SAS Tutorial: Arrays

Peter L. Rikard

Introduction

Arrays as implemented in SAS are good or bad depending on your prior programming experience. Fortunately for most SAS users (non programmers) they are good.

Arrays do NOT provide subscripted variables. Rather they provide the SAS programmer a single name to reference in programming statements that imply the use of a list of variables.

Note: all techniques are using SAS79.5, later versions of SAS may change the assumptions and methods.

The Array Statement

ARRAY arrayname (index) list of variables;

Where arrayname is a SAS name NOT used as a variable name, index while not required, is recommended and the list-of-variables is any list of variables of the same type.

EX: A dataset with numeric variables X1-X20 Y1-Y20 ABC R M Q could have any or all of:

- ARRAY Z (J) X1-X20;
- ARRAY ZZ(K) Y1-Y5;
- ARRAY ZZZ(L) X1 Y2 A X3-X9 M;
- ARRAY ABC X6-X9 X1-X4 X15-X20;
- ARRAY ABCD(J) Y1-Y20;

Notes:

(1) Variables can simultaneously be used in more than one array. (X1 in Z, ZZ, and ABC)

(2) Variables can be specified in any order in an array. (X1-X4 following X6-X9 in ABC)

(3) Arrays without an explicit index have the implied index, _I_.

(4) More than one array may use the same index variable. (J used in Z and ABCD)

(5) Arrays of different length MAY use the same index variable, however DON'T do it.

Using Arrays

Arrays are used as indirect reference to a variable in the list. If you wish to reference the third element of the array Z, you CANNOT say Z(3). The index variable for the array must have the value of 3 and then a reference to Z is a reference to the third element of Z.

1) DO index=beginning value TO ending value BY indexing value;

EX: DO J = 1 TO 20 BY 1;

2) DO index=vl,v2,v3,v4 to v5;

EX: DO K = 1,3,5,7,8 TO 11;

3) DO OVER arrayname;

EX: DO OVER ZZ;

Common use of arrays involve using DO and END statements. The DO/END pair of statements form a loop in the program, with statements between the two being executed one or more times. When execution of the loop stops, SAS goes to the next executable statement following the END statement. There are many forms of the DO, three forms are shown.

1) DO index=beginning value TO ending value BY indexing value;

EX: J=3; (setting the index)
IF Z=99 THEN Z=1000;
("Z" references the third element, X3)

2) DO index=vl,v2,v3,v4 to v5;

EX: DO K = 1,3,5,7,8 TO 11;

3) DO OVER arrayname;

EX: DO OVER ZZ;

There are a numerous uses of array statements, the ones below are for example only, as there may be other or better methods of accomplishing the tasks.
Recoding Variables

The recoding of variables is always a nuisance. This may include changing valid answers for "NON RESPONSE" and "NOT APPLICABLE" in a survey to missing values so they will not be included in statistics, it may include inverting responses to some questions, or any other task of changing the value of a variable to another for any reason. When the recode applies to only a single variable then we include an IF/THEN statement to do the recoding.

IF AGE > 65 THEN AGE = 65;

When there are large numbers of variables to which the same recoding applies then we can use arrays. If in the data above, all X's with a value of 9 are to be changed to missing, then instead of 20 IF/THEN statements we could:

DO J = 1 TO 20;
   IF Z = 9 THEN Z = . ;
END;

(the DO/END loop is executed 20 times, with the reference to Z implying reference successively to the 20 variables in the array list X1-X20. The first time through the loop, the if statement reads as:

IF X1 = 9 THEN X1 = . ;

and so on until the last time as: IF X20 = 9 THEN X20 = . ;

Inversion

If some of X's were coded 1-5, good to bad, and some were coded 1-5, bad to good we could invert one part of the variables so they were all 1-5, good to bad. If the X's to be inverted were X1,X3-X5,X15,X17-X20, then we could recode those with:

ARRAY RECODES X1 X3-X5 X15 X17-X20;
DO OVER RECODES:
   RECODES = 6 - RECODES;
END;

(The DO loop is executed 9 times, once for each of the variables in the array, RECODES in the statement refers successively to X1,X3,X4,X5,X15,X17,X18,X19,X20)

The 8's,9's as missing values must be recoded first.

Accumulating Totals

While this example could be coded using PROC SUMMARY, if the summing logic were not trivial it might best be done with arrays.

EX: With a dataset containing 30 variables X1-X30, responses to a questionnaire (with 9=not applicable, 0=no response) and two demographic variables SEX (1=male, 2=female) and RACE (1=white and 2=black). We wish to get the average response to each question by race and sex.

SET QUESTION END = EOF;
ARRAY X(J) X1-X30;
ARRAY MALESUM(J) T1-T30;
ARRAY FEMSUM(J) TTL-TT30;
ARRAY WSUM(J) TT1-TT30;
ARRAY BSUM(J) TT41-TT430;
(All variables T1--TTT30 are new to contain sums)
RETAIN T1--TTT30 0;

NOTE: You CANNOT RETAIN MALESUM... BSUM, these are arraynames not variables.

DO OVER X; (Or MALESUM...BSUM)
IF (X'=9 & X'='9) THEN DO;
   IF SEX = 1 THEN MALESUM+X;
   ELSE FEMSUM+X;
   IF RACE = 1 THEN WSUM+X;
   ELSE BSUM+X;
END;

(But wait we also need to count the number of good responses, so we must add 4 more arrays for that and 4 more sum statements. But wait, arrays may be lists of other arrays as well as variables (arrays of arrays) So....)

ARRAY ST1(J) S1-S30;
ARRAY ST2(J) S31-S60;
ARRAY ST3(J) S61-S90;
ARRAY ST4(J) S91-SS30;
ARRAY BIG(K) MALESUM FEMSUM
WSUM BSUM ST1 ST2 ST3 ST4;
RETAIN S1--SSSS30 0;
DO OVER X;
  IF X'=8 & X''=9 THEN DO;
    IF SEX=1 THEN K=1;
    ELSE K=2; (set index)
    BIG+X; (sum responses)
    K+4; (move index for count)
    BIG+1; (count responses)
  END;
END;

Setting the index variable, K, "selects" the array within BIG that we wish to sum to. Adding 4, moves the index to the count array for this summation. The DO OVER sets the appropriate value for J, the index within each array. The statement BIG+X, adds the Jth variable in the X array to the Jth variable in the Kth array. (It really is simpler than it looks).

Scoring an Exam

An instructor gives an exam and now must grade the results. Older style processing would imply one logical comparison for each question on the exam.

  IF ANSWER1='A' THEN SCORE=SCORE+1;
  OR
  IF ANSWER1=KEY1 THEN SCORE=SCORE+1;

Depending upon the skill of the programmer, every new test might imply a new program. With arrays the process becomes much more simple.

EX: Exams are given on OPSCAN sheets. The keys are on the first sheet and the exams on the following.

DATA SCORES;
  INFILE EXAMS;
  IF N=1 THEN DO;
    INPUT @10 (KEY1-KEY170)($1.);
    ARRAY KEYS(J) KEY1-KEY170;
    DO OVER KEYS;
      IF KEYS=' ', THEN GOTO ENDX;
    END,
    ENDX: QUESTNUM=J-1;
    RETAIN KEY1-KEY170 QUESTNUM;
    DELETE;
    END;

(we can now score the exams)

INPUT SSN 9.0 +1 (Q1-Q170)($1.);
ARRAY STUDENT(J) Q1-Q170;
SCORE=0;
DO J = 1 TO QUESTNUM;
  IF STUDENT=KEYS THEN SCORE=SCORE+1;
END;
AVERAGE=SCORE/QUESTNUM;
KEEP SSN SCORE AVERAGE;

(This scores any exam, with up to 170 questions with single column answers (character or numeric), where all questions have equal weight and for N questions the first N columns are used, i.e. no spaces between sections)

Recommendations

Arrays really are easy to use. It does take the normal user some time to become comfortable with using arrays. Therefore, you should experiment with using arrays until they become a tool that you can apply easily. If you find that you are coding statements over and over and over arrays may be the answer.)