

03 Linear Algebra

03 Vectors

Watch the well-made videos about linear algebra by “3Blue1Brown” on Youtube! (link attached in Canvas)

- Multiplication of vector with a scalar: $c\mathbf{u} = (cu_1, cu_2, \dots)$
 - Example: $\mathbf{u} = (1,2,3), 3\mathbf{u} = (3 \cdot 1, 3 \cdot 2, 3 \cdot 3) = (3,6,9)$
- Addition of vectors: $\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2, \dots)$
 - Example: $\mathbf{u} = (1,2,3), \mathbf{v} = (4,5,6), \mathbf{u} + \mathbf{v} = (1 + 4, 2 + 5, 3 + 6) = (5,7,9)$
- Combined multiplication and addition: $c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v}$
 - Example: $2(\mathbf{u} + \mathbf{v}) = 2\mathbf{u} + 2\mathbf{v} = (2,4,6) + (8,10,12) = (10,14,18)$
- The length (magnitude) of a vector: $|\mathbf{u}| = \sqrt{\mathbf{u} \cdot \mathbf{u}} = \sqrt{u_1^2 + u_2^2 + \dots}$
 - Example: $\mathbf{u} = (1,2,3), |\mathbf{u}| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$
- Basis vectors in a two-dimensional space: $\hat{i} = (1,0), \hat{j} = (0,1)$
 - Any vector $\mathbf{v} = (a, b)$ in two-dimensional space may be uniquely written as
$$\mathbf{v} = a\hat{i} + b\hat{j}$$

03 Matrices

Watch the well-made videos about linear algebra by “3Blue1Brown” on Youtube! (link attached in Canvas)

- Numerically, matrices provides information of linear transformation.
- In computer science, matrices are commonly perceived as tables of data.
- When doing matrix calculation, it makes more sense to think about linear transformation
- A matrix looks like a table of numbers. An m -by- n matrix A has m rows and n columns
- The elements are indexed as a_{ij} = the element on row i , column j

- Example: A is a 2-by-3 matrix: $A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 8 & 6 \end{bmatrix}$,

$$\text{then } a_{11} = 2, a_{12} = 3, a_{13} = 4, a_{21} = 5, a_{22} = 8, a_{23} = 6$$

- Transpose:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, A^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$$

- Example: $A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 8 & 6 \end{bmatrix}, A^T = \begin{bmatrix} 2 & 5 \\ 3 & 8 \\ 4 & 6 \end{bmatrix}$

03 Matrices

Watch the well-made videos about linear algebra by “3Blue1Brown” on Youtube! (link attached in Canvas)

- Addition: add element-by-element (The number of rows and columns has to match up)

- $\begin{bmatrix} 3 & 2 \\ 4 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 5 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 9 & 1 \end{bmatrix}$

- Multiplication with scalar:

- $2 \begin{bmatrix} 3 & 2 \\ 4 & 0 \end{bmatrix} = \begin{bmatrix} 6 & 4 \\ 8 & 0 \end{bmatrix}$

03 Matrices

Watch the well-made videos about linear algebra by “3Blue1Brown” on Youtube! (link attached in Canvas)

- A matrix can be multiplied with a column vector. The result is a new vector, where each element is the scalar product of a matrix row and the vector. (Note: the number of columns in A must be equal to the number of elements (rows) in x)

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

$$A\mathbf{r} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} a_{11}r_1 + a_{12}r_2 \\ a_{21}r_1 + a_{22}r_2 \end{bmatrix}$$

- Example:

$$\begin{bmatrix} 5 & 2 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \cdot 1 + 2 \cdot 4 \\ 4 \cdot 1 + 1 \cdot 4 \end{bmatrix} = \begin{bmatrix} 13 \\ 8 \end{bmatrix}$$

03 Matrices

I recommend you to watch the well-made videos about linear algebra by “3Blue1Brown” on Youtube! (link attached in Canvas)

- Matrix multiplication with a vector : movement of the vector after linear transformation

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x \begin{bmatrix} 1 \\ 2 \end{bmatrix} + y \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1x + 2y \\ 2x + 1y \end{bmatrix}$$

Where the original vector ends up after the linear transformation

03 Matrices

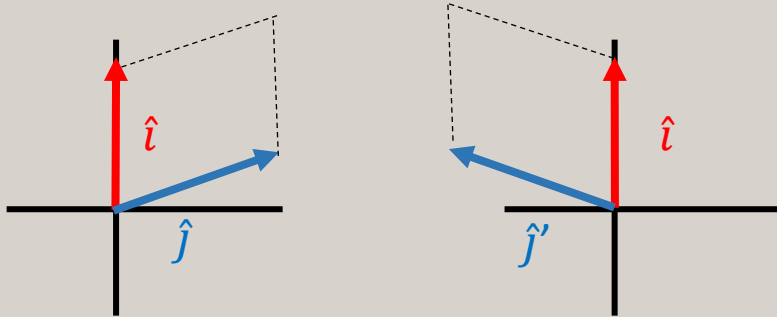
I recommend you to watch the well-made videos about linear algebra by “*3Blue1Brown*” on Youtube! (link attached in Canvas)

- Matrix multiplication with a matrix : two separate steps of linear transformation: so the **sequence** matters!
- $AB \neq BA$
- $A + B = B + A$
- $A(B + C) = AB + AC$
- $(B + C)A = BA + CA$
- $A(BC) = (AB)C = ABC$

03 Matrices

I recommend you to watch the well-made videos about linear algebra by “3Blue1Brown” on Youtube! (link attached in Canvas)

- Determinant: How much is the **area** or **space** squeezed after transformation
 - Two-dimensional system: $\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$
 - Negative determinant indicates orientation flipping:



03 Linear equation systems

- Matrix can also be used to represent the coefficients of linear equation systems:

$$\begin{cases} 4x - 7y = 8 \\ 2x + 3y = 1 \end{cases}$$

$$A = \begin{bmatrix} 4 & -7 \\ 2 & 3 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} 8 \\ 1 \end{bmatrix}$$

Coefficients Variables Constants
(unknown)

$$A\mathbf{u} = \mathbf{v}$$

The inverse of a A , A^{-1} can be used to solve linear equation systems. But it is out of the scope of this course.

03 Modeling dynamical systems in R

- A simple predator-prey dynamics:

In a California redwood forest, the spotted owl is the primary predator of the wood rat. Suppose that the following system models the interaction between owls and rats. Here, $o(t)$ represents the owl population at time t and $r(t)$ represents the rat population at time t :

$$\begin{cases} o(t+1) = 0.5o(t) + 0.3r(t) \\ r(t+1) = -0.2o(t) + 1.2r(t) \end{cases}$$

(1) Let $\vec{x}(t) = \begin{bmatrix} o(t) \\ r(t) \end{bmatrix}$. Find a matrix A so that $\vec{x}(t+1) = A\vec{x}(t)$

(2) Suppose there are currently 2000 owls and 1000 rats. How many owls and rats are there next year? The year after? In t years? (Exercise in R)

04 Linear Algebra in R

04 Vectors in R

```
> x <- c(2,1,4)
```

```
> x
```

```
[1] 2 1 4
```

```
> 3*x
```

```
[1] 6 3 12
```

```
> y <- c(1,0,1)
```

```
> y
```

```
[1] 1 0 1
```

Ignoring the difference
between row and
column vectors

```
> x+y
```

```
[1] 3 1 5
```

```
> 2*(x+y)
```

```
[1] 6 2 10
```

```
> 2*x
```

```
[1] 4 2 8
```

```
> 2*y
```

```
[1] 2 0 2
```

04 Vectors in R

```
> x <- c(2,1,4)
```

```
> y <- c(1,0,1)
```

```
> x*y
```

```
[1] 2 0 4
```

← Multiplying element-by-element

```
> sum(x*y)
```

```
[1] 6
```

← Scalar product
(or $x \%*\% y$)

```
> sqrt(sum(x*x))
```

```
[1] 4.582576
```

← The length of x ($|x|$)

04 R arrays

- An array in R is similar to a vector, but it can have more than one index, so it can be 'multi-dimensional'.

```
> x <- array(dim=c(2,3))  
> x  
      [,1] [,2] [,3]  
[1,]   NA   NA   NA  
[2,]   NA   NA   NA
```

An R array has a dim attribute that defines its dimensions. This creates a 2-by-3 array (filled with NA).

```
> x <- array(c(3,4,5,57), dim=c(2,2))  
> x  
      [,1] [,2]  
[1,]    3    5  
[2,]    4   57
```

creates a 2-by-2 array filled with the given values filled in column by column

04 Arrays

```
> x <- array(c(3,4,5,57), dim=c(2,2))
> x
      [,1] [,2]
[1,]    3    5
[2,]    4   57

> x[1,2]      # [row index, column index]
[1] 5

> x[2,2]
[1] 57

> x[1:2,2]     # [rows 1 and 2, column 2]
[1] 5 57       # the output is a vector!

> x[2,2:1]     # row 2 in reverse order
[1] 57 4

> x[1,]        # leave out an index to get all elements
[1] 3 5         # This is a vector, not an array, somewhat inconsistently
```

I want to extract a single element from the array x, then I need to give the index

You can also give a range, here is from row 1 to row 2 of the column 2

04 Array dimensions

- The dimensions of general arrays are set in the dim attribute

```
> A <- 1:6
```

```
> A
```

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

```
> is.array(A)
```

```
[1] FALSE
```

```
> dim(A)
```

```
NULL
```

This is a vector; they have length but no other dimensions

```
> dim(A) <- 6
```

```
> A
```

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

```
> is.matrix(A)
```

```
[1] FALSE
```

This is a one-dimensional array

```
> dim(A) <- c(1,6)
```

```
> A
```

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

```
> is.matrix(A)
```

```
[1] TRUE
```

A matrix is a two-dimensional array

```
> dim(A) <- c(6,1)
```

```
> A
```

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

```
> dim(A) <- c(2,3)
```

```
> A
```

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

```
> t(A) # transpose!
```

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


04 Matrices

- A Matrix is a two-dimensional array
- Some functions and operators only apply to matrices, not arrays in general:
t(), nrow(), ncol(), %*%, diag(), solve(), eigen()

```
> a <- 1:4
> dim(a) <- c(2,2)

> a
      [,1] [,2]
[1,]    1    3
[2,]    2    4

> is.matrix(a)
[1] TRUE
```

From vector to a matrix

```
> b <- array(1:4, c(2,2))

> b
      [,1] [,2]
[1,]    1    3
[2,]    2    4

> is.matrix(b)
[1] TRUE
```

From array to a matrix

```
> c <- matrix(1:4,2,2)

> c
      [,1] [,2]
[1,]    1    3
[2,]    2    4

> is.matrix(c)
[1] TRUE
```

04 Creating matrices from other matrices or vectors

- cbind: ('column-bind') Merge horizontally, side-by side.
- rbind: ('row-bind') Merge vertically, one on top of the other

```
> cbind(1:4,12:15)
```

```
      [,1] [,2]  
[1,]     1  12  
[2,]     2  13  
[3,]     3  14  
[4,]     4  15
```

```
> rbind(1:4,12:15)
```

```
      [,1] [,2] [,3] [,4]  
[1,]     1     2     3     4  
[2,]    12    13    14    15
```

```
> rbind(1:4,7:8)
```

```
      [,1] [,2] [,3] [,4]  
[1,]     1     2     3     4  
[2,]     7     8     7     8
```

recycling!

```
> rbind(12:15,1:3)
```

```
      [,1] [,2] [,3] [,4]  
[1,]    12    13    14    15  
[2,]     1     2     3     1
```

If two vectors are of different lengths

Warning message:

In rbind(12:15, 1:3) : number of columns of result is not a multiple of vector length (arg 2)

```
> v <- 5:2
```

```
> v
```

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

```
> v <- rbind(v) # easy way to convert a  
vector to a matrix (with only one row)
```

```
> v
```

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

04 Matrix algebra

- Addition: The number of rows and columns must match up because it's adding element-by element

```
> A <-  
matrix(c(3,7,1,8),2,2)  
> A  
      [,1] [,2]  
[1,]    3    1  
[2,]    7    8  
> B <- matrix(c(0,2,2,-  
1),2,2)  
> B  
      [,1] [,2]  
[1,]    0    2  
[2,]    2   -1
```

```
> A+B  
      [,1] [,2]  
[1,]    3    3  
[2,]    9    7
```

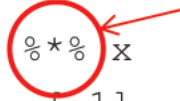
04 Matrix algebra

- Multiplication: %*%

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

x <- matrix(c(2,1),2,1)
> x
      [,1]
[1,]     2
[2,]     1

> A %*% x
      [,1]
[1,]     4
[2,]     6
```



04 Eigenvalues and Eigenvectors

```
> A <- matrix(c(1,-1,2,8),2,2)
> E <- eigen(A)
> E$values
[1] 7.701562 1.298438
      eigenvalue 1 eigenvalue 2
> E$vectors
      [,1] [,2]
[1,] -0.2859743 -0.9890494
[2,] -0.9582373 -0.1475849
      eigenvector 1 eigenvector 2
> v1 <- E$vectors[,1]
> v1
[1] -0.2859743 -0.9582373
> v2 <- E$vectors[,2]
> v2
[1] -0.9890494 -0.1475849
```

← The return value is a list with two components

Testing whether $Av_1 = \lambda v_1$

```
> Av1 <- A %*% v1
> Av1
[,1]
[1,] -2.202449
[2,] -7.379924
> Av1 / v1
[,1]
[1,] 7.701562
[2,] 7.701562
```

Some useful matrix operations in R

<u>Operator or Function</u>	<u>Description</u>
A * B	Element-wise multiplication
A %*% B	Matrix multiplication
t(A)	Transpose
diag(x)	Creates diagonal matrix with elements of x on the diagonal
diag(A)	Returns a vector containing the elements of the diagonal of A
diag(k)	k is an integer. Creates a k x k identity matrix (1 on diagonal, 0 otherwise)
solve(A, b)	Returns vector x in the equation b = Ax (i.e., A⁻¹b)
solve(A)	Inverse of A where A is a square matrix.
y<-eigen(A)	y\$values are the eigenvalues of A , y\$vectors are the eigenvectors of A
cbind(A,B,...)	Combine matrices(vectors) horizontally. Returns a matrix.
rbind(A,B,...)	Combine matrices(vectors) vertically. Returns a matrix.
rowMeans(A)	Returns vector of row means.
rowSums(A)	Returns vector of row sums.
colMeans(A)	Returns vector of column means.
colSums(A)	Returns vector of column sums.
sum(A)	Returns the sum of all elements of A