Note 4. Matrix & Array

Introduction to Statistical Programming

Chanmin Kim

Department of Statistics
Sungkyunkwan University

2022 Spring

Matrix & Array

- Matrix:
  - A vector with two additional attributes (row & column).
  - Thus, vector operations works for matrices.
- Array:
  - A vector with three or more attributes ($p \geq 3$ dimensions).
  - Extension of matrices.
  - Vector operations works.

Creating Matrices

- `matrix(`*`vector, nrow, ncol, byrow=F`*`)` function:
  - Creating a (*nrow* $\times$ *ncol*) matrix from a vector.
  - *byrow=F*: Column-major order in storage (default).

```
> x <- matrix(nrow=2,ncol=3); x
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> x <- 1:12
> matrix(x,3,4)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> matrix(x,3,4,byrow=T)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

Dimension of Matrix

- `length()`: Total number of elements of the matrix.
- `dim()`:
  - It works for both matrix & array objects.
  - It returns the number of elements for each dimension.
- `nrow() & ncol()`: For matrix objects, it returns the numbers of rows and columns, respectively.

```
> A <- matrix(1:6,2,3)
> length(A)
[1] 6
> dim(A)
[1] 2 3
> nrow(A)
[1] 2
> ncol(A)
[1] 3
```

Matrix Operations

- Element-wise operations: Same as vector objects.
  - `+`, `-`, `*`, `/`, `^`, `sqrt()`, `log()`, `exp()`, etc.
- Operations for matrix:
  - %*%: Matrix multiplication.
  - `solve(`*matrix*`)`: Inverse matrix.
  - `eigen(`*matrix*`)`: eigenvalues & eigenvectors.
  - `t(`*matrix*`)`: Transpose.

Example: Matrix Operations

```
> A <- matrix(1:4,2,2); B <- matrix(2:5,2,2)
>
> A * B
     [,1] [,2]
[1,]    2   12
[2,]    6   20
>
> A %*% B
     [,1] [,2]
[1,]   11   19
[2,]   16   28
>
> solve(A)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

Example: Matrix Operations

```
> t(A)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
> eA <- eigen(A)
>
> eA$val
[1]  5.3722813 -0.3722813
>
> eA$vec
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

Matrix Indexing

- Indexing: Picking elements of the matrix or assigning new values to the matrix.

```
> A <- matrix(1:6,2,3)
> A[1,3]
[1] 5
> A[2,]
[1] 2 4 6
> A[,2]
[1] 3 4
> A[1:2,2]
[1] 3 4
> A[2,1:2]
[1] 2 4
> A[,c(1,3)]
     [,1] [,2]
[1,]    1    5
[2,]    2    6
```

Example: Matrix Indexing

```
> A[,-1]
     [,1] [,2]
[1,]    3    5
[2,]    4    6
>
> A  <- matrix(,2,3)
> A[1,] <- rep(0,3); A
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]   NA   NA   NA
> A[1,] <- c(2,3)
Error in A[1, ] <- c(2, 3) :
  number of items to replace is not a multiple of replacement length
> A[,c(1,3)] <- 2
> A
     [,1] [,2] [,3]
[1,]    2    0    2
[2,]    2   NA    2
```

Filtering

- Extracting a part of matrix satisfying a certain condition:
  - Condition for column (row) should be in the position of row (column).

```
> A <- matrix(1:20,4,5); A
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> A[,A[2,]>7]
     [,1] [,2] [,3]
[1,]    9   13   17
[2,]   10   14   18
[3,]   11   15   19
[4,]   12   16   20
> A[2,] > 7
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

Example: Filtering

```
> A[A[,3]>=10 & A[,3]<=11,1:2]
     [,1] [,2]
[1,]    2    6
[2,]    3    7
> A[,3]>=10 & A[,3]<=11
[1] FALSE  TRUE  TRUE FALSE

> x <- c(5,4,7,12)
> A[x %% 2 == 1,]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    3    7   11   15   19

> l <- which(x %% 2 == 1)
> A[l,]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    3    7   11   15   19
```

apply() Function

- apply(*A, p, f, farg*):
  - Applying functions to *p*-dimension of matrix or array.
  - *m*: matrix or array object.
  - *p*: dimension code (e.g., matrix: row=1, column=2; *p*-dimension array: 1,2,...,*p*).
  - *f*: R function.
  - *farg*: an optional set of arguments of *f*.

Example: `apply()` Function

```
> A <- matrix(c(1:3,9,7,8,7,1:8),3,5)
> A
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    9    7    3    6
[2,]    2    7    1    4    7
[3,]    3    8    2    5    8

> apply(A,2,sum)
[1]  6 24 10 12 21

> apply(A,1,mean,trim=0.2)
[1] 5.333333 4.333333 5.333333
```

rbind() & cbind() Functions

- rbind():
  - Combine rows and create a matrix (vector & vector; vector & matrix).
  - Adding rows to the matrix (reassignment).
- cbind(): Combine columns and create a matrix.

```
> x <- c(1,5,2); y <- matrix(1:6,2,3)
> rbind(x,y)
  [,1] [,2] [,3]
x    1    5    2
     1    3    5
     2    4    6
```

```
> x <- matrix(1:6,3,2)
> y <- matrix(9:1,3,3)
> cbind(x,y)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    9    6    3
[2,]    2    5    8    5    2
[3,]    3    6    7    4    1
```

Transformation into Vector or Matrix

- NOTE: If you extract a row or a column of a matrix, then the extracted object is a vector.
- `as.vector()`: Transformation of matrix or array objects into vector objects.
- `as.matrix()`: Transformation of vector or array objects into matrix objects.

```
> A <- matrix(1:6,2,3)
> x <- A[1,]; x
[1] 1 3 5

> is.vector(x)
[1] TRUE
```

Example: Transformation into Vector or Matrix

```
> x <- as.matrix(x); x
     [,1]
[1,]    1
[2,]    3
[3,]    5
> x <- as.vector(x); x
[1] 1 3 5
> A <- array(0,dim=c(2,2,2))
> as.vector(A)
[1] 0 0 0 0 0 0 0 0
> as.matrix(A)
     [,1]
[1,]    0
[2,]    0
[3,]    0
[4,]    0
[5,]    0
[6,]    0
[7,]    0
[8,]    0
```

Names of Matrix Rows and Columns

- `rownames(`*matrix*`) = `*vector*: Names of matrix rows.
- `colnames(`*matrix*`) = `*vector*: Names of matrix columns.

```
> A <- matrix(1:6,2,3); A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> rownames(A) <- c('K','J')
> colnames(A) <- 3:1

> A['J',]
3 2 1
2 4 6

> A[,'3']
K J
1 2
```