[R] R에 사용되는 배열(array)과 리스트(list)의 개념 및 사용되는 함수 — 나무늘보의 개발 블로그

노트북:	blog
만든 날짜:	2020-10-01 오후 9:10
URL:	https://continuous-development.tistory.com/35?category=793392

R

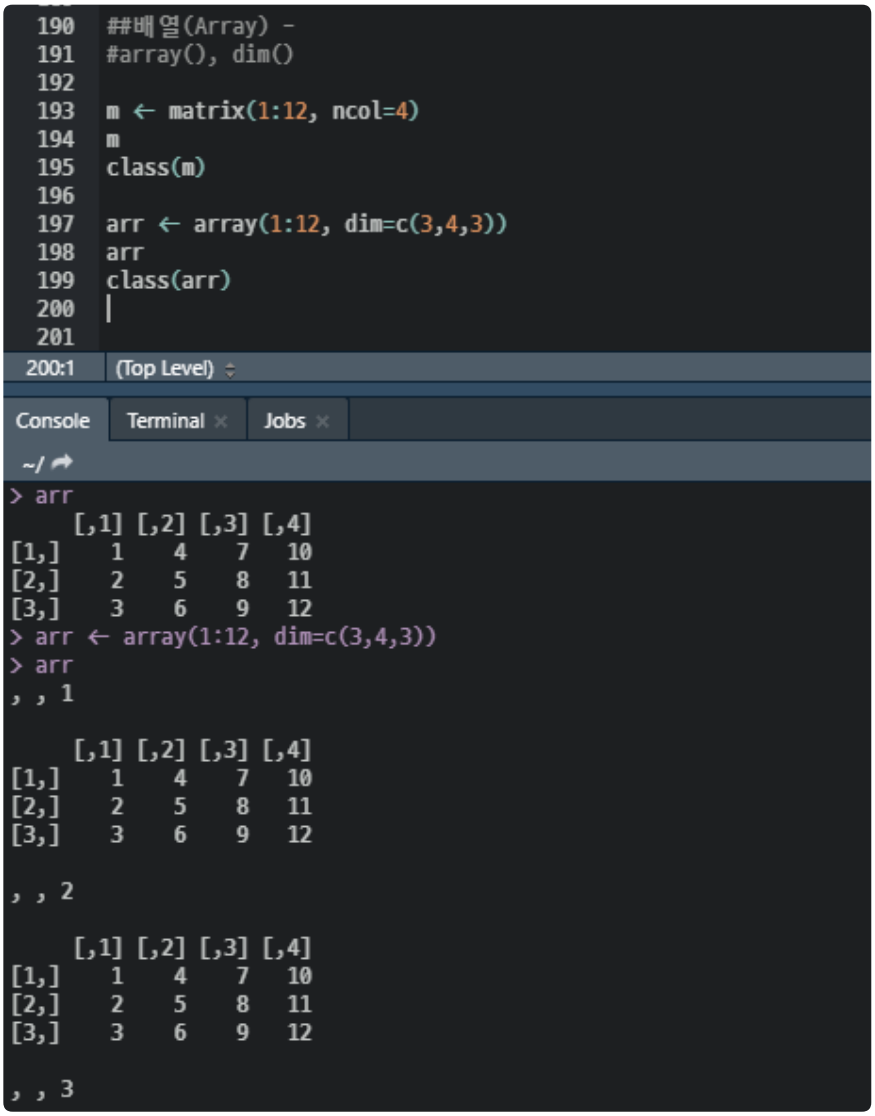
[R] R에 사용되는 배열(array)과 리스트(list)의 개념 및 사용되는 함수

2020. 7. 23. 03:32 수정 삭제 공개

#배열(array)

배열(array) 3차원 벡터이다.
동일 타입의 데이터만 저장 가능
인덱싱 : [행의 인덱싱, 열의 인덱싱, 층(면)의 인덱스]

#배열 생성 array(value , dim=c(value))



```
190  ##배열(Array) -
191  #array(), dim()
192
193  m ← matrix(1:12, ncol=4)
194  m
195  class(m)
196
197  arr ← array(1:12, dim=c(3,4,3))
198  arr
199  class(arr)
200
201
```

200:1 (Top Level)

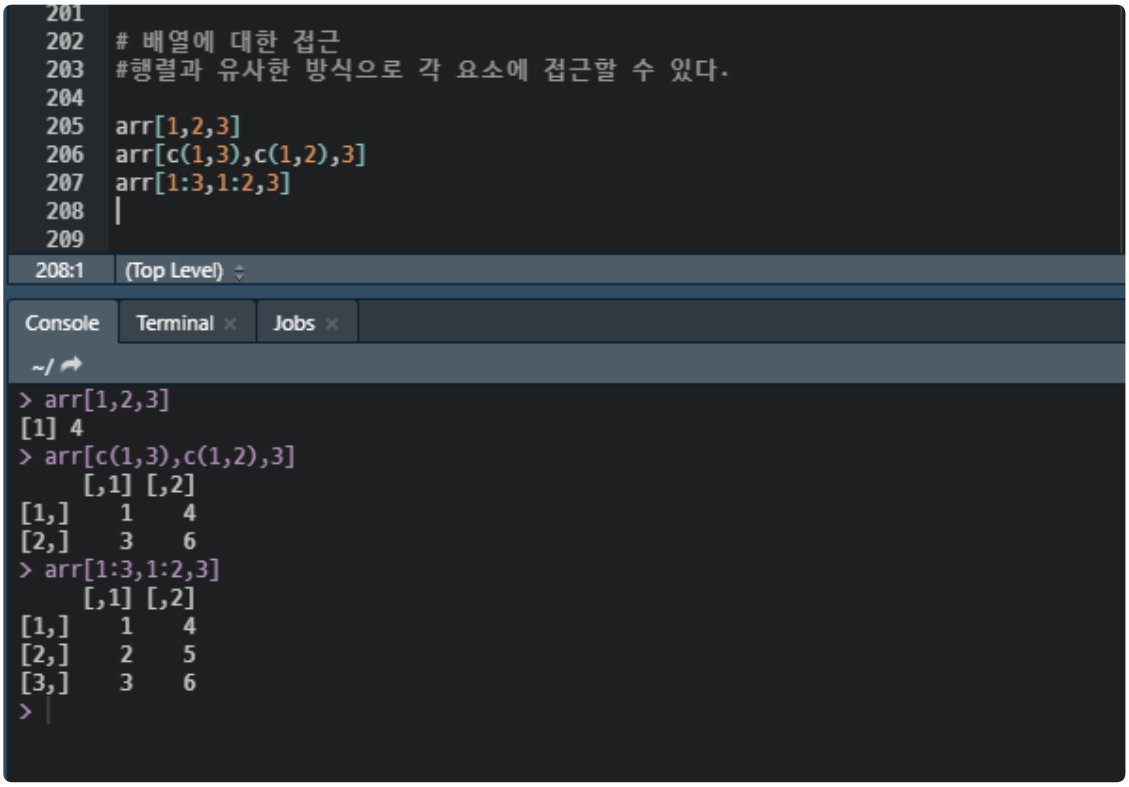
Console Terminal Jobs

~/

```
> arr
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> arr ← array(1:12, dim=c(3,4,3))
> arr
, , 1

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

, , 2

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

, , 3
```

3행 / 4열 /3층 을 가진 배열

#배열에 대한 접근



#리스트(list)

저장 가능한 데이터의 타입, 데이터 샛의 종류에 제한이 없다. 벡터, 행렬, 배열, 데이터 프레임 등의 서로 다른 구조의 데이터를 하나로 묶을 수 있는 자료구조이다. list() 함수로 리스트를 생성하고, [, [[, $ 을 통해 부분집합을 뽑아낸다.

#리스트 - list(키, 값) 형태의 데이터를 담는 연관 배열이다.

```r
#list(카, 값) 형태의 데이터를 담는 연관배열이다. - 데이터 셋의 종류에 제한은 없다 $하위 리스트가 포함한 원소 추출 / 계층수준 한단계 저하
#(key와 valus를 담는 형태 (파이썬에서는 딕셔너리))

#list()
#apply → lapply()-키 밸류를 담아서 보낸다 ,sapply() value만 보낸다.,


list ← list()
list

exList ← list(name="jslim", height = 70)
exList$name
exList&height

#name = 키 valses

simpleList ← list(1:4, rep(3,5) , "cat")
simpleList


newList ← c(list(1, 2, simpleList) , c(3,4))
newList
```

```
> list ← list()
> list
list()
> exList ← list(name="jslim", height = 70)
> exList$name
[1] "jslim"
> exList$height
[1] 70
> simpleList ← list(1:4, rep(3,5) , "cat")
> simpleList
[[1]]
[1] 1 2 3 4

[[2]]
[1] 3 3 3 3 3

[[3]]
[1] "cat"
```

```
256
257 #리스트안에 리스트 중첩
258 list(a = list(c(1,2,3) ),
259      b = list(c(1,2,3,4)))
260
261 overList←list(a = list(c(1,2,3) ),
262           b = list(c(1,2,3,4)))
263
264 overList
265
266 #어느위치로 갈지 정하고 거기서 몇번째 리스트를 가져올지 하고 그다음 몇번째 인덱싱을 할지를 본다
267 overList$a[[1]][2]
268 overList$b[[1]][3]
269
270
271
```

261:36 (Top Level)

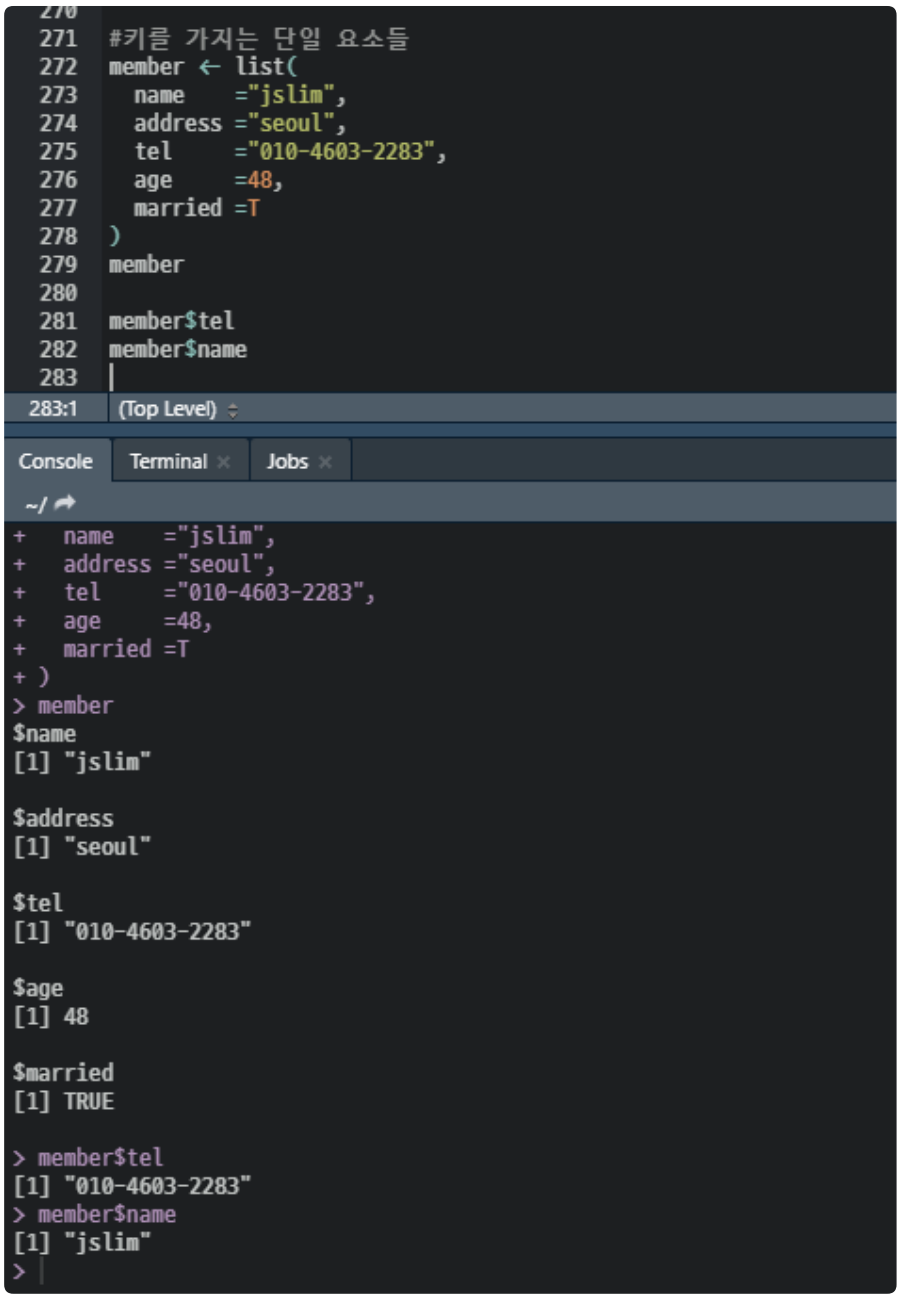
Console Terminal Jobs

~/

```
> #리스트안에 리스트 중첩
> list(a = list(c(1,2,3) ),
+      b = list(c(1,2,3,4)))
$a
$a[[1]]
[1] 1 2 3


$b
$b[[1]]
[1] 1 2 3 4


>
> overList←list(a = list(c(1,2,3) ),
+           b = list(c(1,2,3,4)))
>
> overList
$a
$a[[1]]
[1] 1 2 3


$b
$b[[1]]
[1] 1 2 3 4
```

```
>
> #어느위치로 갈지 정하고 거기서 몇번째 리스트를 가져올지 하고 그다음 몇번째 인덱싱을 할지를 본다
> overList$a[[1]][2]
[1] 2
> overList$b[[1]][3]
[1] 3
>
```

#키에 대해 단일로 값을 가지는 요소들



```r
#키를 가지는 단일 요소들
member ← list(
  name    ="jslim",
  address ="seoul",
  tel     ="010-4603-2283",
  age     =48,
  married =T
)
member

member$tel
member$name

```

283:1 (Top Level)

Console Terminal Jobs

~/

```
+   name    ="jslim",
+   address ="seoul",
+   tel     ="010-4603-2283",
+   age     =48,
+   married =T
+ )
> member
$name
[1] "jslim"

$address
[1] "seoul"

$tel
[1] "010-4603-2283"

$age
[1] 48

$married
[1] TRUE

> member$tel
[1] "010-4603-2283"
> member$name
[1] "jslim"
>
```

#키에 대해 다중으로 값을 가지는 요소들

```r
#키에대해 다중으로 값을 가지는 요소들
member <- list(
  name    = c("섭섭해","임섭순"),
  address = c("seoul","gwangu"),
  age     = c("48","29"),
  gender = c("남자","여자")
)

member$name
member$name[2]
member$age[1] <-38
member$id <- c("jslim","admin")
member
member$id <- NULL
member$address[2] <- "seoul"
member
```

```
> member$id <- c("jslim","admin")
> member
$name
[1] "섭섭해" "임섭순"

$address
[1] "seoul" "seoul"

$age
[1] "38" "29"

$gender
[1] "남자" "여자"

$id
[1] "jslim" "admin"

> member$id <- NULL
> member$address[2] <- "seoul"
> member
$name
[1] "섭섭해" "임섭순"

$address
[1] "seoul" "seoul"

$age
[1] "38" "29"

$gender
[1] "남자" "여자"
```

#서로 다른 자료구조를 가지는 리스트

```r
#서로 다른 자료구조(vector,matrix,array)를 가지는 리스트

multiList <- list(
  one = c("one","two","three"),
  second = matrix(1:9 , nrow = 3),
  third = array(1:12, dim = c(2,3,2)) # 2행 3열에 계층은 2개
)
multiList

multiList$one[1]
multiList$one[2]
multiList$second[2,2]
multiList$second[3,3]
multiList$third[1,3,2]

```

```
[3,]    3    6    9

$third
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12


>
> multiList$one[1]
[1] "one"
> multiList$one[2]
[1] "two"
> multiList$second[2,2]
[1] 5
> multiList$second[3,3]
[1] 9
> multiList$third[1,3,2]
[1] 11
>
```

#unlist() - 리스트를 벡터로 형 변환

```r
# unlist() list → 리스트를 벡터로 형변환
x ← list(1:5)
x
class(x)

vec ← unlist(x)
vec
class(vec)

matrixList ← list(
  row1 = list(1,2,3),
  row2 = list(10,20,30),
  row3 = list(100,200,300)
)

matrixList
class(matrixList)
```

```
> x ← list(1:5)
> x
[[1]]
[1] 1 2 3 4 5

> class(x)
[1] "list"
>
> vec ← unlist(x)
> class(vec)
[1] "integer"
```

#do.call -리스트를 행렬로 바꾸는 함수

```r
# do.call(func, data) - 리스트를 행렬로 바꾸는 명령 (rbind,cbind 만 가능)
matrixList ← list(
  row1 = list(1,2,3),
  row2 = list(10,20,30),
  row3 = list(100,200,300)
)

matrixList
class(matrixList)

row_mat ← do.call(rbind,matrixList)
row_mat

row_mat ← do.call(cbind,matrixList)
row_mat
```

```
> # do.call(func, data) - 리스트를 행렬로 바꾸는 명령 (rbind,cbind 만 가능)
> matrixList ← list(
+   row1 = list(1,2,3),
+   row2 = list(10,20,30),
+   row3 = list(100,200,300)
+ )
> matrixList
$row1
$row1[[1]]
[1] 1

$row1[[2]]
[1] 2

$row1[[3]]
[1] 3


$row2
$row2[[1]]
[1] 10
```

```
> class(matrixList)
[1] "list"
> row_mat ← do.call(rbind,matrixList)
> row_mat
     [,1] [,2] [,3]
row1 1    2    3
row2 10   20   30
row3 100  200  300
> row_mat ← do.call(cbind,matrixList)
> row_mat
     row1 row2 row3
[1,] 1    10   100
[2,] 2    20   200
[3,] 3    30   300
```

#list의 값의 길이 확인

```
344
345 listLength ← list(1:5, list("This is my First time R",c(T,F,T)))
346 listLength
347 length(listLength)
348 length(listLength[[1]])
349 length(listLength[[2]])
350 str_length(listLength[[2]][1])
351 str_length(listLength[[2]][2])
352
```

```
> length(listLength)
[1] 2
> length(listLength[[1]])
[1] 5
> length(listLength[[2]])
[1] 2
> str_length(listLength[[2]][1])
[1] 23
> str_length(listLength[[2]][2])
[1] 20
경고메시지(들):
In stri_length(string) : argument is not an atomic vector; coercing
>
```

#list 처리 함수

```
354
355 # list 처리함수
356 #lapply() : list , key = value  - list 형식
357 #sapply() : list , value        - vactor 형식
358
359
360 x ← list(1:5)
361 y ← list(6:10)
362
363 lapply(c(x,y),FUN = sum)
364 class(lapply(c(x,y),FUN = sum))
365
366 sapply(c(x,y),FUN = sum)
367 class(sapply(c(x,y),FUN = sum))
368
```

```
> x ← list(1:5)
> y ← list(6:10)
> lapply(c(x,y),FUN = sum)
[[1]]
[1] 15

[[2]]
[1] 40

> class(lapply(c(x,y),FUN = sum))
[1] "list"
> sapply(c(x,y),FUN = sum)
[1] 15 40
> class(sapply(c(x,y),FUN = sum))
[1] "integer"
>
```

#리스트 -> 백터 -> 행렬 -> 데이터 프레임 변환 예제

```r
l_value← lapply(iris[ ,1:4], mean)
s_value←sapply(iris[ ,1:4], mean)


# unlist() 리스트를 벡터로 변환
vec ← unlist(l_value)
vec
class(vec)
# matrix() 벡터를 행렬로 변환
mat ← matrix(vec,ncol=4)
mat
class(mat)
#as.data.frame() 행렬을 데이터 프레임으로 변환
dataFrame ← as.data.frame(mat)
dataFrame
class(dataFrame)
#names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여
rename←names(s_value)
re_dataFrame←rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
re_dataFrame
```

```r

# unlist() 리스트를 벡터로 변환
vec ← unlist(l_value)
vec
class(vec)

# matrix() 벡터를 행렬로 변환
mat ← matrix(vec,ncol=4, byrow =T)
mat
class(mat)

#as.data.frame() 행렬을 데이터 프레임으로 변환
dataFrame ← as.data.frame(mat)
dataFrame
class(dataFrame)

#names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여 /rename도 가능
names(dataFrame) ←names(iris[, 1:4])
#names(dataFrame) ←names(s_value)
dataFrame

rename←names(s_value)
re_dataFrame←rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
re_dataFrame
```

```r
l_value← lapply(iris[ ,1:4], mean)
s_value←sapply(iris[ ,1:4], mean)


# unlist() 리스트를 벡터로 변환
vec ← unlist(l_value)
vec
class(vec)
# matrix() 벡터를 행렬로 변환
mat ← matrix(vec,ncol=4)
mat
class(mat)
#as.data.frame() 행렬을 데이터 프레임으로 변환
dataFrame ← as.data.frame(mat)
dataFrame
class(dataFrame)
#names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여
rename←names(s_value)
re_dataFrame←rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
re_dataFrame

```

394:1 (Top Level)

Console Terminal Jobs

~/

```
> vec
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
> class(vec)
[1] "numeric"
> # matrix() 벡터를 행렬로 변환
> mat ← matrix(vec,ncol=4)
> mat
         [,1]     [,2]  [,3]     [,4]
[1,] 5.843333 3.057333 3.758 1.199333
> class(mat)
[1] "matrix" "array"
> #as.data.frame() 행렬을 데이터 프레임으로 변환
> dataFrame ← as.data.frame(mat)
> dataFrame
       V1       V2    V3       V4
1 5.843333 3.057333 3.758 1.199333
> class(dataFrame)
[1] "data.frame"
> #names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여
> rename←names(s_value)
> re_dataFrame←rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
> re_dataFrame
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.843333    3.057333        3.758    1.199333
```

이 과정을 짧게 나타내면

```r

data.frame(do.call(cbind,lapply(iris[,1:4], mean)))

class(data.frame(do.call(cbind,lapply(iris[,1:4], mean))))

```

이렇게 된다.

'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)

array array 함수 do.call list list 함수 R list R 배열 unlist함수 리스트 행렬

꾸까꾸

혼자 끄적끄적하는 블로그 입니다.