Displays for Statistics 5401

Lecture 3

September 12, 2005

Christopher Bingham, Instructor

612-625-1024, kb@umn.edu
372 Ford Hall

Class Web Page

http://www.stat.umn.edu/~kb/classes/5401

© 2005 by Christopher Bingham

MacAnova Operations on Columns

Note: These differ from R and S-plus

```
Cmd> a # previously entered 4 by 3 matrix
(1,1)            1            2           -3
(2,1)            0           11            4     4 by 3 matrix
(3,1)            7           17           12
(4,1)           -3            4            5
```

Sum down columns

```
Cmd> sum(a)#sum down columns; row vector (1 by 4 matrix)
(1,1)            5           34           18

Cmd> sum(a,margin:2) # sum down columns; ordinary vector
(1)              5           34           18
```

(1,1) indicates result has 2 dimensions
(1) indicates result has 1 dimensions

Sum across rows

```
Cmd> sum(a') #  sum across rows, row vector result
(1,1)            0           15           36            6

Cmd> sum(a')' #  sum across rows, column vector sums
(1,1)            0
(2,1)           15
(3,1)           36
(4,1)            6

Cmd> sum(a,margin:1) # sum across rows, ordinary vector result
(1)              0           15           36            6
```

Products down columns

```
Cmd> prod(a) # multiply down columns
(1,1)            0         1496         -720
```

Extremes of columns

```
Cmd> min(a) # row vector result
(1,1)           -3            2           -3

Cmd> max(a) # row vector result
(1,1)            7           17           12
```


Ordering columns by sort()

```
Cmd> sort(a) # Each column sorted in increasing order
(1,1)           -3            2           -3
(2,1)            0            4            4
(3,1)            1           11            5
(4,1)            7           17           12

Cmd> sort(a,down:T) # decreasing order
(1,1)            7           17           12
(2,1)            1           11            5
(3,1)            0            4            4
(4,1)           -3            2           -3
```

grade() is related to sort():

```
Cmd> grade(a)
(1,1)            4            1            1 Row #s of col minima
(2,1)            2            4            2 Row #s of 2nd
(3,1)            1            2            4 Row #s of 3rd
(4,1)            3            3            3 Row #s of col maxima
```

The numbers in row i are the row numbers of the i^{th} smallest entries in each column. You can also use down:T.

```
Cmd> grade(a,down:T) # row numbers of elements in down order
(1,1)            3            3            3 Row #s of col maxima
(2,1)            1            2            4
(3,1)            2            4            2
(4,1)            4            1            1 Row #s of col minima
```

Reorder the rows of **a** so that column 1 is increasing.

```
Cmd> J <- grade(a[,1]) # grade() applied to just column 1 of a

Cmd> a[J,] # Rows now permuted so column 1 is sorted
(1,1)           -3            4            5
(2,1)            0           11            4
(3,1)            1            2           -3
(4,1)            7           17           12
```


Matrix Multiplication A × C = A C

MacAnova: a %*% b *not* a * b

```
Cmd> c <- matrix(enter(1 3 5 2 4 6),3) # 3x2

Cmd> c
(1,1)            1            2
(2,1)            3            4                     3 by 2
(3,1)            5            6

Cmd> b <- matrix(run(12),nrows(a)) # 4 by 3

Cmd> b
(1,1)            1            5            9
(2,1)            2            6           10        4 by 3
(3,1)            3            7           11
(4,1)            4            8           12
```

A × C = A C (A times C)

```
Cmd> a %*% c # (4 by 3) %*% (3 by 2)
(1,1)           -8           -8
(2,1)           53           68                     4 by 2
(3,1)          118          154
(4,1)           34           40
```

A × B′ = A B′ (A times transpose of B)

```
Cmd> a %*% b' #  4 by 3  %*%  3 by 4
(1,1)          -16          -16          -16          -16
(2,1)           91          106          121          136   4 by 4
(3,1)          200          236          272          308
(4,1)           62           68           74           80
```

For A B to be defined, number of columns of A must = number of rows of B:

```
Cmd> a %*% b #   4 by 3   %*%  4 by 3  is an error
ERROR: Dimension mismatch: 4 by 3 %*% 4 by 3 near
a %*% b # 4 by 3  %*% 4 by 3
```

This produced an error because
ncols(a) ≠ nrows(b)

Two *short cuts*:

a %c% b is the same as a' %*% b

a %C% b is the same as a %*% b'

```
Cmd> equal(a %c% b, a' %*% b)
(1) T            T means True, a %c% b equals a' %*% b

Cmd> equal(a %C% b, a %*% b')
(1) T
```

equal(x,y) has value T (True) if and only if x and y are identical. Otherwise it has value F (False).

Help topic **arithmetic** summarizes the arithmetic operations and details of their use.

Help topic **matrices** has info on matrix multiplication and most functions that compute things from matrices such as eigenvalues and determinants.

To read these topics, you can

- use **Help** on the **Help** menu and then click on the topic, or
- type help(matrices) or help(arithmetic).

Notational Conventions

You can view a matrix in several ways.

Suppose $\mathbf{A} = [a_{ij}]_{1\leq i\leq n, 1\leq j\leq p}$ is an n by p matrix, that is, a rectangular table:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ a_{31} & a_{32} & \dots & a_{3p} \\ \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \dots & a_{np} \end{bmatrix} \text{ n rows, p columns}$$

- You can view A as a set of n 1 by p *row* vectors, $\mathbf{a}_i' = [a_{i1}, a_{i2}, \dots, a_{ip}]$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1' \\ \mathbf{a}_2' \\ \dots \\ \mathbf{a}_n' \end{bmatrix} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n]',$$

where $\mathbf{a}_i = \begin{bmatrix} a_{i1} \\ a_{i2} \\ \dots \\ a_{ip} \end{bmatrix}$, $i = 1,\dots,n$

$\mathbf{a}_i'$ is 1 by p

- You can view **A** as a set of p n × 1 *column* vectors

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_p]$$

where $\mathbf{A}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ a_{3j} \\ \cdot\ \cdot \\ a_{nj} \end{bmatrix} = [a_{1j}, a_{2j}, \dots, a_{nj}]'$

I often use the convention:

- Lower case letters such as **a** stand for the rows of a matrix
- Uppercase letters such as **A** stand for the columns.

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1' \\ \mathbf{a}_2' \\ \dots \\ \mathbf{a}_n' \end{bmatrix} = [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_p].$$

When **A** is a data matrix for n cases and p variables, $\mathbf{a}_i$ has the data for *case* i and $\mathbf{A}_j$ has the data for *variable* j.

Vocabulary: Inner product

When **a** and **b** are vectors with the *same length p*, their *inner product* is

$$\sum_{1\leq i\leq p} a_i b_i$$

that is, an inner product is a *sum of products*. This is a useful operation because you use a lot of sums of products in statistics.

```
Cmd> x <- vector(3.32,3.00,1.61,2.53,3.61)

Cmd> y <- vector(11.21,10.64,8.16,9.47,10.41)

Cmd> innerxy <- sum(x*y); innerxy # x*y is elementwise product
(1)      143.81          Inner product of x and y
```

Inner product of **a** with itself is

$$\|\mathbf{a}\|^2 \equiv \sum_i a_i^2, \text{ a sum of } squares.$$

You use a lot of sums of squares, too.

Vocabulary: $\|\mathbf{a}\| = \sqrt{\{\sum_i a_i^2\}}$ is the *norm* or *length* of **a**.

```
Cmd> normx <- sqrt(sum(x^2)); normx
(1)      6.4844
```

This is not the same as length(a) = number of elements of **a** in MacAnova.

Matrix product of row and column vectors

Suppose **a** and **b** are vectors of the same length. Then **a**′ is a **row** vector and **b** is a **column** vector.

The matrix *products* **a**′×**b** = **a**′**b** and **b**′×**a** = **b**′**a** have the same value.

It is *defined to be* the inner product of **a** and **b**. That is

$$\mathbf{a}'\mathbf{b} = \mathbf{b}'\mathbf{a} = \sum_i a_i b_i$$

```
Cmd> x' %*% y
(1,1)      143.81
Cmd> y' %*% x
(1,1)      143.81
Cmd> innerxy # computed previously as sum(x*y)
(1)      143.81
```

A particular case is

$$\mathbf{a}'\mathbf{a} = \sum_i a_i^2 = \|\mathbf{a}\|^2$$

So you can represent both sums of products and sums of squares by matrix products.

```
Cmd> x' %*% x # same as sum(x^2)
(1,1)      42.047
```


Note that products **a**′**b** and **b**′**a** have the form

(1 by p matrix) × (p by 1 matrix)

The number of columns on the left = the number of rows on the right.

```
Cmd> a <- vector(78.2, 69.5, 32.4, 52.0, 66.2)
Cmd> b <- vector(56.5, 26.9, 54.5, 38.9, 67.3)
Cmd> sum(a*b) # inner product
(1)       14532
Cmd> a' %*% b # matrix product  of a %c% b
(1,1)       14532
Cmd> b' %*% a # matrix product  or b %c% a
(1,1)       14532

Cmd> a %*% b' # this is ab', not yet defined; differs from b'%*%a
(1,1)     4418.3       2103.6       4261.9        3042       5262.9
(2,1)     3926.8       1869.5       3787.8      2703.5       4677.3
(3,1)     1830.6       871.56       1765.8      1260.4       2180.5
(4,1)       2938       1398.8         2834      2022.8       3499.6
(5,1)     3740.3       1780.8       3607.9      2575.2       4455.3
```

This last illustrates that in general

$$\mathbf{AB} \neq \mathbf{BA}$$

Of course, I haven't yet even defined what **ab**′ is.

In fact, the i,j element of **ab**′ is $a_i b_j$.

Matrix multiplication in terms of inner products

You can define matrix multiplication for larger matrices in terms of inner products $\mathbf{a}_i'\mathbf{B}_j$.

Suppose **A** is m by n and **B** is n by q

$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m]'$, with $\mathbf{a}_i$ n by 1

$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_q]$, with $\mathbf{B}_i$ n by 1

The matrix product $\mathbf{C} = [c_{ij}] = \mathbf{A}\,\mathbf{B}$ is *defined* by the inner products $c_{ij} \equiv \mathbf{a}_i'\mathbf{B}_j$

That is, c_{ij} = (*inner product* of $\mathbf{a}_i$ and $\mathbf{B}_j$) = (matrix product of row i of **A** ($\mathbf{a}_i'$) and column j of **B** ($\mathbf{B}_j$)).

In more familiar terms

$$c_{ij} = \sum_{1\leq k\leq n} a_{ik} b_{kj},\ i = 1, \ldots, m,\ j = 1,\ldots,q$$

An important case

Suppose n×p $\mathbf{X} = [\mathbf{X}_1\ \mathbf{X}_2 \ldots \mathbf{X}_p]$ is a matrix of *data* on p variables for n cases.

Then the columns of **X** and the rows of **X**′ are the vectors $\mathbf{X}_j$, j = 1,...,p, each of which have n elements. Therefore

$$\mathbf{X}'\mathbf{X} = [\mathbf{X}_j'\mathbf{X}_k]_{1\leq j,k\leq p}.$$

But

$$\mathbf{X}_j'\mathbf{X}_k = \sum_{1\leq i\leq n} x_{ij}x_{ik} \text{ and } \mathbf{X}_j'\mathbf{X}_j = \sum_{1\leq i\leq n} x_{ij}^2$$

are **sums of products and squares.**

Similarly, if $\mathbf{Y} = [\mathbf{Y}_1\ \mathbf{Y}_2 \ldots \mathbf{Y}_q]$ is n×q,

$$\mathbf{X}'\mathbf{Y} = [\mathbf{X}_j'\mathbf{Y}_k]_{1\leq j\leq p,1\leq k\leq q}$$

where $\mathbf{X}_j'\mathbf{Y}_k = \sum_{1\leq i\leq n} x_{ij}y_{ik}$ is a **sum of products**.

Matrix Multiplication in terms of outer products

Suppose

$\mathbf{x} = [x_i]_{1\leq i\leq m}$ a m-vector (m by 1 matrix)

$\mathbf{y} = [y_i]_{1\leq i\leq n}$ a n-vector (n by 1 matrix)

m and n may be different.

Vocabulary: The *outer product* $\mathbf{x} \otimes \mathbf{y}$ is defined to be the m×n matrix

$$\mathbf{x} \otimes \mathbf{y} = \begin{bmatrix} x_1y_1 & x_1y_2 & \dots & x_1y_n \\ x_2y_1 & x_2y_2 & \dots & x_2y_n \\ \dots & \dots & \dots & \dots \\ x_my_1 & x_my_2 & \dots & x_my_m \end{bmatrix} = \underset{m\times 1}{\mathbf{x}}\ \underset{1\times n}{\mathbf{y}'}$$

```
Cmd> x <- vector(29,4,-2,-58) # m = 4

Cmd> y <- vector(70,45,40,34,-147) # n = 5

Cmd> outer(x,y) # or x %*% y'  or x %C% y
(1,1)        2030         1305         1160          986        -4263
(2,1)         280          180          160          136         -588
(3,1)        -140          -90          -80          -68          294
(4,1)       -4060        -2610        -2320        -1972         8526

Cmd> x %*% y'
(1,1)        2030         1305         1160          986        -4263
(2,1)         280          180          160          136         -588
(3,1)        -140          -90          -80          -68          294
(4,1)       -4060        -2610        -2320        -1972         8526

Cmd> x[4]*y[5] # check row 4 and column 5 of result
(1)         8526
```


Suppose **A** is m by n and **B** is n by q.

You can define the matrix product **AB** in terms of the *outer products* $\mathbf{A}_j\mathbf{b}_j'$ of the *columns* $\mathbf{A}_j$ of **A** and the *rows* $\mathbf{b}_j'$ of **B**. ,

$$\mathbf{AB} = \sum_{1\leq j\leq n}\mathbf{A}_j\mathbf{b}_j' = \mathbf{A}_1\mathbf{b}_1' + \mathbf{A}_2\mathbf{b}_2' + \dots + \mathbf{A}_n\mathbf{b}_n'$$

= sum of n outer products, all m by q

```
Cmd> a <- matrix(vector(1.04,0.696,-0.651,0.13,1.5,1.61), 2); a
(1,1)        1.04      -0.651          1.5    2 by 3
(2,1)       0.696        0.13         1.61

Cmd> b <- matrix(vector(0.367,0.658,0.796,\
           0.024,-0.784,-0.851, 0.352,-0.082,1.143),3); b
(1,1)       0.367       0.024        0.352
(2,1)       0.658      -0.784       -0.082    3 by 3
(3,1)       0.796      -0.851        1.143

Cmd> a %*% b # matrix product
(1,1)      1.1473    -0.74116        2.134    2 by 3
(2,1)      1.6225     -1.4553       2.0746

Cmd> # Here is same thing as sum of outer prods

Cmd> a[,1] %*% b[1,] + a[,2] %*% b[2,] + a[,3] %*% b[3,]
(1,1)      1.1473    -0.74116        2.134
(2,1)      1.6225     -1.4553       2.0746
```

For example, `a[,2] %*% b[2,]` = outer product of column 2 of **a** and row 2 of **b**

This might be a good time to show how "looping" can be helpful. I computed the same product by using a **for** loop.

I started by creating a 2 by 3 matrix of 0's to hold the sum of outer products

```
Cmd> s <- matrix(rep(0,6),2);s # set initial value of s to 0
(1,1)           0           0           0
(2,1)           0           0           0

Cmd> for(i,1,3){ # same thing using a loop
        s <- s + a[,i] %*% b[i,];;
     }

Cmd> s
(1,1)      1.1473    -0.74116        2.134
(2,1)      1.6225     -1.4553       2.0746
```

The "loop" was repeated 3 times, with i = 1, i = 2 and i = 3.

Note the ";;" after the last (and only) command in the loop. If that is omitted, output may be produced on every trip around a loop.

Note: Except in exceptional cases

$$\mathbf{AB} \neq \mathbf{BA}$$

That is, order of multiplication matters.

Notational conventions again

I will always use the conventions

- **Single** multivariate observation is column vector **x**
- **Column** of data matrix ⇐=⇒ **Variable**
- **Row** of data matrix ⇐=⇒ **Case** notated $\mathbf{x}_i'$, where $\mathbf{x}_i$ is column vector

Linear Combinations of Variables

Often in statistics you are interested in a *linear combination* of variables in a multivariate vector $\mathbf{x} = [x_1,\ldots,x_p]'$.

Vocabulary: A **linear combination** has the form

$$y = c_1x_1 + c_2x_2 + \ldots + c_px_p = \sum_{1\leq i\leq p} c_ix_i$$

Example:

$$\begin{aligned} y &= x_3 - x_1 \\ &= (-1)x_1 + (0)x_2 + (1)x_3 + \ldots + (0)x_p, \end{aligned}$$

This is the inner product of

$$\mathbf{c} = [c_1, c_2, \ldots, c_p]' = [-1, 0, 1, \ldots, 0]'$$

and

$$\mathbf{x} = [x_1, x_2, \ldots x_p]'$$

You can write y as the matrix product

$$y = \mathbf{c}'\mathbf{x}$$

You should learn to recognize $\mathbf{c}'\mathbf{x}$ as representing a linear combination.

Symmetric Matrices

Vocabulary: Square, symmetric

If **A** is a p by p matrix, then we say it is *square*.

A square matrix **A** is *symmetric* when $a_{ji} = a_{ij}$ all i and j.

$$\mathbf{A} = \begin{bmatrix} 15.3 & 11.0 & 16.2 & 10.3 \\ 11.0 & 13.2 & 12.0 & 11.0 \\ 16.2 & 12.0 & 17.3 & 11.2 \\ 10.3 & 11.0 & 11.2 & 9.4 \end{bmatrix}$$

Equal values are connected by arrows.

When **A** is square, then $.5*(\mathbf{A} + \mathbf{A}')$ is always symmetric.

Diagonal Matrices

When **B** is square and $b_{ij} = 0$ when $i \neq j$, then **B** is a **diagonal** matrix:

Example

$$\mathbf{B} = \begin{bmatrix} 15.3 & 0.0 & 0.0 & 0.0 \\ 0.0 & 13.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 17.3 & 0.0 \\ 0.0 & 0.0 & 0.0 & 9.4 \end{bmatrix}$$

This is sometimes written

$$\mathbf{B} = \text{diag}[b_{11}, b_{22}, \ldots, b_{pp}].$$

Create **B** in MacAnova using `dmat()`:

```
Cmd> b <- dmat(vector(15.3,13.2,17.3,9.4)); b
(1,1)        15.3           0           0           0
(2,1)           0        13.2           0           0
(3,1)           0           0        17.3           0
(4,1)           0           0           0         9.4
```


You use `diag()` to extract the diagonal of a matrix:

```
Cmd> diag(b)
(1)        15.3        13.2        17.3         9.4

Cmd> a <- matrix(vector(15.3,11,16.2,10.3, 11,13.2,12,11,\
     16.2,12,17.3,11.2, 10.3,11,11.2,9.4),4)

Cmd> a
(1,1)        15.3          11        16.2        10.3
(2,1)          11        13.2          12          11
(3,1)        16.2          12        17.3        11.2
(4,1)        10.3          11        11.2         9.4

Cmd> diag(a)
(1)        15.3        13.2        17.3         9.4
```

m by m identity matrix I_m

$$I_m = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

with 1's down the diagonal and 0's everywhere else. I_m is square and diagonal.

Create in MacAnova by, for example,

```
Cmd> dmat(rep(1,4)) # or dmat(vector(1,1,1,1))
(1,1)          1          0          0          0
(2,1)          0          1          0          0
(3,1)          0          0          1          0
(4,1)          0          0          0          1

Cmd> rep(1,4)
(1)          1          1          1          1
```

A shortcut form is

```
Cmd> dmat(4, 1)
(1,1)          1          0          0          0
(2,1)          0          1          0          0
(3,1)          0          0          1          0
(4,1)          0          0          0          1
```

dmat(m,c) creates a **m** by **m** matrix with m values of **c** on the diagonal.

Multiplying a matrix by an identity matrix doesn't change it.

If **A** is m by n,

- $I_m A = A$, that is $I_m A$ is *identical* to **A**
- $A I_n = A$, that is $A I_n$ is *identical* to **A**

MacAnova

```
Cmd> a # m = 2, n = 3
(1,1)        1.04     -0.651        1.5
(2,1)       0.696       0.13       1.61

Cmd> i2 <- dmat(2, 1) # 2 by 2 identity matrix

Cmd> i3 <- dmat(3, 1) # 3 by 3 identity matrix

Cmd> i2 %*% a # identity matrix on left; same as a
(1,1)        1.04     -0.651        1.5
(2,1)       0.696       0.13       1.61

Cmd> a %*% i3 # identity matrix on right; same as a
(1,1)        1.04     -0.651        1.5
(2,1)       0.696       0.13       1.61
```