R Syntax 2: List, Matrix, and Array

Pilsung Kang
School of Industrial Management Engineering
Korea University

Handling List

Scalar	Vector	List	Matrix	Array	Factor	Data.frame

- Lists are heterogeneous
  - ✓ Element in a list can have different modes
  - ✓ List can have other structured object such as dataframe as its element
- Elements in a list are referred by their index
- Elements in a list can have their names

```
91  # Part 1-1: Data Handling (List) -------------
92
93  # Example of a list
94  listA <- list(1, 2, "a")
95  print(listA)
96  listA[[1]]
97  listA[c(1,2)]
98  names(listA)
99  names(listA) <- c("First", "Second", "Third")
100
101 listA[["Third"]]
102 listA$Third
```

Handling List

- Creating a list
  - ✓ Use list() or vector() function
    - Element names can be assigned using tags

```
Console ~/
> A <- list(name="Kang", salary = 10000, union = TRUE)
> A
$name
[1] "Kang"

$salary
[1] 10000

$union
[1] TRUE

> A$name
[1] "Kang"
>
```

```
Console ~/
> B <- list("Kang", 10000, TRUE)
> B
[[1]]
[1] "Kang"

[[2]]
[1] 10000

[[3]]
[1] TRUE

> B[[1]]
[1] "Kang"
>
```

```
Console ~/
> C <- vector(mode="list")
> C[["name"]] <- "Kang"
> C[["salary"]] <- 10000
> C[["union"]] <- TRUE
> C
$name
[1] "Kang"

$salary
[1] 10000

$union
[1] TRUE
```

Handling List

- List operations
  - ✓ List indexing
    - Three ways of accessing the elements in a list
    - list$element_name, list[["element_name"]], list[[element's index]]
    - A list is returned if [ ] is used
  - ✓ Add/remove element in a list
    - Add: use a new name
    - Remove: use NULL

```
Console ~/
> C$name
[1] "Kang"
> C[["name"]]
[1] "Kang"
> C[[1]]
[1] "Kang"
>
```

```
Console ~/
> C1 <- C[[1]]
> class(C1)
[1] "character"
> C1
[1] "Kang"
> C2 <- C[1]
> class(C2)
[1] "list"
> C2
$name
[1] "Kang"
```

```
Console ~/
> C$office <- "frontier"
> C
$name
[1] "Kang"

$salary
[1] 10000

$union
[1] TRUE

$office
[1] "frontier"
```

```
Console ~/
> C$salary <- NULL
> C
$name
[1] "Kang"

$union
[1] TRUE

$office
[1] "frontier"
```

Handling List

- List operations (cont')
  - ✓ Unlist returns a vector with a single mode values

```
Console ~/
> tmplist <- list(a = list(1:5, c("a","b","c")), b = "Z", c = NA)
> tmplist
$a
$a[[1]]
[1] 1 2 3 4 5

$a[[2]]
[1] "a" "b" "c"


$b
[1] "Z"

$c
[1] NA

> unlist(tmplist)
 a1  a2  a3  a4  a5  a6  a7  a8   b   c
"1" "2" "3" "4" "5" "a" "b" "c" "Z"  NA
> unlist(tmplist, use.names = FALSE)
 [1] "1" "2" "3" "4" "5" "a" "b" "c" "Z" NA
>
```

Handling List

- Applying functions to list
  - ✓ lapply( ) returns a list while sapply( ) returns a vector

```
Console ~/
> A <- list(1:3,25:29)
> A
[[1]]
[1] 1 2 3

[[2]]
[1] 25 26 27 28 29

> lapply(A,median)
[[1]]
[1] 2

[[2]]
[1] 27

> sapply(A,median)
[1]  2 27
>
```

Handling Matrix and Array

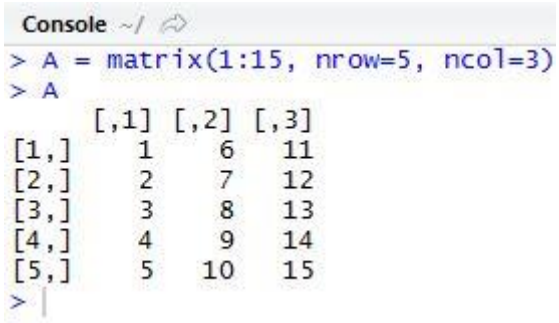
Scalar	Vector	List	Matrix	Array	Factor	Data.frame

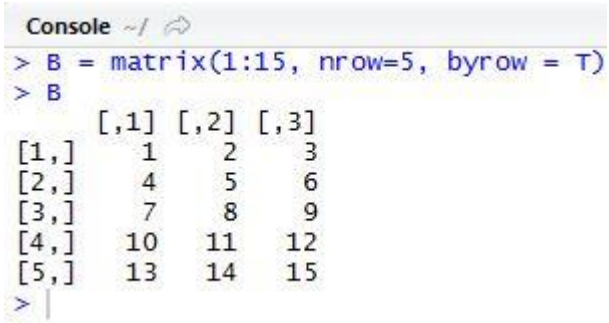
- Matrix
  - ✓ Matrix is a vector with dimensions
    - Vectors and lists can be transformed into a matrix
- Array
  - ✓ Matrix can be extended to n-dimensions
    - Indexed by multiple locations and returns subvectors

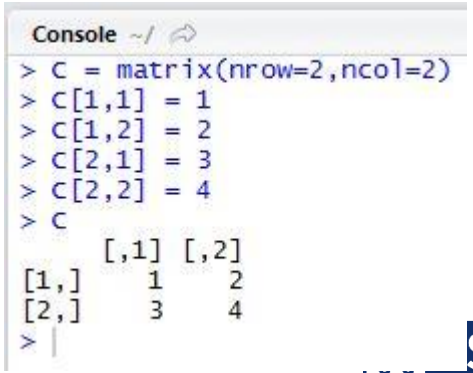
```
148 # Part 1-3: Data Handling (Matrix)
149
150 # Example of a matrix
151 A <- 1:6
152 dim(A)
153 print(A)
154
155 dim(A) <- c(2,3)
156 print(A)
157
158 B <- list(1,2,3,4,5,6)
159 print(B)
160 dim(B)
161 dim(B) <- c(2,3)
162 print(B)
163
164 D <- 1:12
165 dim(D) <- c(2,3,2)
166 print(D)
```

Handling Matrix and Array

- Features of matrix in R
  - ✓ Index begins with 1 (0 for python)
  - ✓ Column-major order
- Create a matrix: matrix( )
  - ✓ Method 1: provide all elements and assign the number of columns and rows (column first)
  - ✓ Method 2: provide all elements and assign the number of columns and rows (use row first option)
  - ✓ Method 3: Create an empty matrix and fill each element in







Handling Matrix and Array

- Matrix operations
  - ✓ Linear algebra of matrix: matrix multiplication, matrix-constant multiplication, etc.
  - ✓ Indexing and filtering

```
Console ~/
> A = matrix(1:4, nrow=2, ncol=2)
> B = matrix(seq(from=2,to=8,by=2), nrow=2, ncol=2)
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> B
     [,1] [,2]
[1,]    2    6
[2,]    4    8
> A*B # 행렬 원소간 곱셈
     [,1] [,2]
[1,]    2   18
[2,]    8   32
> A %*% B # 행렬간 곱셈
     [,1] [,2]
[1,]   14   30
[2,]   20   44
> A*3 # 행렬*상수
     [,1] [,2]
[1,]    3    9
[2,]    6   12
> A+B # 행렬간 합
     [,1] [,2]
[1,]    3    9
[2,]    6   12
>
```

```
Console ~/
> C = matrix(1:15, nrow=5, ncol=3)
> C
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> C[3,2]
[1] 8
> C[2,]
[1]  2  7 12
> C[,3]
[1] 11 12 13 14 15
> C[2:4,2:3]
     [,1] [,2]
[1,]    7   12
[2,]    8   13
[3,]    9   14
> C[-1,]
     [,1] [,2] [,3]
[1,]    2    7   12
[2,]    3    8   13
[3,]    4    9   14
[4,]    5   10   15
> C[1,] <- c(10, 11, 12)
> C
     [,1] [,2] [,3]
[1,]   10   11   12
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
>
```

Handling Matrix and Array

- Applying functions to the rows/columns of matrix
  - ✓ Use apply( ) function family: apply( ), sapply( ), tapply( ), lapply( ), etc.
    - apply(m, dimcode, f, fargs)
      - m: matrix
      - dimcode: dimension to apply (1: row, 2: column)
      - f: function
      - fargs: arguments needed to execute f

```
Console ~/
> A <- matrix(c(1:6), nrow=3, ncol=2)
> apply(A,1,mean)
[1] 2.5 3.5 4.5
> apply(A,2,mean)
[1] 2 5
>
```

Handling Matrix and Array

- Modifying matrix
  - ✓ rbind( ) & cbind( ): combine two matrices
  - ✓ rbind( ): combine two matrices with the same column names (top and bottom)
  - ✓ cbind( ): combine two matrices with the same row names (left and right)

```
Console ~/
> A <- matrix(c(1:6), nrow=3, ncol=2)
> B <- matrix(c(11:16), nrow=3, ncol=2)
> A
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> B
     [,1] [,2]
[1,]   11   14
[2,]   12   15
[3,]   13   16
>
```

```
Console ~/
> rbind(A,B)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
[4,]   11   14
[5,]   12   15
[6,]   13   16
> cbind(A,B)
     [,1] [,2] [,3] [,4]
[1,]    1    4   11   14
[2,]    2    5   12   15
[3,]    3    6   13   16
> cbind(A[,1],B[,2])
     [,1] [,2]
[1,]    1   14
[2,]    2   15
[3,]    3   16
>
```

Handling Matrix and Array

- Assign names for matrix columns/rows
  - ✓ Use colnames( ) and rownames( )

```
Console ~/
> A <- matrix(c(1:6), nrow=3, ncol=2)
> colnames(A)
NULL
> rownames(A)
NULL
> colnames(A) <- c("1st","2nd")
> colnames(A)
[1] "1st" "2nd"
> rownames(A) <- c("First","Second","Third")
> rownames(A)
[1] "First"  "Second" "Third"
> A[,"1st",drop=FALSE]
       1st
First    1
Second   2
Third    3
>
```

Handling Matrix and Array

- High dimensional array
  - ✓ Use array( ) function

```
Console ~/
> A <- matrix(c(1:15), nrow=5, ncol=3)
> B <- matrix(c(11:25), nrow=5, ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> B
     [,1] [,2] [,3]
[1,]   11   16   21
[2,]   12   17   22
[3,]   13   18   23
[4,]   14   19   24
[5,]   15   20   25
> C <- array(data=c(A,B),dim=c(3,2,2))
> C
, , 1

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

, , 2

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


ANY
questions?