



3주차: R의 데이터형과 연산

ChulSoo Park

School of Computer Engineering & Information Technology

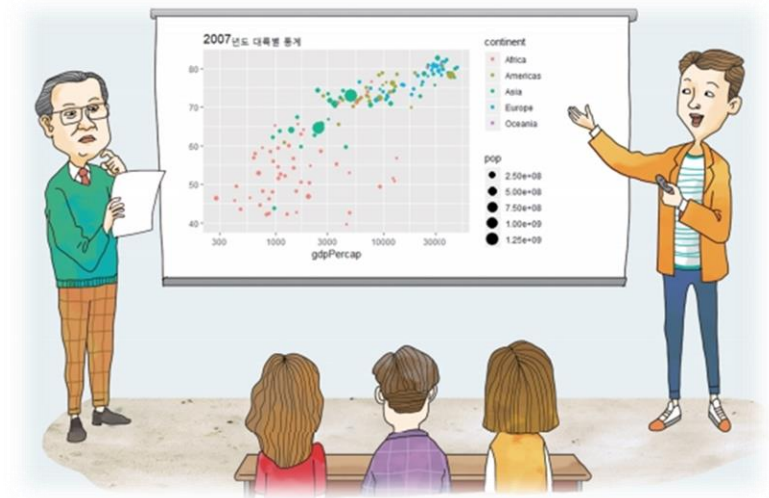
Korea National University of Transportation



03

CHAPTER

R의 데이터형과 연산



CONTENTS

- 3.1 데이터 저장과 처리
- 3.2 변수
- 3.3 데이터형
- 3.4 연산자
- 3.5 벡터
- 3.6 배열(행렬)
- 3.7 데이터 프레임
- 3.8 리스트
 - 요약



3.3 데이터형

■ R의 기본 데이터형

R에서는 데이터형을 지정하지 않고, 변수에 어떤 값을 저장하느냐에 따라 데이터 형이 결정 된다.

데이터형	종류
숫자형	정수(integer), 실수(numeric), 복소수(complex)
문자형	character : 작은따옴표나 큰따옴표로 묶어 표기
범주형	factor : 레벨(level)에 따라 분류된 형태
논리형	TRUE(T), FALSE(F)
특수 상수	NULL : 정의 되지 않은 값 NA(Not Available) : 결측 값 -Inf(음의 무한대), Inf(양의 무한대) NaN(Not a Number : 0/0, Inf/Inf 등과 같이 연산불가능함 값 표시)



3.3 데이터형

■ R의 기본 데이터형

Console C:/RSources/ ↗

```
> x=c(1, NA, 0, 2)
> x
[1] 1 NA 0 2
> x[1]=x[1]+10
> x[1]
[1] 11
> x[2]=x[2]+10
> x[2]
X[2] ???
>
>
> x=c()
> x
NULL
> x=x+1
> x
X ???
> |
```



3.4 연산자

■ 연산자의 종류

■ 산술연산자

연산자	설명	예
+	덧셈	
-	뺄셈	
*	곱셈	
/	나눗셈	
^ 또는 **	지수승	

$x \% y$	x를y로 나눈 나머지 (정수 나눗셈 나머지)	$5 \% 2 \rightarrow 1$
----------	-----------------------------	------------------------

$x \%/\% y$	X를 y로 나눈 몫 (정수 나눗셈 몫)	$5 \%/\% 2 \rightarrow 2$
-------------	--------------------------	---------------------------

RGui (64-bit)

파일 편집 보기 기타 패키지들 윈도우즈 도움말

R Console

```

> x=2+5
> x
[1] 7
> y=5-2
> y
[1] 3
> x=2*5
> x
[1] 10
> y=5/2
> y
[1] 2.5
> x=5^2
> x
[1] 25
> y=5%%2
> y
[1] 1
> x=5%%/2
> x
[1] 2
> y=2.5
> y <- as.integer(y)
> y
[1] 2
  
```



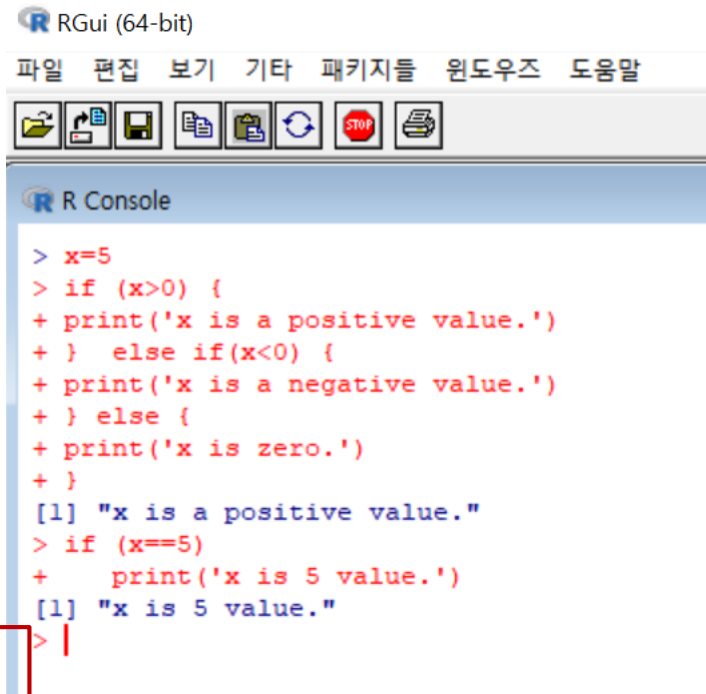
3.4 연산자

■ 연산자의 종류

- 비교연산자, 논리연산자 : 이상(>) 이하(<) 등호 표시가 "=" 앞에 나와 함

표 3-5 비교 연산자와 논리 연산자

연산자	설명	예
<	좌변이 작은(미만)	5 < 5 → FALSE
<=	좌변이 작거나 같은(이하)	5 <= 5 → TRUE
>	좌변이 큰(초과)	5 > 5 → FALSE
>=	좌변이 크거나 같은(이상)	5 >= 5 → TRUE
==	좌변과 우변이 같은	5 == 5 → TRUE
!=	좌변과 우변이 다른	5 != 5 → FALSE
!	부정(not)	!TRUE → FALSE
x y, x y	x or y(또는, 합집합)	TRUE FALSE → TRUE
x & y, x && y	x and y(그리고, 교집합)	TRUE & FALSE → FALSE
isTRUE(x)	x 의 TRUE 여부 판단	isTRUE(TRUE) → TRUE



```

RGui (64-bit)
파일 편집 보기 기타 패키지를 윈도우즈 도움말

> x=5
> if (x>0) {
+ print('x is a positive value.')
+ } else if(x<0) {
+ print('x is a negative value.')
+ } else {
+ print('x is zero.')
+ }
[1] "x is a positive value."
> if (x==5)
+ print('x is 5 value.')
[1] "x is 5 value."
> |
  
```



3.4 연산자

■ 연산자 우선순위

표 3-6 연산자 우선순위

연산자	설명	우선순위
\wedge , **	지수승	높음 ↑
+, -	단항 플러스와 마이너스	
%any%	%%, %/% 등 연산자	
*, /	곱셈, 나눗셈	
+, -	덧셈, 뺄셈	낮음 ↓
==, !=, <, >, <=, >=	비교 연산자	
!	논리 부정(not)	
&, &&	논리 and	
,	논리 or	

RGui (64-bit)

파일 편집 보기 기타 패키지들 윈도우즈 도움말

R Console

```

> x=5+2^3/2
> x
[1] 9
> y=-5^2*3
> y
[1] 2
> y=-5^2
> y
[1] -25
> y=5^2*3
> y
[1] 1
> y=2+5*3^2
> y
[1] 7
> |
  
```



■ 연산자 우선순위

Console C:/RSources/ 

```
> aa=-5**2  
> b=-5  
> bb=b**2  
> cc=(-5)**2  
>  
>
```

aa = ??

bb = ??

cc = ??



3.5 벡터

- 여러 단일 값을 하나의 변수 명으로 저장 가능
 - 지금까지 단일 값을 단일 변수에 저장하는 방법을 배웠다.
 - 단일 값을 하나의 변수로 저장하면 값이 많은 경우 변수의 수도 증가하게 됨.
 - 하나의 벡터 변수로 여러 단일(형태의)값을 저장할 수 있음.

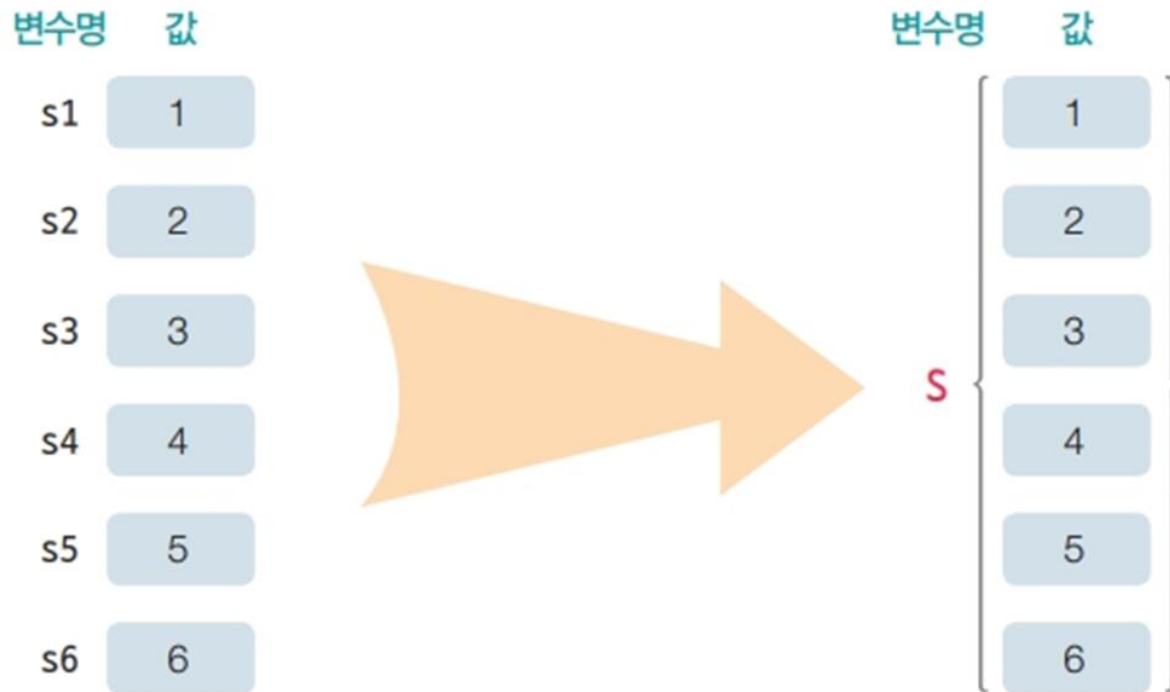


그림 3-2 단일값들의 조합으로 구성된 벡터



- 여러 단일 값을 하나의 변수 명으로 저장 가능
 - 하나의 벡터 변수로 여러 단일값을 저장할 수 있음.

Console C:/RSources/ ↗

```
> xy=c()  
> xy  
NULL  
> xy[1]=1  
> xy[1]=1  
> xy[1]  
[1] 1  
> class(xy[1])  
[1] "numeric"  
> xy[2]=2  
> xy[2]  
[1] 2  
> class(xy[2])  
[1] "numeric"  
> xy[3]='AA'
```

class(xy[1]) ??



① 벡터 생성

- 벡터 생성 연산자 ':' 이용

```
> x=(1:10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> y=(10:1)
> y
[1] 10 9 8 7 6 5 4 3 2 1
> y[3]
[1] 8
```

- vector 함수 이용
 - 요소의 개수가 n 개인 빈 벡터 생성

```
> z=vector(length=5)
> z
[1] FALSE FALSE FALSE FALSE FALSE
> z[3]=50
> z
[1] 0 0 50 0 0
> z[3]
[1] 50
> y=vector(length=7)
> y
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> y[6]=('AA')
> y
[1] "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "AA" "FALSE"
```

요소의 개수



■ 벡터 생성 대표적인 함수

1. c함수 : 일반 벡터 생성

(예) : $x = c(1,2,3,4,5)$

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

2. seq 함수 : 순열 벡터 함수

(예) : $x = seq(from=1, to=10, 3)$, $x=seq(1,9,3(by=3))$

```
[1] 1 4 7
```

3. rep 함수 : 반복 벡터 함수

(예) $x1=rep(c(1:3),times=2)$, $x2=rep(c(1:3),each=3)$

```
[1] 1 2 3 1 2 3, [1] 1 1 1 2 2 2 3 3 3
```



① 벡터 생성

- C함수 : 일반 벡터 생성

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

3chapter.R* x R data sets x

```

6 x=c(1,2,3)
7 x
8 y=c() # 빈 벡터로 생성
9 y
10 y=c(y,c(1:3)) # 기존 빈 벡터 y에 값입력 방법
11 y
12 class(y)
13

```

13:1 (Top Level) R Script

Environment History Connections Tutorial

Import Dataset

R Global Environment

values	
x	num [1:3] 1 2 3
y	int [1:3] 1 2 3

Files Plots Packages Help Viewer

Zoom Export

Console C:/RSources/

```

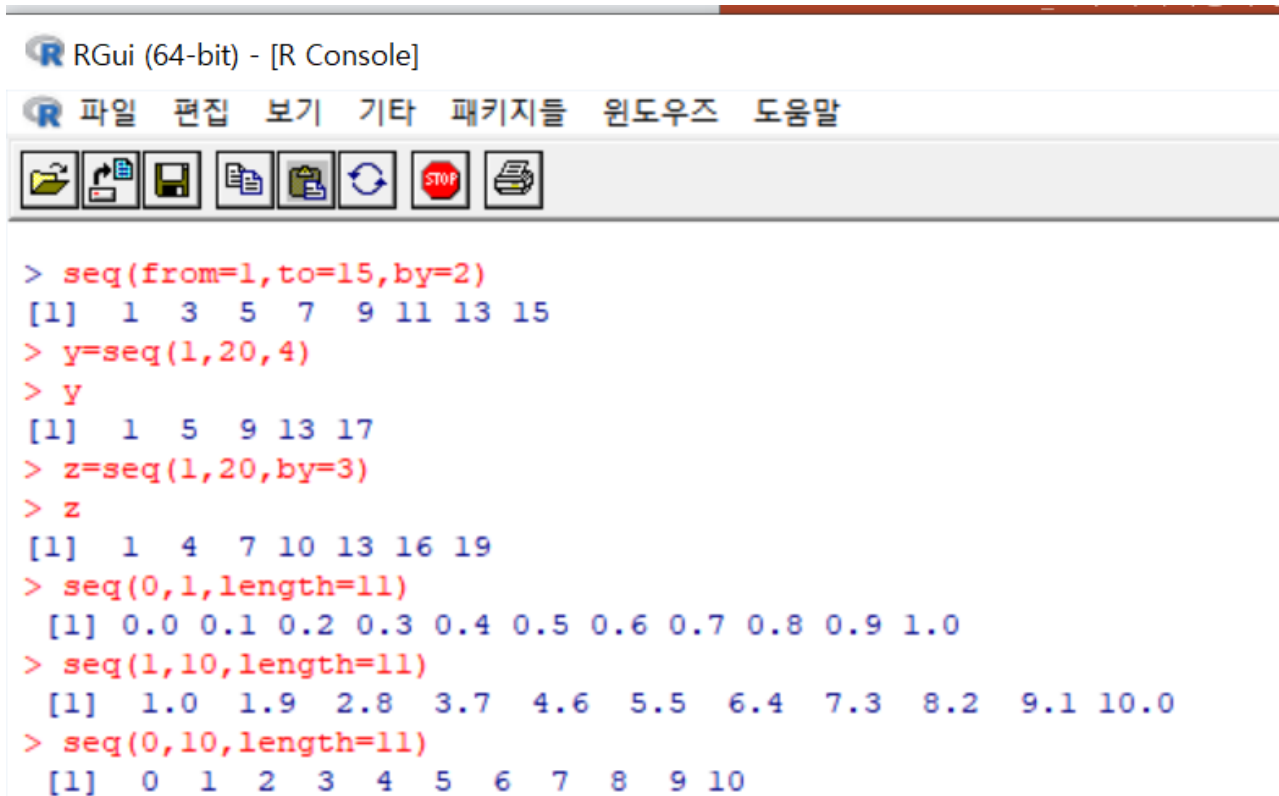
> c(1:5)
[1] 1 2 3 4 5
> c(1,2,3,c(4:6))
[1] 1 2 3 4 5 6
> x=c(1,2,3)
> x
[1] 1 2 3
> y=c() # 빈 벡터로 생성
> y
NULL
> y=c(y,c(1:3)) # 기존 빈 벡터 y에 값입력 방법
> y
[1] 1 2 3
> class(y)
[1] "integer"
>

```



① 벡터 생성

- seq 함수 : 순열 벡터 생성



```
> seq(from=1,to=15,by=2)
[1] 1 3 5 7 9 11 13 15
> y=seq(1,20,4)
> y
[1] 1 5 9 13 17
> z=seq(1,20,by=3)
> z
[1] 1 4 7 10 13 16 19
> seq(0,1,length=11)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(1,10,length=11)
[1] 1.0 1.9 2.8 3.7 4.6 5.5 6.4 7.3 8.2 9.1 10.0
> seq(0,10,length=11)
[1] 0 1 2 3 4 5 6 7 8 9 10
```



① 벡터 생성

- Rep 함수 : 반복 벡터 생성

```

> rep(c(1:5), times=3)
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
> rep(c(1:5), 3)
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
> rep(c(1:5), each=3)
[1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5

```

벡터의 반복 횟수

요소의 반복 횟수



② 벡터 연산

- 벡터 요소 값을 선택하여 출력하기

RGui (64-bit) - [R Console]

파일 편집 보기 기타 패키지를 윈도우즈 도움말



```
> x=seq(2,10,2)
> x
[1] 2 4 6 8 10
> length(x)
[1] 5
> x[3]
[1] 6
> x[1:3]
[1] 2 4 6
> x[1],x[3]
예러: 여기치 않은 ', '입니다 in "x[1],"
> x[c(2:4)]
[1] 4 6 8
> x[-c(1,4)]      # [1], [4]를 제외하고 출력
[1] 4 6 10
> x[1,2,3]
Error in x[1, 2, 3] : incorrect number of dimensions
```



② 벡터 연산

- 벡터 간 연산 : 벡터의 길이가 같거나 요소 개수가 배수 관계에 있을 때 연산이 가능

Console C:/RSources/ ➔

```
> x=c(1:4)
> y=c(5:8)
> z=c(3,4)
> w=c(5:7)
> x+2
[1] 3 4 5 6
> x+y
[1] 6 8 10 12
> x+z
[1] 4 6 6 8
> x+w
[1] 6 8 10 9
경고메시지(들):
In x + w : 두 객체의 길이가 서로 배수관계에 있지 않습니다
> |
```



③ 벡터 연산에 유용한 함수

- all · any 함수: 벡터 내 모든 · 일부 요소의 조건 검토
- all 함수 : 모든 요소가 조건을 만족하는지 비교하여 T/F 로 표시
- any 함수 : 요소 중 일부라도 조건을 만족하는지 check하여 T/F 표시
- head, tail함수는 요소 중 앞,뒤 요소 추출 함수

```
> x=1:10
> x>5
[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
> all(x>5) # x 벡터의 요소값이 5보다 큰지 확인
[1] FALSE
> any(x>5) # x 벡터의 요소 값 중 일부가 5보다 큰지 확인
[1] TRUE
> head(x) # 데이터의 앞 6개 요소 추출
[1] 1 2 3 4 5 6
> head(x,3) # 데이터 중 앞 3개 요소 추출
[1] 1 2 3
> tail(x)
[1] 5 6 7 8 9 10
> tail(x,3) #데이터 중 뒤 3개 요소 추출
[1] 8 9 10
```



③ 벡터 연산에 유용한 함수

- Union, intersect, setdiff, setequal 함수 : 벡터 간 집합 연산

Console C:/RSources/ ↗

```
> x=c(1:3)
> y=c(3:6)
> z=c(3,1,2)
> union(x,y)    # 합집합
[1] 1 2 3 4 5 6
> intersect(x,y) #교집합
[1] 3
> setdiff(x,y)   # 차집합(x에서 y와 동일한 요소 제외)
[1] 1 2
> setdiff(y,x)   # 차집합(y에서 x와 동일한 요소 제외)
[1] 4 5 6
> setequal(x,y)
[1] FALSE
> setequal(z,x)  # x와 z가 동일한지 비교
[1] TRUE
```



3.6 배열(행렬)

- 배열: 열과 행으로 구성된 데이터
- 학생 100명의 DB, OS, Data Science 성적을 한꺼번에 저장하려면
과목은 열(column)로, 학생 별 성적은 행(row)으로 100X3 배열로 생성

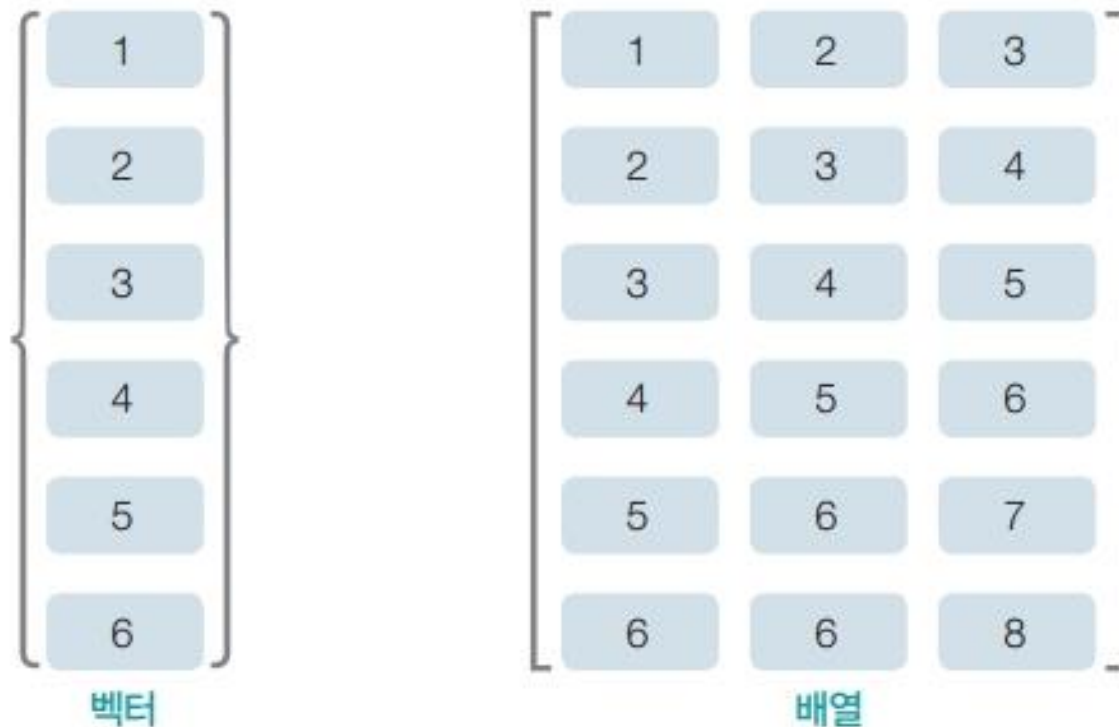


그림 3-4 벡터와 배열의 구성 형태



3.6 배열(행렬)

① 배열 생성 함수

- array 함수: N차원 배열 생성

벡터 Data

차원을 정의하는 벡터

Console C:/RSources/

```
> x=array(1:8,c(2,4))
```

```
> x
```

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

행렬로 구성 할 벡터 Data

```
> y=1:12
```

```
> y
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12
```

행과 열 중 하나 결정

```
> matrix(y,nrow=3,byrow=T)
```

```
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

데이터를 행단위로 배치할지 여부(T/F)

```
> matrix(y,ncol=3,byrow=T)
```

```
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```



3.6 배열(행렬)

① 배열 생성 함수

- cbind · rbind 함수: 열 · 행 단위로 묶어 배열 생성

```
Console C:/RSources/ ➔  
> # 벡터를 묶어 배열 생성  
> vect1=c(1:4)  
> vect2=c(5:8)  
> vect3=c(9:12)  
> cbind(vect1,vect2,vect3) # 열 단위로 묶어 배열 생성  
      vect1 vect2 vect3  
[1,]     1     5     9  
[2,]     2     6    10  
[3,]     3     7    11  
[4,]     4     8    12  
> rbind(vect1,vect2,vect3) # 행단위로 묶어 배열 생성  
      [,1] [,2] [,3] [,4]  
vect1     1     2     3     4  
vect2     5     6     7     8  
vect3     9    10    11    12
```



3.6 배열(행렬)

② 배열 연산

- 배열을 이용한 연산에서는 기본적인 덧셈, 뺄셈, 행렬 곱, 전치 행렬(transposed matrix), 역 행렬(inverse matrix), 행렬식(determinant) 등을 구해야 한다.


표 3-7 행렬 연산자


연산자	설명
+, -	행렬의 덧셈과 뺄셈
*	R에서의 행렬 곱셈(각 열별 곱셈)
%*%	수학적인 행렬 곱셈
t(), aperm()	전치 행렬
solve()	역행렬
det()	행렬식



3.6 배열(행렬)

② 배열 연산(예제)

```
Console C:/RSources/   
> x=array(1:4,c(2,2))  
> y=array(5:8,c(2,2))  
> x  
      [,1] [,2]  
[1,]    1    3  
[2,]    2    4  
> y  
      [,1] [,2]  
[1,]    5    7  
[2,]    6    8  
> x+y  
      [,1] [,2]  
[1,]    6   10  
[2,]    8   12  
> y-x  
      [,1] [,2]  
[1,]    4    4  
[2,]    4    4
```

```
Console C:/RSources/   
> x*y      #각 열별 곱셈  
      [,1] [,2]  
[1,]    5   21  
[2,]   12   32  
> x%*%y    # 수학적인 행렬 곱셈  
      [,1] [,2]  
[1,]   23   31  
[2,]   34   46  
> t(x)      # x의 전치 행렬  
      [,1] [,2]  
[1,]    1    2  
[2,]    3    4  
> solve(x)  # x의 역행렬  
      [,1] [,2]  
[1,]   -2   1.5  
[2,]    1  -0.5  
> det(x)    # x의 행렬식  
[1] -2
```



3.6 배열(행렬)

③ 배열에 유용한 함수

- apply 함수 : 배열의 행 또는 열 별로 함수 적용
- dim 함수 : 배열의 크기(차원의 수)

Console C:/RSources/ ↗

```
> x=array(1:12, c(3,4))
```

```
> x
```

```
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

행

```
> apply(x,1,mean)
```

```
[1] 5.5 6.5 7.5
```

열

```
> apply(x,1,max)
```

```
[1] 10 11 12
```

```
> apply(x,2,mean)
```

```
[1] 2 5 8 11
```

연산을 수행할 함수

```
> apply(x,2,min)
```

```
[1] 1 4 7 10
```

```
> dim(x)
```

```
[1] 3 4
```

```
> apply(x,2,sum)
```

```
[1] 6 15 24 33
```

Files Plots Packages Help Viewer

← → ↩ ↪ 🔍 apply × Refresh Help Topi

R: Apply Functions Over Array Margins ▾ Find in Topic

apply {base} R Documentation

Apply Functions Over Array Margins

Description

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Usage




```
apply(X, MARGIN, FUN, ...)
```



3.6 배열(행렬)

③ 배열에 유용한 함수

- sample 함수 : 벡터나 배열에서 샘플 추출

```
Console C:/RSources/     
> x=array(1:12, c(3,4))  
> sample(x) # 배열의 요소를 임의로 섞어서 추출  
[1] 2 5 7 1 3 11 9 6 12 10 4 8  
> sample(x, 10) # 배열의 요소 중 10개를 골라 추출  
[1] 5 11 4 2 6 8 9 10 1 12  
> sample(x,10,prob=c(1:12)/24) #각 요소별 추출 확률을 달리함  
[1] 7 11 5 4 6 9 12 10 2 8  
> sample(10) # 단순히 숫자만 사용하여 샘플을 만듦  
[1] 2 6 1 7 8 9 5 10 4 3  
> sample(x,10,prob=c(1:12)/36) #각 요소별 추출 확률을 달리함  
[1] 12 2 8 9 6 11 10 4 5 7
```



Thank you

