Linear Algebra with R

Lucas C. França

2022-03-07

Contents

1 Systems of Linear Equations and Matrices

1.1 Matrix and Vector

In this section we discuss matrices and vectors, and then we discuss hoe we can create a matrix and a vector in the R programming environment.

Definition 1 *For positive integers n, an n-**dimensional vector** is a 1-dimensional n array.*

In R, we can use `c()` function to create a vector. For example,

```
v <- c(907,220,625,502)
v
```

```
## [1] 907 220 625 502
```

In R, using ":" you can easily create a vector of a sequence of numbers. In this example, you can type

```
v <- 2:6
v
```

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

Definition 2 *For positive integers m, n, an m x n **matrix** is a 2-dimensional m x n array.*

In R we can create a matrix by calling the `matrix()` function. For example,

```
M <- matrix(c(1,1,1,2,1,3,1,4),nrow = 2, ncol= 4)
M
```

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

Here the `c()` function stores elements in the matrix. The order of the elements in `c()` function matters. As the default setting, it starts from the first element of the first column vector to the last element of the first column vector. Then it goes from the first element of the second column vector to the last element of the second column vector, and so on. "nrow" defines the number of row vectors and "ncol" defines the number of column vectors. For this example the number of row vectors is 2 and the number of column vectors is 4, so we set "nrow=2" and "ncol=4".

Definition 3 Suppose we have an mxn matrix, then $mx1$ matrices are called **collumn vectors** of the matrix. Also $1xn$ matrices are called **row vectors** of the matrix.

Now suppose we want to have just the first row vector of this matrix M, then we type:

```
M[1,]
```

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

Similarly, if we want to have the second column of the matrix M, then we type

```
M[,2]
```

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

Noe we show another way to make a matrix from combining row vectors or column vectors. First, we show how to create a matrix by combining vectors with the function `rbind()`. Let us define r1 and r2 as the first row vector and the second row vector of the matrix. in `R` we can define as

```
r1 <- c(1,1,1,1)
r2 <- c(1,2,3,4)
```

Then we use the `rbind()` function to create a matrix M as

```
M <- rbind(r1,r2)
M
```

```
##    [,1] [,2] [,3] [,4]
## r1    1    1    1    1
## r2    1    2    3    4
```

To create a matrix from column vectors in `R` we can use the function `cbind()`. For this example we first create four column vectors c1,c2,c3 and c4 with a function `c()`.

```
c1 <- c(1,1)
c2 <- c(1,2)
c3 <- c(1,3)
c4 <- c(1,4)
```

Then you use the `cbind()` function to create a matrix as follows:

```
M <- cbind(c1,c2,c3,c4)
M
```

```
##      c1 c2 c3 c4
## [1,]  1  1  1  1
## [2,]  1  2  3  4
```

If you want to extract the element in the 2th row and the 3th column, you can do:

```
M[2,3]
```

```
## c3
##  3
```

Using the `cbind()` function you can also create a new matrix. For example, let us pick the 2th and 4th columns if the matrix. Then let us make a new matrix called M2.

```
M2 <- cbind(M[,2], M[,4])
M2
```

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

If we want to know the number of row vectors and the number of column vectors of a matrix, you can use the `dim()` function in `R`.

```
dim(M)
```

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

The first output of the `dim()` function is the number of row vectors and the second output of the dim() function is the number of column vectors.

Definition 4 A **zero vector** or a **null vector** is an n-dimensional vector with all zeros as its elements.

In `R` you can create the 5-dimensional zero vector using the `rep()` function. The first argument is the value you want to assign as its element and the second argument is the dimension of the vector. The output from Ris as follows:

```
rep(0,5)
```

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

Definition 5 Suppose we have an mxn matrix. If $m = n$, then we call this matrix a **square matrix**.

For example, a $4x4$ zero square matrix:

```
M <- matrix(rep(0,16), ncol =  4)
M
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
## [3,]    0    0    0    0
## [4,]    0    0    0    0
```

Definition 6 The identity matrix, In, of size n is an $nÖn$ square matrix such that all elements in the ith row and ith column equal 1 for all i from 1 to n, and otherwise all 0.

To create the identity matrix of size n in `R` you can use the `diag()` function. If you type 'diag(n), then you create the identity matrix of size n.

```
diag(3)
```

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

2 Matrix Arithmetic

2.1 2.2 Matrix Operations

Matrix operations are the foundation for linear algebra and important for solving a system of linear equations as well as linear transformations.

2.1.1 Matrix addition

First we define **matrix addition**. In order to do so, the number of row vectors of A and B have to be the same and the number of column vectors A and B have to be the same.

In R, we do the following: First we define two vectors:

```
v1 <- c(2, -1, 3)
v2 <- c(-1, 0, 4)
```

Then the sum of these vectors is

```
v1 + v2
```

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

With matrices:

$$A = \begin{pmatrix} 3.00 & 0.00 & -5.00 \\ -1.00 & -3.00 & 4.00 \end{pmatrix}.$$

$$B = \begin{pmatrix} -5.00 & 5.00 & 2.00 \\ 1.00 & -2.00 & 0.00 \end{pmatrix}.$$

In R:

```
A <- matrix(c(3, 0, -5, -1, -3, 4), nrow = 2, ncol = 3, byrow = TRUE)
B <- matrix(c(-5, 5, 2, 1, -2, 0), nrow = 2, ncol = 3, byrow = TRUE)
A+B
```

Then R outputs,

$$A + B = \begin{pmatrix} -2.00 & 5.00 & -3.00 \\ 0.00 & -5.00 & 4.00 \end{pmatrix}.$$

2.1.2 Matrix multiplication

2.1.2.1 Scalar Multiplication **Definition** Suppose we have a real number c and an $A = mxn$ matrix, the **scalar multiplication** of c to A is each entry in the matrix multiplied by the given scalar.

In R, we can do the scalar multiplication as follows: First we define a matrix A:

```
A <- matrix(c(3, 0, -5, -1, -3, 4), nrow = 2, ncol = 3, byrow = TRUE)
A
```

```
##      [,1] [,2] [,3]
## [1,]    3    0   -5
## [2,]   -1   -3    4
```

Then we can do the scalar multiplication in Ras

```
-3 * A
```

```
##      [,1] [,2] [,3]
## [1,]   -9    0   15
## [2,]    3    9  -12
```

2.1.2.2 Multiplication of matrices When we multiply two matrices, we have to be very careful of their dimension. The number of column vectors of the left matrix has to be the same as the number of row vectors of the right matrix. The multiplication of an mÖn matrix and an nÖk matrix is an mÖk matrix.

2.1.2.2.1 Dot Product The **dot product** is an algebraic operation that takes two equal-length sequences of numbers (usually coordinate vectors), and returns a single number. In Eucledean geometry, the dot product of the Cartesian coordinates of two vectors is widely used. It is often called "the" **inner product** (or rarely **projection product**) of Euclidean space.

In R, first, we define these vectors

```
v1 <- c(2, -1, 3)
v2 <- c(-1, 0, 4)
```

Then the dot product of these vectors in Ris

```
v1 %*% v2
```

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

2.1.2.2.2 Dot Product of two matrices The multiplication of two matrices can be seen as a generalization of this dot product of two vectors.

InRwe do the following: First wr define two matrices:

```
A <- matrix(c(3, 0, -5, -1, -3, 4), nrow = 2, ncol = 3, byrow = TRUE)
B <- matrix(c(-5, 5, 2, 1, -2, 0), nrow = 3, ncol = 2, byrow = TRUE)
```

The matrices are:

$$A = \begin{pmatrix} 3.00 & 0.00 & -5.00 \\ -1.00 & -3.00 & 4.00 \end{pmatrix}.$$

and,

$$B = \begin{pmatrix} -5.00 & 5.00 \\ 2.00 & 1.00 \\ -2.00 & 0.00 \end{pmatrix}.$$

Then the for product of these matrices can be computed by

```
A %*% B
```

```
##      [,1] [,2]
## [1,]   -5   15
## [2,]   -9   -8
```

2.1.2.2.3 Transpose One of the important operations in matrices is called the **transpose** of a matrix. With this operation, we can be more flexible with other matrix operations.

Example Suppose we have a 2x3 matrix:

$$A = \begin{pmatrix} 4.00 & -5.00 & 1.00 \\ -1.00 & 0.00 & -2.00 \end{pmatrix}.$$

The transpose of the matrix A is a $3x2$ matrix

$$A^T = \begin{pmatrix} 4.00 & 0.00 \\ -1.00 & 1.00 \\ -5.00 & -2.00 \end{pmatrix}.$$

We can use Rto compute the transpose of a matrix using the `t()` function. With the matrix A from the example above, we first create a matrix using the `matrix()` function:

```
A <- matrix(c(4, -1, -5, 0, 1, -2), 2, 3, byrow = TRUE)
```

Then, we type in R

```
t(A)
```

```
##      [,1] [,2]
## [1,]    4    0
## [2,]   -1    1
## [3,]   -5   -2
```

2.1.2.2.4 Inverse One of the most important concepts is the **inverse** of a matrix. This concept is very important for solving a system of linear equations.

Definition Suppose A is a square matrix, *i.e.*, an mxm matrix. Then the **inverse** of a matrix A is an mxm matrix A^{-1} such that

$$

A . A^{-1} = A^{-1} . A = Im

$$ Where Im is the identity matrix of size m. If a matrix A has the inverse, then we say a matrix A is invertible.

In R, if we want to compute the inverse of a matrix we can use the `inv()` function from the `matlab` package. For example, if we have a matrix

$$A = \begin{pmatrix} 1.00 & 2.00 & 3.00 \\ -2.00 & 3.00 & -2.00 \\ -1.00 & 2.00 & 1.00 \end{pmatrix}.$$

then we can type:

```
library(matlib)
A <- matrix(c(1,-2,-1,2,3,2,3,-2,1), nrow = 3, ncol = 3)
matlib::inv(A)
```

```
##             [,1]       [,2]       [,3]
## [1,]  0.58333333  0.3333333 -1.0833333
## [2,]  0.33333333  0.3333333 -0.3333333
## [3,] -0.08333333 -0.3333333  0.5833333
```

If we want to see the output in terms of rational numbers, then we can use the `fractions()`function from the `MAAS`package:

```
library(MASS)
MASS::fractions(inv(A))
```

```
##      [,1]   [,2]   [,3]
## [1,]   7/12    1/3 -13/12
## [2,]    1/3    1/3   -1/3
## [3,]  -1/12   -1/3   7/12
```

In R we can also use the `solve()` function to find the inverse of a matrix instead of the `inv()` function. For example:

```
A <- matrix(c(1,-2,-1,2,3,2,3,-2,1), nrow = 3, ncol = 3)
solve(A)
```

```
##             [,1]       [,2]       [,3]
## [1,]  0.58333333  0.3333333 -1.0833333
## [2,]  0.33333333  0.3333333 -0.3333333
## [3,] -0.08333333 -0.3333333  0.5833333
```

3 Determinants

A **determinant** of a square matrix is a numerical summary of the matrix. This can be used for computing the inverse of a matrix if it is invertible and can also be used for solving a system of linear equations. In particular, the determinante is nonzero if and only if the matrix is invertible and the linear map represented by the matrix is an isomorphism. A matrix is often used to represent the coefficients in a system of linear quations, and determinants can be used to solve these equations (Cramer's rule), although other methods of solution are computationally much more efficient.

In R we can use the `det()` function to compute the determinant of a square matrix.

First we define a matrix in R:

```r
A <- matrix(c(-2, 4, -5, -1, -1, 1, -5, 0, -3), nrow = 3, ncol = 3)
A
```

```
##      [,1] [,2] [,3]
## [1,]   -2   -1   -5
## [2,]    4   -1    0
## [3,]   -5    1   -3
```

Then we use the `det()`function:

```r
det(A)
```

```
## [1] -13
```

4 Vector Spaces

A vector space is a foundation in many areas. A **vector space** is a set of objects called vectors, which may be added together and multiplied ("scaled") by numbers called **scalars**. The operations of vector addition and scalar multiplication must satisfy certain requirements, called vector axioms.

Definition 1 A **Scalar multiplication** is an operation such that $c * v$, where c is a scalar and v is an element of a set V.

Definition 2 A **vector space** V is a non-empy set with two operations: **addition** and **scalar multiplication** such that the *vector axioms* are true.

4.1 Inner Product Space

An inner product is an operation which defines a vector space. This operation is key to conduct any analysis in a vector space including a Euclidean space. This defines a magnitude or length of a vector in a vector space as well as the angle between two vectors in a vector space.

5 Eigen Values and Eigen Vectors

6 Linear Repression

7 Linear Programming

8 Network Analysis

9 References

1. Yoshida, R. (2021). Linear Algebra and Its Applications with R (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781003042259