데이터 사이언티스트를 위한 R-3: 연산자 및 함수

Jinseog Kim
Dongguk University
jinseog.kim@gmail.com

2017-03-28

R 연산자

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

R 연산자 예제

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

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

```
x%/% 3
```

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

```
x > 3
```

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

```
y <- c(3, 5, 2, 1)
x>y
```

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

```
z <- TRUE
!z
```

```
# [1] FALSE
```

R 연산자 예제

```
x1 <- c(1, 1, 0, 0)
y1 <- c(1, 0, 1, 0)
x1 | y1
```

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

```
x1 || y1
```

```
# [1] TRUE
```

```
x1 & y1
```

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

```
x1 && y1
```

```
# [1] TRUE
```

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

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
```

```
  floor(x1)
```

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

```
  trunc(x1)
```

내장함수 예제

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

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

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

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

기타 내장함수들

- print(): Prints a single R object

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

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

- cat(): Prints multiple objects, one after the other

```
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(): Gives the vector of distinct values

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

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

순환문

순환문의 표현으로는 다음 세가지 표현을 사용한다.

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

for

```
for(i in 1:3){
    print(i)
}
```

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

for

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

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

while

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

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

repeat

```
repeat {
  x <- runif(1)
  if(x < 0.1) break else print(x)
}
```

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) “‘

조건문

```
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
```

사용자 함수의 작성 및 호출

- 함수의 작성

```
function_name <- function(arg_1, arg_2, ...){
    # expression...;
    return(x)
}
```

- '...' : extra arguments
- function call
  - function_name(arg_1=1)
  - function_name(arg_2=1, arg_1=3)
  - function_name(3, 1)

함수 예제

```
exftn <- function(x, y) {
  if(length(x) != length(y)) stop("x and y must have same length.")
  o1 <- x^2
  o2 <- x*y
  list(square_x = o1, product=o2, n=length(x))
}
exftn(1:3, 4:6)
```

```
# $square_x
# [1] 1 4 9
#
# $product
# [1]  4 10 18
#
# $n
# [1] 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)
```

- 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
```

Foreign Function Interface

```
#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);
}
```

- R CMD SHLIB csum.c : build .so from source code
- .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
```