벡터

Jeon Jong-June

Monday, March 09, 2015

벡터의 생성

```
x <- 3
x
```

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

x 라는 변수는 미리 선언하지 않아도 R 프로그램에서는 무방하다. C의 경우는 다음과 같이 벡터를 선언한다.

```
# C program
int x;
x = 1;
```

int x는 x가 정수값을 저장하는 변수임을 미리 선언하는 구문이다. 위 구문 없이 바로 x=1을 실행하면 error를 발생시킨다. R에서 벡터를 생성하는 방법으로 함수 c 를 사용할 수 있다.

```
x <- c(3)
x
```

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

```
x <- c(88, 15, 12, 13)
x
```

```
## [1] 88 15 12 13
```

```
x1 <- 1:3
x2 <- 10:5
x3 <- c(x2,x1)
x1
```

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

```
x2
```

```
## [1] 10  9  8  7  6  5
```

```
x3
```

```
## [1] 10  9  8  7  6  5  1  2  3
```

콜론 연산자를 이용하여 연속한 숫자벡터를 생성할 수 있다.

```
x <- 5:8
x1 <- x[1:3]
x2 <- c(x1[1:3], 168, x[4])
x2
```

```
## [1]   5   6   7 168   8
```

벡터의 길이

```
length(x2)
```

```
## [1] 5
```

length는 벡터의 길이를 계산하는 함수다.

벡터의 연산

사칙연산

벡터는 벡터간 사칙연산이 가능한다. 벡터간의 사칙연산은 벡터 원소간에 적용되며 재사용 규칙을 따른다. 다음 예제를 보자

```
x1 <- c(5,0, -4)
x2 <- c(1, 2, 2)
```

```
x1+x2
```

```
## [1]  6  2 -2
```

```
x1*x2
```

```
## [1]  5  0 -8
```

```
x1/x2
```

```
## [1]  5  0 -2
```

위의 실행결과를 보면 벡터간의 사칙연산이 벡터 원소간에 이루어졌음을 알 수 있다.

```
x1 + 3
```

```
## [1]  8  3 -1
```

```
x1*2
```

```
## [1] 10  0 -8
```

위 예에서는 사칙연산에서 스칼라가 벡터 원소마다 재사용되어 계산되었음을 알 수 있다. 다음은 벡터의 원소가 어떻게 재사용되는지 알 수 있는 예제다.

```
x1 <- c(5,0,-4,2)
x2 <- c(2,1)
x1/x2
```

```
## [1]  2.5  0.0 -2.0  2.0
```

벡터의 인덱싱

저장된 벡터에서 특정한 원소 값을 추출할 수 있다.

```
y <- c(1.2, 3.9, 0.4, 0.12)
y[c(1, 3)]
```

```
## [1] 1.2 0.4
```

```
y[2:3]
```

```
## [1] 3.9 0.4
```

```
v <- 3:4
y[v]
```

```
## [1] 0.40 0.12
```

```
y[c(1, 1, 2)]
```

```
## [1] 1.2 1.2 3.9
```

저장된 벡터에서 특정한 원소값을 제외하고 추출할 수 있다.

```
y[-c(1,2)]
```

```
## [1] 0.40 0.12
```

```
y[-length(y)]
```

```
## [1] 1.2 3.9 0.4
```

벡터연산의 순서

```
y <- 1:10-1
y
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```r
y <- 1:(10-1)
y
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

seq 함수

```r
y <- seq(from = 12, to = 30, by = 2)
y
```

```
##  [1] 12 14 16 18 20 22 24 26 28 30
```

```r
y <- seq(12, 30, length = 19)
y
```

```
##  [1] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

rep 함수

```r
x <- rep(8, 4)
x
```

```
## [1] 8 8 8 8
```

```r
x <- rep( c(5, 12, 13), 3)
x
```

```
## [1]  5 12 13  5 12 13  5 12 13
```

```r
x <- rep( c(5, 12, 13), each = 3)
x
```

```
## [1]  5  5  5 12 12 12 13 13 13
```

유용한 연산

지수, 몫과 나머지

```r
x <- 11
x^2
```

```
## [1] 121
```

```
x%/%5
```

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

```
x%%5
```

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

벡터화 연산(Vectorized operation)

```
u <- c(5, 2, 8)
v <- c(1, 3, 9)
u>v
```

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

재사용 규칙에도 벡터화 연산이 이루어진다.

```
u + 5
```

```
## [1] 10  7 13
```

NA(not available) 과 NULL

```
x <- c(88, NA, 12, 168, 13)
mean(x)
```

```
## [1] NA
```

```
mean(x, na.rm = TRUE)
```

```
## [1] 70.25
```

mean은 x벡터의 평균을 계산하는 함수, na.rm 은 NA 데이터를 remove 하고 계산할 것인지 정하는 옵션.

```
x <- NULL
x
```

```
## NULL
```

```
x <- c(x, 2)
x
```

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

NULL은 변수를 초기화 시킬때 유용한 값의 형태다.

필터링

```
z <- c(5, 2, -3, 8)
w <- z[z*z > 8]
w
```

```
## [1]  5 -3  8
```

위 구문에서 작동방식을 살펴보자.

```
z*z>8
```

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

```
z[ c(TRUE, FALSE, TRUE, TRUE)]
```

```
## [1]  5 -3  8
```

위 결과에 따른면 R에서는 벡터길이와 같은 논리연산자 벡터를 이용해서 필터링 할 수 있다.

```
z
```

```
## [1]  5  2 -3  8
```

```
z[z>3] <- 0
z
```

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

위와 같이 특정조건을 만족시키는 벡터의 위치에 어떤 값을 치환할 수도 있다.

subset 함수

subset 함수는 NA를 포함한 벡터에서 특정 원소를 필터링하는데 유용한 함수다.

```
x <- c(6, 1:3, NA, 12)
x
```

```
## [1]  6  1  2  3 NA 12
```

```
x[x>5]
```

```
## [1]  6 NA 12
```

```
subset(x, x>5)
```

```
## [1]  6 12
```

is.na

NA 값을 판별하는 함수

```r
is.na(x)
```

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

subset을 이용하지 않으면 is.na 함수를 사용해 NA 값을 지우고 작업을 진행할 수도 있을 것이다.

which 함수

벡터내의 어떤 값이 조건에 맞는지 확인하고 그 위치를 알고 싶을 때 사용하는 함수

```r
x
```

```
## [1]  6  1  2  3 NA 12
```

```r
which(x>5)
```

```
## [1] 1 6
```

벡터의 원소값이 5보다 큰 위치를 반환해준다. which 안에 필요한 조건이 들어갈 수 있다.

```r
x <- c(3,1,4,1)
which(x>5)
```

```
## integer(0)
```

```r
y <- which(x>5)
length(y)
```

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

위 처럼 조건에 맞는 벡터내의 원소가 없는 경우 integer(0)를 반환한다. 이를 저장한 경우 그 벡터의 길이는 0 이다. 이를 이용해서 조건을 만족시키는 원소가 하나라도 있는지 확인할 수 있다. 이는 다음 any 함수를 이용해서 확인 할 수도 있다.

```r
x <- c(3,1,4,1)
any(x>5)
```

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

%in% 함수

```r
x <- c(3,1,4,1)
x%in% c(2,1,4)
```

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

x 의 각 원소가 어떤 벡터에 포함이 되어 있나?

match 함수

```
x <- c(3,1,4,1)
match(x ,c(2,1,4))
```

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

x 의 각 원소가 벡터에 포함된 위치가 어디인가?