
DOT (CHAR=*nesting level character*, INDEX=*variable*, VALUE=*variable*)
list of sector names or strings ;

Function:

DOT is the first statement in a DOT loop, which is like a regular DO loop, except that the values of the index are a series of character strings (names). Each of these names is substituted in turn each time through the loop wherever the symbol dot (.) appears in a variable name. The dot may appear anywhere in the variable name.

Usage:

The primary use of DOT loops in TSP is the processing of multisectoral data, where the same group of statements is to be repeated on each sector. The names on the DOT command are the names of the sectors. They may also be integer numbers, which will be treated as the corresponding character string. Use the form ‘01’, ‘02’, etc. if you wish to include leading zeroes in the numbers.

Each DOT loop must be terminated by an ENDDOT statement. Any number of statements may appear between DOT and ENDDOT statements. TSP will cycle through them as many times as there are sectors on the DOT statement.

DOT loops may be nested up to five deep and more than one dot included in each variable name (i.e., VAR..). The names substituted for the dots are taken in the order that the DOT statements appear, one from each statement. See the second example below. If there are fewer dots in a name than loops, the innermost loop is used first.

The DOT procedure cannot be used in a LOAD section; however, it can be used when reading data from files.

Legal dotted variable names are A. , . , .VAR , .D. , and AL.B .

Illegal dotted variable names are .1 , 2. (numbers), and .EQ. (logical operator).

Options:

CHAR= special character to be used in conjunction with . in nested DOT loops, in order to specify explicitly which DOT statement is used to expand the period. It is normally used only to make a single . refer to the *outer* DOT loop when inside an *inner* loop. It must be a valid name character, such as A-Z, _ # % @ .

INDEX= scalar variable name to hold the values 1,2,3, ... during the DOT loop. This is like having a DOT loop and a DO loop at the same time.

VALUE=scalar variable name to hold the values of numeric DOT sectors.

Examples:

Suppose that you have data on the rate of change of wages (DW), the unemployment rate (U), and prices (P) for each of four countries: 1 (United States), 2 (England), 3 (Sweden), and 4 (Germany). When you create and load the data, give the series names like DW1, DW2, DW3, DW4, U1, U2, etc. Then you can easily run the same set of TSP statements on all four countries by means of a DOT loop. For example, to normalize the series to the same year compute the rate of change in prices, and run a regression of DW on the unemployment and rate of change of prices, use the following:

DOT

```
DOT 1-4 ;
    NORMAL P. 72 100 ;
    GENR DP. = LOG(P./P.(-1)) ;
    OLSQ DW. C U. DP. ;
    FRML EQ. DW.= A+B*U.+G*DP.**RHO ;
    PARAM A B G RHO ;
    LSQ EQ. ;
ENDDOT ;
```

If you wanted to work with data for the same four countries on prices and quantities of commodities, for example, food, housing, energy, etc., you could use the double dot construction:

```
DOT FOOD HOUS OIL ;
    DOT 1-4 ;    GENR S..=P..*Q..  ;     ENDDOT ;
    PRINT S.1-S.4 ;
ENDDOT ;
```

This example generates 12 series with the names SFOOD1, SFOOD2, SFOOD3, SFOOD4, SHOUS1-SHOUS4, and SOIL1-SOIL4, in that order. Each series is equal to the product of the corresponding price and quantity series. In the outer DOT loop, a table of the series for all the countries of each commodity is printed (note the use of the imbedded dot).

Here is another example with numbered sectors:

```
DOT(VALUE=J)  0-9 ;
    SELECT  COUNTRY=J;  OLSQ FISH C X ;   FIT= @FIT ;   SET B.=@COEF(2);
ENDDOT ;
SELECT 1;    PRINT COUNTRY FIT;   PRINT B0-B9;
```

This example regresses the same dependent variable, FISH, on C and X in separate samples for each of 10 countries (same as PANEL(BYID,ID=COUNTRY) FISH C X;). The fitted values are saved and printed together.

Here is another examples, showing the use of numeric character strings and double dots:

```
DOT '01' '10' '11' ;
    DOT FOOD HOUSE OIL ;
        GENR S.. = P. * Q.. ;
    ENDDOT ;
ENDDOT ;
```

In this case the variables S01FOOD, S10FOOD, ..., S01HOUSE, etc. are created using a single price index PFOOD, PHOUSE, POIL, for each set of series Q01FOOD, Q10FOOD, etc. Note the use of a single dot for the P variable.

This example takes a list of variables called VARS and regresses each of them on the others in the list one by one. Note the use of the INDEX and CHAR options.

```
LIST VARS X Y Z W ;
DOT(INDEX=I,CHAR=%) VARS ;
    DOT(INDEX=J) VARS ;
        IF I >= J ;  THEN ;     OLSQ .%  C .  ;
    ENDDOT ;
ENDDOT ;
```