2 | VECTORS AND MATRICES

Jay Klopper MD MMed(Surgery)Cum Laude

The George Washington University

Table of Contents

Introduction

Data are often captured and saved in tabular format, such as in a spreadsheet. We have rows and columns of data. If each row of data is that of a specific subject and if each column is of a defined variable, then a spreadsheet is said to be **long-format tidy data**. This is the most convenient format and the format that we will consider here. The table below is long-format tidy data. Each row is a subject and each column only contains data for a specific variable (and of a specific level of measurement).

Age	Systolic blood pressure	Diastolic blood pressure
77	134	89
76	137	91
81	129	88
76	138	87
76	139	90
69	141	88

If we ignore the column headers (names of the variables) we have six rows and three columns. A rectangular array of numbers such as this is a **matrix**. We can also consider the first row of values $[77\ 134\ 89]$ (or indeed any of the other rows) as a **row vector**. Any of the columns is a **column vector**. Below, we see the first column of numbers as a column vector. Note the use of (square) brackets. Parentheses are also used.

$$\begin{bmatrix} 77 \\ 76 \\ 81 \\ 76 \\ 76 \\ 69 \end{bmatrix}$$

A vector is simply an ordered list of numbers. MATLAB (named for Matrix Laboratory) is an excellent programming language for matrix analysis.

In this chapter we explore the use of vectors and matrices. In later chapters we will import data directly from spreadsheet files. While we explore some vector and matrix arithmetic, knowledge of linear algebra is not absolutely required.

Vectors

Vectors are ordered lists of numbers. We can create both row and column vectors. Vector are typically assigned boldface lowercase letters. Below we create the row vector $\mathbf{u} = [1 \quad 2 \quad 3]$. Note that any row vector has a single row and various columns. This is the size of a vector.

```
clear variables % Delete variables from the workspace
% Create the row vector u with elements 1, 2, and 3
u=[1 2 3] % Elements are separated by spaces by commas can also be used as in [1,2,3]
```

```
u = 1×3
     1     2     3
```

The `size` function returns the number of rows and columns (as a row vector).

```
% Calculate the size of u
size(u) % Returns a row vector with elements indicating the number of rows and the number of columns
```

```
ans = 1×2
     1     3
```

In simple terms, the transpose of a vector $\mathbf{u}$, written as $\mathbf{u}^{\mathbf{T}}$, turns a row vector into a column vector and *vice versa*. Below, we calculate $\mathbf{u}^{T}$.

```
% Transpose of u is calculated using an apostrophe
u'
```

```
ans = 3×1
     1
     2
     3
```

A column vector is created by separating the elements by a semicolon. We create the vector below.

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
% Create the column vector v with elements 1, 2, and 3
v = [1;2;3] % Elements are separated by semicolons
```

```
v = 3×1
     1
     2
     3
```

The transpose $\mathbf{v}^{T}$ creates a row vector.

```
% Transpose of v
v'
```

```
ans = 1×3
     1     2     3
```

Vectors can be added if they have the same shape.

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{bmatrix}$$

or

$$\begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} + \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = \begin{bmatrix} u_1 + v_1 & u_2 + v_2 & \dots & u_n + v_n \end{bmatrix}$$

```
% Add u and v
u+v'
```

```
ans = 1×3
     2     4     6
```

Scalar vector multiplication is defined below for a scalar $c \in \mathbb{R}$.

$$c\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} c\,u_1 \\ c\,u_2 \\ \vdots \\ c\,u_n \end{bmatrix}$$

or

$$c\begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} = \begin{bmatrix} c\,u_1 & c\,u_2 & \dots & c\,u_n \end{bmatrix}$$

```
% Scalar-vector multiplication
3*u
```

```
ans = 1×3
     3     6     9
```

Using the scalar $c = -1$ we can define the negative of a vector and vector subtraction, shown below.

$-\mathbf{v} = (-1)\mathbf{v}$

```
% Calculate the negative of vector v
-v
```

```
ans = 3×1
    -1
    -2
    -3
```

$\mathbf{u} - \mathbf{v} = \mathbf{u} + (-1)\mathbf{v}$

```
% Calculate u transpose minus v (to have two column vectors)
u'-v
```

```
ans = 3×1
     0
     0
     0
```

```
% Calculate u minus the transpose of v
u-v'
```

```
ans = 1×3
     0     0     0
```

There are more ways in MATLAB to created ordered lists of numbers. A unit range states the first and last integer separated by a colon.

```
% Create a row vector of the integers 1 through 10
1:10
```

```
ans = 1×10
     1     2     3     4     5     6     7     8     9    10
```

A step size can be added.

```
% Add a step size of 2
1:2:11
```

```
ans = 1×6
     1     3     5     7     9    11
```

The `linspace` function takes a start and end value, as well as an optional number of elements. The default is 100 equally space numbers between the start and end value.

```
% Create a row matrix of 10 numbers between 1 and 10
linspace(1,10,10)
```

```
ans = 1×10
     1     2     3     4     5     6     7     8     9    10
```

Matrices

Matrices are ordered lists of numbers arranged into rows and columns. We create the matrix below. Matrices are typically assigned to uppercase letters.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
clear variables
% Create the matrix A with 2 rows and three columns
A=[1 2 3;4 5 6]
```

```
A = 2×3
     1     2     3
     4     5     6
```

The `reshape` function can also be used to create a matrix. The matrix A below is created with the elements listed as a row vector. We specify the number of rows and columns.

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

```
% Creation by reshaping a vector
A=reshape([1 2 3 4 5 6 7 8 9],3,3)
```

```
A = 3×3
     1     4     7
     2     5     8
     3     6     9
```

The transpose again interchanges the rows and columns.

```
% Transpose of A
A'
```

```
ans = 3×3
     1     2     3
     4     5     6
     7     8     9
```

There are special matrices such as matrices of all 0's, or all 1's.

```
% Matrix with three rows and four columns of 0's assigned to the variable A
A=zeros(3,4)
```

```
A = 3×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
% Matrix with three rows and four columns of 1's assigned to the variable A
A=ones(3,4)
```

```
A = 3×4
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

The identity matrix is a square matrix with 1's along the diagonal from top left to bottom right and 0's everywhere else.

```
% Create the identity matrix with 4 rows and columns assigned to the variable I
I=eye(4)
```

```
I = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

A diagonal matrix has non-zero entries along the diagonal and 0's everywhere else.

```
% Diagonal matrix with entries 3, 4, and 5 assigned to the variable A
diag([3 4 5])
```

```
ans = 3×3
     3     0     0
     0     4     0
     0     0     5
```

```
% Use the square 4x4 matrix of ones and turn it into an upper triangular
% matrix assigned to the variable A
A=triu(ones(4))
```

```
A = 4×4
     1     1     1     1
     0     1     1     1
     0     0     1     1
     0     0     0     1
```

```
% Use the square 4x4 matrix of ones and turn it into a lower triangular
% matrix assigned to the variable A
A=tril(ones(4))
```

```
A = 4×4
     1     0     0     0
     1     1     0     0
     1     1     1     0
     1     1     1     1
```

The inverse of a square matrix A is A^{-1} such that $AA^{-1} = A^{-1}A = I$.

```
% Create a new square matrix assigned to the variable A
A=[1 2 3;-4 5 2;3 2 1]
```

```
A = 3×3
     1     2     3
    -4     5     2
     3     2     1
```

```
% Inverse of a square matrix
inv(A)
```

```
ans = 3×3
   -0.0208   -0.0833    0.2292
   -0.2083    0.1667    0.2917
    0.4792   -0.0833   -0.2708
```

```
% Multiply A and the inverse of A
A*A'
```

```
ans = 3×3
    14    12    10
    12    45     0
    10     0    14
```

The determinant of a square matrix is calculates using the `det` function.

```
% Determinant
det(A)
```

```
ans = -48.0000
```

Matrices of similar sizes can be added. The operation is element wise.

```
% Matrix-matrix addition
A = eye(3);
B = eye(3);
A + B
```

```
ans = 3×3
     2     0     0
     0     2     0
     0     0     2
```

Scalar-matrix multiplication is also element wise. Below, we multiply the matrix A by the scalar $c = 3$.

```
% Scalar-matrix multiplication
3 * A
```

```
ans = 3×3
     3     0     0
     0     3     0
     0     0     3
```

Using $c = -1$ we define matrix subtraction (of same sized matrices only).

$A - B = A + (-1)B$

Indexing

Every entry in a vector or matrix has an index (address).

```
clear variables
% Create a new 3 row, 4 column matrix assigned to the variable A
A = [3 4 2 8; 9 1 -3 7; -4 -5 0, 5]
```

```
A = 3×4
     3     4     2     8
     9     1    -3     7
    -4    -5     0     5
```

We use row-column indexing to return the value of the element in row 2 and column 2 of A. Note the use of parentheses for indexing.

```
% Row 2, column 2
A(2,2)
```

```
ans = 1
```

The colon symbol is shorthand that indicates that we want all the elements in the rows and or column. Below we index row 2 and all the columns in that row. and the all the rows in column 3.

```
% All of row 2
A(2,:)
```

```
ans = 1×4
     9     1    -3     7
```

```
% All of column 3
A(:,3)
```

```
ans = 3×1
     2
    -3
     0
```

Next we index row 2 and columns 2 through 4 using a unit range object.

```
% Row 2, columns 2 through 4
A(2,2:4)
```

```
ans = 1×3
     1    -3     7
```

Finally we index the elements in row two for columns 2 and 4. The latter are passed as a row matrix.

```
% Row 2, columns 1 and 4
A(2,[1 4])
```

```
ans = 1×2
     9     7
```

Vector dot product

The dot product (or scalar inner product) of two vectors $\mathbf{u}$ and $\mathbf{v}$ both member of $\mathbb{R}^n$ is defined as $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v}$, shown below, where u_i and v_i are the i^{th}. elements of $\mathbf{u}$ and $\mathbf{v}$ respectively.

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$$

```
clear variables
% Create a vector u and a vector v
u=[1 2 3]
```

```
u = 1×3
     1     2     3
```

```
v=[3 2 -1]
```

```
v = 1×3
     3     2    -1
```

MATLAB will calculate the scalar dot product of two vectors with the same number of entries. Both can be row vectors and the first will be transposed automatically by MATLAB.

```
% Scalar dot product of u and v
dot(u,v)
```

```
ans = 4
```

The arithmetic mean of a list of number can be calculated as a vector dot product. We need a weight vector $\mathbf{w}$. Each element of $\mathbf{w}$ is 1 divided by the sample size. Below, we calculate the arithmetic mean of the five numbers $33, 56, 53, 39$ and 49.

```
% Create a weight-vector assigned to the variable w
w=repmat(1/5, 1, 5)
```

```
w = 1×5
    0.2000    0.2000    0.2000    0.2000    0.2000
```

```
% Create a vector of subject ages assigned to the variable a
a=[33,56,53,39,49]
```

```
a = 1×5
    33    56    53    39    49
```

Mean age is $\mathbf{w} \cdot \mathbf{a}$.

```
% Dot product of weight vector and age vector
dot(w,a)
```

```
ans = 46.0000
```

The `mean` function confirms that we have calculated the arithmetic mean.

```
% Mean of ages
mean(a)
```

```
ans = 46
```

Matrix-vector multiplication

Matrix-vector multiplication $A\mathbf{u}$ is defined for matrices $A \in \mathbb{R}^{m,n}$ and $\mathbf{u} \in \mathbb{R}^n$. The result is a vector $(A\mathbf{u}) \in \mathbb{R}^m$. Each element of the result is the dot product of the i^{th} rows of A and the vector $\mathbf{u}$.

```
clear variables
% Create a 4 row and 3 column matrix assigned to the variable A
A = [33,9,0.4;34,10,0.5;31,8,0.5;1,0,-1]
```

```
A = 4×3
   33.0000    9.0000    0.4000
   34.0000   10.0000    0.5000
   31.0000    8.0000    0.5000
    1.0000         0   -1.0000
```

```
% Create a vector with 3 entries assigned to the variable v
v = [3;4;1]
```

```
v = 3×1
     3
     4
     1
```

The `mtimes` function performs the matrix-vector multiplication of A and $\mathbf{u}$, assigned to the variables `A` and `u`. The result is a column vector that has four rows.

```
% The shape of the matrix and vector must be correct for matrix-vector
% multiplication
% A times v
mtimes(A,v)
```

```
ans = 4×1
  135.4000
  142.5000
  125.5000
    2.0000
```

Matrix-matrix multiplication

Matrix-matrix multiplication is defined for matrices $A \in \mathbb{R}^{m,p}$ and $B \in \mathbb{R}^{p,n}$. The result is the matrix $C \in \mathbb{R}^{m,n}$. Each element $c_{ij} \in C$ is the dot product of the i^{th} row of A and the j^{th} column of B.

```
clear variables
% Create matrices A and B
A = [1 3 5; 2 4 7];
B = [-5 8 11; 3 9 21; 4 0 8];
% Matrix-matrix multiplication
A * B
```

```
ans = 2×3
    24    35   114
    30    52   162
```

In general matrix-matrix multiplication does not commute. We can always calculate AA^T or A^TA for any matrix.

```
% Calculate A times the transpose of A
A * A'
```

```
ans = 2×2
    35    49
    49    69
```

```
% Calculate the transpose of A times A
A' * A
```

```
ans = 3×3
     5    11    19
    11    25    43
    19    43    74
```

Note the difference in size of the two matrix-matrix products.