# *Vectors and Matrices in R*

*Arnab Maity*

*NCSU Department of Statistics ~ 5240 SAS Hall ~ 919-515-1937 ~ amaity[at]ncsu.edu*

*Contents*

## *Introduction*

Let us consider the first five rows and the first two columns of the `iris` dataset in R.

```
iris[1:5, 1:2]
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
```

The first column containing five numbers is an example of a vector. The entire table with five rows and two columns is an example of a $5 \times 2$ matrix. These data structures are very common is both multivariate and longitudinal data analysis.

## *Vectors*

A vector is an **array of numbers**. Specifically, we will write

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

and call it a *column vector*. We often write $\boldsymbol{x} \in \mathbb{R}^p$. Similarly, a *row vector* is written as

$$\boldsymbol{x}^T = (x_1, x_2, \ldots, x_p).$$

Note that the notation $\boldsymbol{x}^T$ denotes "transpose" of $\boldsymbol{x}$[1]

In our `iris` data example above, consider the first column of the table (corresponding to `Sepal.Length`). This is an example of a $6 \times 1$ (column) vector

$$\boldsymbol{x} = \begin{pmatrix} 5.1 \\ 4.9 \\ 4.7 \\ 4.6 \\ 5.0 \end{pmatrix}.$$

To create this vector in R, we can use the command:

```
x = c(5.1, 4.9, 4.7, 4.6, 5)
x
```

[1] **Note:** In this course, **we will always take a vector as a column vector by convention, and will always use the transpose to denote a row vector**. Thus the statement "$\boldsymbol{a}$ is a vector" will imply that "$\boldsymbol{a}$ is a *column* vector."

```
## [1] 5.1 4.9 4.7 4.6 5.0
```

Even though R prints the vector x using a single line but it still considers x as a column vector. To see this, try to view x in a matrix form using as.matrix():

```
as.matrix(x)
```

```
##      [,1]
## [1,]  5.1
## [2,]  4.9
## [3,]  4.7
## [4,]  4.6
## [5,]  5.0
```

If we take the transpose using t() function, we obtain a row vector:[2]

[2] What happens when you take transpose of a row vetor? Try it here.

```
t(x)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  5.1  4.9  4.7  4.6    5
```

*Addition and subtraction of two vectors*

For two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$, the sum is defined as[3]

[3] Similarly, the difference is defined as

$$\boldsymbol{a} - \boldsymbol{b} = \begin{pmatrix} a_1 - b_1 \\ a_2 - b_2 \\ \vdots \\ a_p - b_p \end{pmatrix}$$

$$\boldsymbol{a} + \boldsymbol{b} = \begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_p + b_p \end{pmatrix},$$

that is, a vector of same dimension as of $\boldsymbol{a}$ and $\boldsymbol{b}$, where each element is the sum of corresponding elements of $\boldsymbol{a}$ and $\boldsymbol{b}$.

Consider the two vectors as follows.[4]

[4] Note that to add (or subtract) $\boldsymbol{a}$ and $\boldsymbol{b}$, the two vectors have to have the same number of elements.

```
a = c(5.1, 4.9, 4.7, 4.6, 5)
b = c(3.5, 3, 3.2, 3.1, 3.6)
```

Their sum is:

```
a + b
```

```
## [1] 8.6 7.9 7.9 7.7 8.6
```

Their difference is:

```
a - b
```

```
## [1] 1.6 1.9 1.5 1.5 1.4
```

*Vector multiplication*

A vector $\boldsymbol{a}$ can be multiplied by a scalar $k$ by simply multiplying each element of $\boldsymbol{a}$ by $k$:

$$k\boldsymbol{a} = k \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} ka_1 \\ ka_2 \\ \vdots \\ ka_p \end{pmatrix}$$

In R, we can use the $*$ operator:[5]

[5] We can similarly divide a vector by a scalar by using the / operator.

```
a
```

```
## [1] 5.1 4.9 4.7 4.6 5.0
```

```
2 * a
```

```
## [1] 10.2  9.8  9.4  9.2 10.0
```

Multiplication between two vectors is a little more involved. Here we need to define the *inner product* of two vectors. For two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$, the inner product is defined as:

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^T \boldsymbol{b} = \begin{pmatrix} a_1 & a_2 & \ldots & a_p \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix} = a_1 b_1 + a_2 b_2 + \ldots + a_p b_p = \sum_{j=1}^{p} a_j b_j.$$

Note that *the result is a scalar*.

As an example, suppose $\boldsymbol{a}^T = (1, 0, 2, 5)$ and $\boldsymbol{b} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 6 \end{pmatrix}$. Then we have

$$\boldsymbol{a}^T \boldsymbol{b} = \begin{pmatrix} 1 & 0 & 2 & 5 \end{pmatrix} \times \begin{pmatrix} 2 \\ 3 \\ 1 \\ 6 \end{pmatrix} = (1 \times 2) + (0 \times 3) + (2 \times 1) + (5 \times 6) = 34$$

In R, we can use the `%*%` operator to compute the inner product (or matrix multiplication in general). In this example[6]

```
a <- c(1, 0, 2, 5)
b <- c(2, 3, 1, 6)

t(a) %*% b
```

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

[6] **Note:** Be careful to use `%*%`. Be sure to put the `%` signs properly. Just using `*` without the `%` signs would give you elementwise product:

$$\boldsymbol{a} * \boldsymbol{b} = \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_p b_p \end{pmatrix}.$$

In matrix algebra this is referred to as *Hadamard product*.

### *Norm/Length of a Vector*

The *length* of a vector is defined as its distance from the vector $\mathbf{0}$, the origin. It is defined as

$$||\boldsymbol{x}|| = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{1/2} = \left( x_1^2 + \ldots + x_p^2 \right)^{1/2}.$$

In other words, the length of a vector $\boldsymbol{x}$ is the square root of the inner product of $\boldsymbol{x}$ with itself.

Try to compute length of `a` defined before:[7]

```
sqrt(sum(a^2))
```

```
## [1] 5.477226
```

[7] Another way to compute this is to use `sqrt(t(a) %*% a)`

If a vector has norm one (unity), that is, $||\boldsymbol{x}|| = 1$, then the vector is called *unit vector*.

### *Orthogonal vectors*

Two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ (that have the same number of elements) are said to be *orthogonal* if $\boldsymbol{a}^T \boldsymbol{b} = 0$. In other words, two vectors are orthogonal if their inner product is zero.

Recall the vectors `a` and `b` defined before. Are they orthogonal? Are they orthonormal?

## *Matrices*

Matrices are array of numbers. In the example in the Introduction section we defined the matrix

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
```

This is an example of a $5 \times 2$ matrix.[8] We can write this matrix as

$$\mathbf{M} = \begin{pmatrix} 5.1 & 3.5 \\ 4.9 & 3.0 \\ 4.7 & 3.2 \\ 4.6 & 3.1 \\ 5.0 & 3.6 \end{pmatrix}.$$

[8] The size of the matrix $\mathbf{M}$ is $5 \times 2$ as it has 5 rows and 2 columns. In general, a matrix can have any number of rows and columns.

To create the matrix **M** in R, and then to print, we use the following commands:[9]

[9] The command `cbind()` takes the column vectors, and puts them side by side. We can also use `rbind()` to concatinate row by row.

```
M = cbind(c(5.1, 4.9, 4.7, 4.6, 5), c(3.5, 3, 3.2, 3.1, 3.6))
M
```

```
##      [,1] [,2]
## [1,]  5.1  3.5
## [2,]  4.9  3.0
## [3,]  4.7  3.2
## [4,]  4.6  3.1
## [5,]  5.0  3.6
```

This way of creating matrix is essentially taking each column and then joining them together.

One could also try the command `matrix()`.[10]

[10] By default this command fills the matrix by columns. One could try to fill the matrix by rows by including the argument `byrow = TRUE` in the call to `matrix()`.

```
mydata = c(5.1, 4.9, 4.7, 4.6, 5, 3.5, 3, 3.2, 3.1, 3.6)
M = matrix(mydata, nrow = 5, ncol = 2, byrow = F)
M
```

```
##      [,1] [,2]
## [1,]  5.1  3.5
## [2,]  4.9  3.0
## [3,]  4.7  3.2
## [4,]  4.6  3.1
## [5,]  5.0  3.6
```

One could also read the matrix into R from an external file:

```
M = read.table(file="mydata.txt", header=FALSE)
```

where `mydata.txt` is an external file containing the values of the matrix with no column names (and hence `header=FALSE`). If column names are included in the file on top of each column, then use `header=TRUE` in the argument.

### *Transpose*

Transposing matrices involves turning the first column into the first row, second column into second row and so on. We write $\mathbf{M}^T$ as the transpose of $\mathbf{M}$.

We can use `t()` to take a transpose in R:

```
Mt = t(M)
Mt
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  5.1  4.9  4.7  4.6  5.0
## [2,]  3.5  3.0  3.2  3.1  3.6
```

The dimensions of any matrix can be checked with `dim()`.

```
dim(M)
```

```
## [1] 5 2
```

```
dim(Mt)
```

```
## [1] 2 5
```

We can also access certain elements of the matrix. For example $\mathbf{M}_{12}$ denotes the element of $\mathbf{M}$ which is in the 1st row and 2nd column of the matrix:

```
M[1, 2]
```

```
## [1] 3.5
```

### *Addition and subtraction*

Addition and subtraction of matrices can be done if the matrices have the *same size*. The sum of two matrices `A` and `B` (of same size) is another matrix (of the same size) where each element is the sum of the corresponding elements of `A` and `B`.

```
A = cbind(c(0.71, 0.61, 0.72, 0.83, 0.92), c(0.63, 0.69, 0.77,
    0.8, 1))
A
```

```
##      [,1] [,2]
## [1,] 0.71 0.63
## [2,] 0.61 0.69
## [3,] 0.72 0.77
## [4,] 0.83 0.80
## [5,] 0.92 1.00
```

```
B = matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 5, 2)
B
```

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

```
# Summing two matrices
A + B
```

```
##      [,1]  [,2]
## [1,] 1.71  6.63
## [2,] 2.61  7.69
## [3,] 3.72  8.77
## [4,] 4.83  9.80
## [5,] 5.92 11.00
```

```
# Subtracting
A - B
```

```
##       [,1]  [,2]
## [1,] -0.29 -5.37
## [2,] -1.39 -6.31
## [3,] -2.28 -7.23
## [4,] -3.17 -8.20
## [5,] -4.08 -9.00
```

Matrix addition satisfies the usual commutative and associative laws.

$$\begin{aligned} \text{Commutative law:} \quad & \mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \\ \text{Associative law:} \quad & \mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C} \end{aligned}$$

*Equality of two matrices*

Two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are equal, that is, $\boldsymbol{A} = \boldsymbol{B}$ if any only if:

1. $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same size, and
2. the $(i, j)$-th element of $\boldsymbol{A}$ is equal to the $ij$th element of $\boldsymbol{A}$ for all $1 \leq i \leq r$ and $1 \leq j \leq n$.

Therefore the following two zero matrices are not equal:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \neq \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

*Multiplication*

*Multiplication of a matrix by a scalar* is done by simply multiplying every element in the matrix by the scalar. So if $k = 0.4$, and

$$\mathbf{A} = \begin{pmatrix} 1 & 5 & 8 \\ 1 & 2 & 3 \end{pmatrix},$$

we can calculate $k\mathbf{A}$ as:

$$k\mathbf{A} = 0.4 \times \begin{pmatrix} 1 & 5 & 8 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 0.4 & 2 & 3.2 \\ 0.4 & 0.8 & 1.6 \end{pmatrix}.$$

Matrix multiplication however follows vector multiplication, and therefore does not follow the same rules as basic multiplication. To multiply two matrices `A` and `B`, one must first check that the *number of columns in `A` is exactly the same as the number of rows in `B`*. Otherwise, we can not multiply these two matrices. More generally,

$$\boldsymbol{A}_{\boldsymbol{m}\times\boldsymbol{n}} \times \boldsymbol{B}_{\boldsymbol{n}\times\boldsymbol{p}} = \boldsymbol{C}_{\boldsymbol{m}\times\boldsymbol{p}}.$$

Let $\boldsymbol{A}$ be of size $m \times n$; represent $\boldsymbol{A}$ using its row vectors $\boldsymbol{a}_1^T, \boldsymbol{a}_2^T, \ldots, \boldsymbol{a}_m^T$. Let $\boldsymbol{B}$ be of size $n \times p$; represent $\boldsymbol{B}$ using its columns vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_p$. The multiplication operation for matrices is defined as:

$$\mathbf{AB} = \begin{pmatrix} a_1^T \\ a_2^T \\ \ldots \\ a_m^T \end{pmatrix} \begin{pmatrix} b_1 & b_2 & \ldots & b_p \end{pmatrix} = \begin{pmatrix} a_1^T b_1 & a_1^T b_2 & \ldots & a_1^T b_p \\ a_2^T b_1 & a_2^T b_2 & \ldots & a_2^T b_p \\ \vdots & \vdots & & \vdots \\ a_m^T b_1 & a_m^T b_2 & \ldots & a_m^T b_p \end{pmatrix}$$

Thus, $(i, j)$-th element of $\mathbf{AB}$ is the inner product of $i$-th row of $\boldsymbol{A}$ and $j$-th column of $\boldsymbol{B}$.

Consider the following example.

```
A = cbind(c(0.71, 0.61, 0.72, 0.83, 0.92), c(0.63, 0.69, 0.77,
    0.8, 1))
A
```

```
##      [,1] [,2]
## [1,] 0.71 0.63
## [2,] 0.61 0.69
## [3,] 0.72 0.77
## [4,] 0.83 0.80
## [5,] 0.92 1.00
```

```
B = matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 2, 5)
B
```

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

Here `A` has 2 columns and `B` has two rows, and hence we can multiply `A` with `B`. In R, we only need to use the `%*%` operator to ensure we are getting matrix multiplication:

```
C = A %*% B
C
```

```
##      [,1] [,2]  [,3]  [,4]  [,5]
## [1,] 1.97 4.65  7.33 10.01 12.69
## [2,] 1.99 4.59  7.19  9.79 12.39
## [3,] 2.26 5.24  8.22 11.20 14.18
## [4,] 2.43 5.69  8.95 12.21 15.47
## [5,] 2.92 6.76 10.60 14.44 18.28
```

Just to check, look at $\boldsymbol{C}_{23}$, the $(2,3)$-th element of $\boldsymbol{C}$.

$$\boldsymbol{C}_{23} = 7.19 = (0.61, 0.69)\begin{pmatrix} 5 \\ 6 \end{pmatrix} = (5 \times 0.61) + (6 \times 0.69) = 7.19.$$

You will get an error message if you multiply non-conformable matrices.[11]

```
B %*% t(A)
```

```
## Error in B %*% t(A): non-conformable arguments
```

Unlike addition, matrix multiplication is not commutative:

[11] Dimesion of `B` is $2 \times 5$ but dimension of `t(A)` is $2 \times 5$. Thus number of columns in `B` is not the same as number of columns in `t(A)`.

$$\begin{aligned} \text{(non-commutative)} \quad & \mathbf{AB} \neq \mathbf{BA} \\ \text{Associative law} \quad & \mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} \end{aligned}$$

The distributive laws of multiplication over addition still apply.

$$\begin{aligned} \mathbf{A}(\mathbf{B}+\mathbf{C}) &= \mathbf{AB}+\mathbf{AC} \\ (\mathbf{A}+\mathbf{B})\mathbf{C} &= \mathbf{AC}+\mathbf{BC} \end{aligned}$$

We have the following rules for transposes.

$$\begin{aligned} (\mathbf{A}+\mathbf{B})^T &= \mathbf{A}^T+\mathbf{B}^T \\ (\mathbf{AB})^T &= \mathbf{B}^T\mathbf{A}^T \end{aligned}$$

*Some special matrices*

There are some matrices which have particular structure or properties of interest. We will use the following matrices often in this course.

(a) Identity Matrix: An identity matrix (of any size), is a diagonal matrix with 1 as each diagonal entry. For example, $\mathbf{I}_3$ is defined as

$$\mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

```r
diag(3)
```

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

(b) Ones: We also need to define a vector of ones; $\mathbf{1}_p$, a $p \times 1$ matrix containing only the value 1. There is no inbuilt function in `{R} to create this vector, it is easily added:`

```r
ones <- rep(1, 3)
ones
```

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

(c) Zero matrix: $\mathbf{0}$ denotes the zero matrix, a matrix of zeros. Unlike the previously mentioned matrices this matrix can be any shape you want. So, for example:

$$\mathbf{0}_{2\times 3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```r
matrix(0, nrow = 2, ncol = 3)
```

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

(d) Diagonal Matrices: A diagonal matrix is a square matrix in which all the "off diagonal" elements are zero. An example of diagonal matrix is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

```r
diag(c(1:3))
```

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

(e) Symmetric matrices: A matrix $\mathbf{A}$ is called a *symmetric* matrix if $A_{ij} = A_{ji}$, that is, $\mathbf{A} = \mathbf{A}^T$. As a consequence, symmetric matrix has to square, that is, they has to have the same number of rows and columns. For example, the following is a symmetric matrix:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix}.$$

### *Rank of a matrix*

Rank denotes the number of linearly independent rows or columns. For example:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

is $3 \times 3$ matrix with rank 2 since the first column can be found from the other two columns as $\boldsymbol{a_1} = \boldsymbol{a_2} + \boldsymbol{a_3}$.

If all the rows and columns of a *square matrix* are linearly independent it is said to be of full rank and non-singular. Otherwise it is said to be singular.

### *Matrix inversion*

Suppose $\boldsymbol{A}$ is a non-singular (full rank) $p \times p$ matrix. There is a unique matrix $\boldsymbol{B}$ such that $\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{I}_p$. We call the matrix $\boldsymbol{B}$ the inverse of $\boldsymbol{A}$, and denote by $\boldsymbol{A}^{-1}$. A singular matrix has no inverse.

In R, we use `solve()` to invert a matrix.

```
D <- matrix(c(5, 3, 9, 6), 2, 2)
D
```

```
##      [,1] [,2]
## [1,]    5    9
## [2,]    3    6
```

```
solve(D)
```

```
##      [,1]      [,2]
## [1,]    2 -3.000000
## [2,]   -1  1.666667
```

Reference: Multivariate Statistics with R by Paul J. Hewson