R언어의 데이터 형태: 벡터, 행렬, 배열, 데이터프레임, 리스트

1차원: 벡터, 리스트

2차원: 행렬, 데이터프레임

필요한만큼: 배열

동질적: 벡터, 행렬, 배열

이질적: 리스트, 데이터 프레임

벡터: 다른 언어의 배열과 비슷하며 한가지의 스칼라 데이터 타입의 데이터를 저장할 수 있다.

Ex) 세개의 숫자 1, 4, 6을 포함하는 벡터

> C(1,4,6)

[1] 1 4 6

**벡터 결합**

벡터는 함수 c 를 통해 결합 될 수 있습니다 . 예를 들어, 다음 두 벡터 n 과 s 는 두 벡터의 요소가 포함 된 새 벡터로 결합됩니다.

> n = c (2, 3, 5)   
> s = c ( "aa", "bb", "cc", "dd", "ee")   
> c (n, s)   
[1] "2" "3" "5" "aa" "bb" "cc" "dd" "ee"

# 벡터 산술

벡터의 산술 연산은 멤버별로, *즉* 멤버별로 수행 됩니다.

예를 들어 두 개의 벡터 a 와 b 가 있다고 가정 합니다.

> a = c (1, 3, 5, 7)   
> b = c (1, 2, 4, 8)

그런 다음, a 를 5로 곱 하면 각 구성원에 5를 곱한 벡터를 얻습니다.

> 5  \*  a   
[1] 5 15 25 35

그리고 만약 a 와 b를 합친다면, 그 합은 a 와 b 로부터의 대응하는 멤버들의 합인 벡터가 될 것 입니다.

> a + b   
[1] 2 5 9 15

마찬가지로 뺄셈, 곱셈 및 나눗셈에 대해 우리는 memberwise 연산을 통해 새로운 벡터를 얻습니다.

> a  -  b   
[1] 0 1 1  - 1   
   
> a  \*  b   
[1] 1 6 20 56   
   
> a / b   
[1] 1.000 1.500 1.250 0.875

#### 재활용 규칙

두 벡터의 길이가 다른 경우, 더 긴 벡터와 일치시키기 위해 더 짧은 벡터가 재활용됩니다. 예를 들어, 다음의 벡터 u 와 v 는 다른 길이를 가지며 그 합은 짧은 벡터 u의 값을 재활용하여 계산됩니다 .

> u = c (10, 20, 30)   
> v = c (1, 2, 3, 4, 5, 6, 7, 8, 9)   
> u + v   
[1] 11 22 33 14 25 36 17 28 39

# 벡터 색인

*하나의 대괄호*"[]" 연산자 내에서 색인을 선언하여 벡터의 값을 검색합니다 .

예를 들어, 다음은 벡터 멤버를 검색하는 방법을 보여줍니다. 벡터 인덱스는 1부터 시작하므로 세 번째 멤버를 검색 할 때 인덱스 위치 3 을 사용합니다 .

> s = c ( "aa", "bb", "cc", "dd", "ee")   
> s [3]   
[1] "cc"

다른 프로그래밍 언어와 달리 대괄호 연산자는 개별 구성원 이상을 반환합니다. 사실 대괄호 연산자의 결과는 다른 벡터이며, s [3] 은 단일 멤버 "cc"를 포함하는 벡터 **슬라이스** 입니다.

#### 음수 색인

인덱스가 음수이면 위치가 음수 인덱스와 절대 값이 같은 멤버를 제거합니다. 예를 들어, 다음은 세 번째 멤버가 제거 된 벡터 슬라이스를 만듭니다.

> s [ - 3]   
[1] "aa" "bb" "dd" "ee"

#### 범위를 벗어난 색인

색인이 범위를 벗어난 경우 누락 된 값은 기호 NA 를 통해보고됩니다 .

> s [10]   
[1] NA

행렬: 동일한 스칼라 데이터 타입의 2차원 데이터 구조를 말한다.

EX) **행렬** 이차원 직사각형 레이아웃으로 배열 된 데이터 요소의 집합이다. 다음은 2 행 3 열의 행렬의 예입니다.

    [] 2 4 3 A = 1 5 7


행렬 함수를 사용하여 행렬 의 메모리 표현을 R로 재현 합니다. 데이터 요소는 동일한 기본 유형이어야합니다.

> A = 행렬 (   
+ c (2, 4, 3, 1, 5, 7), # 데이터 요소   
+ nrow = 2, # 행 수   
+ ncol = 3, # 열 수   
+ byrow = TRUE) 행 단위로 행렬 채우기   
   
> A # 행렬을 인쇄하십시오   
     [, 1] [, 2] [, 3]   
[1,] 2 4 3   
[2,] 1 5 7

상기 소자 *m 번째* 열, *N 번째* 의 열 A를 발현에 의해 액세스 될 수 A [m, n]을 .

> 2 행 3 열의 A [2, 3] # 요소   
[1] 7

전체 *제 m* 행 A 는 A [m,] 로서 추출 될 수있다 .

> A [2,] # 2 번째 행   
[1] 1 5 7

유사하게, 전체 *n 번째* 열 A 는 A [, n] 으로 추출 될 수있다 .

> A [, 3] # 세 번째 열   
[1] 3 7

한 번에 둘 이상의 행 또는 열을 추출 할 수도 있습니다.

> A [, c (1,3)] # 첫 번째와 세 번째 열   
     [, 1] [, 2]   
[1,] 2 3   
[2,] 1 7

행렬의 행과 열에 이름을 할당하면 이름별로 요소에 액세스 할 수 있습니다.

> 희미한 이름 (A) = 목록 (   
+ c ( "row1", "row2"), # 행 이름   
+ c ( "col1", "col2", "col3")) # 열 이름   
   
> A # print A   
     col1 col2 col3   
행 1 2 4 3   
행 2 1 5 7   
   
> A [ "row2", "col3"] # 2 행 3 열의 요소   
[1] 7

행렬을 구성하는 다양한 방법이 있습니다. 데이터 요소로 직접 행렬을 구성하면 행렬 내용이 기본적으로 열 방향을 따라 채워집니다. 예를 들어 다음 코드 단편에서 B 의 내용은 연속적으로 열을 따라 채워집니다.

> B = 매트릭스 (   
+ c (2, 4, 3, 1, 5, 7),   
+ nrow = 3,   
+ ncol = 2)   
   
> B # B는 3 행 2 열   
     [, 1] [, 2]   
[1,] 2 1   
[2,] 4 5   
[3,] 3 7

#### 바꾸어 놓다

행렬을 함수 t 로 교체하여 행렬 의 **전치** 를 만듭니다.

> t (B) # B의 전치   
     [, 1] [, 2] [, 3]   
[1,] 2 4 3   
[2,] 1 5 7

#### 행렬 결합

동일한 수의 행을 갖는 2 개의 행렬의 열은 더 큰 행렬로 결합 될 수있다. 예를 들어 다른 행렬 C 에도 3 행 이 있다고 가정 합니다.

> C = 매트릭스 (   
+ c (7, 4, 2),   
+ nrow = 3,   
+ ncol = 1)   
   
> C # C에는 3 개의 행이 있습니다.   
     [,1]   
[1,] 7   
[2,] 4   
[3,] 2

그럼 우리의 열을 결합 할 수 있습니다 B 와 C를 함께 cbind .

> cbind (B, C)   
     [, 1] [, 2] [, 3]   
[1,] 2 1 7   
[2,] 4 5 4   
[3,] 3 7 2

비슷하게 rbind 함수를 사용하여 같은 수의 열을 가진다면 두 행렬의 행을 결합 할 수 있습니다 .

> D = 행렬 (   
+ c (6, 2),   
+ nrow = 1,   
+ ncol = 2)   
   
> D # D에는 2 개의 열이 있습니다.   
     [, 1] [, 2]   
[1,] 6 2   
   
> rbind (B, D)   
     [, 1] [, 2]   
[1,] 2 1   
[2,] 4 5   
[3,] 3 7   
[4,] 6 2

#### 탈구축

모든 열 벡터를 하나로 결합 하는 c 함수 를 적용하여 행렬을 분해 할 수 있습니다 .

> c (B)   
[1] 2 4 3 1 5 7

리스트: 다른 개체를 포함하는 일반적인 벡터이다.

Ex) 예를 들어, 다음 변수 x 는 세 개의 벡터 n , s , b 및 숫자 값 3 의 사본을 포함하는 목록 입니다.

> n = c (2, 3, 5)   
> s = c ( "aa", "bb", "cc", "dd", "ee")   
> b = c (TRUE, FALSE, TRUE, FALSE, FALSE)   
> x = list (n, s, b, 3) # x는 n, s, b의 복사본을 포함합니다.

#### 목록 분할

*하나의 대괄호*"[]" 연산자를 사용 하여 목록 슬라이스를 검색합니다 . 다음은 s 의 복사본 인 x 의 두 번째 멤버를 포함하는 슬라이스 입니다.

> x [2]   
[[1]]   
[1] "aa" "bb" "cc" "dd" "ee"

인덱스 벡터를 사용하면 여러 멤버가있는 슬라이스를 검색 할 수 있습니다. x 의 두 번째와 네 번째 멤버가 포함 된 슬라이스 입니다.

> x [c (2, 4)]   
[[1]]   
[1] "aa" "bb" "cc" "dd" "ee"   
   
[[2]]   
[1] 3

#### 회원 참고서

직접리스트 멤버를 참조하기 위해, 우리는 사용해야 *이중 대괄호*"[]" 연산자. 다음 객체 x [2] 는 x 의 두 번째 멤버입니다 . 즉, x [[2]] 는 s 의 사본 이지만 s 또는 사본을 포함하는 슬라이스 는 *아닙니다* .

> x [[2]]   
[1] "aa" "bb" "cc" "dd" "ee"

우리는 그 내용을 직접 수정할 수 있습니다.

> x [[2]] [1] = "ta"   
> x [[2]]   
[1] "ta" "bb" "cc" "dd" "ee"   
> s   
[1] "aa" "bb" "cc" "dd" "ee"# s는 영향을받지 않습니다.

데이터 프레임: 스칼라 데이터 타입에 상관없이 2차원 형태의 데이터 구조를 말한다.

Ex) **데이터 프레임은** 데이터 테이블을 저장하기 위해 사용된다. 동일한 길이의 벡터 목록입니다. 예를 들어, 다음 변수 df 는 세 개의 벡터 n , s , b를 포함하는 데이터 프레임 입니다.

> n = c (2, 3, 5)   
> s = c ( "aa", "bb", "cc")   
> b = c (TRUE, FALSE, TRUE)   
> df = data.frame (n, s, b) # df는 데이터 프레임

#### 내장 데이터 프레임

우리는 튜토리얼을 위해 R에 내장 된 데이터 프레임을 사용합니다. 예를 들어, 여기에 **mtcars** 라고하는 R의 내장 데이터 프레임이 있습니다.

> mtcars   
               mpg cyl disp hp drat wt ...   
마쯔다 RX4 21.0 6 160 110 3.90 2.62 ...   
Mazda RX4 Wag 21.0 6 160 110 3.90 2.88 ...   
Datsun 710 22.8 4 108 93 3.85 2.32 ...   
               ............

**헤더** 라고하는 표의 맨 위 줄 에는 열 이름이 들어 있습니다. 이후 각 수평선은 **데이터 행을** 나타내며 **행** 의 이름으로 시작하여 실제 데이터가옵니다. 한 행의 각 데이터 멤버를 **셀** 이라고합니다 .

셀의 데이터를 검색하려면 행 및 열 좌표를 *단일 대괄호*"[]" 연산자에 입력합니다. 두 좌표는 쉼표로 구분됩니다. 즉, 좌표는 행 위치로 시작한 다음 쉼표가 따라오고 열 위치로 끝납니다. 순서가 중요합니다.

다음은 mtcars 의 첫 번째 행, 두 번째 열의 셀 값입니다 .

> mtcars [1, 2]   
[1] 6

또한 숫자 좌표 대신 행 및 열 이름을 사용할 수 있습니다.

> mtcars [ "Mazda RX4", "cyl"]   
[1] 6

마지막으로 데이터 프레임의 데이터 행 수는 nrow 함수에 의해 지정됩니다 .

> nrow (mtcars) # 데이터 행 수   
[1] 32

그리고 데이터 프레임의 컬럼 수는 ncol 함수에 의해 주어진다 .

> ncol (mtcars) # 열 수   
[1] 11

mtcars 데이터 세트 에 대한 자세한 내용은 R 문서를 참조하십시오.

> 도움 (mtcars)

#### 시사

전체 데이터 프레임을 인쇄하는 대신 미리 헤드 기능으로 미리 보기하는 것이 바람직 합니다.

> 머리 (mtcars)   
               mpg cyl disp hp drat wt ...   
마쯔다 RX4 21.0 6 160 110 3.90 2.62 ...   
               ............

배열: N차원 데이터이다. 하나의 스칼라 데이터 타입만을 가진다.