MATRICKS: Matrix algebra for Applesoft BASIC

JOHN R. VOKEY
University of Lethbridge, Lethbridge, Alberta, Canada

Many of the statistical techniques commonly used in psychology are most easily represented as matrix-algebraic equations. For example, the least squares solution for the coefficients of a multiple linear regression may be expressed succinctly as (e.g., Pedhazur, 1982)

\[ B = (X'X)^{-1}X'Y, \]

in which \( B \) is the column vector of unknown coefficients, \( X \) is the matrix of raw scores on the independent variables, \( Y \) is the column vector of the corresponding scores on the dependent variable, and \( X' \) is the transpose of \( X \). Unfortunately, most versions of the BASIC programming language available for microcomputers, including the floating-point version for the Apple II (Applesoft BASIC), do not include matrix-algebraic operations as intrinsic procedures of the language. Although it is possible to express these procedures in terms of the basic algebraic statements acceptable to the language, the resulting routines are long, require extensive debugging, and are often tied to a specific application or data set. Worse yet, they inevitably are slow, even for relatively small matrices.

MATRICKS, a collection of machine language routines, extends the Applesoft BASIC language to include the primary matrix-algebraic procedures. In addition, it includes other routines to facilitate matrix operations in general, and the programming of major statistical techniques in particular. It is compatible with the complete line of Apple II computers (i.e., the II with Applesoft resident in a language card, the II+, IIe, and IIE) running under either the DOS or the ProDOS operating system.

MATRICKS was designed with two purposes in mind: to allow for the direct expression of matrix-algebraic equations in Applesoft BASIC and for speed. It is interfaced to Applesoft BASIC via the ampersand (&) vector so that MATRICKS statements may be intermixed with standard Applesoft BASIC statements, even on the same program line, and may be used in either immediate or deferred mode. With respect to speed, procedures such as matrix inversion and multiplication are up to 5 times faster than their comparably coded BASIC counterparts, and other procedures, such as assigning a constant to all elements of a matrix or equating two matrices, can be over 10 times faster. Moreover, the ability to express matrix-algebraic equations directly results in a marked reduction in the length of the code.

MATRICKS DEFINITIONS AND COMMANDS

All MATRICKS commands are in the form of an equation, prefaced by the ampersand (&) character. The presence of this character in a command line (in either immediate or deferred mode) results in the Applesoft BASIC interpreter's initiating an unconditional jump to location $3F5, which, in turn, given that MATRICKS has been installed, passes control to the MATRICKS interpreter. MATRICKS then processes the equation and returns control to the BASIC interpreter.

Definitions

Every MATRICKS equation has the following general form:

\[ \&A = \text{(operation)} \]

in which "(operation)" represents the specific MATRICKS operation to be performed, and \( A \) is the matrix to which the result of the operation is to be assigned. Except where explicitly excluded, the same matrix may occur on both sides of the equation. To facilitate the exposition of the commands that follows, the definitions listed in Table 1 are used.

All matrices in a MATRICKS equation must be real and previously dimensioned to (only) two dimensions; the actual values of the dimensions are immaterial, except insofar as they are compatible with the equation and are not stated in the MATRICKS equation. Thus, for example, the equation 

\[ \&A = \text{IDN} \]

creates the identity matrix in the square matrix \( A \) regardless of whether \( A \) is dimensioned as, for instance, 2 rows and columns or 100 rows and columns. The first dimension dictates the number of rows of the matrix, and the second dimension dictates the number of columns. Neither dimension may exceed 255, although either or both may be zero; a matrix in which the first dimension is zero defines a row vector, and one in which the second dimension is zero defines a column vector. A matrix in which both dimensions are zero (defining a single element) may be used to receive the single-

<table>
<thead>
<tr>
<th>Table 1 MATRICKS Syntactic Definitions and Abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{op} = +, -, *, /, ^; \text{arithmetic operators.}</td>
</tr>
<tr>
<td>\text{A} = \text{B}, \ldots \text{any real, two-dimensional array in which the dimensions}</td>
</tr>
<tr>
<td>\text{are greater than or equal to 0, and less than or equal to 255}</td>
</tr>
<tr>
<td>\text{(expr)} = \text{arithmetic expression; any legal Applesoft BASIC expression that evaluates to a number; the expression must be enclosed in parentheses.}</td>
</tr>
<tr>
<td>\text{FNC} = \text{SGN, INT, ABS, USR, SQRT, RND, LOG, COS, SIN, TAN, ATN, Applesoft BASIC functions.}</td>
</tr>
<tr>
<td>\text{op} = +, -, NOT, unary operators.</td>
</tr>
</tbody>
</table>

Preparation of this article was supported in part by an operating grant from the Natural Sciences and Engineering Research Council of Canada to the author. Requests for reprints or the MATRICKS package should be addressed to J R Vokey, Department of Psychology, University of Lethbridge, Lethbridge, Alberta, Canada T1K 3M4.
number matrix product (i.e., the dot product) of a row and a column vector.

**Commands**

MATRICKS adds 13 new commands to the language. The majority of these are designed to simplify array handling and to replace the more cumbersome (and time-consuming) code required for the same operations in Applesoft BASIC. Of the remaining commands, 4 institute the primary matrix-algebraic procedures (i.e., identity, transpose, inverse, and matrix multiplication), and 2 are designed to facilitate construction of matrices common to many statistical techniques.

**Array-handling commands**

The following 7 MATRICKS commands are designed to simplify array handling in Applesoft BASIC:

1. The equation "&A = (expr)" assigns the result of "(expr)" to each element of the specified matrix. For example, the equation "&A = (0)" sets all elements of A to zero.

2. The equation "&A = op B" assigns the result of the specified unary operation on each element of B to the corresponding element of A. For example, the equation "&A = NOT B" assigns the logical NOT of each element of B to the corresponding element of A. Similarly, either "&A = B" or "&A = + B" may be used to equate two matrices.

3. The equation "&A = B aop (expr)" computes the result of the arithmetic operation and the "(expr)" on each element of B and assigns the result to the corresponding element of A. For example, the equation "&A = A / (2)" divides each element of the matrix by two.

4. The equation "&A = B aop C" performs the specified arithmetic operation between the two matrices on an element-by-element basis. For example, the equation "&A = B + C" assigns the arithmetic product of the corresponding elements of B and C to the corresponding elements of A.

5. The equation "&A = FNC(B)" computes the specified function unary operation of each element of the matrix. For example, the equation "&A = COS(B)" assigns the cosine of each element of B to the corresponding elements of A.

6. The equation "&A = PRNT, expr1, expr2" dumps the specified matrix to the current output device. Each row of the matrix is terminated by a carriage return. Printed field width of each element, which may be up to 127 characters in length, is specified by "expr1," and the number of displayed decimal places is specified by "expr2." The "print-using" algorithm is adapted from Daviduck (1985).

7. The equation "&A = DISP" clears the specified matrix from memory, leaving all other variables and arrays intact and freeing the memory it occupied for other data. The command name is short for "dispose," the Pascal language command with similar effect found in the Jensen and Wirth standard, and the algorithm is adapted from a more general utility of the same name developed by Kaner and Vokey (1985).

**Matrix-algebraic commands**

The following 4 MATRICKS commands provide for the primary matrix-algebraic procedures in Applesoft BASIC:

8. The equation "&A = IDN" creates the identity matrix in A.

9. The equation "&A = TRN(B)" computes the transpose of B and assigns the result to A.

10. The equation "&A = INV(B)" computes the inverse of B and assigns the result to A; B is returned as the identity matrix. The inversion is performed by a modified Gauss-Jordan elimination method that obviates the need for calculation of the determinant, and functions with zeros along the diagonal (e.g., Pachner, 1984).

11. The equation "&A = MULT(B,C)" computes the matrix-product of B and C and assigns the result to A.

**Statistical commands**

12. The equation "&A = SSCP(B)" updates the sums of squares of deviations/sums of squares of cross-products of deviations (SSCP) matrix A with the new data input from row vector B. The element A(0,0) contains the current number of cases in the SSCP matrix; the zero-order row element A(0,k) contains the current mean, and the zero-order column element A(k,0) contains the current variance, for each of the k variables in the matrix. The contents of A are updated using a provisional means algorithm in which the means, variances, sums of squares of deviations, and sums of squares of cross products of deviations of the variables are computed recursively (Spicer, 1972). This recursive calculation allows for the SSCP matrix to be constructed with only a single pass through the data and obviates the need to store the raw data for many common statistical procedures. Moreover, subsequent cases may be added to the matrix at any time, and the number that may be added is limited only by the resolution of Applesoft BASIC's floating-point number representation.

13. The equation "&A = CRR(B)" computes a correlation matrix from the SSCP matrix B and assigns the result to A.

**Error Messages**

For the most part, MATRICKS generates standard Applesoft BASIC error messages (e.g., "SYNTAX ERROR" if a MATRICKS statement is syntactically incorrect), particularly when it calls on internal Applesoft routines. However, certain errors are unique to MATRICKS. If MATRICKS fails to locate a specified matrix, for example, rather than dimensioning one (as Applesoft BASIC does), MATRICKS responds with an "OUT OF DATA ERROR." In the remaining cases, MATRICKS generates error messages that are composed of parts of standard Applesoft BASIC messages, flagging