Introduction to R software

- Matrix Operations

Matrix Calculations

- Matrix Operations

Matrix

A matrix is a rectangular array with **`m rows`** and **`n columns`**.

An element in the i-th row and j-th column is denoted by **`Xij`** in text, or **`X[i,j]`** in programs, where

`i = 1,2,...,m,`

`j = 1,2,...,n`.

We consider only numerical matrices, whose elements are generally real numbers.

m x n matrix

$$A_{(mxn)} = \begin{bmatrix} a11 & a12 & . & . & . & a1n \\ a21 & a22 & . & . & . & a2n \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ am1 & am2 & . & . & . & amn \end{bmatrix}$$

Matrix

In R, a 4 × 2-matrix *X* can be created with a following command:

```
> x = matrix( nrow=4, ncol=2, data=c(1,2,3,
4,5,6,7,8) )
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Matrix

- The parameter `nrow` defines the row number of a matrix.
- The parameter `ncol` defines the column number of a matrix.
- The parameter `data` assigns specified values to the matrix elements.
- The values from the parameters are written column-wise in matrix.

Matrix

Access to single element:

```
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

One can access a single element of a matrix with `x[i,j]`

```
> x[3,2]
[1] 7
```

If we leave out either one of the subscripts, we'll get the entire row or column of the matrix, depending on which subscript we leave out.

Matrix: Access to rows, and columns

```
> x <- matrix(c(1:10), nrow=5, ncol=2)
> x
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

> x[4,]
[1] 4 9

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

Matrix: Access to rows, columns or submatrices:

```
> x <- matrix(c(1:8), nrow = 4, ncol = 2, byrow = T)
> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8

> x[1:3, 1:2]          # submatrix from rows 1-3, and columns 1-2
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

Matrix – read data from a file

When data is being read from a file, you can imbed a call to scan into a call to matrix. Suppose we have a file called **matrix.dat** with the following contents:

```
7 12 19 4
18 7 12 3
9 5 8 42
```

We could create a 3x4 matrix, read in by rows, with the following command:

```
> A <- matrix(scan('matrix.dat'), nrow=3, byrow=TRUE)
> A
     [,1] [,2] [,3] [,4]
[1,]    7   12   19    4
[2,]   18    7   12    3
[3,]    9    5    8   42
```

Matrix

In case, the data has to be entered row wise, then a 4 × 2-matrix *X*

can be created with

```
> x = matrix( nrow=4, ncol=2, data=c(1,2,3,4,
5,6,7,8), byrow = TRUE)
> x
     [,1]  [,2]
[1,]    1     2
[2,]    3     4
[3,]    5     6
[4,]    7     8
```

Transpose of matrix in R

Transpose of a matrix *X*: ***X'***

Consider the matrix

$$x = \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix}$$

```
> x = matrix( nrow=4, ncol=2, data=c(1,2,3,4,
5,6,7,8), byrow = FALSE)

> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Transpose of matrix in R – contd.

```
> xt <- t(x)
> xt
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Multiplication of a matrix with a constant

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

> 5*x
  [,1] [,2]
[1,]    5   10
[2,]   15   20
[3,]   25   30
[4,]   35   40
```

Matrix multiplication: operator %*%

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

```
> xt <- t(x)
> xt
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

```
> xtx = t(x) %*% x

> xtx
     [,1] [,2]
[1,]   84  100
[2,]  100  120
```

2x4 matrix % * % 4x2 matrix

2x2 matrix

Special matrices – constant matrix

Matrix where all m rows and n columns are filled by a single constant ‘k’.

Command : **matrix(k,m,n)**

```
> matrix(7, 4, 3)                    # k=7, m=4, n=3
     [,1] [,2] [,3]
[1,]    7    7    7
[2,]    7    7    7
[3,]    7    7    7
[4,]    7    7    7
```

Unit Matrix

Command : **matrix(1,m,n)**

```
> matrix(1, 4, 3)                    # k=1, m=4, n=3
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
[4,]    1    1    1
```

Zero Matrix

Command : **matrix(0,m,n)**

```
> matrix(0, 4, 3)                    # k=0, m=4, n=3
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
[4,]    0    0    0
```

Diagonal matrix

Command: **`diag(k,m,n)`**

First parameter k as constant/array

```
> diag(5, 3, 4)                    # k=5, m=3, n=4
     [,1] [,2] [,3] [,4]
[1,]    5    0    0    0
[2,]    0    5    0    0
[3,]    0    0    5    0
```

Identity matrix

The identity matrix is a special case of diagonal matrix where k =1, and m=n

Command: **`diag(1,n,n)`**

```
> diag(1,4,4)
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

Symmetric Matrix

```
> C <- matrix(c(2,1,5,1,3,4,5,4,-2),3,3)
> C
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]    1    3    4
[3,]    5    4   -2
> CT <- t(C)
> CT                          # Observe that C = CT
>      [,1] [,2] [,3]
> [1,]    2    1    5
> [2,]    1    3    4
> [3,]    5    4   -2
```

Note: *A symmetric matrix is a square matrix that is equal to its transpose*.

Inverse of a Matrix

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

> AI <- solve(A)
> AI
      [,1] [,2] [,3]
[1,]  1.0 -0.5  0.5
[2,] -4.0  2.5 -3.0
[3,]  2.5 -1.5  2.0
```

Inverse of a Matrix – contd.

```
> A %*% AI
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

```
> AI %*% A
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

The inverse of A is A^{-1} only when:

$$AA^{-1} = A^{-1}A = \mathbf{I}$$

Names attribute

```
> vec <- 1:6

> vec

[1] 1 2 3 4 5 6

> names(vec)

NULL

> names(vec) <- c("one", "two", "three", "four", "five", "six")

> vec

  one   two three  four  five   six

    1     2     3     4     5     6
```

Names attribute

To remove the names attribute, set it to NULL:

```
> names(vec) <- NULL
> vec
[1] 1 2 3 4 5 6
```

Dim attribute

You can transform an atomic vector into an n-dimensional array by giving it a dimensions attribute with **dim**. To do this, set the dim attribute to a numeric vector of length n. R will reorganize the elements of the vector into n dimensions. Each dimension will have as many rows (or columns, etc.) as the nth value of the dim vector.

Dim attribute

```
> vec <- 1:6
> vec
[1] 1 2 3 4 5 6

> dim(vec) <- c(2, 3)
> vec
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```