R프로그램 기초: 연산자 및 함수

Jinseog Kim
Dongguk University
jskim1986@gmail.com

2018-04-12

Contents

1 R 연산자

1.1 R 연산자의 종류

Table 1: R operators

Operator	Descriptions
-,+,*,/	Minus,Plus, Multiplication, Division
%%	Modulus(나머지연산)
%/%	Integer division(정수나누기의 몫)
<	Less than
>	Greater than
==	Equal to
>=	Greater than or equal to
<=	Less than or equal to
!	Unary not
^	Exponentiation
&	And, vectorized
&&	And
¦	Or, vectorized
¦¦	Or

Operator	Descriptions
<-	Left assignment
=	Left ssignment
->	Right assignment
<<-	global assignment(함수 외부의 변수값 지정)

1.2 R 연산자 예제

나머지 연산

```
x <- c(1, 10, 13, 3)
x %% 2 # 나머지연산
```

```
# [1] 1 0 1 1
```

```
x%/% 3 # 정수나누기의 몫
```

```
# [1] 0 3 4 1
```

부등식/부정

```
x > 3  # 부등식
```

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

```r
y <- c(3, 5, 2, 1)
x > y    # 부등식 - 벡터간 비교
```

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

```r
z <- TRUE
!z      # Not
```

```
# [1] FALSE
```

 논리연산

```r
x1 <- c(T, T, F, F)
y1 <- c(F, F, T, F)
x1 | y1  # or 벡터연산
```

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

```r
T || F   # 처음 값만 비교: 단일값
```

```
# [1] TRUE
```

```r
x1 &  y1  # and 벡터연산
```

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

```r
x1 && y1  # 처음 값만 비교: 단일값
```

```
# [1] FALSE
```

2 R 함수

1. 내장함수
2. 사용자 함수의 작성

2.1 내장함수(built-in functions)

Table 2: 내장함수

함수	R 함수
제곱근	sqrt
지수함수	exp
로그함수	log(5), log2(5), log10(5), log(5, base=3)
최대값	max, pmax
최소값	min, pmin
합	sum
평균	mean
절대값	abs
누적연산	cummax, cummin, cumprod, cumsum
삼각함수	sin, cos, tan
올림,반올림..	ceiling, round, trunc, floor

2.2 내장함수 예제

```
  a <- 1:5
  sqrt(a)
```

```
# [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
  exp(a)
```

```
# [1]   2.718282   7.389056  20.085537  54.598150 148.413159
```

```
  out <- (a + sqrt(a))/(exp(2)+1); out
```

```
# [1] 0.2384058 0.4069842 0.5640743 0.7152175 0.8625604
```

```
  x1 <- seq(-2, 4, by = .5); x1
```

```
#  [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0
```

```r
floor(x1)
```

```
#  [1] -2 -2 -1 -1  0  0  1  1  2  2  3  3  4
```

```r
trunc(x1)
```

```
#  [1] -2 -1 -1  0  0  0  1  1  2  2  3  3  4
```

```r
a <- c(1,-2,3,-4)
b <- c(-1,2,-3,4)
min(a,b)
```

```
# [1] -4
```

```r
pmin(a,b)
```

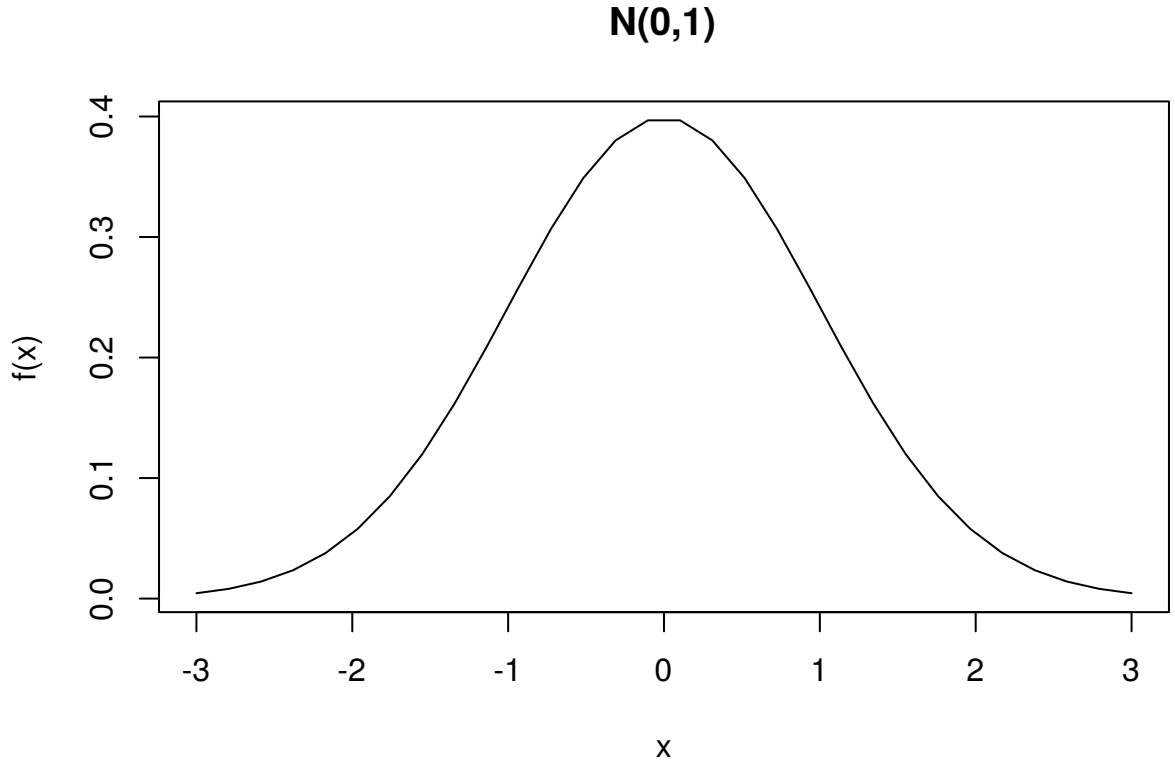
```
# [1] -1 -2 -3 -4
```

2.3 확률분포와 관련된 함수

1. 정규분포

 dnorm(x, mean=0, sd=1): 확률밀도함수

```
x <- seq(-3,3, length=30)
y <- dnorm(x)
plot(x, y, type='l', main="N(0,1)", ylab="f(x)")
```



 pnorm(q, mean=0, sd=1): 누적분포함수

$$P(Z \leq 1.96) = ?, \text{ where } Z \sim N(0, 1^2)$$

```
pnorm(1.96)
```

```
# [1] 0.9750021
```

 qnorm(p, mean=0, sd=1): p-백분위수

```
qnorm(0.9) # 90%-백분위수
```

```
# [1] 1.281552
```

 rnorm(n, mean=0, sd=1): 난수(랜덤넘버)생성

```
rnorm(10)
```

```
#  [1]  0.15153062  1.73578979 -0.25685337  0.30040463 -0.58096749
#  [6]  0.70640652 -0.03707681 -0.62760903  1.29114088 -0.66933968
```

2. 이항분포

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

2.4 기타 내장함수들

 print: 객체의 값을 화면에 출력

```
  a <- c(5,3,6,2,4)
  print(a)
```

```
# [1] 5 3 6 2 4
```

 cat: 문자열 및 연산결과를 동시에 출력 화면에 프린트

```
  cat("mean of a is ",mean(a), "variance of a is ", var(a),"\n")
```

```
# mean of a is  4 variance of a is  2.5
```

 unique: 서로 다른 원소값들

```r
x <- c(1,5,1,3,5,7,5)
unique(x)
```

```
# [1] 1 5 3 7
```

*. substr(x, start, stop): 문자열에서 일부 추출

```r
x <- c("노무현","이명박", "박근혜", "문재인")
substr(x, 1, 1)
```

```
# [1] "노" "이" "박" "문"
```

  paste(..., sep=""): 문자열의 조립

```r
paste("x",1:3,sep="")
```

```
# [1] "x1" "x2" "x3"
```

```r
paste("x",1:3,sep="M")
```

```
# [1] "xM1" "xM2" "xM3"
```

```r
paste("Today is", date())
```

```
# [1] "Today is Thu Apr 12 10:33:34 2018"
```

3 R 제어문

1. 순환문
2. 제어문

3.1 순환문

 순환문의 표현으로는 다음 세가지 표현을 사용

```
while ( cond ) expr
repeat expr
for ( var in list ) expr
break :  while, repeat, for에서 순환문을 끝내는 구문
next : 이후의 문장을 건너뛰고 다음 순환
```

1. for문

```
for(i in 1:3){ # i가 1,2,3값을 취하면서 아래의 코드를 반복함
    print(i)
}
```

```
# [1] 1
# [1] 2
# [1] 3
```

```r
x <- c("a", "b", "c")
for(i in x) print(i)
```

```
# [1] "a"
# [1] "b"
# [1] "c"
```

2. while문

```r
count <- 0
while(count < 3) {
  cat(count, "\n")
  count <- count + 1
}
```

```
# 0
# 1
# 2
```

3. repeat문

```r
repeat {
  x <- runif(1) # (0,1)사이의 균등분포에서의 난수
  if(x < 0.1) break else print(x)
}
```

3.2 조건문

 조건문에 해당되는 표현

```
if ( cond ) expr

if ( cond ) expr1 else expr2

if ( cond1 ) expr1
else if( cond2 ) expr2
else expr3

ifelse(cond, TRUE인 경우 반환값, FALSE인 경우 반환값)
```

1. 조건문 예제

```
 x <- runif(10)
 y <- rep(NA, length(x))
 for(i in seq_along(x)){
   if(i > 0.5) {
     y[i] <- T
   } else {
```

```
    y[i] <- F
  }
}
y
```

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

4 사용자 함수의 작성 및 호출

4.1 함수의 형식

1. 아래의 수학 함수를 고려

$$y = f(x_1, x_2)$$

- f: 함수이름
- x_1, x_2는 입력 값
- y는 결과값

2. 프로그램에서는 수학함수와 동일한 개념, 용어는 다소 상이

- f: 함수이름
- x_1, x_2를 인수(input arguments)
- y는 결과값 (반환값, return value)

3. R 에서 함수 정의 방법

```
함수이름  <- function(인수1, 인수2, ...){
  # 여기에 함수 계산 코드 삽입
  return(반환값)
}
```

4. 함수가 작성되면 호출하는 방법

- function_name(arg_1=1)
- function_name(arg_2=1, arg_1=3)
- function_name(3, 1)

4.2 함수 작성 예제

1. 아래의 수학 함수에서 $f(3,4)$, $f(1,2)$를 계산해 보자

$$f(x_1, x_2) = x_1^2 + x_2^2$$

1. R 함수 정의

```
f <- function(x1, x2) {
  y <-  x1^2 + x2^2
  return(y)
}
```

2. 함수 호출

```
f(x1=1, x2=2)
```

```
# [1] 5
```

```
f(3, 4)
```

```
# [1] 25
```

3. 아래의 결과는 ?

```r
f(x1=c(1,2), x2=c(3,4))
```

```
# [1] 10 20
```

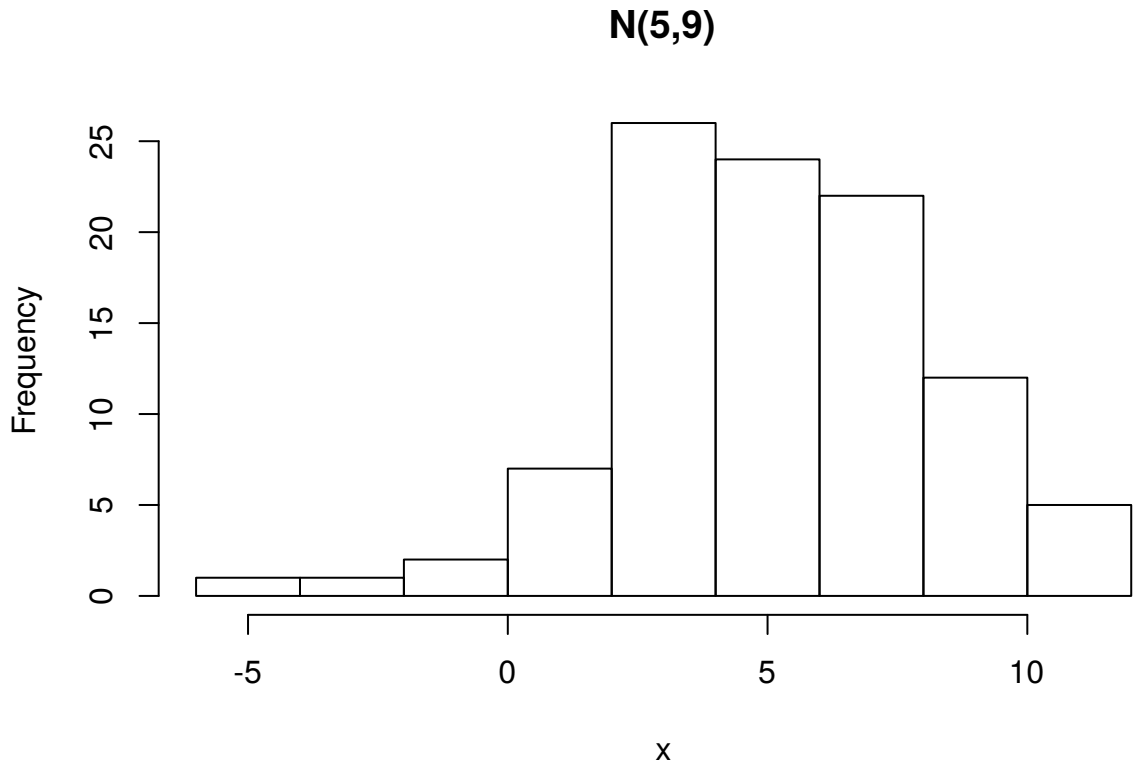
2. 모평균이 m이고 모표준편차가 s인 정규분포에서 난수 n개를 생성하여 히스토그램과 상자그림을 그리는 함수

    1. R 함수 정의

```r
normal_hist <- function(m, s, n) {
  x <-  rnorm(n, mean=m, sd=s)
  hist(x, main=paste0("N(",m, ",", s^2, ")"))
}
```

    2. 함수 호출

```r
normal_hist(m=5, s=3, n=100)
```


N(5,9)
Frequency
0
5
10
15
20
25
-5
0
5
10
x

4.3 함수 작성시 필요한 유틸리티

- 함수의 디버깅

```
#exftn(1:3, 4:7)
## Error in exftn(1:3, 4:7) : x and y must have same length.

#traceback()
##2: stop("x and y must have same length.") at #2
##1: exftn(1:3, 4:7)
```

- 함수의 수행시간 (runing time)
  - user time: CPU time of user program
  - system time: CPU time of system
  - Elapse time: wall clock time

```
# system.time({module....})
## user system elapsed
## 0.004 0.002 0.431
```

4.4 외부언어의 호출

- C 언어 예제

```
#include <R.h>
void csum(double *val, int *n, double *csum){
  int i;  csum=0;
  for(i=0; i < *n; i++) csum += val[i];
  Rprintf("sum=%f\n", csum);
}
```

- 소스코드의 컴파일 및 빌드

```
R CMD SHLIB csum.c
```

  - 위의 결과 : csum.so가 만들어짐

- .C(), .Call()

```
dyn.load("csum.so")
Rsum <- function(val) {
  .C("csum", as.double(val), as.integer(length(val)), double(1))
```

```
}
Rsum(1:10)
### sum=55.000000
### [[1]]
###  [1]  1  2  3  4  5  6  7  8  9 10
### [[2]]
### [1] 10
### [[3]]
### [1] 55
```