R

[R] R에 사용되는 행렬(matrix)의 개념 및 사용되는 함수

2020. 7. 23. 02:26 수정 삭제 공개

-행렬(matrix)의 개념

2차원 벡터로서 동일 타입의 데이터만 저장 가능
인덱싱 : [행의 인덱싱,열의 인덱싱]
행렬 생성방법 :
matrix(data=벡터, nrow=행의 개수, ncol=열의 개수)
matrix(data=벡터, nrow=행의갯수, ncol=열의 개수, byrow=TRUE)
rbind(백터), cbind(벡터) dim

-행렬 생성하는 예제

#matrix() - 행렬 생성

```r
## 행렬(matrix)
#단일유형인 스칼라 타입을 담는다.
#matrix(), rbing(), cbind() - 행렬 만드는 함수
#apply(data, margin, function) - 행이나 열을 나타낸다. 함수가 함수를 매개변수로 받아 실행하는 함수


x ← c(1,2,3,4,5,6,7,8,9)
mat ← matrix(x) #들어 있는 값을 행에 넣어주는 형식
mat
class(mat)

matrix(x,nrow=3)
matrix(x,ncol=3)
matrix(x,nrow=3,ncol=3,byrow =T)

matrix( 0, nrow=2, ncol = 3)
```

```
> x ← c(1,2,3,4,5,6,7,8,9)
> mat ← matrix(x) #들어 있는 값을 행에 넣어주는 형식
> mat
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
[6,]    6
[7,]    7
[8,]    8
[9,]    9
> class(mat)
[1] "matrix" "array"
>
```

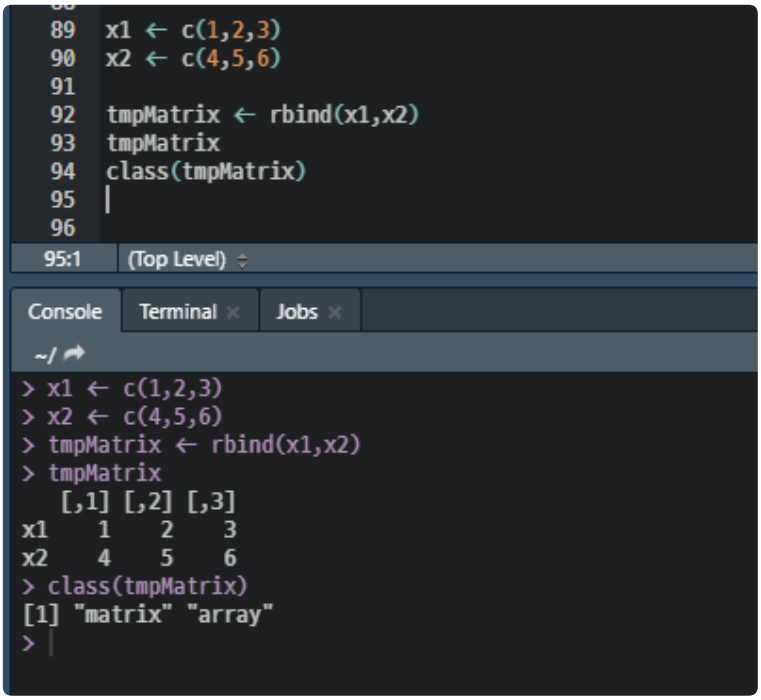
```
> matrix(x,nrow=3)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> matrix(x,ncol=3)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> matrix(x,nrow=3,ncol=3,byrow =T)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> matrix( 0, nrow=2, ncol = 3)
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
>
```

#diag - 정방향의 행렬을 만들때 사용하는 함수

```r
#정방향으로 만들때 사용하는 함수
matD ←diag(0,3)
class(matD)
```

```
> matD
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
>
```

#rbind (Row 기준으로 행렬 생성)



#cbind (col 기준으로 행렬 생성)

```
91
92  tmpMatrix ← rbind(x1,x2)
93  tmpMatrix
94  class(tmpMatrix)
95
96  tmpMatrix ← cbind(x1,x2)
97  tmpMatrix
98  class(tmpMatrix)
99
100
```

99:1 (Top Level)

Console Terminal Jobs

~/

```
> tmpMatrix ← cbind(x1,x2)
> tmpMatrix
     x1 x2
[1,]  1  4
[2,]  2  5
[3,]  3  6
> class(tmpMatrix)
[1] "matrix" "array"
>
>
```

-행렬 사용되는 함수

#t() -#전치행렬 - 행이 열로 바뀌는 함수

```
22
23  #전치행렬 - 행이 열로 바뀌는 함수
24  x ← matrix( c(1,2,3,4,5,6) , 2 , 3)
25  x
26
27  t(x)
28
```

```
> #전치행렬 - 행이 열로 바뀌는 함수
> x ← matrix( c(1,2,3,4,5,6) , 2 , 3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> t(x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
>
```

```r
#데이터 접근([행 인덱스, 열 인덱스])

x ← matrix(x,3,3)
x

# row에 대한 인덱스
row(x)

#col 대한 인덱스
col(x)
```

```
> x ← matrix(x,3,3)
> x
     [,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
[3,]    3    6    3
> # row에 대한 인덱스
> row(x)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
[3,]    3    3    3
> #col 대한 인덱스
> col(x)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    2    3
[3,]    1    2    3
>
```

#현재 행렬에 대한 인덱스 접근

```
> x
     [,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
[3,]    3    6    3
>
```

```r
#데이터 접근
x[1,1]
x[3,3]

x[1:2,2]
x[1:2,1:3]

x[-3,]
```

```
> #데이터 접근
> x[1,1]
[1] 1
> x[3,3]
[1] 3
>
> x[1:2,2]
[1] 4 5
> x[1:2,1:3]
     [,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
>
> x[-3,]
     [,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
>
```

#행렬을 만들때

```
81  #행렬을 만들때 요소의 개수가 맞지 않을때
82  x1 ← c(1,2,3)
83  x2 ← c(4,5,6,7)
84
85  tmpMatrix ← rbind(x1,x2)
86  tmpMatrix
87  class(tmpMatrix)
```

```
~/
> #행렬을 만들때 요소의 개수가 맞지 않을때
> x1 ← c(1,2,3)
> x2 ← c(4,5,6,7)
>
> tmpMatrix ← rbind(x1,x2)
경고메시지(들):
In rbind(x1, x2) :
  number of columns of result is not a multiple of vector length (arg 1)
> tmpMatrix
   [,1] [,2] [,3] [,4]
x1    1    2    3    1
x2    4    5    6    7
> class(tmpMatrix)
[1] "matrix" "array"
>
```

#dimnames - 행 이름과 열 이름 지정

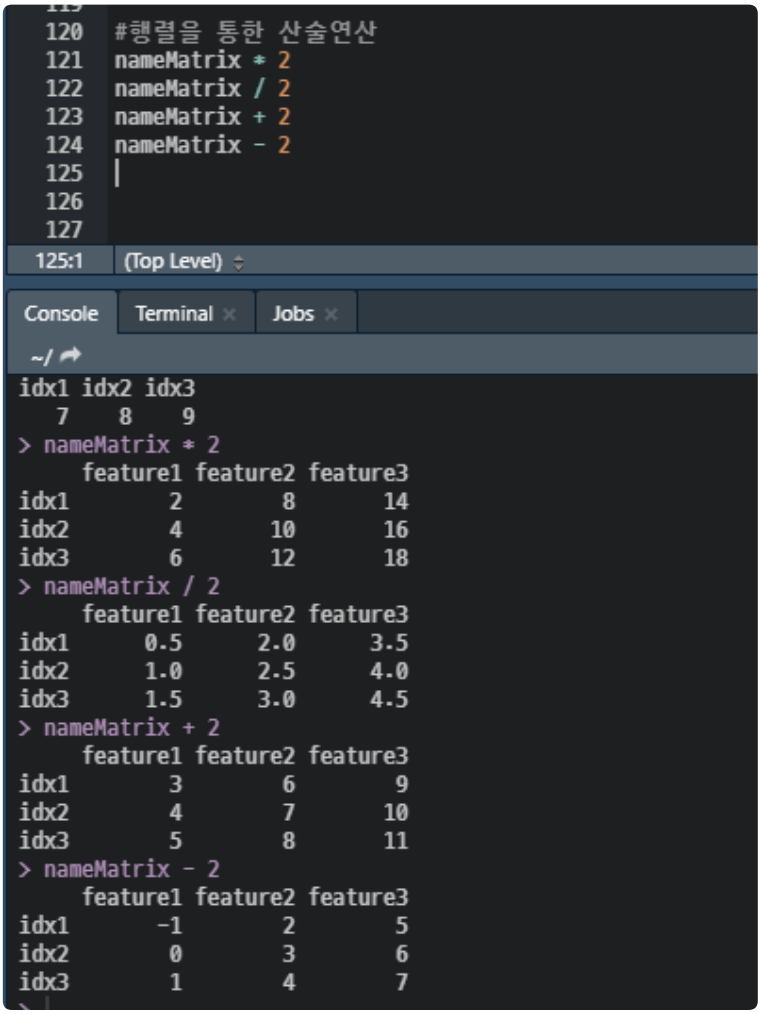
```
104
105  #matrix() 함수에 dimnames 옵션을 활용하면 행이름, 열이름을 지정할 수 있고 이를 활용하여 인덱싱이 가능하다.
106
107  nameMatrix ← matrix( c(1,2,3,4,5,6,7,8,9),
108                       nrow = 3,
109                       dimnames = list(c("idx1","idx2","idx3"),
110                                       c("feature1","feature2","feature3")) )
111  nameMatrix
112
```

```
113  nameMatrix["idx1",]
114  nameMatrix[,"feature2"]
115
115:1  (Top Level)

Console  Terminal  Jobs
~/
> nameMatrix ← matrix( c(1,2,3,4,5,6,7,8,9),
+                      nrow = 3,
+                      dimnames = list(c("idx1","idx2","idx3"),
+                                      c("feature1","feature2","feature3")) )
> nameMatrix
     feature1 feature2 feature3
idx1        1        4        7
idx2        2        5        8
idx3        3        6        9
> nameMatrix[idx1,]
에러: 객체 'idx1'를 찾을 수 없습니다
> nameMatrix[idx1,]
에러: 객체 'idx1'를 찾을 수 없습니다
> nameMatrix["idx1",]
feature1 feature2 feature3
       1        4        7
> nameMatrix[,"feature2"]
idx1 idx2 idx3
   4    5    6
>
```

이런 식으로 행과 열의 이름으로 접근이 가능하다.

#행렬을 통한 산술 연산



```
120  #행렬을 통한 산술연산
121  nameMatrix * 2
122  nameMatrix / 2
123  nameMatrix + 2
124  nameMatrix - 2
125
126
127
125:1  (Top Level)
```

Console | Terminal | Jobs

```
~/
idx1 idx2 idx3
   7    8    9
> nameMatrix * 2
     feature1 feature2 feature3
idx1        2        8       14
idx2        4       10       16
idx3        6       12       18
> nameMatrix / 2
     feature1 feature2 feature3
idx1      0.5      2.0      3.5
idx2      1.0      2.5      4.0
idx3      1.5      3.0      4.5
> nameMatrix + 2
     feature1 feature2 feature3
idx1        3        6        9
idx2        4        7       10
idx3        5        8       11
> nameMatrix - 2
     feature1 feature2 feature3
idx1       -1        2        5
idx2        0        3        6
idx3        1        4        7
```

apply() - 행이나 열의 방향으로 특수한 함수를 적용한다

```r
# apply() - 행이나 열 방향으로 특수한 함수를 적용한다.
# margin 1 = row, 2 = col

(x ← matrix(1:4,2,2))


# 열의 합
colSums(x)


# 행의 합
rowSums(x)


sumApply←apply(x, 2, sum)
sumApply
class(sumApply)

```

```
> (x ← matrix(1:4,2,2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
>
> # 열의 합
> colSums(x)
[1] 3 7
>
>
> # 행의 합
> rowSums(x)
[1] 4 6
>
>
> sumApply←apply(x, 2, sum)
> sumApply
[1] 3 7
```

```r

sumApply←apply(x, 1, sum)
sumApply
class(sumApply)


```

```
> sumApply←apply(x, 1, sum)
> sumApply
[1] 4 6
> class(sumApply)
[1] "integer"
> sumApply
[1] 4 6
> class(sumApply)
[1] "integer"
>
```

```
158 #apply()함수를 적용해서 컬럼의 요약정보를 확인해보세요 II
159 sl <-iris[,1]
160 sl
161 str(sl)
162
163 apply(iris[,1:4] ,2,sum)
164 apply(iris[,1:4] ,2,mean)
165 apply(iris[,1:4] ,2,median)
166 apply(iris[,1:4] ,2,max)
167 apply(iris[,1:4] ,2,min)
168
169
```

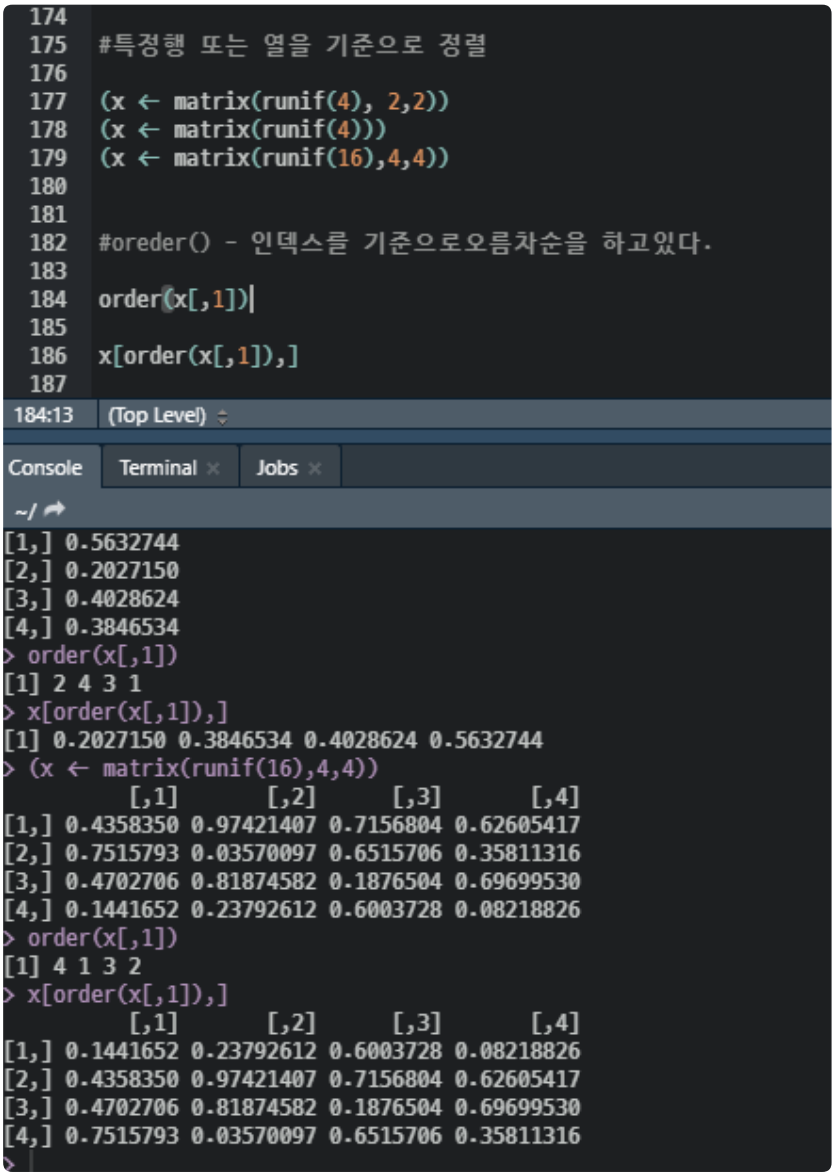
167:25 (Top Level) R Scrip

Console Terminal Jobs

~/

```
 [36] 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6
 [71] 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5
[106] 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9
[141] 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
> str(sl)
 num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
>
> apply(iris[,1:4] ,2,sum)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       876.5        458.6        563.7        179.9
> apply(iris[,1:4] ,2,mean)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
> apply(iris[,1:4] ,2,median)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
        5.80         3.00         4.35         1.30
> apply(iris[,1:4] ,2,max)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
         7.9          4.4          6.9          2.5
> apply(iris[,1:4] ,2,min)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
         4.3          2.0          1.0          0.1
>
```

#order(value) - 기준을 통해 그 부분을 정렬하는 함수



'**R**' 카테고리의 다른 글□

[R] R로 만드는 제어문 (if, else if, for)과 예제□

[R] R에서 사용되는 Data.frame 과 Factor 에 사용되는 다양한 함수□

[R] R에 사용되는 배열(array)과 리스트(list)의 개념 및 사용되는 함수□

[R] R에 사용되는 행렬(matrix)의 개념 및 사용되는 함수□

[R] R에서 사용되는 정규표현식(Regex) 표현 방법과 함수를 통한 사용 예제□

[R] R에 사용되는 벡터(matrix)의 개념 및 사용되는 함수(출력,인덱싱,길이반환,문자열비교 등등)□

apply함수 cbind cbind함수 Matrix matrix함수 r rbind rbind함수 행렬

꾸까꾸

혼자 끄적끄적하는 블로그 입니다.