PROC

PROC *procedure name* [*list of arguments*] ;

Function:

PROC defines a TSP user procedure. It is always the first statement in a procedure and must be matched by a corresponding ENDPROC statement.

Usage:

PROC gives the procedure name (any legal TSP name) and the list of "dummy" arguments for the procedure. When the procedure is called by specifying its name somewhere else in the TSP run, any dummy arguments which are used in the procedure are replaced by the actual arguments on the statement which invokes the procedure.

Always include an ENDPROC statement at the end of your procedure.

A user procedure can use any of the TSP variables which are at a higher level, that is, any variables in the procedure(s) which call it, or in the main TSP program. Variables which are created in a lower level procedure are not available after leaving that procedure unless they are the names of dummy arguments. In programming terminology, they are local variables.

When you enter a procedure, the last SMPL processed is in force. When you leave be careful to restore the SMPL if you have changed it during the procedure, or you may get unpredictable results if you call the PROC from different parts of the program.

If the name of a list is used as an argument to a PROC, the list is not expanded until it is used inside the PROC. This allows the number of items in the list to vary between PROC uses. The number of items in the list for a particular call can be determined by the LENGTH command. All PROC arguments are actually interpreted as lists within the PROC. This means if you print an argument, its original name will be shown (instead of the name of the argument within the PROC); it also helps in handling lagged arguments. If a PROC has a single argument, but is called with multiple arguments, the multiple arguments are automatically converted into a single list variable. You can also pass such "implicit lists" to PROCs with multiple arguments, but you have to separate each list/argument with a vertical bar (|).

If you have PROCs that you use in several different programs, you can use the INPUT or INPUT(NOPRINT) command to read them from separate files. TSP will also automatically search for such input files in a batch job, if you have used commands which are not defined. For example, if you have used the command KFHP X; but you have not explicitly defined a PROC KFHP, TSP will look for the file KFHP.TSP , and INPUT it if it is found. Besides the current directory, TSP will look in the same directories it searches for LOGIN.TSP (home directory on unix; install directory on PC).

Examples:

This simple example creates a dummy variable over a sample specified by START and STOP, which is one every SKIPth observation and zero elsewhere. This is an inefficient way to create a seasonal dummy, or year dummies for panel data

(it is better to use a repeating TREND and the DUMMY command). Note how the SMPL and FREQ are saved and restored on exit from the PROC.

```
PROC SKIPDUM START STOP SKIP VAR ;
   COPY @SMPL SMPSAV ; COPY @FREQ FRQSAV;
   SMPL START STOP ;
   GENR VAR = 0 ;
   DO I = START TO STOP BY SKIP ;
         SET VAR(I) = 1.0 ;
   ENDD ;
   FREQ FRQSAV;  SMPL SMPSAV ;
ENDPROC ;
```

The next example is a procedure to compute the prediction error for the classical linear regression model. The formula for the error variance is

$$s^2 (I + X_0(X'X)^{-1}X_0')$$

where s^2 is the estimate of the residual variance, X'X is the moment matrix for the data in the regression, and X_0 is the matrix of data for the prediction interval. This PROC is invoked by the statements

```
LIST VARLIST VAR1 VAR2 VAR3 ; ? Example with 3 variables
VARFORC ERROR VARLIST ;
```

Same as above, with an implicit list:

```
VARFORC ERROR | VAR1-VAR3;
```

The sample over which you wish the forecast and error must be specified before invoking VARFORC. VARFORC expects that the estimation of the model for which you are doing the forecast has just been executed and that @S2 and @VCOV contain the estimated variances of the residuals and coefficients. The series ERROR contains the prediction error for each observation on return. This example also passes a list as an argument.

```
PROC VARFORC PREDERR XLIST ;
   MMAKE X XLIST ;
   MAT  PREDERR = SER(SQRT(@S2 + VECH(DIAG(X*@VCOV*X'))));
ENDPROC ;
```

Here is a user procedure to compute Theil's inequality coefficient (U) as a normalized measure of forecast error:

```
PROC THEILU ACT PRED U ;
   GENR RESID = ACT-PRED ;
   MAT U = SQRT(RESID′RESID)/(ACT′ACT) ;
ENDPROC ;
```

Output:

PROC produces no output, unless your procedure generates output.