MATRIX

MATRIX *matrix* = *matrix equation* ;

Function:

MATRIX processes matrix algebra expressions. Operations on matrices are specified in matrix equations preceded by the word MAT; these equations are just like the variable transformations performed by GENR, except for two things: they do not operate under control of the current SMPL and the results are stored as a matrix. The MAT procedure checks the matrices for conformability of the operations and gives an error message if the operation specified is not possible. Often printing the matrices in question will reveal why the operation cannot be performed.

Usage:

All the ordinary operators and functions used in TSP equations can also be used in the MAT command. They operate on an element-by-element basis (and hence require conforming matrices if they are binary operators). There is one important exception to this, the multiply operator * . For simplicity, this operator denotes the usual matrix multiplication, and element-by-element multiplication (the Hadamard product) is denoted by the operator % .

In the descriptions of the matrix operators that follow, we use the following symbols to denote the inputs and outputs of operations:

s = scalar or subscript variable.
i = integer scalar.
m = any matrix (if scalar, treated as 1 by 1 matrix).
qm = square matrix, N by N.
sm = symmetric matrix, assumed positive semi-definite.
dm = diagonal matrix, assumed positive semi-definite.
tm = upper-triangular matrix, assumed positive semi-definite.
v = column vector, N by 1.

Here are the symbolic operators understood by the MAT command (in addition to the ordinary operators used also in GENR). Remember that the operands *must* be conformable for the operations that you request; TSP will check the dimensions for you and refuse to perform the computation if this condition is violated.

m = *m***m*	matrix product
m = *m***s*	scalar multiplication (or s*m)
m = *m'*	matrix transpose
m = *m'm*	matrix transpose with implied matrix product
m = *qm"*	matrix inverse
m = *qm"m*	matrix inverse with implied matrix product
m = *m*#*m*	Kronecker product ($\otimes$)
m = *m*%*m*	Hadamard product (element by element)

When TSP processes a MAT command, it recognizes several operations where great savings of computation time can be made by eliminating duplicate calculations. These situations include, but are not limited to, the cross-product operation (which generates a symmetric matrix) and the calculation of a quadratic form (the expression A*B*A'). This occurs even when the arguments to these expressions are complicated expressions themselves. Thus, you should be careful to express any such complex arguments the same way whenever they appear in the matrix expression.

MATRIX

The following functions take matrices as their input and produce scalars as output. They may be used anywhere in a MAT statement where scalars are allowed, keeping in mind that a scalar is also a 1 by 1 matrix.

s = DET*(qm)*	determinant (truncated to zero when <1.E-37)
s = LOGDET*(qm)*	log of (positive) determinant, no truncation
s = TR*(qm)*	trace (sum of diagonal elements)
s = MIN*(m)*	element with minimum value
s = MAX*(m)*	element with maximum value
s = SUM*(m)*	sum of elements
i = NROW*(m)*	number of rows
i = NCOL*(m)*	number of columns
i = RANK*(m)*	rank (number of linearly independent columns or rows)

The following functions are matrix-to-matrix; that is, they take a matrix, perform some computation on it, and produce another matrix as output. They can be used anywhere in a MAT equation.

tm	= CHOL*(sm)*	Choleski factorization (matrix square root)
sm	= YINV*(sm)*	Positive semi-def. inverse via CHOL() for Hausman test
dm	= IDENT*(i)*	Creates an identity matrix of order *i*
v	= EIGVAL*(qm)*	Computes the vector of eigenvalues of *qm.* If *qm* is not symmetric positive semi-de finite, the imaginary parts of the eigenvalues are stored as @EIGVALI. Real eigenvalues are sorted in decreasing order. Complex eigenvalues are sorted by their norm.
qm	= EIGVEC*(qm)*	Computes the matrix of eigenvectors (columns). If *qm* is not symmetric positive semi-definite, the imaginary parts of the eigenvectors are stored as @EIGVECI
v	= VEC*(m)*	Creates a vector of all the elements of *m*, column by column
v	= VECH*(m)*	Creates a vector of all the unique elements of *m* column by column: *qm*: N*N elements *sm, tm*: N*(N+1)/2 elements *dm*: N elements
dm	= DIAG*(m)*	Creates a diagonal matrix from a matrix *qm, sm, tm*: take the diagonal from input matrix *v*: convert the vector to a diagonal matrix *s*: illegal, use s*IDENT(*i*) to create a diagonal matrix with s on the diagonal
sm	= SYM*(qm)*	Creates a symmetric matrix from a square matrix (the upper triangular elements are ignored)
m	= GEN*(qm)*	Creates a general matrix from a symmetric diagonal matrix
series	= SER*(v)*	Creates a series from a vector – same as UNMAKE v series;

Examples:

```
MAT B = (X'X)"X'Y;    produces OLS regression coefficients  (not a very accurate way to do this)
```

This example computes the Eicker-White estimate of the variance-covariance of the estimated coefficients after a regression:

```
OLSQ Y C X ;
MMAKE XMAT C X ;
MAT XXI =  (XMAT ' XMAT) " ;
MAT VCOV = (XMAT*XXI) ' DIAG(@RES**2) * (XMAT*XXI) ;
```

Output:

MATRIX

MATRIX produces no printed output. Typically, one matrix is stored in data storage.