Matrix Operations

Rafa

January 31, 2015

Matrix Notation

Matrix Notation

Here we introduce the basics of matrix notation. Initially this may seem over-complicated, but once we discuss examples, you will appreciate the power of using this notation to both explain and derive solutions, as well as implement them as R code.

Solving Systems of Equations

Solving Systems of Equations

Linear algebra was created by mathematicians to solve systems of linear equations such as this:

$$\begin{aligned} a + b + c &= 6 \\ 3a - 2b + c &= 2 \\ 2a + b - c &= 1 \end{aligned}$$

It provides very useful machinery to solve these problems generally. We will learn how we can write and solve this system using matrix algebra notation:

$$\begin{pmatrix} 1 & 1 & 1 \\ 3 & -2 & 1 \\ 2 & 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix} \implies \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 3 & -2 & 1 \\ 2 & 1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix}$$

This section explains the notation used above. It turns out that we can borrow this notation for linear models in statistics as well.

Matrix Operations

Matrix Operations

Let's approach the use of matrix algebra with this system of equations:

$$\begin{aligned} a + b + c &= 6 \\ 3a - 2b + c &= 2 \\ 2a + b - c &= 1 \end{aligned}$$

Matrix Operations

Matrix Operations

We described how this system can be rewritten and solved using matrix algebra:

$$\begin{pmatrix} 1 & 1 & 1 \\ 3 & -2 & 1 \\ 2 & 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix} \implies \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 3 & -2 & 1 \\ 2 & 1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix}$$

Having described matrix notation, we will explain the operation we perform with them. For example, above we have matrix multiplication and we also have a symbol representing the inverse of a matrix. The importance of these operations and others will become clear once we present specific examples related to data analysis.

Multiplying by a scalar

We start with one of the simplest operations: scalar multiplication. If a is scalar and $\mathbf{X}$ is a matrix, then:

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ x_{2,1} & \dots & x_{2,p} \\ & \vdots & \\ x_{N,1} & \dots & x_{N,p} \end{pmatrix} \implies a\mathbf{X} = \begin{pmatrix} ax_{1,1} & \dots & ax_{1,p} \\ ax_{2,1} & \dots & ax_{2,p} \\ & \vdots & \\ ax_{N,1} & \dots & ax_{N,p} \end{pmatrix}$$

R automatically follows this rule when we multiply a number by a matrix using `*`:

```r
X <- matrix(1:12,4,3)
print(X)
```

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

```r
a <- 2
print(a*X)
```

```
##      [,1] [,2] [,3]
## [1,]    2   10   18
## [2,]    4   12   20
## [3,]    6   14   22
## [4,]    8   16   24
```

The transpose

The transpose is an operation that simply changes columns to rows. We use a $\top$ to denote a transpose. The technical definition is as follows: if X is as we defined it above, here is the transpose which will be $p \times N$:

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ x_{2,1} & \dots & x_{2,p} \\ & \vdots & \\ x_{N,1} & \dots & x_{N,p} \end{pmatrix} \implies \mathbf{X}^\top = \begin{pmatrix} x_{1,1} & \dots & x_{p,1} \\ x_{1,2} & \dots & x_{p,2} \\ & \vdots & \\ x_{1,N} & \dots & x_{p,N} \end{pmatrix}$$

In R we simply use `t`:

```
X <- matrix(1:12,4,3)
X
```

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

```
t(X)
```

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

Matrix multiplication

We start by describing the matrix multiplication shown in the original system of equations example:

$$\begin{aligned} a + b + c &= 6 \\ 3a - 2b + c &= 2 \\ 2a + b - c &= 1 \end{aligned}$$

What we are doing is multiplying the rows of the first matrix by the columns of the second. Since the second matrix only has one column, we perform this multiplication by doing the following:

$$\begin{pmatrix} 1 & 1 & 1 \\ 3 & -2 & 1 \\ 2 & 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a + b + c \\ 3a - 2b + c \\ 2a + b - c \end{pmatrix}$$

Here is a simple example. We can check to see if `abc=c(3,2,1)` is a solution:

```
X  <- matrix(c(1,3,2,1,-2,1,1,1,-1),3,3)
abc <- c(3,2,1) #use as an example
rbind( sum(X[1,]*abc), sum(X[2,]*abc), sum(X[3,]*abc))
```

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

We can use the `%*%` to perform the matrix multiplication and make this much more compact:

```
X%*%abc
```

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

We can see that `c(3,2,1)` is not a solution as the answer here is not the required `c(6,2,1)`.

To get the solution, we will need to invert the matrix on the left, a concept we learn about below.

Here is the general definition of matrix multiplication of matrices A and X:

$$\mathbf{AX} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ & & \vdots & \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{pmatrix} \begin{pmatrix} x_{1,1} & \cdots & x_{1,p} \\ x_{2,1} & \cdots & x_{2,p} \\ & \vdots & \\ x_{N,1} & \cdots & x_{N,p} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^{N} a_{1,i}x_{i,1} & \cdots & \sum_{i=1}^{N} a_{1,i}x_{i,p} \\ & \vdots & \\ \sum_{i=1}^{N} a_{M,i}x_{i,1} & \cdots & \sum_{i=1}^{N} a_{M,i}x_{i,p} \end{pmatrix}$$

You can only take the product if the number of columns of the first matrix A equals the number of rows of the second one X. Also, the final matrix has the same row numbers as the first A and the same column numbers as the second X. After you study the example below, you may want to come back and re-read the sections above.

The identity matrix

The identity matrix is analogous to the number 1: if you multiply the identity matrix by another matrix, you get the same matrix. For this to happen, we need it to be like this:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}$$

By this definition, the identity always has to have the same number of rows as columns or be what we call a square matrix.

If you follow the matrix multiplication rule above, you notice this works out:

$$\mathbf{XI} = \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ & \vdots & \\ x_{N,1} & \dots & x_{N,p} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ & & & \vdots & & \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} = \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ & \vdots & \\ x_{N,1} & \dots & x_{N,p} \end{pmatrix}$$

In R you can form an identity matrix this way:

```r
n <- 5 #pick dimensions
diag(n)
```

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

The inverse

The inverse of matrix X, denoted with X^{-1}, has the property that, when multiplied, gives you the identity $X^{-1}X = I$. Of course, not all matrices have inverses. For example, a 2×2 matrix with 1s in all its entries does not have an inverse.

As we will see when we get to the section on applications to linear models, being able to compute the inverse of a matrix is quite useful. A very convenient aspect of R is that it includes a predefined function `solve` to do this. Here is how we would use it to solve the linear of equations:

```r
X <- matrix(c(1,3,2,1,-2,1,1,1,-1),3,3)
y <- matrix(c(6,2,1),3,1)
solve(X)%*%y #equivalent to solve(X,y)
```

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

Please note that `solve` is a function that should be used with caution as it is not generally numerically stable.