Paper 2102-2015

Reversing the In Operator

Can Tongur, Statistics Sweden

ABSTRACT

The IN operator within the DATA step is used for searching a specific variable for some values, either numeric or character (for example, "if X in (2) then..."). This brief note explains how the opposite situation can be managed. That is, how to search for a specific value/-s in multiple variables by applying an ARRAY and the IN operator together.

INTRODUCTION

The IN operator in for numeric comparisons is well used in *Base SAS*®. For instance, a search for the existence of any specific values, say (0, 2 or 3), in a variable **X** in order to execute an action is denoted " IF X IN (0, 2, 3) THEN…". This returns an output IF-conditional on the existence of any argument specified through a listing (in parenthesis) to the IN-operator. The output is defined by some action syntax following the THEN statement. If THEN is not specified, the IN operator will serve as a logical expression, conditioning the entire output on the arguments to the IN operator.

A set search with the IN operator can also be done without explicit IF/THEN conditioning:
Y = X IN (0, 2, 3). In this case, IN operates as a Boolean type with binary response **Y**=1 if exists a 0, 2 or 3 in the variable **X**, else **Y** = 0.

The IN operator could thus be described as a search for multiple arguments in one single variable, i.e. a many-to-one search.

The IN operator cannot be reversed such that " IF (2) IN (X, Y, Z)…" since the input arguments to the IN operator are merely constant elements in an enclosed string that contains either numeric or character data. Hence, the IN operator cannot reference any argument with a wider scope, i.e. variables.

What about the opposite situation of when a single value is to be searched for in a set of variables? Or, when multiple values are to be searched for in a set of variables?

We will look at these two cases and simply refer to them as a "One-to-Many set search" and a "Many-to-Many sets search", respectively. An example based on numeric variables is given for each case although the operations are identical for character variables.

ONE-TO-MANY SET SEARCH

A viable solution to the problem of finding a single value in a set of variables is to use the ARRAY statement in the DATA step to set up an array that contains the variables to be searched for. Then, the IN operator is applied to search the variables included in the array. The IN operator is used without the explicit IF/THEN conditioning, as mentioned above, with binary response in some outcome variable. This approach is applicable to both numeric and character data and could simply be described as reversing the IN operator.

EXAMPLE 1

An array named for instance "**VARS**" is specified in a DATA step and contains some numeric variables **X**, **Y** and **Z**, which are to be searched for <u>one</u> specific value, let us say the value 2. The result of the following arrayed search is a binary response **1** in a specified outcome variable **W** if <u>any</u> of the input variables listed in the ARRAY statement contains the specified value, in this case the arbitrarily chosen value 2. Else ways, the outcome variable **W** is set to **0** if <u>none</u> of the listed variables contains the value specified.

For an input data set containing two observations identified by **OBS_ID**, and the numeric variables **X**, **Y** and **Z** according to

OBS_ID	X	Y	Z
1001	1	4	3
1002	8	2	6

a possible code syntax for a one-to-many arrayed search for the value 2 would be the following:

```
data Search_O2M;
  set InputData;
    ARRAY VARS X Y Z;
     W = (2) in VARS;
    put " W is " W;
run;
```

This operation will return the binary response 0 in the variable **W** for the first observation (**OBS_ID** = 1001) and the binary response 1 in the variable **W** for the second observation (**OBS_ID** = 1002).

Output 1 is from Example 1 and given by the PRINT procedure on the data set Search_O2M. It shows the indication of the searched value which is found in the second observation.

OBS_ID	X	Y	Z	W
1001	1	4	3	0
1002	8	2	6	1

Output 1. Output from a PROC PRINT statement with NOOBS for the DATA set Search_O2M

MANY-TO-MANY SETS SEARCH

A variety of the preceding case is to search for multiple values in multiple variables. In this case, two arrays must be specified: one array holding the variables that are to be searched in and one array containing the values to be searched for.

EXAMPLE 2

In the DATA step, an ARRAY named for instance "**VARS**" is specified and holds the numeric variables **X**, **Y** and **Z,** which may or may not contain the searched values of interest. Additionally, an ARRAY named for instance "**VALS**" is specified and contains some set of values, in this case numeric, say 9, 66, 12, 7, 5, 2 and 23, to be searched for in "**VARS**". Also, an outcome variable **W** is specified and assigned an arbitrarily chosen value, say 99, as default. Now, if any of the variables in the array "**VARS**" contains any of the values in the array "**VALS**", then **W** is assigned some indicator value, say 1.
Using the same input data as in Example 1, which contains the variables **X**, **Y** and **Z**, the code syntax for this operation could look like this:

```
data Search_M2M;
  set InputData;
    ARRAY VARS X Y Z;
    ARRAY VALS (7) (9 66 12 7 5 2 23);
    W = 99;
    do over VARS;
       if VARS{_I_} IN VALS then W = 1;
    end;
    put " W is " W;
   DROP VALS:;
run;
```

This operation will return the default value **W** = 99 for the first observation (**OBS_ID** = 1001) and the indicator value **W** = 1 for the second observation (**OBS_ID** = 1002) since one of the searched values specified in the array **VALS**, the value 2, is found in the variable **Y** only for the second observation.

Output 2 contains the result of the PRINT PROCEDURE on the data set Search_M2M.

OBS_ID	X	Y	Z	W
1001	1	4	3	99
1002	8	2	6	1

Output 2. Data set Search_M2M from the PROC PRINT statement with NOOBS

CAUTION ON THE ARRAY NAME

Note the syntax DROP VALS: used in Example 2. Since arrays are not variables but rather a kind of containers for variables, they cannot be used with DROP/KEEP/RENAME statements as these statements only apply to variables. The defined array VALS in Example 2 is specified to carry seven elements, so *SAS*® assigns each element a variable name with the prefix VALS; VALS1, VALS2, VALS3,…,VALS7. The DROP statement was applied to the prefix together with a semi-colon, VALS: and resulted in dropping only the variables created by *SAS*® for the array **VALS**.
Since the data set InputData did not contain any other variables named with the prefix VALS, the many-to-many set search was not affected by this as there were no input variables named VALS1, VALS2, VALS3,…,VALS7 that accidentally collided with the searched values in **VALS**. The array name should thus be chosen with precaution to other variable names in the data step.

CONCLUSION

The often encountered problem of searching for a value in multiple variables is solved by applying an ARRAY statement together with the IN operator in a DATA step. This approach can also be applied when searching for multiple values in multiple variables by using two arrays, one for the values and one for the variables. The solutions presented here apply to both numeric and character data.

REFERENCES

Matthews, G. (1999). Using ARRAYs: The Basics. *Proceedings of the Twenty-Fourth Annual SAS® Users Group International Conference*, SAS Institute Inc., Cary, NC.
Available at http://www2.sas.com/proceedings/sugi24/Posters/p215-24.pdf

First, S. and Schudrowitz, T. (2005). Arrays Made Easy: An Introduction to Arrays and Array Processing. *Proceedings of the Thirtieth Annual SAS® Users Group International Conference,* SAS Institute Inc., Cary, NC. Available at http://www2.sas.com/proceedings/sugi30/242-30.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Can Tongur
Statistics Sweden
Can.Tongur@scb.se
www.scb.se