R Lab 3: Matrix Algebra with R

Ikuma Ogura

August 22, 2019

Today

- Vector and matrix algebra with R
- Accessing to vector/matrix elements with R
- (Time permitting) `apply()` function

Creating Vectors and Matrices

- As covered yesterday, in R we create vectors using `c()` command.
- We can create matrices with `matrix()` command

Usage

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

where

- `data`: vector of matrix elements
- `nrow`, `ncol`: number of rows/columns
- `byrow`: if `TRUE`, the matrix is filled by rows; if `FALSE`, it is filled with columns

Creating Vectors and Matrices: Example

```
# Creating matrices
A <- matrix(data = c(1, 4, 3, 5), nrow = 2, byrow = TRUE)
B <- matrix(data = c(1, 4, 3, 5), nrow = 2, byrow = FALSE)
C <- matrix(data = c(9, 7, 6, 2, 1, 3), nrow = 2,
            byrow = TRUE)
D <- matrix(data = c(2, 4, 5, 7, 1, 2), nrow = 3,
            byrow = TRUE)
# Print
A
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    3    5
```

```
B
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    4    5
```

Creating Vectors and Matrices: Example (cont.)

```
C
```

```
##      [,1] [,2] [,3]
## [1,]    9    7    6
## [2,]    2    1    3
```

```
D
```

```
##      [,1] [,2]
## [1,]    2    4
## [2,]    5    7
## [3,]    1    2
```

Vector Operations

```r
# Creating vectors
a <- c(1, 4, 5, 3, 7)
b <- c(3, 2, 4, 7, 1)
c <- c(8, -2, -4)
# Vector operations
a + b # vector addition
```

```
## [1]  4  6  9 10  8
```

```r
3 * a # scalar product
```

```
## [1]  3 12 15  9 21
```

```r
a + 3 # !?
```

```
## [1]  4  7  8  6 10
```

Vector operations (cont.)

```r
a %*% b # dot/inner product
```

```
##      [,1]
## [1,]   59
```

```r
a * b # different from above!
```

```
## [1]  3  8 20 21  7
```

Matrix operations

```
A - B # matrix addition/subtraction
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]   -1    0
```

```
3 * C # scalar product
```

```
##      [,1] [,2] [,3]
## [1,]   27   21   18
## [2,]    6    3    9
```

```
A + 2
```

```
##      [,1] [,2]
## [1,]    3    6
## [2,]    5    7
```

Matrix operations (cont.)

```r
A %*% B # matrix product
```

```
##      [,1] [,2]
## [1,]   17   23
## [2,]   23   34
```

```r
A * B # different from above!
```

```
##      [,1] [,2]
## [1,]    1   12
## [2,]   12   25
```

Matrix operations (cont.)

```
t(C) %*% B # t() to transpose matrices
```

```
##      [,1] [,2]
## [1,]   17   37
## [2,]   11   26
## [3,]   18   33
```

```
C %*% c # vectors are treated as the k by 1 matrices
```

```
##      [,1]
## [1,]   34
## [2,]    2
```

```
t(c) %*% D
```

```
##      [,1] [,2]
## [1,]    2   10
```

Determinant & Inverse

```r
det(A)
```

```
## [1] -7
```

```r
solve(B)
```

```
##            [,1]       [,2]
## [1,] -0.7142857  0.4285714
## [2,]  0.5714286 -0.1428571
```

Solving System of Equations

- We can also use `solve()` command
- Example: Let's solve the following system of equations with R.

$$x + 3y = 7$$
$$2x + 5y = 10$$

```r
coefs <- matrix(c(1, 3, 2, 5), nrow = 2, byrow = TRUE)
rhs <- c(7, 10)
solve(coefs, rhs)
```

```
## [1] -5  4
```

Vector/Matrix Operations: Summary

Command	Meaning
`+`	Summation
`-`	Subtraction
`*`	Element-wise product (Adamar product)
`%*%`	Matrix/Vector product
`length()`	(For vectors) Vector length
`dim()`	(For matrices) Matrix dimension
`t()`	Transpose
`det()`	Determinant
`solve()`	Inverse
`diag(A)`	($\boldsymbol{A}$ is a square matrix) Extract diagonal elements
`diag(k)`	(k is a scalar) Create a $k \times k$ identity matrix
`eigen()`	Compute eigenvalues and eigenvectors

Accessing to Vector/Matrix Elements

- For vectors: `vectorname[i]` extracts the *i*th element of the vector
  - We can put in a vector within `[]` to extract multiple elements
  - If we specify negative numbers within `[]`, R deletes corresponding elements
- For matrices: `matrixname[i, j]` extracts the element in *i*th row and *j*th column
  - `matrixname[i,]` extracts all the elements in *i*th row as a vector
  - `matrixname[, j]` extracts all the elements in *j*th column as a vector

Accessing to Vector/Matrix Elements: Example

```
b[3]
```

```
## [1] 4
```

```
a[c(2, 4)]
```

```
## [1] 4 3
```

```
b[c(-1, -5)]
```

```
## [1] 2 4 7
```

```
C[2, 1]
```

```
## [1] 2
```

```
D[-2,]
```

```
##      [,1] [,2]
## [1,]    2    4
## [2,]    1    2
```

Accessing to Vector/Matrix Elements: Example (cont.)

```r
B[1, 1] <- 9
B
```

```
##      [,1] [,2]
## [1,]    9    3
## [2,]    4    5
```

```r
C[, 2] <- c(8, 3)
C
```

```
##      [,1] [,2] [,3]
## [1,]    9    8    6
## [2,]    2    3    3
```

Excersices!

For the following matrix

$$\boldsymbol{A} = \begin{pmatrix} 7 & -3 & 0 \\ -2 & 6 & 1 \\ 0 & -5 & 6 \end{pmatrix},$$

1. find the determinant
2. calculate the inverse
3. replace the third row to $(-4, 2, -1)$ and recompute the determinant
4. delete the first row and third column and find its inverse matrix

apply()

- We use `apply()` function when we apply a command/function to each row/column

Usage

```
apply(X, MARGIN, FUN...)
```

where

- `X`: a matrix we apply a function
- `MARGIN`: set 1 when we want to apply a function to each row; set 2 when we apply the function to each column
- `FUN`: function to apply

apply(): Example

```r
apply(A, 1, sum)
```

```
## [1] 5 8
```

```r
apply(C, 2, prod)
```

```
## [1] 18 24 18
```

Tommorrow

- Introduction to
  - loading data into R
  - data preprocessing with R
  - summarizing data with R