

데이터과학을 위한 **R**프로그래밍

2주차. 벡터, 행렬의 연산 및 함수



이혜선 교수

포항공과대학교 산업경영공학과



목차

2주차. 벡터, 행렬의 연산 및 함수

1차시

벡터 및 행렬 생성

2차시

벡터와 행렬의 연산

3차시

간단한 함수 생성 및 루프



2주차

1차시

벡터 및 행렬 생성

● 벡터 생성

✓ 벡터의 생성 (lec2_1.r)

프로그램 편집 창

```
# lec2_1.r

# vector
x<-c(1,3,5,7,9)
x[3]

# subset of vector :
x[-1]

# subset of vector :
x1<-x[-c(1,2)]
x1

# subset of vector :
x2<-x[-c(1:3)]
x2
```

컨솔창

```
> # vector
> x<-c(1,3,5,7,9)
> x[3]
[1] 5
>
> # subset of vector : delete the first element
> x[-1]
[1] 3 5 7 9
>
> # subset of vector : delete the first two element
> x1<-x[-c(1,2)]
> x1
[1] 5 7 9
> # subset of vector : delete the 1st to the 3rd element
> x2<-x[-c(1:3)]
> x2
[1] 7 9
```

x[-1]은 첫번째 값을 삭제하라는 의미



x[-(1,2)]은 첫번째, 두번째값 삭제



x[-(1:3)]은 첫번째부터 세번째값까지 삭제



● 벡터 생성

☑ 벡터 (seq 함수 사용)

sequence

```
# create vector using 'seq'  
# sequence of 20 values  
y1<-seq(0,10, length=20)  
# sequence of (1 to 10) by 0.5  
y2<-seq(0,10, by=0.5)
```

y1: 0부터 10까지, 20개의 값을 생성

y2: 0부터 10까지 0.5씩 간격을 두고 값을 생성

```
> y1<-seq(0,10, length=20)  
> y1  
[1] 0.0000000 0.5263158 1.0526316 1.5789474 2.1052632  
[6] 2.6315789 3.1578947 3.6842105 4.2105263 4.7368421  
[11] 5.2631579 5.7894737 6.3157895 6.8421053 7.3684211  
[16] 7.8947368 8.4210526 8.9473684 9.4736842 10.0000000  
> y2<-seq(0,10, by=0.5)  
> y2  
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0  
[12] 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
```

● 벡터 생성

✓ 벡터 (rep 함수 사용)

replication

```
# using rep  
z1<-rep(1:4, 2)  
z1
```

1부터 4까지 두번을 반복하여 생성하라는 의미

```
> z1<-rep(1:4, 2)  
> z1  
[1] 1 2 3 4 1 2 3 4
```

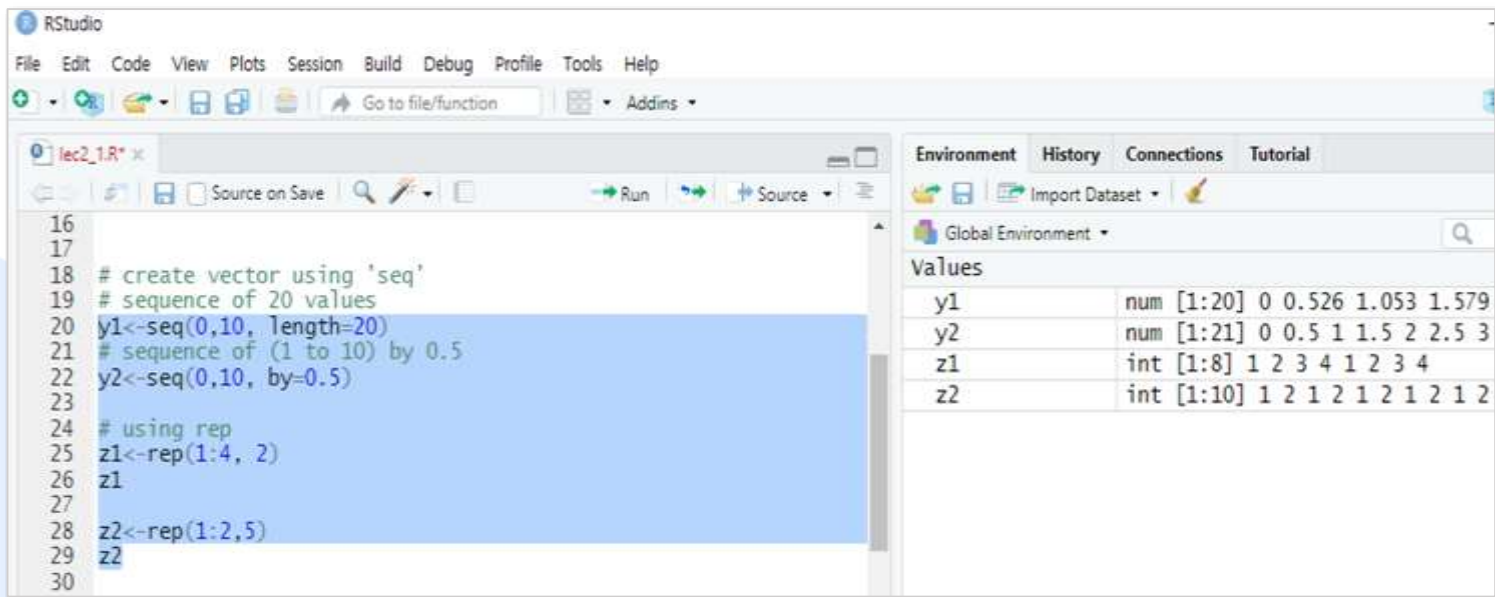
```
z2<-rep(1:2,5)  
z2
```

1부터 2까지 다섯번을 반복하여 생성하라는 의미

```
> z2<-rep(1:2,5)  
> z2  
[1] 1 2 1 2 1 2 1 2 1 2
```

● 벡터 생성

☑ 생성된 데이터 확인



The screenshot shows the RStudio interface. The script editor on the left contains the following R code:

```
16  
17  
18 # create vector using 'seq'  
19 # sequence of 20 values  
20 y1<-seq(0,10, length=20)  
21 # sequence of (1 to 10) by 0.5  
22 y2<-seq(0,10, by=0.5)  
23  
24 # using rep  
25 z1<-rep(1:4, 2)  
26 z1  
27  
28 z2<-rep(1:2,5)  
29 z2  
30
```

The Environment pane on the right shows the following variables and their values:

Values	
y1	num [1:20] 0 0.526 1.053 1.579
y2	num [1:21] 0 0.5 1 1.5 2 2.5 3
z1	int [1:8] 1 2 3 4 1 2 3 4
z2	int [1:10] 1 2 1 2 1 2 1 2 1 2

● 벡터 생성

☑ 벡터 결합 (행과 열을 기준)

➤ cbind : column bind (열 기준으로 결합)

```
# combine vectors in a row or column  
c1<-c(2,4,6,8,10)  
c2<-cbind(x, c1)  
c2
```

(5*2)인 행렬이 됨

```
> x<-c(1,3,5,7,9)  
> c1<-c(2,4,6,8,10)  
> c2<-cbind(x, c1)  
> c2  
      x c1  
[1,] 1  2  
[2,] 3  4  
[3,] 5  6  
[4,] 7  8  
[5,] 9 10
```

➤ rbind : row bind (행으로 결합)

```
c3<-rbind(x,c1)  
c3
```

(2*5)인 행렬이 됨

```
> c3<-rbind(x,c1)  
> c3  
      [,1] [,2] [,3] [,4] [,5]  
x       1   3   5   7   9  
c1      2   4   6   8  10
```



● 벡터 이름 정의

☑ 벡터이름주기 (names)

➤ (0,1)값을 갖는 벡터 gender에 0=female, 1=male이라는 값을 부여

```
# Give name to a vector  
gender<-c(0,1)  
names(gender)<-c("female", "male")  
gender
```

```
length(gender)
```



```
> gender<-c(0,1)  
> names(gender)<-c("female", "male")  
> gender  
female    male  
      0      1
```

범주형 변수 생성

범주형 변수 생성 (factor 사용)

size라는 변수를 생성 : (S, M, L, XL)의 값을 갖는 범주형 변수(factor)를 생성

```
# categorical variables : factor  
size<-c("S", "M", "L", "XL")  
# define size as a factor (categorical)  
size_factor<-factor(size)  
  
size_factor
```

factor()는 범주형 변수로 정의하는 함수

```
> size<-c("S", "M", "L", "XL")  
> # define size as a factor (categorical)  
> size_factor<-factor(size)  
>  
> size_factor  
[1] S M L XL  
Levels: L M S XL
```

순서가 없음!

질문: size_factor가 범주형 변수인가?

답: size_factor는 범주형 변수이다(위에서 factor로 정의했음)

```
is.factor(size_factor)
```

```
> is.factor(size_factor)  
[1] TRUE
```

범주형 변수 생성

✓ 범주형 변수 생성 (factor 사용) - 순서를 정의한 factor 생성

➤ size_factor3 : (S, M, L, XL)의 값을 가지며, $S < M < L < XL$ 의 순서가 정의된 factor !!

```
# give order for categorical variable  
size_factor3 <- factor(size, ordered = TRUE,  
                       levels = c("S", "M", "L", "XL"))  
size_factor3
```

→

```
> size_factor3  
[1] S M L XL  
Levels: S < M < L < XL
```

행렬의 생성

✓ 행렬의 생성 (matrix 함수 이용) - 행의 수, 열의 수 입력

▶ matrix 함수를 이용하여 1부터 10까지의 숫자로 2행의 행렬을 생성

```
# create matrix  
# two row matrix with 1 to 10  
m1<-matrix(1:10, nrow=2)  
m1
```



```
> m1<-matrix(1:10, nrow=2)  
> m1  
      [,1] [,2] [,3] [,4] [,5]  
[1,]    1    3    5    7    9  
[2,]    2    4    6    8   10
```

▶ matrix 함수를 이용하여 1부터 6까지의 숫자로 3개 열의 행렬을 생성, 1열부터 채우는 것이 default

```
# three columns matrix with 1:6  
m2<-matrix(1:6, ncol=3)  
m2
```




```
> m2<-matrix(1:6, ncol=3)  
> m2  
      [,1] [,2] [,3]  
[1,]    1    3    5  
[2,]    2    4    6
```

행렬의 생성

✓ 행렬의 생성 (matrix 함수 이용) - 행의 수, 열의 수 입력

- matrix 함수를 이용하여 1부터 6까지의 숫자로 2개 행의 행렬을 생성,
1열부터 채우는 것이 default, 여기서는 byrow=T이므로 1행부터 채워서 생성

```
# matrix filled by rows, default: filled by columns  
m3<-matrix(1:6, nrow=2, byrow=T)  
m3  
# help (matrix)|
```



```
> m3<-matrix(1:6, nrow=2, byrow=T)  
> m3  
      [,1] [,2] [,3]  
[1,]    1    2    3  
[2,]    4    5    6
```

행렬의 생성

☑ 고차원 행렬 (array를 이용하여 생성)

```
# higher order of array  
a1<-array(c(1:18), dim=c(3,3,2))  
a1
```



```
> a1<-array(c(1:18), dim=c(3,3,2))  
> a1  
, , 1  
  
    [,1] [,2] [,3]  
[1,]    1    4    7  
[2,]    2    5    8  
[3,]    3    6    9  
  
, , 2  
  
    [,1] [,2] [,3]  
[1,]   10   13   16  
[2,]   11   14   17  
[3,]   12   15   18
```

```
a1[, ,1]  
a1[, ,2]
```



```
> a1[, ,1]  
    [,1] [,2] [,3]  
[1,]    1    4    7  
[2,]    2    5    8  
[3,]    3    6    9
```