Programming in Data science
Intro to R Matrix

Asad Waqar Malik

Department of Information Systems
Faculty of Computer Science and Information Technology
University of Malaya
Malaysia
asad.malik@um.edu.my

Sept 19, 2019

Lecture Outline

1. Matrices
   - R Matrix
   - Element Access
   - Create Matrices from Vector
2. Operations on Matrix
   - Modify element
3. Exercises
   - Practise Questions

Definition – Matrices

- A matrix is a collection of data elements arranged in a two-dimensional rectangular layout. The following is an example of a matrix with 2 rows and 3 columns.

Sample

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$$

How to Create Matrix in R?

- Using the `matrix(..)` function, we will create our first matrix in R. The basic syntax for creating a matrix in R is as follows:

```
matrix(data, nrow, ncol, byrow, dimnames)
```

Where:

```
Data is the input vector, can be a list or an expression.
Nrow is the number of rows needed in our matrix.
Ncol is the number of columns in our matrix.
Byrow is a logical attribute which is FALSE by default. Setting it true will arrange the input vectors by row.
Dimnames allows you to name rows and columns in a matrix.
```

How to Create Matrix in R?

- Matrices are vectors with a dimension attributes.
- The dimension attribute is itself an integer vector of length 2 (number of rows, number of columns).

Sample Code

```
> m <- matrix(nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA

> dim(m)
[1] 2 3
```

How to Create Matrix in R?

- Matrices are constructed column-wise, so entries can be thought of starting in the "upper left" corner and running down the columns.

Sample Code

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> dim(m)
[1] 2 3
```

Direct access – Matrices

- An element at the mth row, nth column of A can be accessed by the expression A[m, n].

Try it out!

```
1  > A[2, 3]        # element at 2nd row, 3rd column
2  > A[2, ]         #  the 2nd row
3  > A[ ,3]         # the 3rd column
```

Formulating Matrices from Vector

- Use the byrow argument to specify how the matrix is filled.

Try it out!

```
mdat <- matrix(c(1, 2, 3, 11, 12, 13),
               nrow = 2,
               ncol = 3,
               byrow = TRUE)

mdat
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]   11   12   13
```

- Creating R matrix by arranging elements sequentially by column i.e. byrow = FALSE.

Defining names of columns and rows in a matrix

- In order to define rows and column names, you can create two vectors of different names, one for row and other for a column. Then, using the Dimnames attribute, you can name them appropriately:

```
1  rows = c("row1", "row2", "row3", "row4")      # ↩
       Creating vector of row names
2  cols = c("colm1", "colm2", "colm3")           #↩
       Creating character vector of column names
3  mat <- matrix(c(4:15), nrow = 4, byrow = TRUE, ↩
       dimnames = list(rows, cols) )
4  print(mat)                        #Printing our matrix
```

Defining Matrices from dimension!

- Matrices can also be created directly from vectors by adding a dimension attribute.

Try it out!

```
> dim(m) <- c(2, 5)
     [,1] [,2] [,3] [4,] [5,]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Define Matrices using `cbing(..)` and `rbing(..)`

- Matrices can be created by column-binding or row-binding with the `cbind(..)` and `rbind(..)` functions.

Try it out!

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12

> rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

How to Modify Matrix in R?

- Assign a Single Element.

```
1  > mat[2,3] <- 20  #Assigning value  to element at 2nd row and 3rd column
```

- Use of Relation Operators – Here, we use $==$ operator to replace the value that is equal to 4 with 0. Similarly, we can use $<$ operator to replace values that are less than 10 with 0.

```
1  > mat[mat == 4] <- 0     #Replacing elements that are equal to 4 with 0
```

Addition of Rows and Columns!

- Another method of modifying an R matrix is through the addition of rows and columns using the rbind() and cbind() function respectively.

```
1  > new_mat = matrix(1:12, nrow = 3, ncol = 3)
2  > new_mat
3  > cbind(new_mat, c(1,2,3))
4  > rbind(new_mat, c(1,2,3)
```

- Line 3: add a column to our matrix 'new_mat' using `cbind(..)` function.
- Line 4: add a row using the `rbind(..)` function.

Addition

- In order to perform addition on matrices in R, create two matrices 'mat1' and 'mat2' with four rows and four columns as follows

```
> mat1 <- matrix(data = 1:8, nrow = 4, ncol = 4)
> mat2 <- matrix(data = 1:16, nrow = 4, ncol = 4)
```

- In order to perform addition on A and B, we simply use '+' as follows:

```
sum <- mat1 + mat2 #Adding our two matrices
print(sum) #Printing the sum
```

Arithmetic Operations

- Similarly, subtraction and division can be performed as:

```
1  > diff <- mat1 - mat2
2  > div <- mat1 / mat2
```

- Multiplication with a constant.

```
1  > prod <- mat1*4
```

- Matrix Multiplication

```
1  > prod <- mat1*mat2
```

Exercise

- Write a R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.

Question

```
1  a<-c(1,2,3)
2  b<-c(4,5,6)
3  c<-c(7,8,9)
```

Exercise

- R has a full suite of matrix-vector and matrix-matrix arithmetic operations, let

Question

```
1  x <- matrix (1:30, 10)
2  y <- matrix (1:10, 2)
3  a <- 1:3
4  b <- 1:5
```

- Experiment with these different operations, such as x+a, x*x, y+b.

Excercise

- A square matrix A is said to be invertible if there exists a matrix B such that $AB = BA = I$, Where I is the identity matrix (1 along the diagonal, 0 elsewhere). Use R to compute the inverse of this matrix:

Excercise

```
1  set.seed (1234)
2  x <- matrix( rnorm(25), nrow=5, ncol=5)
```

Excercise

- Create a vector with 12 integers. Convert the vector to a 4*3 matrix B using matrix(). The argument byrow in matrix() is set to be FALSE by default. Please change it to TRUE and print B to see the differences.
- Please obtain the transpose matrix of B named tB. Hint, see ?t().
- Now tB is a 3×4 matrix. By the rule of matrix multiplication in algebra, can we perform tB*tB in R language? (Is a 3×4 matrix multiplied by a 3×4 allowed?) What result would we get?
- Extract a sub-matrix from B named subB . It should be a 3×3 matrix which includes the last three rows of matrix B and their corresponding columns.Hint, see example on next slide.

Sub Matrix example

```
1  Write a R program to extract the submatrix whose ↩
       rows have column value > 7 from a given matrix.
2  Sample Solution:
3  row_names = c("row1", "row2", "row3", "row4")
4  col_names = c("col1", "col2", "col3", "col4")
5  M = matrix(c(1:16), nrow = 4, byrow = TRUE, ↩
       dimnames = list(row_names, col_names))
6  print("Original Matrix:")
7  print(M)
8  result = M[M[,3] > 7,]
9  print("New submatrix:")
10 print(result)
```