Matrices and Arrays

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Vector: 1-dimensional object
Matrix: 2-dimensional object
Array: n-dimensional object

Making Matrix and Array

- `dim()`
- `matrix()`
- `array()`
  `matrix()` and `array()` are commonly used.

```
x <- 1:8

# 2x4
dim(x) <- c(2, 4)
x
```

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

```
x <- 1:8

# 2x2x2
dim(x) <- c(2, 2, 2)
x
```

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

```
# 3x4
matrix(1:12, nrow = 3, ncol = 4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# 2x6
matrix(1:12, nrow = 2, ncol = 6,
       byrow = TRUE, # fill elements from left to right
       dimnames = list(c("A", "B"), c("col1", "col2", "col3", "col4", "col5", "col6"))) # labels
```

```
##   col1 col2 col3 col4 col5 col6
## A    1    2    3    4    5    6
## B    7    8    9   10   11   12
```

```
# 2x2x2
array(1:12, dim = c(2, 2, 3))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
##
## , , 3
##
##      [,1] [,2]
## [1,]    9   11
## [2,]   10   12
```

They will give an error if the number of elements in vector doesn't match the number of cells.

Extract Elements of Matrices

- `name_of_matrix[row_index, col_index]`: specific cell.
- `name_of_matrix[row_index,]`: specific row.
- `name_of_matrix[, col_index]`: specific column.

```
mat <- matrix(1:12, nrow = 3, ncol = 4)
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# first row, second column
mat[1, 2]
```

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

```
# second row
mat[2,]
```

```
## [1]  2  5  8 11
```

```
# third column
mat[, 3]
```

```
## [1] 7 8 9
```

```
# odd rows, even columns
mat[seq(1, 3, by = 2), seq(2, 4, by = 2)]
```

```
##      [,1] [,2]
## [1,]    4   10
## [2,]    6   12
```

```
# no second row, no first column
mat[-2, -1]
```

```
##      [,1] [,2] [,3]
## [1,]    4    7   10
## [2,]    6    9   12
```

Binding Rows and Columns

- `rbind()`: binds rows.
- `cbind()`: binds columns.

```r
mat2 <- matrix(1:8, nrow = 2, ncol = 4)
mat2
```

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

```r
# combine two matrices by row
rbind(mat, mat2)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
## [4,]    1    3    5    7
## [5,]    2    4    6    8
```

```r
mat3 <- matrix(1:6, nrow = 3, ncol =2)
mat3
```

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

```r
# combine two matrices by column
cbind(mat, mat3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7   10    1    4
## [2,]    2    5    8   11    2    5
## [3,]    3    6    9   12    3    6
```

Mathematical Operation on Matrices

1. Transpose X^T: `t()`
   $X_{n\times m} \Rightarrow X_{m\times n}$

```r
# mat is 3x4 matrix
# its transpose is 4x3 matrix
t(mat)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

2. Addition
   Matrices must be the same dimension.
   $\begin{bmatrix}1 & 3 & 5\\2 & 4 & 6\end{bmatrix} + \begin{bmatrix}7 & 9 & 8\\8 & 7 & 9\end{bmatrix} = \begin{bmatrix}8 & 12 & 13\\10 & 11 & 15\end{bmatrix}$

```
A <- matrix(1:6, 2, 3)
B <- matrix(7:9, 2, 3)

A + B
```

```
##      [,1] [,2] [,3]
## [1,]    8   12   13
## [2,]   10   11   15
```

3. Scalar Multiplication

```
1/2 * A
```

```
##      [,1] [,2] [,3]
## [1,]  0.5  1.5  2.5
## [2,]  1.0  2.0  3.0
```

4. Matrices Multiplication: `%*%` $X_{a\times b} \times X_{b\times c} = X_{a\times c}$

```
C <- matrix(7:9, 3, 2)
A %*% C
```

```
##      [,1] [,2]
## [1,]   76   76
## [2,]  100  100
```

5. Multiplication by Elements: `*` Matrices must be the same dimension.

```
A * A
```

```
##      [,1] [,2] [,3]
## [1,]    1    9   25
## [2,]    4   16   36
```

6. Cross Product X^TY or XY^T: `crossprod()`, `tcrossprod()`

```
# t(A) %*% A
crossprod(A, A)
```

```
##      [,1] [,2] [,3]
## [1,]    5   11   17
## [2,]   11   25   39
## [3,]   17   39   61
```

```r
# A %*% t(A)
tcrossprod(A, A)
```

```
##      [,1] [,2]
## [1,]   35   44
## [2,]   44   56
```

7. Linear Combination
   Multiplication by vector and matrix.

```r
X <- matrix(1:12, 3, 4)
a <- seq(0.25, 1, by = 0.25)
b <- 1:3

X %*% a
```

```
##      [,1]
## [1,] 17.5
## [2,] 20.0
## [3,] 22.5
```

```r
b %*% X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   14   32   50   68
```

Other Mathematical Functions of Matrices

1. `det()`: Find determinant.

```r
X2 <- rbind(c(-1,4,2), c(3,-2,1), c(1,1,1))
det(X2)
```

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

2. `diag()`: Extract or replace the diagonal elements.

```r
# 3x3 identity matrix
diag(3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```r
# 3x3 diagonal elements 1, 2, 3
diag(1:3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    0
## [3,]    0    0    3
```

```
# change diagonal elements to 5
I <- diag(3)
diag(I) <- 5
I
```

```
##      [,1] [,2] [,3]
## [1,]    5    0    0
## [2,]    0    5    0
## [3,]    0    0    5
```

3. `solve()`: Solve the system of equations.

```
X3 <- rbind(c(-1,4,2), c(3,-2,1), c(1,1,1))
c <- c(1,2,-1)

solve(X3, c)
```

```
## [1] -3 -3  5
```

4. `eigen()`: Eigen decomposition.

```
v <- matrix(c(8, 4, 1, 5), 2, 2)
v
```

```
##      [,1] [,2]
## [1,]    8    1
## [2,]    4    5
```

```
eigen(v)
```

```
## eigen() decomposition
## $values
## [1] 9 4
##
## $vectors
##           [,1]       [,2]
## [1,] 0.7071068 -0.2425356
## [2,] 0.7071068  0.9701425
```

5. `svd()`: singular value decomposition.
6. `qr()`: QR decomposition.
7. `chol()`: Choleski decomposition.

Exercises

1) Given the following vector `hp`, how would you create the matrix displayed below?

```
# vector of names
hp <- c("Harry", "Potter", "Ron", "Weasley", "Hermione", "Granger")
```

```
     [,1]     [,2]
[1,] "Harry"  "Weasley"
[2,] "Potter" "Hermione"
[3,] "Ron"    "Granger"
```

```
matrix(hp, 3, 2)
```

```
##      [,1]     [,2]
## [1,] "Harry"  "Weasley"
## [2,] "Potter" "Hermione"
## [3,] "Ron"    "Granger"
```

2) Given the following vector `sw`, how would you create the matrix displayed below?

```
# vector of names
sw <- c("Luke", "Skywalker", "Leia", "Organa", "Han", "Solo")
```

```
     [,1]   [,2]
[1,] "Luke" "Skywalker"
[2,] "Leia" "Organa"
[3,] "Han"  "Solo"
```

```
matrix(sw, 3, 2, byrow = TRUE)
```

```
##      [,1]   [,2]
## [1,] "Luke" "Skywalker"
## [2,] "Leia" "Organa"
## [3,] "Han"  "Solo"
```

3) Consider the following vectors `a1`, `a2`, `a3`:

```
a1 <- c(2, 3, 6, 7, 10)
a2 <- c(1.88, 2.05, 1.70, 1.60, 1.78)
a3 <- c(80, 90, 70, 50, 75)
```

Column-bind the vectors a1, a2, a3 to form the following matrix `M`:

```
#>   a1   a2 a3
#> 1  2 1.88 80
#> 2  3 2.05 90
#> 3  6 1.70 70
#> 4  7 1.60 50
#> 5 10 1.78 75
```

```
M <- cbind(a1, a2, a3)
rownames(M) <- 1:5
M
```

```
##   a1   a2 a3
## 1  2 1.88 80
## 2  3 2.05 90
## 3  6 1.70 70
## 4  7 1.60 50
## 5 10 1.78 75
```

4) Consider the following vectors `b1`, `b2`, `b3`:

```
b1 <- c(1, 4, 5, 8, 9)
b2 <- c(1.22, 1.05, 3.60, 0.40, 2.54)
b3 <- c(20, 40, 30, 80, 100)
```

Row-bind the vectors b1, b2, b3 to form the following matrix `M2`:

```
#>        1     2    3    4      5
#> b1  1.00  4.00  5.0  8.0   9.00
#> b2  1.22  1.05  3.6  0.4   2.54
#> b3 20.00 40.00 30.0 80.0 100.00
```

```
M2 <- rbind(b1, b2, b3)
colnames(M2) <- 1:5
M2
```

```
##        1     2    3    4      5
## b1  1.00  4.00  5.0  8.0   9.00
## b2  1.22  1.05  3.6  0.4   2.54
## b3 20.00 40.00 30.0 80.0 100.00
```

5) With matrices `M` and `M2` created above, use the matrix-multiplication operator `%*%` and the transpose function `t()` to compute the matrix products:

a. MM2

```
M %*% M2
```

```
##          1        2        3        4        5
## 1 1604.294 3209.974 2416.768 6416.752 8022.775
## 2 1805.501 3614.153 2722.380 7224.820 9032.207
## 3 1408.074 2825.785 2136.120 5648.680 7058.318
## 4 1008.952 2029.680 1540.760 4056.640 5067.064
## 5 1512.172 3041.869 2306.408 6080.712 7594.521
```

b. M2M

```
M2 %*% M
```

```
##         a1       a2      a3
## b1  190.00  47.4000  1865.0
## b2   55.39  15.7273   654.6
## b3 1900.00 476.6000 18800.0
```

c. $M^T M2^T$

```
t(M) %*% t(M2)
```

```
##        b1       b2      b3
## a1  190.0  55.3900  1900.0
## a2   47.4  15.7273   476.6
## a3 1865.0 654.6000 18800.0
```

d. $M2^T M^T$

```
t(M2) %*% t(M)
```

```
##          1        2        3        4        5
## 1 1604.294 1805.501 1408.074 1008.952 1512.172
## 2 3209.974 3614.153 2825.785 2029.680 3041.869
## 3 2416.768 2722.380 2136.120 1540.760 2306.408
## 4 6416.752 7224.820 5648.680 4056.640 6080.712
## 5 8022.775 9032.207 7058.318 5067.064 7594.521
```