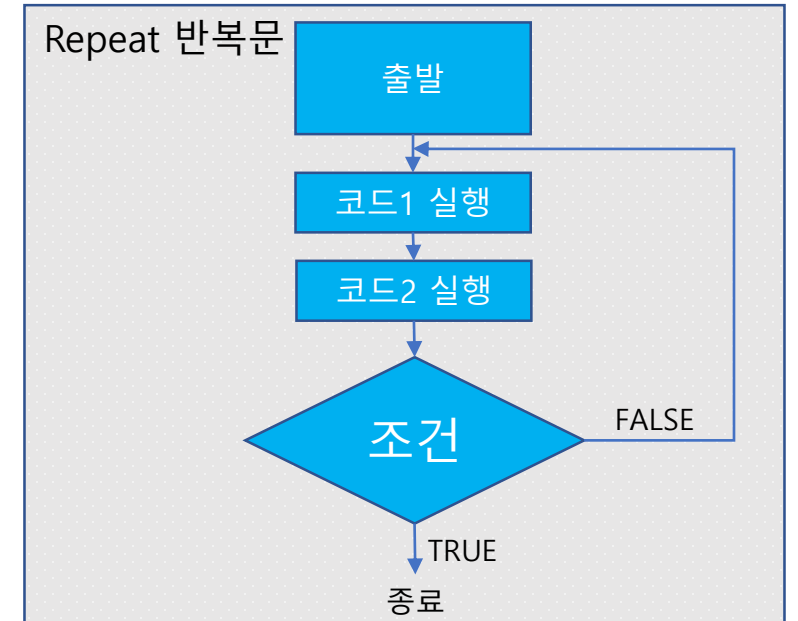
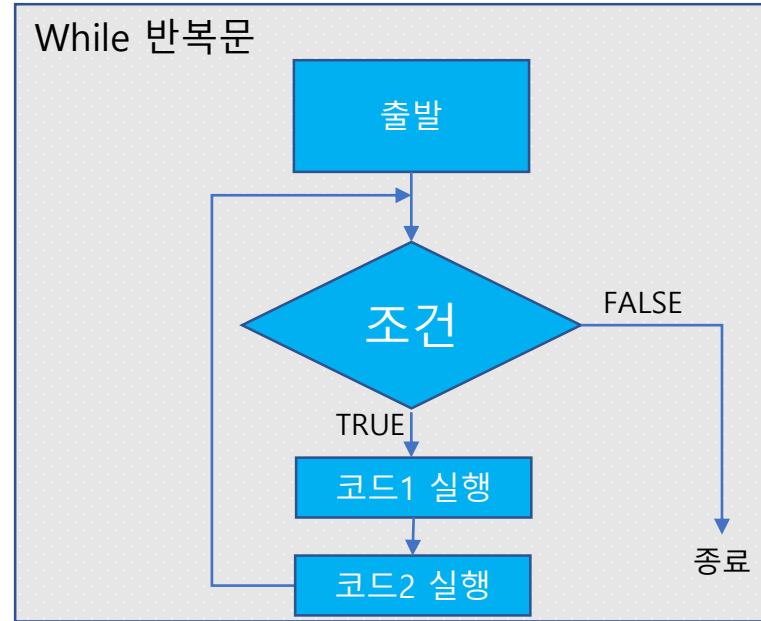
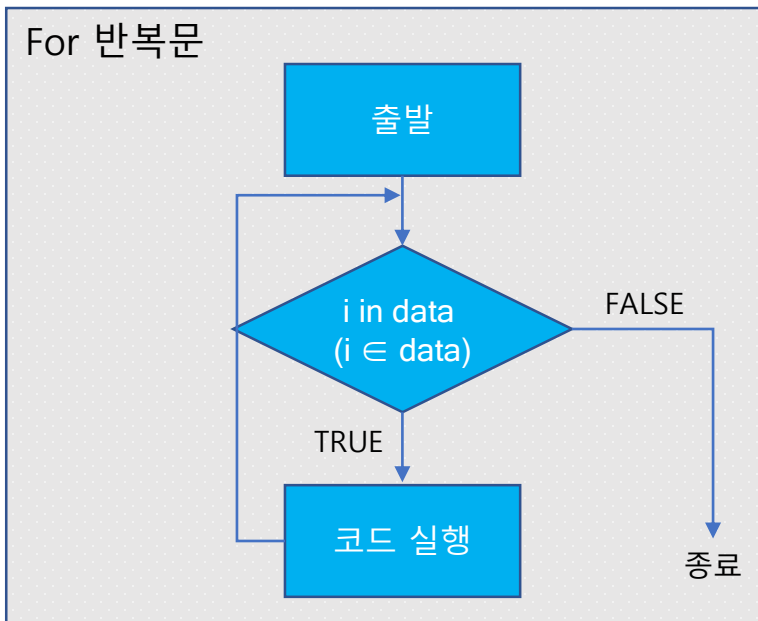
```
> xy <- data.frame(x=c(1:10), y=c(14:5))
> xy
  x  y
1  1 14
2  2 13
3  3 12
4  4 11
5  5 10
6  6  9
7  7  8
8  8  7
9  9  6
10 10 5
> xyz <- ifelse(xy$x >=5 & xy$y >=8, "f1",
+           ifelse(xy$x <5 & xy$y >=8, "f2", "f3"))
> xyz
[1] "f2" "f2" "f2" "f2" "f1" "f1" "f1" "f3" "f3" "f3"
```

# R 프로그래밍 기초

## R 반복문

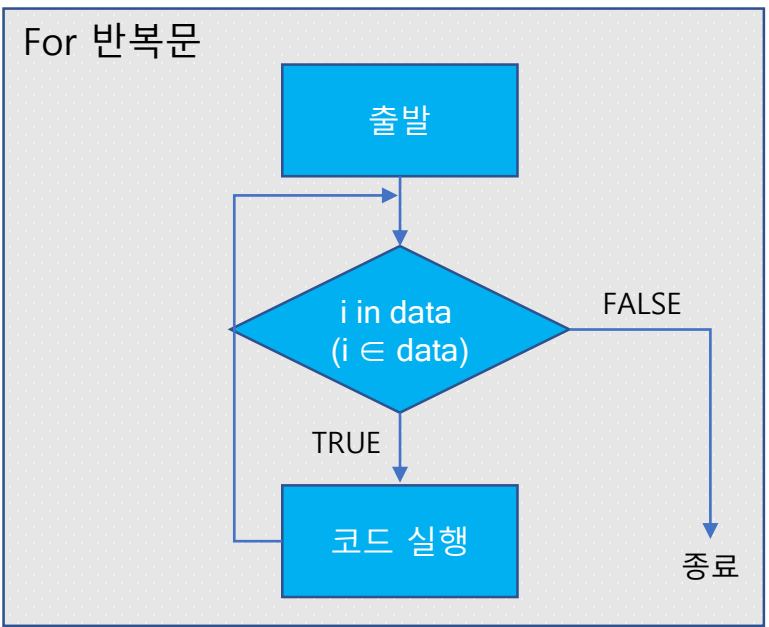
- for문, while문, repeat문
- 반복문내 실행 내용 조정: break, next
  - break: 반복문 종료
  - next: 현재 수행 중인 블록의 반복문 수행 중단 및 다음 반복 실행

문법	의미
for (i in data) { 코드 }	Data에 해당하는 각각의 값을 변수 i에 할당하면서 블록({ })안에 있는 코드를 실행
while (조건) { 코드 }	조건이 참일 때 블록({ })안에 있는 코드를 실행
repeat { 코드 }	블록({ })안의 문장을 반복해서 실행하며, 다른 언어에서의 do-while에 해당함



## R 반복문

- For 반복문
  - 1부터 10까지 숫자 벡터인 객체 x에 1씩 더하면서  $y = 10 + 5*x$  값을 순차적으로 프린터하기



```
> for (x in 1:10) {  
+   y = 10 + 5 * x  
+   print(y)  
+ }  
[1] 15  
[1] 20  
[1] 25  
[1] 30  
[1] 35  
[1] 40  
[1] 45  
[1] 50  
[1] 55  
[1] 60
```

## R 반복문

- while 반복문

- while(TRUE 또는 FALSE) : { if () break } → break

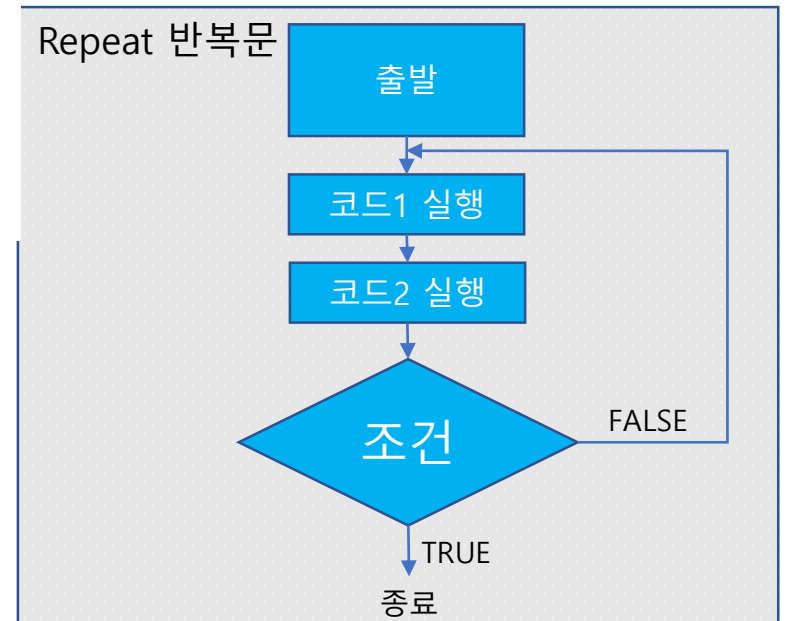
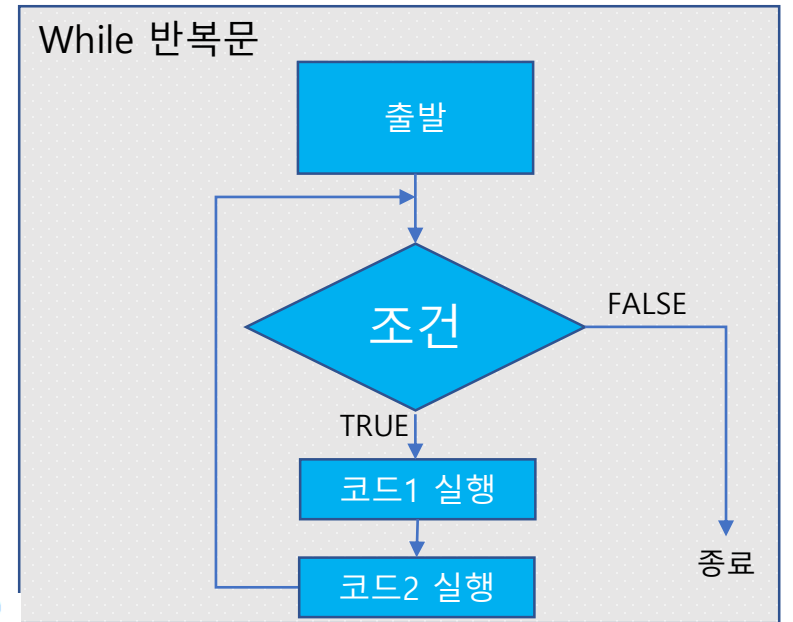
- 예: i의 값이 30보다 적거나 같으면, i에 숫자 3을 더하여 출력

```
> ##### while(TRUE 또는 FALSE): { if () break } => stop
> j <- 1
> while(TRUE) {
+   j <- j+3
+   if (j > 25) break
+ }
> j
[1] 28
```

- repeat 반복문

- repeat { if ( ) break } → break

```
> # repeat { if ( _) break } => stop
> k <- 1
> repeat {
+   k <- k+5
+   if (k > 25) break
+ }
> k
[1] 26
```



# R 반복문

- 반복문내 실행 내용 조정: break, next
  - break: 반복문 종료
  - next: 현재 수행 중인 블록의 반복문 수행 중단 및 다음 반복 실행

```
> ##### next
> 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
```

```
> i <- 1
> while(i<=10){
+   i <- i + 1
+   if(i %% 2 !=0)
+     {next # print()를 실행하지 않고 다시 while 문으로 반복
+   }
+   print(i)
+ }
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```

## 연습문제 09

- if문을 활용하여 어떠한 숫자 또는 객체를 항상 양수이 되도록 하기 위한 조건문을 작성하시오.

## 연습문제 10

- $z$ 가 홀수이면  $x$ 와  $y$ 를 덧셈하고, 짝수이면 뺄셈을 실행하는 나만의 함수를 정의하고, 아래와 같은 경우의 결과 값을 적으시오.
  - $z\_xy(z=5, x=2, y=3)$
  - $z\_xy(z=4, x=2, y=3)$
  - $z\_xy(z=7, x=2, y=3)$



# R 패키지와 스크립트

## R 패키지

- R의 함수들
  - 기본R(base R)
    - R 프로그램에 내장되어 있는 함수
    - R을 열 때 자동으로 불러오는 함수들
  - R packages
    - 내장되어 있지 않은 함수들
    - Packages = 함수 + 도움말 파일 + 데이터 세트
  - 왜 미리 내장하지 않을까?
    - R의 속도 향상(꼭 필요한 것만 사용함으로써)

- R Packages 설치
  - `install.packages("패키지 이름")`
    - 예:
      - `install.packages("ggplot2")`
      - `install.packages(c("ggplot2", "dplyr"))`
      - 온라인 미리 사이트 선택
- R package 불러오기
  - 설치 후 불러오기
    - `library(패키지 이름)`
      - " "가 없음에 주의
      - `library(ggplot)`
      - `library(ggplot2, dplyr)`
    - `request()`
      - 잘 사용하지 않음

```
> ?install.packages()
> install.packages(c('ggplot2', 'dplyr'))
WARNING: Rtools is required to build R packages before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing packages into 'C:/Users/hgs_hom
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/Rtools/
Content type 'application/zip' length 3959
downloaded 3.8 MB

trying URL 'https://cran.rstudio.com/bin/windows/Rtools/
Content type 'application/zip' length 3255
downloaded 3.1 MB

package 'ggplot2' successfully unpacked and
package 'dplyr' successfully unpacked and

The downloaded binary packages are in
C:\Users\Public\Documents\ESTsoft\
> library(ggplot2, dplyr)
```

## R 패키지

- R 패키지를 찾는 방법
  - 메일링 리스트
    - <http://stat.ethz.ch/mailman/listinfo/r~packages>
    - 새로운 패키지 소개 및 지난 뉴스 보관
  - R 블로그 활용
    - [www.r-bloggers.com](http://www.r-bloggers.com)
    - <http://supprot.rstudio.com> # Getting Started 섹션
  - 주제별 분류 사이트
    - <http://cran.r-project.org/web/views>

## R 패키지

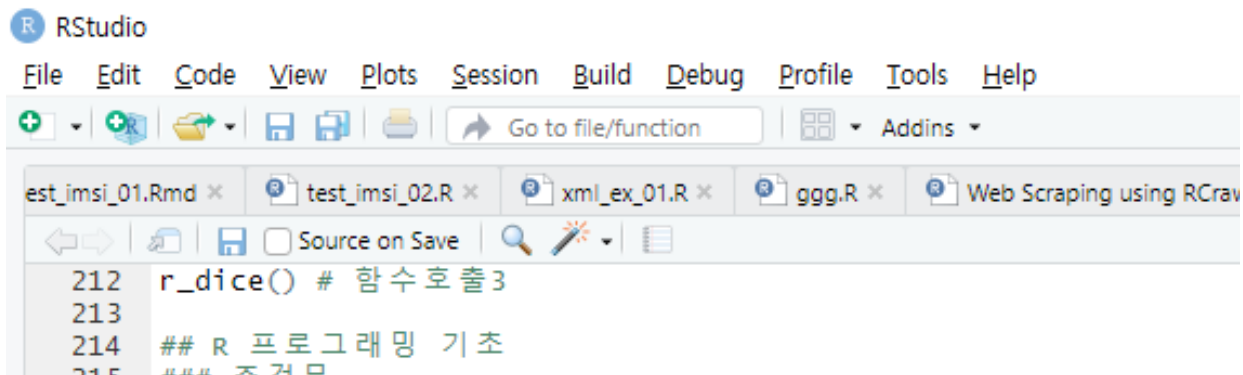
- R 패키지 업데이트
  - R packages는 진화 중
    - R 코어 개발팀은 계속해서 버그 찾고, 성능 개선노력
    - 예상하지 못한 버그 직면할 경우, 최신 버전으로 패키지 업데이트 필요
  - 패키지 업데이트
    - `update.packages` 함수로 체크 및 최신 버전 설치

# R 패키지와 스크립트

## R 스크립트(Script)

- R 스크립트 사용시의 장점
  - 반복성
    - 언제든지 동일한 작업 수행
  - 재현성
    - 누구나 동일한 작업 수행
  - 확장성
    - 누군가가 한 작업을 R 스크립트에서 수정하여 유사한 작업을 수행

- 명령의 실행 및 반복, 수정 후 재실행
  - 코드 재입력??????
  - R 스크립트 활용
    - R Script란?
      - 실행할 명령어들을 모아놓은(또는 미리 작성한) 파일
      - R 코드를 저장한 텍스트 파일
      - 스크립트로 저장한 파일을 열어 재실행 및 재분석 가능
      - 스크립트로 저장하면, 다른 사람도 동일한 실행결과와 처리를 공유하게 됨
- 열기:
  - 파일-> 새파일-> R 스크립트 메뉴 선택
- 저장하기
  - 파일-> 다른 이름으로 저장



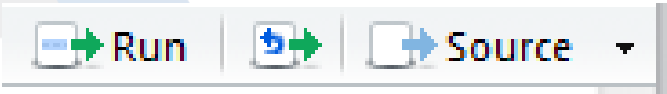
The screenshot shows the RStudio interface. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and a search bar. The script editor shows the following R code:

```
212 r_dice() # 함수호출3
213
214 ## R 프로그래밍 기초
215 ### 주커모
```

# R 패키지와 스크립트

## R 스크립트 (Script)

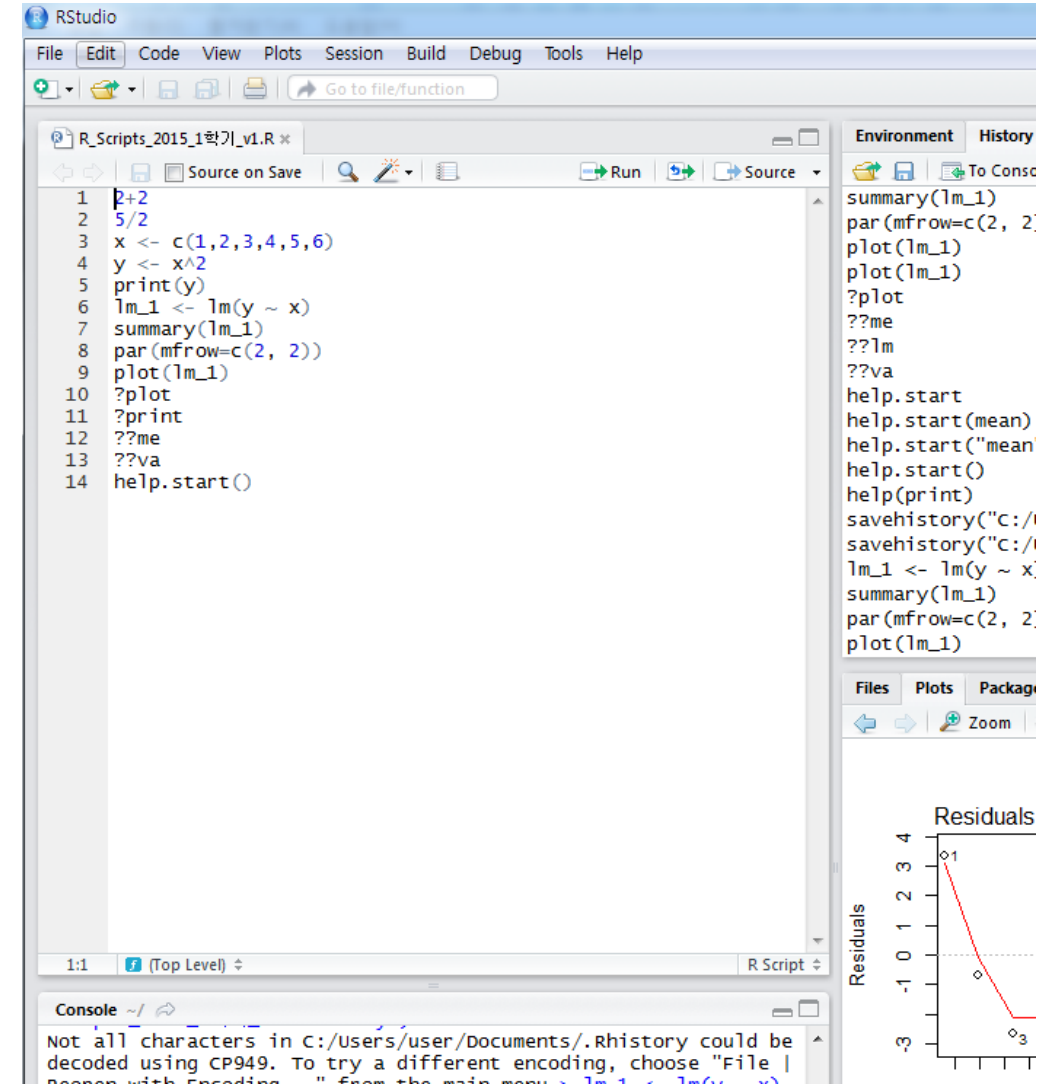
- 실행(Run)



- 실행 버튼을 누르면 자동으로 코드를 한 줄씩 실행
  - 커서가 위치한 행의 코드 실행
- 구간 실행
  - 마우스로 구간 선택 후 실행버튼
  - 선택된 구간의 코드를 모두 실행

```
231 z_xy(z=5, x=2, y=3)
232 z_xy(z=4, x=2, y=3)
233 z_xy(z=7, x=2, y=3)
234
```

- 전체 실행
  - 소스(source)버튼
- 실행버튼 대신 Ctrl+Enter 단축키 활용가능








## 연습문제 11

- 'tidytext'라는 R 패키지를 설치하고, 불러와서 실행하고자 한다. 순서와 내용이 옳은 것은?

- ① `install.packages(tidytext) → library(tidytext)`
- ② `library(tidytext) → install.packages(tidytext)`
- ③ `install.packages("tidytext") → library("tidytext")`
- ④ `install.packages("tidytext") → library(tidytext)`

## 연습문제 12

- R 스크립트에서 한 줄 씩 실행하고자 할 때의 버튼에 해당하는 것은?

- ①  Source 버튼
- ②  Run 버튼
- ③  버튼
- ④  버튼
- ⑤  버튼

# 요약

## R 기초

- 연산
- 객체
- 객체 연산
- 객체의 유형(종류)와 속성

## R 함수

- 함수
- 내장 함수
- 함수 만들기(사용자 정의 함수)

## R 프로그래밍 기초

- 프로그래밍 개요
- 조건문
- 반복문

## R 패키지와 스크립트

- R 패키지
- R 스크립트



- 질의와 토의(Question & Discussion)
  - 이번 강의 내용을 시청하고, 실행하면서 궁금한 점이나 어려운 점에 대하여 토의해봅시다.
- 다음 주 강의주제
  - R 데이터 마이닝
  - dplyr 패키지