**Introduction to matrix algebra**

**The basics**

A matrix is a two-dimensional grouping of elements put into rows and columns.

Example:



The *dimension* or *size* of a matrix: # rows × # columns = r×c

Example: 2×3 for previous matrix

Rather than using numbers, we can represent parts of a matrix (and the matrix itself) by symbols.

Example:



where aij is the row i and column j element of **A**

a11 is often called the “(1,1) element” of **A**, a12 is called the “(1,2) element” of **A**,…

a11 = 1 from the previous example

Notice that the matrix **A** was bolded. When bolding is not possible (writing on a piece of paper or chalkboard), the letter is underlined - A

Example: Matrix of size r×c



Example: Square matrix is r×c where r = c

Example: HS and College GPA

From the introduction to R notes, we can start putting parts of the observed data into a matrix form. For example,



The above 20×2 matrix contains the HS GPAs in the second column. It will be clear shortly why we have a leading column of 1’s.

Vector – a r×1 (column vector) or 1×c (row vector) matrix – special case of a matrix

Example: Symbolic representation of a 3×1 column vector



Example: HS and College GPA



The above 20×1 vector contains the College GPAs.

Transpose – Interchange the rows and columns of a matrix or vector

Example:

 and 

**A** is 2×3 and **A**′ is 3×2

The ′ symbol indicates a transpose, and it is said as the word “prime”. Thus, the transpose of **A** is “A prime”.

Example: HS and College GPA



**Matrix addition and subtraction**

Add or subtract the corresponding elements of matrices with the same dimension.

Example:

Suppose .

Then .

Example: Using R (BasicMatrixAlgebra.R)

> A <- matrix(data = c(1, 2, 3,

4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)

> A

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

> class(A)

[1] "matrix"

> B <- matrix(data = c(-1, 10, -1, 5, 5, 8), nrow = 2, ncol = 3, byrow = TRUE)

> A + B

[,1] [,2] [,3]

[1,] 0 12 2

[2,] 9 10 14

> A - B

[,1] [,2] [,3]

[1,] 2 -8 4

[2,] -1 0 -2

Notes:

1. Be careful with the byrow option. By default, this is set to FALSE. Thus, the numbers would be entered into the matrix by columns. For example,

> matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

1. The class of these objects is “matrix”.
2. A vector can be represented as a “matrix” class type or a type of its own.

> y <- matrix(data = c(1,2,3), nrow = 3, ncol = 1, byrow = TRUE)

> y

[,1]

[1,] 1

[2,] 2

[3,] 3

> class(y)

[1] "matrix"

> x <- c(1,2,3)

> x

[1] 1 2 3

> class(x)

[1] "numeric"

> is.vector(x)

[1] TRUE

This can present some confusion when vectors are multiplied with other vectors or matrices because no specific row or column dimensions are given. More on this shortly.

1. A transpose of a matrix can be done using the t() function. For example,

> t(A)

[,1] [,2]

[1,] 1 4

[2,] 2 5

[3,] 3 6

**Matrix multiplication**

Scalar - 1×1 matrix

Example: Matrix multiplied by a scalar

 where c is a scalar

Let  and c = 2. Then .

Multiplying two matrices

Suppose you want to multiply the matrices **A** and **B**; i.e., **A**\***B** or **AB**. In order to do this, you need the number of columns of **A** to be the same as the number of rows as **B**. For example, suppose **A** is 2×3 and **B** is 3×10. You can multiply these matrices. However, if **B** is 4×10 instead, these matrices cannot be multiplied.

The resulting dimension of **C** = **AB**:

1. The number of rows of **A** is the number of rows of **C**.
2. The number of columns of **B** is the number of rows of **C**.
3. In other words,  where the dimension of the matrices are shown below them.

How to multiply two matrices – an example

Suppose . Notice that **A** is 2×3 and **B** is 3×2 so **C** = **AB** can be done.



The “cross product” of the rows of **A** and the columns of **B** are taken to form **C**.

In the above example, **D** = **BA** ≠ **AB** where **BA** is:



In general for a 2×3 matrix times a 3×2 matrix:



Example: Using R (BasicMatrixAlgebra.R)

> A <- matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)

> B <- matrix(data = c(3, 0, 1, 2, 0, 1), nrow = 3, ncol = 2, byrow = TRUE)

> C <- A%\*%B

> D <- B%\*%A

> C

[,1] [,2]

[1,] 5 7

[2,] 17 16

> D

[,1] [,2] [,3]

[1,] 3 6 9

[2,] 9 12 15

[3,] 4 5 6

> # What is A\*B?

> A\*B

Error in A \* B : non-conformable arrays

Notes:

1. %\*% is used for multiplying matrices and/or vectors
2. \* means to perform elementwise multiplications. Here is an example where this can be done:

> E <- A

> A\*E

[,1] [,2] [,3]

[1,] 1 4 9

[2,] 16 25 36

The (i,j) elements of each matrix are multiplied together.

1. Multiplying vectors with other vectors or matrices in R can be confusing because no row or column dimensions are given for a vector object. For example, suppose **x** = , a 3×1 vector.

> x <- c(1,2,3)

> x%\*%x

[,1]

[1,] 14

> A%\*%x

[,1]

[1,] 14

[2,] 32

How does R know that we want **x**′**x** (1×1) instead of **xx**′ (3×3) when we have not told R that **x** is 3×1? Similarly, how does R know that **Ax** is 2×1? From the R help for %\*% in the Base package:

Multiplies two matrices, if they are conformable. If one argument is a vector, it will be promoted to either a row or column matrix to make the two arguments conformable. If both are vectors it will return the inner product.

An inner product produces a scalar value. If you wanted **xx**′ (3×3), one can use the outer product %o%

> x%o%x # outer product

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 2 4 6

[3,] 3 6 9

We will only need to use %\*% in this class.

Example: HS and College GPA (HScollegeGPA-MA.R)

 and 

Find **X**′**X**, **X**′**Y**, and **Y**′**Y**.

> #Read in the data

> gpa <- read.csv(file = "c:\\data\\gpa.csv")

> head(gpa)

HSGPA CollegeGPA

1 3.04 3.1

2 2.35 2.3

3 2.70 3.0

4 2.05 1.9

5 2.83 2.5

6 4.32 3.7

> X <- cbind(1, gpa$HSGPA)

> Y <- gpa$CollegeGPA

> X

[,1] [,2]

[1,] 1 3.04

[2,] 1 2.35

[3,] 1 2.70

[4,] 1 2.55

[5,] 1 2.83

[6,] 1 4.32

[7,] 1 3.39

[8,] 1 2.32

[9,] 1 2.69

[10,] 1 2.83

[11,] 1 2.39

[12,] 1 3.65

[13,] 1 2.85

[14,] 1 3.83

[15,] 1 2.22

[16,] 1 1.98

[17,] 1 2.88

[18,] 1 4.00

[19,] 1 2.28

[20,] 1 2.88

> Y

[1] 3.10 2.30 3.00 2.45 2.50 3.70 3.40 2.60 2.80 3.60 2.00

[12] 2.90 3.30 3.20 2.80 2.40 2.60 3.80 2.20 2.60

> t(X)%\*%X

[,1] [,2]

[1,] 20.00 57.9800

[2,] 57.98 175.8094

> t(X)%\*%Y

[,1]

[1,] 57.2500

[2,] 170.6995

> t(Y)%\*%Y

[,1]

[1,] 168.9025

Notes:

1. The cbind() function combines items by “c”olumns. Because 1 is only one element, it will replicate (called “recycling” in R) itself for all elements that you are combining so that one full matrix is formed. There is also a rbind() function that combines by rows. Thus, rbind(a,b) forms a matrix with a above b.
2. 
3.  because x11 = = xn1 = 1
4. **X**′**X**  
   
5. Here’s another way to get the **X** matrix

> mod.fit <- lm(formula = CollegeGPA ~ HSGPA, data = gpa)

> model.matrix(object = mod.fit)

(Intercept) HS.GPA

1 1 3.04

2 1 2.35

3 1 2.70

4 1 2.55

5 1 2.83

6 1 4.32

7 1 3.39

8 1 2.32

9 1 2.69

10 1 2.83

11 1 2.39

12 1 3.65

13 1 2.85

14 1 3.83

15 1 2.22

16 1 1.98

17 1 2.88

18 1 4.00

19 1 2.28

20 1 2.88

attr(,"assign")

[1] 0 1

**Special types of matrices**

Symmetric matrix: If **A** = **A**′, then **A** is symmetric.

Example: 

Diagonal matrix: A square matrix whose “off-diagonal” elements are 0. The diagonal of a matrix are the (1,1), (2,2), … elements.

Example: 

Identity matrix: A diagonal matrix with 1’s on the diagonal.

Example: 

Note that “**I**” (the letter **I**, not the number one) usually denotes the identity matrix.

Vector of 0’s: 

**Linear dependence and rank of matrix**

Let . Think of each column of **A** as a vector; i.e., **A** = [**a**1, **a**2, **a**3]. Note that 3**a**2 = **a**3. This means the columns of **A** are “linearly dependent.”

Formally, a set of column vectors are linearly dependent if there exists constants λ1, λ2,…, λc (not all zero) such that λ1**a**1 + λ2**a**2 + +λc**a**c = **0**. A set of column vectors are linearly independent if λ1**a**1 + λ2**a**2+ + λc**a**c = **0** only for λ1 = λ2 = = λc = **0**.

The rank of a matrix is the maximum number of linearly independent columns in the matrix.

rank(**A**) = 2

If a matrix does not have a rank equal to the number of its columns, this can cause problems in some of our calculations this semester. More on this later…

**Inverse of a matrix**

Note that the inverse of a scalar, say b, is b-1. For example, the inverse of b = 3 is 3-1 = 1/3. Also, b\*b-1 = 1. In matrix algebra, the inverse of a matrix is another matrix. For example, the inverse of **A** is **A**-1, and **AA**-1 = **A**-1**A** = **I**. Note that **A** must be a square matrix.

Example: 

Check: 

Finding the inverse

For a 2×2 matrix, there is a simple formula. Let . Then

.

Verify **AA**-1 = **I** on your own.

For larger matrices, the calculations are more difficult. We will rely on R for these calculations. Matrix algebra textbooks provide the calculation details for those who have an interest in them.

Example: Using R (BasicMatrixAlgebra.R~~HScollegeGPA-MA.R~~)

> A <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE)

> solve(A)

[,1] [,2]

[1,] -2.0 1.0

[2,] 1.5 -0.5

> A%\*%solve(A)

[,1] [,2]

[1,] 1 1.110223e-16

[2,] 0 1.000000e+00

> solve(A)%\*%A

[,1] [,2]

[1,] 1.000000e+00 0

[2,] 1.110223e-16 1

> round(solve(A)%\*%A, 2)

[,1] [,2]

[1,] 1 0

[2,] 0 1

The solve()function inverts a matrix in R. The solve(A,b) function can also be used to “solve” for **x** in **Ax** = **b** since **A**-1**Ax** = **A**-1**b** ⇒ **Ix** = **A**-1**b** ⇒ **x** = **A**-1**b**

Example: HS and College GPA (HScollegeGPA-MA.R)

Remember that  and 

Find (**X**′**X**)-1 and (**X**′**X**)-1**X**′**Y**

> solve(t(X)%\*%X)

[,1] [,2]

[1,] 1.1378689 -0.3752566

[2,] -0.3752566 0.1294435

> solve(t(X)%\*%X) %\*% t(X)%\*%Y

[,1]

[1,] 1.0868795

[2,] 0.6124941

From previous output:

> mod.fit <- lm(formula = CollegeGPA ~ HSGPA, data = gpa)

> mod.fit$coefficients

(Intercept) HS.GPA

1.0868795 0.6124941

Thus, (**X**′**X**)-1**X**′**Y** = , where  and  are the estimated intercept and slope coefficients in a simple linear regression model.

**Trace and Determinant**

Suppose we have a p×p matrix **A**:



Notice this is a square matrix!

### Trace – The trace of a square matrix **A** is defined as ; i.e., the sum of a square matrix’s diagonal elements.

Example: Using R (BasicMatrixAlgebra.R)

> A <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE)

> A

[,1] [,2]

[1,] 1 2

[2,] 3 4

> diag(A)

[1] 1 4

> sum(diag(A))

[1] 5

Determinant – The determinant of a square matrix **A** is  where **A**1j = (-1)1+j|**A**1j| and **A**1j is obtained from **A** by deleting its first row and its jth column.

The determinant for a 22 matrix is defined as



and the determinant for a 33 matrix can be defined as



Example: Using R (BasicMatrixAlgebra.R)

> A <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE)

> det(A)

[1] -2

**Eigenvalues and eigenvectors**

Suppose we have a p×p matrix **A**:



Notice this is a square matrix!

Eigenvalues – The roots of the polynomial equation defined by  where **I** is an identity matrix.

If p = 2, then the eigenvalues are the roots of





Using the quadratic formula, we obtain



In general, the eigenvalues are the p roots of

c1λp + c2λp-1 + c3λp-2 + … + cpλ + cp+1 = 0 where ci for i = 1, …, p+1 denote constants.

When **A** is a symmetric matrix, the eigenvalues are real numbers and can be ordered from largest to smallest as λ1 ≥ λ2 ≥ ≥ λp where λ1 is the largest.

Example: Using R (BasicMatrixAlgebra.R)

> A <- matrix(data = c(1, 0.5, 0.5, 1.25), nrow = 2, ncol = 2, byrow = TRUE)

> A

[,1] [,2]

[1,] 1.0 0.50

[2,] 0.5 1.25

> eigen(A)

$values

[1] 1.6403882 0.6096118

$vectors

[,1] [,2]

[1,] 0.6154122 -0.7882054

[2,] 0.7882054 0.6154122

> save.eig <- eigen(A)

> save.eig$values

[1] 1.6403882 0.6096118

> save.eig$vectors

[,1] [,2]

[1,] 0.6154122 -0.7882054

[2,] 0.7882054 0.6154122



Eigenvectors – Each eigenvalue of **A** has a corresponding nonzero vector **b** called an eigenvector that satisfies **Ab** = λ**b**.

Notes:

* Eigenvectors for a particular eigenvalue are not unique. They are also given as having a length of 1 (see two pages ahead).
* When two eigenvalues are not equal, their corresponding eigenvectors are orthogonal (i.e., ).
* 
* 

Example: Using R (BasicMatrixAlgebra.R)

The eigenvectors of **A** satisfy



for **b** = [b1, b2]′, λ1 = 1.6404, and λ2 = 0.6096.

Two possible vectors are:

 for λ1 = 1.6404 and  for λ2 = 0.6096 (see the previous example’s output).

Note that



and



Below is the verification from R:

> A%\*%save.eig$vectors[,1]

[,1]

[1,] 1.009515

[2,] 1.292963

> save.eig$values[1]\*save.eig$vectors[,1]

[1] 1.009515 1.292963

> A%\*%save.eig$vectors[,2]

[,1]

[1,] -0.4804993

[2,] 0.3751625

> save.eig$values[2]\*save.eig$vectors[,2]

[1] -0.4804993 0.3751625

Vector length – Suppose **b** = [b1, …, bp]′ is a vector. The length of a vector is .

R chooses eigenvectors so that they have a length of 1.

Example: Using R (BasicMatrixAlgebra.R)

> sqrt(sum(save.eig$vectors[,1]^2))

[1] 1

> sqrt(sum(save.eig$vectors[,2]^2))

[1] 1

Below is a plot of the vectors:

> # Square plot (default is "m" which means maximal region)

> par(pty = "s")

> # Set up some dummy values for plot

> b1 <- c(-1,1)

> b2 <- c(-1,1)

> plot(x = b1, y = b2, type = "n", main = expression(paste("Eigenvectors of ", bold(A))), xlab = expression(b[1]), ylab = expression(b[2]), panel.first = grid(col = "gray", lty = "dotted"))

> # Run demo(plotmath) for help on mathematical notation

> # draw line on plot - h specifies a horizontal line

> abline(h = 0, lty = "solid", lwd = 2)

> # v specifies a vertical line

> abline(v = 0, lty = "solid", lwd = 2)

> arrows(x0 = 0, y0 = 0, x1 = save.eig$vectors[1,1], y1 = save.eig$vectors[2,1], col = "red", lty = "solid")

> arrows(x0 = 0, y0 = 0, x1 = save.eig$vectors[1,2], y1 = save.eig$vectors[2,2], col = "red", lty = "solid")



Notes:

* The vectors are orthogonal.
* Other eigenvectors do exist! For example, you can multiply one of the vectors by any scalar value. If the scalar was -10, this is what we obtain:

> new.vec <- -10\*save.eig$vectors[,1]

> A%\*%new.vec

[,1]

[1,] -10.09515

[2,] -12.92963

> save.eig$values[1]\*new.vec

[1] -10.09515 -12.92963

> sqrt(sum(new.vec^2))

[1] 10

### **Positive Definite and Positive Semidefinite Matrices**

Definition 4.5: If a symmetric matrix has all of its eigenvalues positive, the matrix is called a *positive definite matrix*.

Definition 4.6: If a symmetric matrix has all nonnegative eigenvalues and if at least one of the eigenvalues is actually 0, then the matrix is called a *positive semidefinite matrix*.

Definition 4.7: If a matrix is either positive definite or positive semidefinite, the matrix is defined to be a nonnegative matrix.

If you are familiar with covariance and correlation matrices, they are always nonnegative!

Example: Using R (BasicMatrixAlgebra.R)

All eigenvalues are positive, so **A** is a positive definite matrix.

For a matrix **A**, |**A**| > 0 means **A** is positive definite and its inverse exists.