R컴퓨팅

7강 프로그래밍 구조 I

정보통계학과 장영재 교수

1 프로그래밍 언어로서의 R

2 연산자

1 프로그래밍 언어로서의 R

1 프로그래밍 언어로서의 R

- R은 반복문, 조건문 등을 이용하여 다양한 프로그래밍이 가능한 프로그래밍 언어

기본적인 프로그래밍 구조를 적절히 활용하여
효과적인 작업수행을 위한 도구를 만들어
내는 것이 R 사용자의 목표

2 연산자

2 연산자

1 산술연산자

›보기 7-1 : 수치형 자료의 사칙연산과 관련된 예제

① 덧셈연산자 : +

```
〉 1+2                  #  일반적인  숫자의  덧셈
[1] 3
〉 x〈–3            #  값을  변수에  할당한  후에  덧셈
〉 y〈–2
〉 x+y
[1] 5
〉 vec1〈–c(1,3)                  # 벡터 vec1을 정의
〉 vec2〈–c(2,4)                  # 벡터 vec2를 정의
```

2 연산자

1 산술연산자

```
> vec1+vec2                    # 벡터의 덧셈
[1] 3 7
> A<-matrix(c(5,10,2,1), ncol=2)          # 행렬 A를 정의
> B<-matrix(c(3,4,5,6), ncol=2)           # 행렬 B를 정의
> A+B                  # 행렬의 덧셈
[,1] [,2]
[1,] 8 7
[2,] 14 7
```

2 연산자

1 산술연산자

② 뺄셈연산자 : -

```
> 4-2
[1] 2
> x<-5
> y<-2
> x-y
[1] 3
> vec1<-c(4,3)          # 벡터 vec1을 정의
> vec2<-c(2,1)          # 벡터 vec2를 정의
> vec1-vec2                   # 벡터의 뺄셈
[1] 2 2
```

2 연산자

1 산술연산자

```
> A<-matrix(c(5,10,2,1), ncol=2)        # 행렬 A를 정의
> B<-matrix(c(3,4,5,6), ncol=2)         # 행렬 B를 정의
> A-B                                   # 행렬의 뺄셈
[,1] [,2]
[1,] 2 -3
[2,] 6 -5
```

2 연산자

1 산술연산자

③ 곱셈연산자 : *

```
> 2*5
[1] 10
> x<-4
> y<-3
> x*y
[1] 12
> vec1<-c(2,3)              # 벡터 vec1을 정의
> vec2<-c(4,5)              # 벡터 vec2를 정의
> vec1*vec2        # 곱셈연산자 *에 의한 곱셈은 각 벡터의 동일 위치 원소 간의 곱
[1] 8 15
```

2 연산자

1 산술연산자

```
> A<-matrix(c(5,10,2,1), ncol=2)      # 행렬 A를 정의
> B<-matrix(c(3,4,5,6), ncol=2)       # 행렬 B를 정의
> A*B          # 곱셈연산자 *에 의한 곱셈은 각 행렬의 동일 위치 원소 간의 곱
     [,1] [,2]
[1,]   15   10
[2,]   40    6
```

2 연산자

1 산술연산자

④ 나눗셈연산자 : /

```
> 10/5
[1] 2
> x<-4
> y<-2
> x/y
[1] 2
> vec1<-c(6,4)          # 벡터 vec1을 정의
> vec2<-c(2,2)          # 벡터 vec2를 정의
> vec1/vec2             # 각 벡터의 동일 위치 원소 간의 나눗셈
[1] 3 2
```

2 연산자

1 산술연산자

```
> A<-matrix(c(5,10,2,1), ncol=2)        # 행렬 A를 정의
> B<-matrix(c(3,4,5,6), ncol=2)         # 행렬 B를 정의
> A/B                  # 각 행렬의 동일 위치 원소 간의 나눗셈
     [,1]      [,2]
[1,] 1.666667 0.4000000
[2,] 2.500000 0.1666667
```

2 연산자

1 산술연산자

› 보기 7-2 : 수치형 원소로 이루어진 행렬의 제곱, 정수 나눗셈 및 행렬의 곱셈과 관련된 예제

① 제곱연산자 : ^

```
〉 2^3          # 일반적인 수치형 자료의 제곱 2의 세 제곱
[1] 8
〉 A〈-matrix(c(5,10,2,1), ncol=2)        # 행렬 A를 정의
〉 B〈-matrix(c(2,2,2,2), ncol=2)          # 행렬 B를 정의
〉 A^B          # 행렬의 각 원소 간 제곱연산 수행
[,1] [,2]
[1,] 25 4
[2,] 100 1
```

2 연산자

1 산술연산자

② 정수 나눗셈연산자 : %/%

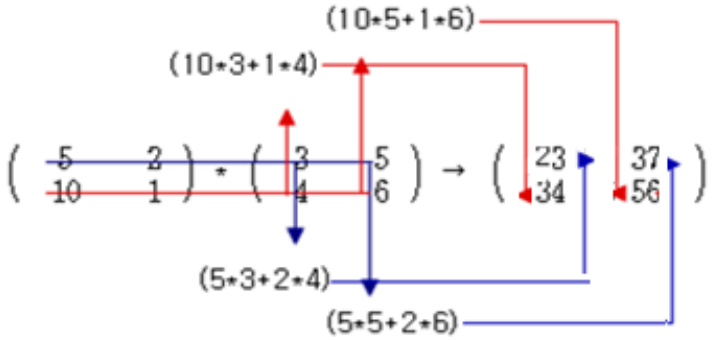
```
> x<-3
> y<-2
> x%/%y          # 정수 나눗셈은 몫의 정수부분만을 출력
[1] 1            # 나눗셈 결과는 1.50이지만, 정수값인 1만을 출력
> A<-matrix(c(5,10,2,1), ncol=2)          # 행렬 A를 정의
> B<-matrix(c(3,4,5,6), ncol=2)           # 행렬 B를 정의
> A%/%B              # 행렬의 각 원소간 정수 나눗셈을 실시
[,1] [,2]
[1,] 1 0
[2,] 2 0
```

2 연산자

1 산술연산자

③ 행렬의 곱셈연산자 : %*%

```
> A<-matrix(c(5,10,2,1), ncol=2)
> B<-matrix(c(3,4,5,6), ncol=2)
> A%*%B
[,1] [,2]
[1,] 23 37
[2,] 34 56
```



2 연산자

2 비교연산자와 논리연산자

- 비교연산자는 대상이 되는 두 객체를 비교하여 비교의 결과가 참인지 거짓인지를 밝히는 연산을 수행하며 결과값으로 논리값을 출력

- 논리연산자는 논리값을 결합하여 논리 구조를 설정하는 연산을 수행

2 연산자

2 비교연산자와 논리연산자

- 보기 7-3 : 비교연산자와 논리연산자의 사용법

① 비교연산자

- == : 비교되는 두 항이 같은지를 비교. 같을 경우 True, 다를 경우 False

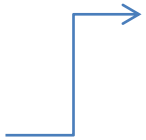
```
> 1==2
[1] FALSE
> x<-2
> y<-3
> x==y
[1] FALSE
```

2 연산자

2 비교연산자와 논리연산자

- != : 비교되는 두 항이 다른지를 비교. 같을 경우 False, 다를 경우 True

```
> 1!=2
[1] TRUE
> x<-2
```



```
> y<-3
> x!=y
[1] TRUE
```

- <= : 왼쪽 항이 오른쪽 항보다 작거나 같은지를 판단. 작거나 같으면 True, 크면 False

```
> 1<=2
[1] TRUE
```

```
> x<-2
> y<-2
> x<=y
[1] TRUE
```

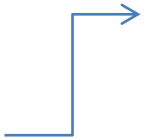
```
> x<-3
> y<-2
> x<=y
[1] FALSE
```

2 연산자

2 비교연산자와 논리연산자

- < : 왼쪽 항이 오른쪽 항보다 작은지 판단. 작으면 True, 크면 False

```
> 1<2
[1] TRUE
> x<-3
```



```
> y<-2
> x<y
[1] FALSE
```

- > : 왼쪽 항이 오른쪽 항보다 큰지 판단. 크면 True, 작으면 False

```
> 1>2
[1] FALSE
```

```
> x<-3
> y<-2
> x>y
[1] TRUE
```

2 연산자

2 비교연산자와 논리연산자

- >= : 왼쪽 항이 오른쪽 항보다 크거나 같은지 판단. 크거나 같으면 True, 작으면 False

```
> 1>=2
[1] FALSE
```

```
> x<-2
> y<-2
> x>=y
[1] TRUE
```

```
> x<-3
> y<-2
> x>=y
[1] TRUE
```

2 연산자

2 비교연산자와 논리연산자

② 논리연산자

- &&, & : &&는 일반적인 and 논리연산자이고 &는 벡터에서의 and 논리연산자

```
> 2==2 && 3>4
[1] FALSE

> 2==2 && c(2==2, 3>4)
[1] TRUE

> 3==3 && 3>2
[1] TRUE
```

```
> 2==2 & c(2==2, 3>4)
[1] TRUE FALSE
```

2 연산자

2 비교연산자와 논리연산자

- ||, | : ||는 일반적인 or 논리연산자이고 |는 벡터에서의 or 논리연산자

```
> 2==2 || 3>4
[1] TRUE

> 3!=3 || 3>2
[1] TRUE

> 3!=3 || 3>4
[1] FALSE
```

```
> 2!=2 || c(2==2, 3>4)
[1] TRUE

> 2!=2 | c(2==2, 3>4)
[1] TRUE FALSE
```

2 연산자

2 비교연산자와 논리연산자

- 주의할 점은 &&, || 연산자의 경우 일반적인 논리 연산자이므로 벡터에 사용할 경우 결과는 출력되지만, 원하는 결과를 얻을 수 없다는 것

- 벡터에 and나 or 논리연산자를 적용할 때에는 &나 |로 사용하고 벡터의 각 원소 위치 별로 연산을 수행한다는 것을 꼭 기억할 필요

2 연산자

2 비교연산자와 논리연산자

③ 집합연산자

- 벡터를 원소들로 이루어진 집합으로 볼 때, 수학적인 정의에 따른 다양한 집합연산을 실시할 수 있음

2 연산자

2 비교연산자와 논리연산자

- 보기 7-4 : 다음은 주요 집합연산자의 사용법에 관한 예제이다.

- union과 intersect : union(x,y)는 집합 x와 y의 합집합을 구하고 intersect(x,y)는 집합 x와 y의 교집합을 산출

```
〉x 〈–c(1,2,5)
〉y 〈–c(5,1,8,9)
〉union(x,y)
[1] 1 2 5 8 9
```

```
〉 intersect(x,y)
[1] 1 5
```

2 연산자

2 비교연산자와 논리연산자

- setdiff와 setequal : setdiff(x,y)는 집합 x와 y의 차집합을 구하고 setequal(x,y)는 집합 x와 집합 y가 같은지 여부를 판단

- 집합 x와 집합 y는 위의 예에서 정의된 집합과 동일하다고 가정하면,

```
〉 setdiff(x,y)
[1] 2

〉 setdiff(y,x)
[1] 8 9
```

```
〉setequal(x,y)
[1] FALSE

〉setequal(x,y)
[1] FALSE
```

2 연산자

2 비교연산자와 논리연산자

- %in%와 choose(n,k) : c%in%x는 원소 c가 집합 x에 속하는지를 판단하고 choose(n,k)는 n개의 원소로 이루어진 집합에서 추출 가능한 k개의 원소로 이루어진 부분집합의 수를 계산

```
> 2 %in% x
[1] TRUE

> 2 %in% y
[1] FALSE
```

```
> choose(5,2)
[1] 10
```

R컴퓨팅