Using SAS to Reformat Data Records from One to Several

Last revised: 08/19/99

Researchers frequently wish to analyze secondary data in a format that is different from the publicly released version. In the following example, data from a roster of household members is reformatted to separate records for each child aged 0 to 17 years. The data are taken from the National Survey of Families and Households, Wave 2. The household roster included information on each person in the household:

- Name, sex, age, marital status and whether married before, relation to R
- Whether each of R's biological children is also a biological child of spouse/partner
- Education, working full or part time, enrolled in last four months and school type, economic situation, number of children (and number living here)
- If absent spouse: reason for absence.

The variables used in this example are:

MA1	Main respondent's ID number (5 digits)
MA7	Main respondent's sex (1=Male, 2=Female)
MA8	Main respondent's age (in years)
MB1NUM	Number of additional people in household
MB4Pxx	Sex of household member (1=Male, 2=Female, xx ranges 1 to 16)
MB5Pxx	Age of household member (in years, xx ranges 1 to 16)

The first three cases look like:

MA1	MA7	MA8	MB1NUM	MB4P01	MB5P01	MB4P02	MB5P02	MB4P03	MB5P03	MB4P04	MB5P04	MB4P05	MB5P05	MB4P06-16	MB5P06-16
3	1	56	5	2	58	1	39	1	22	1	9	1	10	9	99
11	2	85	0	9	99	9	99	9	99	9	99	9	99	9	99
29	2	62	2	2	6	2	5	9	99	9	99	9	99	9	99

where all data for the 6th through 16th household members are missing (MB4Pxx (sex) is coded 9 and MB5Pxx (age) is coded 99).

The result of reformatting the data will look like:

MA1	MA7	MA8	MB1NUM	SEX	AGE
3	1	56	5	1	9
3	1	56	5	1	10
29	2	62	2	2	6
29	2	62	2	2	5

where there is one record for each household member aged 0 to 17 years that includes selected characteristics of the main respondent. In the example shown, there are two records each for main respondent (MA1) 3 and 29. Main respondent (MA1) 11 lives alone (MB1NUM = 0). Note that for main respondent (MA1) 3 only 2 of the 5 additional household members are aged 0 to 17 and included in the reformatted data. Both additional household members are in this age group for main respondent (MA1) 29 and included in the reformatted data.

The following SAS program uses Arrays and Do-loops in a Data step to process the roster data and reformat the data. The arrays serve as shortcuts to the lists of personal characteristics that are repeated for each household member and the do-loops control the processing of data for each household member in turn.

Here is the data step for reformatting the data that has already been read into a SAS data set called NSFH2:

```
data rkids (keep=ma1 ma7 ma8 mb1num sex age); Î
     set nsfh2;
     array asex {16} mb4p01 - mb4p16;     * sex;Ï
     array aage {16} mb5p01 - mb5p16;     * age;

     do i=1 to mb1num;Ð
          sex = asex{i};Ñ
          age = aage{i};

*** select records for hh members between age 0 and 17 (other than
R);

          if (0 le age le 17) then output;Ò
     end; Ó
run;
```

Î On the new data set, only the information for the main respondent and the sex and age of the additional household members is kept. That is, we are dropping 32 variables (16 sex variables MB4P01 – MB4P16 and 16 age variables MB5P01 – MB5P16).

Ï The ARRAY statement defines a nickname for a list of variables (or values). The array names (asex and aage in this example) cannot be the same as any variable name used in the data set. This example uses the “a” prefix on the sex and age arrays, so that variables named “sex” and “age” can be created below. After the array name the

ARRAY statement includes the number of array elements (variables) included in the array inside braces {}. This is followed by the list of variables that make up the array.

Ð The DO statement starts the looping process where all of the characteristics (here, sex and age) are processed for each household member listed in turn. An index variable, i, is used to refer to the array elements and counts the number of times the loop is executed. The loop is set to be executed once for each member of the household (other than the main respondent). Note that for main respondent 11 this DO statement evaluates to:

```
do i=1 to 0;
```

without generating any errors or warnings.

If the data did not contain an indicator variable for the number of household members (MB1NUM), this command could have been written:

```
do i=1 to 16;
```

and the loop would have been executed 16 times for every case. While this would work in this situation, it is less efficient than limiting the number of times the loop is executed to the number of eligible household members.

Ñ Here two new variables are created, sex and age, for the current household member. The variables are equal to the current value of the array for the current iteration of the loop. This will be shown in more detail below.

Ò The OUTPUT statement within the loop allows one output record for each iteration of the loop, even though only one record was read. Here it is conditional on the age criteria that household members be between 0 (less than 1 year of age) and 17 years of age. All of the variables listed on the KEEP= statement (see **Î**) are written to the output file, so the household or main respondent characteristics are written out to each record with the variables (age and sex) for the current household member being processed.

Ó END closes the DO loop. Every DO loop must match to an END or your program will not run.

Let's take the first case, main respondent 3, and track the looping process. For this record MB1NUM = 5, thus, the DO command is evaluated as:

```
do i=1 to 5;
```

so the loop is executed five times.

When i=1:

ASEX{1} = MB4P01 → SEX = 2
AAGE{1} = MB5P01 → AGE = 58 → OUTPUT = No.

When i=2:

ASEX{2} = MB4P02 → SEX = 1
AAGE{2} = MB5P02 → AGE = 39 → OUTPUT = No.

When i=3:

ASEX{3} = MB4P03 → SEX = 1
AAGE{3} = MB5P03 → AGE = 22 → OUTPUT = No.

When i=4:

ASEX{4} = MB4P04 → SEX = 1
AAGE{4} = MB5P04 → AGE = 9 → OUTPUT = Yes.

When i=5:

ASEX{5} = MB4P05 → SEX = 1
AAGE{5} = MB5P05 → AGE = 10 → OUTPUT = Yes.

After the loop is executed the designated number of times the next record is read from the input file and the program continues until all the input records are processed. This results in a data file like the one shown at the bottom of page 1.

As a final note, the example shown here uses the most basic array specification. As shown, these were one-dimensional arrays. Advanced programmers can achieve further efficiency gains by using multi-dimensional arrays.

Recommended Sources

SAS Language: Reference, Version 6, First Edition.
SAS Applications Guide, 1987 Edition.
Combining and Modifying SAS Data Sets: Examples, Version 6.
SAS Programming Tips: A Guide to Efficient SAS Processing.
SAS Programming by Example (Cody and Pass, 1995).

SSCC has a complete set of SAS documentation. These documents are circulated by the CDE Print/Virtual Library in 4457 Social Science.