Matrix Algebra

Mike Strube

August 29, 2018

1 Overview

Multivariate statistics are often represented in matrix form. This provides a convenient shorthand for what ordinarily could be very cumbersome equations. In addition, there are matrix characteristics (e.g., singularity, determinant) that play important roles in interpreting multivariate statistics. And, transformations of data and means are common in multivariate statistics. These are easily represented in matrix form. For these reasons, and to obtain some street cred among statisticians, it is useful to know a few matrix algebra basics.

2 Preliminaries

In this section, the RStudio workspace and console panes are cleared of old output, variables, and other miscellaneous debris. Packages are loaded and any required data files are retrieved.

```
options(replace.assign = TRUE, width = 65, digits = 4, scipen = 4, fig.width = 4,
    fig.height = 4)
# Clear the workspace and console.
rm(list = ls(all = TRUE))
cat("\f")
```

```
# Turn off showing of significance asterisks.
options(show.signif.stars = F)
# Set the contrast option; important for ANOVAs.
options(contrasts = c("contr.sum", "contr.poly"))
how_long <- Sys.time()
set.seed(123)
library(knitr)
```

```
# Load the psych package.
library(psych)
library(MASS)
```

3 A Symmetric Matrix

A symmetric matrix has an equal number of rows and columns.

3.1 Define a Matrix

```
F <- matrix(c(1, 3, -4, 3, 11, 7, -4, 7, 2), nrow = 3, ncol = 3, byrow = TRUE)
F
```

```
##      [,1] [,2] [,3]
## [1,]    1    3   -4
## [2,]    3   11    7
## [3,]   -4    7    2
```

4 Trace of the Matrix

The sum of the elements on the main diagonal of a symmetric matrix is called the trace. The trace can be obtained using the tr() function from the psych package.

```
t_F <- tr(F)
```

The trace for matrix, F, is 14.

5 Diagonal Matrix

If all elements of a matrix except the main diagonal are zero, the matrix is a diagonal matrix.

```
# A diagonal matrix can be created in the usual way:
F <- matrix(c(1, 0, 0, 0, 11, 0, 0, 0, 2), nrow = 3, ncol = 3, byrow = TRUE)
# Or, it can be created using the diag() function. This is easier
# for large matrices.
F <- diag(c(1, 11, 2))
F
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0   11    0
## [3,]    0    0    2
```

6 Matrix Addition and Subtraction

Matrices can be added and subtracted, element by element, provided they are of the same order (i.e., have the same numbers of rows and columns).

6.1 Define Matrices

```
F <- matrix(c(1, 3, 3, 11, -4, 7), nrow = 3, ncol = 2, byrow = TRUE)
H <- matrix(c(4, -1, 6, 2, 12, 8), nrow = 3, ncol = 2, byrow = TRUE)
F
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    3   11
## [3,]   -4    7
```

```
H
```

```
##      [,1] [,2]
## [1,]    4   -1
## [2,]    6    2
## [3,]   12    8
```

6.2 Define a New Matrix: K = F + H

```
K <- F + H
K
```

```
##      [,1] [,2]
## [1,]    5    2
## [2,]    9   13
## [3,]    8   15
```

6.3 Define a New Matrix: K = F - H

```
K <- F - H
K
```

```
##      [,1] [,2]
## [1,]   -3    4
## [2,]   -3    9
## [3,]  -16   -1
```

7 Zero Matrix

It is occasionally useful to have a matrix of zeros. This is created in the usual way.

```
Z <- matrix(0, nrow = 3, ncol = 5)
Z
```

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

8 Scalar Multiplication

A matrix multiplied by a scalar is equivalent to multiplying each matrix element by the scalar.

8.1 Define a Matrix

```
F <- matrix(c(1, 3, -4, 3, 11, 7, -4, 7, 2), nrow = 3, ncol = 3, byrow = TRUE)
F
```

```
##      [,1] [,2] [,3]
## [1,]    1    3   -4
## [2,]    3   11    7
## [3,]   -4    7    2
```

8.2 Define a New Matrix: K = 5*F

```
K <- 5 * F
K
```

```
##      [,1] [,2] [,3]
## [1,]    5   15  -20
## [2,]   15   55   35
## [3,]  -20   35   10
```

9 Matrix Multiplication

Two matrices can be multiplied provided the number of columns in the first matrix is equal to the number of rows in the second matrix. Order of multiplication matters. To multiply matrices we use the matrix multiplication operator, %*%.

9.1 Define Matrices

```
A <- matrix(c(4, -1, 7, 0, 6, 2), nrow = 3, ncol = 2, byrow = TRUE)
B <- matrix(c(1, 7, 0, 4, 2, 6), nrow = 2, ncol = 3, byrow = TRUE)
A
```

```
##      [,1] [,2]
## [1,]    4   -1
## [2,]    7    0
## [3,]    6    2
```

```
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    7    0
## [2,]    4    2    6
```

9.2 Define a New Matrix: C = A*B

```
C <- A %*% B
C
```

```
##      [,1] [,2] [,3]
## [1,]    0   26   -6
## [2,]    7   49    0
## [3,]   14   46   12
```

9.3 AB May Not Equal BA

```
A %*% B

##      [,1] [,2] [,3]
## [1,]    0   26   -6
## [2,]    7   49    0
## [3,]   14   46   12

B %*% A

##      [,1] [,2]
## [1,]   53   -1
## [2,]   66    8
```

10 Matrix Transpose

Every matrix has a transpose that is obtained by exchanging the rows and columns.

10.1 Define Matrix

```
X <- matrix(c(4, -1, 7, 0, 6, 2), nrow = 3, ncol = 2, byrow = TRUE)
X

##      [,1] [,2]
## [1,]    4   -1
## [2,]    7    0
## [3,]    6    2
```

10.2 Transpose

To get the transpose of a matrix, use the t() function.

```
t(X)

##      [,1] [,2] [,3]
## [1,]    4    7    6
## [2,]   -1    0    2
```

10.3 The Transpose in Action

Transposes are useful for arranging a matrix so that matrix multiplication is possible. For example, we frequently need to generate the sums of squares and cross-products for a data matrix. If $X_{n,v}$ is a matrix of deviation scores, then $X_{n,v}X_{n,v}$ is not possible. But, $X_{v,n}X_{n,v}$ can be carried out. The matrix, $X_{v,n}$, is the transpose of $X_{n,v}$ and can be symbolized as $X'_{n,v}$ or as $X^T_{n,v}$.

Here we will create a matrix of random numbers to simulate some data. We will pretend we have 100 cases and 4 variables and will draw the variables from a multivariate normal distribution with means of 0 and a covariance matrix of

$$\begin{bmatrix} 1.0 & 1.2 & 0.3 & 0.4 \\ 1.2 & 2.0 & 0.2 & 0.3 \\ 0.3 & 0.2 & 3.0 & 0.4 \\ 0.4 & 0.3 & 0.4 & 4.0 \end{bmatrix}$$

10.4 Define Matrix

```
mu <- matrix(0, nrow = 1, ncol = 4)
VC <- matrix(c(1, 1.2, 0.3, 0.4, 1.2, 2, 0.2, 0.3, 0.3, 0.2, 3, 0.4,
    0.4, 0.3, 0.4, 4), nrow = 4, ncol = 4, byrow = TRUE)
X <- mvrnorm(100, mu, VC)
head(X)

##         [,1]    [,2]    [,3]    [,4]
## [1,] -1.2889 -1.7473  3.0924  0.9601
## [2,] -1.1915 -1.3785  0.8433  0.9307
## [3,] -0.7982 -0.1738 -0.9763 -3.0777
## [4,] -0.6704 -0.2480  0.8656 -0.2650
## [5,]  0.4306  1.0186  0.8992 -1.0780
## [6,] -0.3379 -0.4127 -1.5537 -3.2714
```

10.5 Create Deviation Scores

We need deviation scores. We might think we already have them because we drew the sample from a population with means of 0. The finite sample means will likely not be zero exactly, however. We can make them so by centering the variables.

```
describe(X)

##    vars   n  mean   sd median trimmed  mad   min  max range
## X1    1 100 -0.10 0.93  -0.09   -0.09 0.86 -2.87 2.06  4.93
## X2    2 100 -0.13 1.28  -0.14   -0.09 1.25 -3.52 2.53  6.06
## X3    3 100  0.18 1.65   0.14    0.17 1.65 -3.94 4.19  8.13
## X4    4 100 -0.22 1.90  -0.10   -0.22 1.82 -5.41 4.71 10.12
##     skew kurtosis   se
## X1 -0.15    -0.13 0.09
## X2 -0.31     0.03 0.13
## X3  0.03    -0.47 0.16
## X4 -0.05    -0.08 0.19
```

```
X <- scale(X, center = TRUE, scale = FALSE)
describe(X)

##    vars   n mean   sd median trimmed  mad   min  max range  skew
## X1    1 100    0 0.93   0.01    0.01 0.86 -2.77 2.16  4.93 -0.15
## X2    2 100    0 1.28  -0.02    0.04 1.25 -3.40 2.66  6.06 -0.31
## X3    3 100    0 1.65  -0.04    0.00 1.65 -4.11 4.02  8.13  0.03
## X4    4 100    0 1.90   0.13    0.00 1.82 -5.19 4.93 10.12 -0.05
##    kurtosis   se
## X1    -0.13 0.09
## X2     0.03 0.13
## X3    -0.47 0.16
## X4    -0.08 0.19
```

10.6 Sums of Squares and Cross-Products

Multiplication using the transpose produces the desired matrix.

```
SSCP <- t(X) %*% X
```

10.7 Variance-Covariance Matrix

The sum of squares and cross-products matrix is one step away from the variance-covariance matrix. All we need to do is divide each element of the matrix by the degrees of freedom, N-1. That is simply a scalar operation.

```
VC_Sample <- SSCP/(99)
VC_Sample

##        [,1]    [,2]   [,3]    [,4]
## [1,] 0.8638  0.9652 0.2011  0.1620
## [2,] 0.9652  1.6481 0.1284 -0.1134
## [3,] 0.2011  0.1284 2.7139  0.3338
## [4,] 0.1620 -0.1134 0.3338  3.6255
```

We did this set of operations for demonstration purposes. The variance-covariance matrix can be obtained directly using the cov() function.

```
cov(X)

##        [,1]    [,2]   [,3]    [,4]
## [1,] 0.8638  0.9652 0.2011  0.1620
## [2,] 0.9652  1.6481 0.1284 -0.1134
## [3,] 0.2011  0.1284 2.7139  0.3338
## [4,] 0.1620 -0.1134 0.3338  3.6255
```

We can convert the variance-covariance matrix to a correlation matrix using the cov2cor() function.

```
cov2cor(VC_Sample)

##         [,1]     [,2]    [,3]     [,4]
## [1,] 1.00000  0.80893 0.13132  0.09157
```

```
## [2,] 0.80893  1.00000 0.06072 -0.04639
## [3,] 0.13132  0.06072 1.00000  0.10643
## [4,] 0.09157 -0.04639 0.10643  1.00000
```

The correlation matrix can be thought of as a standardized variance-covariance matrix. If we standarize the variables and calculate the variance-covariance matrix, it will equal the correlation matrix for the original variable.

```
Z <- scale(X, scale = TRUE, center = TRUE)
cov(Z)

##         [,1]     [,2]    [,3]     [,4]
## [1,] 1.00000  0.80893 0.13132  0.09157
## [2,] 0.80893  1.00000 0.06072 -0.04639
## [3,] 0.13132  0.06072 1.00000  0.10643
## [4,] 0.09157 -0.04639 0.10643  1.00000

cor(X)

##         [,1]     [,2]    [,3]     [,4]
## [1,] 1.00000  0.80893 0.13132  0.09157
## [2,] 0.80893  1.00000 0.06072 -0.04639
## [3,] 0.13132  0.06072 1.00000  0.10643
## [4,] 0.09157 -0.04639 0.10643  1.00000
```

11 Identity matrix

The identity matrix is a diagonal matrix with ones on the main diagonal. We can create it in two steps by first creating a square matrix of 0s and then replacing the main diagonal with 1s.

```
I <- matrix(0, nrow = 4, ncol = 4)
diag(I) <- 1
I

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

12 Multiplication By Diagonal Matrices

A diagonal matrix has zero values for all non-diagonal elements. It can be used to multiply the rows or columns (or both) of another matrix by constants.

12.1 Define Matrices

```
X <- matrix(c(4, 7, 7, 6), nrow = 2, ncol = 2, byrow = TRUE)
D <- diag(c(3, 2))
D
```

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

```
X
```

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    7    6
```

12.2 Post-Multiplication By A Diagonal Matrix

Post-multiplication of a matrix, X, by a diagonal matrix, D, results in the columns of X being multiplied by the corresponding diagonal element in D.

```
Y <- X %*% D
Y
```

```
##      [,1] [,2]
## [1,]   12   14
## [2,]   21   12
```

12.3 Pre-Multiplication By A Diagonal Matrix

Pre-multiplication of a matrix, X, by a diagonal matrix, D, results in the rows of X being multiplied by the corresponding diagonal element in D.

```
Y <- D %*% X
Y
```

```
##      [,1] [,2]
## [1,]   12   21
## [2,]   14   12
```

12.4 Scalar Multiplication Revisited

Scalar multiplication is just multiplication by a diagonal matrix with a constant in the diagonal.

```
X <- matrix(c(4, 7, 7, 6), nrow = 2, ncol = 2, byrow = TRUE)
D <- matrix(c(3, 0, 0, 3), nrow = 2, ncol = 2, byrow = TRUE)
X
```

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    7    6
```

```
D
```

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

3 * X

##      [,1] [,2]
## [1,]   12   21
## [2,]   21   18

X %*% D

##      [,1] [,2]
## [1,]   12   21
## [2,]   21   18

D %*% X

##      [,1] [,2]
## [1,]   12   21
## [2,]   21   18
```

13 The Determinant

Variance-covariance matrices and correlation matrices can be characterized by a single number called the determinant that represents the "generalized variance." For the correlation matrix, this number can take on values from 0 to 1. When all variables are independent (an identity matrix), the determinant is 1. As variables increase in their interdependence, the determinant approaches 0. A singular matrix has a determinant of 0. The determinant thus indexes the redundancy among variables in a correlation matrix. We can obtain the determinant using the det() function, here demonstrated using the sample variance-covariance matrix from an earlier example. The determinant of matrix A is symbolized as |A|. The context is usually sufficient to not confuse this with absolute value. If there is potential confusion, then $\det A$ ***would be used.***

```
det(VC_Sample)

## [1] 4.461

det(cov2cor((VC_Sample)))

## [1] 0.3185
```

14 The Inverse

Some square matrices have an inverse such that $AA^{-1} = I$***. The inverse is useful in solving matrix equations and is obtained in R by using the solve() function. We will use it here to obtain the regression coefficients from a multiple regression using matrix operations.***

First we will create some data consisting of 100 cases and 4 variables. The first variable will be treated as the outcome and the remaining variables treated as predictors.

```
mu <- matrix(0, nrow = 1, ncol = 4)
VC <- matrix(c(1, 1.2, 0.3, 0.4, 1.2, 2, 0.2, 0.3, 0.3, 0.2, 3, 0.4,
    0.4, 0.3, 0.4, 4), nrow = 4, ncol = 4, byrow = TRUE)
Data <- mvrnorm(100, mu, VC)
Data <- as.data.frame(Data)
names(Data) <- c("DV", "IV1", "IV2", "IV3")
```

Now we conduct a simple multiple regression using the lm() function. Our goal is to reproduce the regression coefficients using matrix operations.

```
fit <- lm(DV ~ 1 + IV1 + IV2 + IV3, data = Data)
summary(fit)

##
## Call:
## lm(formula = DV ~ 1 + IV1 + IV2 + IV3, data = Data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3336 -0.2178  0.0281  0.3333  1.4937
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0755     0.0521    1.45  0.15078
## IV1           0.5849     0.0364   16.06  < 2e-16
## IV2           0.0352     0.0293    1.20  0.23296
## IV3           0.1095     0.0298    3.67  0.00039
##
## Residual standard error: 0.513 on 96 degrees of freedom
## Multiple R-squared:  0.761,Adjusted R-squared:  0.753
## F-statistic:  102 on 3 and 96 DF,  p-value: <2e-16
```

The matrix equation for the regression coefficients is:

$$B = (X'X)^{-1}X'Y$$

in which X is the matrix of predictor values with a column of 1s added to account for the constant and Y is the vector of Y values. Note that the equation contains an inverse, in this case an inverse of a matrix formed by multiplying the transpose of X by X. The matrix operations produce identical values for the regression coefficients.

```
X <- Data[, 2:4]
X <- cbind(matrix(1, nrow = 100, ncol = 1), X)
X <- as.matrix(X)
colnames(X) <- c("Constant", "X1", "X2", "X3")
Y <- Data[, 1]
Y <- as.matrix(Y)
colnames(Y) <- c("Y")
B <- solve(t(X) %*% X) %*% t(X) %*% Y
B

##                Y
## Constant 0.07550
## X1       0.58494
```

```
## X2       0.03519
## X3       0.10946
```

15 Variances and Covariances of Linear Combinations

Quite frequently in statistics we will form linear combinations of variables and will need to know the variances and covariances of those linear combinations. Of course, one simple way to do that is to carry out the linear combination calculations, producing new variables, and then calculating the variances and covariances of those new variables. There is, however, a very useful matrix solution. If Y = XW, in which X is a matrix of data transformed by the weight matrix, W, to produce linear combinations, then the variance-covariance matrix for the transformed variables, Y, can be found with the matrix calculation: $W^{'}\Sigma_X W$, in which Σ_X is the variance-covariance matrix for the original variables.

To demonstrate, we first create some sample data consisting of 100 cases and 3 variables from a multivariate standard normal distribution with all intercorrelations assumed to be .2.

```
mu <- matrix(0, nrow = 1, ncol = 3)
VC <- matrix(c(1, 0.2, 0.2, 0.2, 1, 0.2, 0.2, 0.2, 1), nrow = 3, ncol = 3,
    byrow = TRUE)
X <- mvrnorm(100, mu, VC)
X <- as.matrix(X)
colnames(X) <- c("V1", "V2", "V3")
```

Next we create a matrix of weights, W, that will be used to transform the original variables.

$$W = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & -2 \\ 1 & -1 & 1 \end{pmatrix}$$

Because the data matrix is post-multiplied by W, this will create three new variables consisting of the sum of the original variables, the difference between the first and third variables, and the difference between the second variable and the sum of the first and third variables.

```
W <- matrix(c(1, 1, 1, 1, 0, -2, 1, -1, 1), nrow = 3, ncol = 3)
Y <- X %*% W
```

First, we can obtain the variances and covariances for the new variables using the cov() function.

```
cov(Y)

##        [,1]   [,2]   [,3]
## [1,]  4.646 -1.722  1.832
## [2,] -1.722  3.916 -1.129
## [3,]  1.832 -1.129  3.339
```

But, we can obtain those variances and covariances by transforming the variance-covariance matrix of the original variables (X) using the weights that create the new variables.

```
t(W) %*% cov(X) %*% W
```

```
##         [,1]   [,2]   [,3]
## [1,]  4.646 -1.722  1.832
## [2,] -1.722  3.916 -1.129
## [3,]  1.832 -1.129  3.339
```

This may seem like a trivial difference in the way the variances and covariances are derived, but it speaks to a very important link between original and transformed variables. Transformations do not add or subtract information but merely represent it differently. Whatever variability and correlation was present in the original variables is also present in the transformed variables, but it is distributed differently.