PhUSE 2014

Paper CC08

The Use of ARRAYs in SAS

Ruurd Bennink, OCS Consulting B.V., 's-Hertogenbosch, the Netherlands

ABSTRACT

This paper shows 5 situations on which the use of ARRAYs is quite practical, and in my personal opinion the most appropriate. Each purpose is explained by one or more examples of actual SAS code. All examples are gained from daily life experience as a statistical SAS programmer in the Life Sciences industry.

The reason that I came up with this topic for the PhUSE Conference is that I'm quite impressed by the elegancy of using ARRAYs in SAS since the day I started as a SAS programmer almost 14 years ago. When I started I was trained on the job and my tutor at the time also emphasised that ARRAYs are such a nice tool of SAS. It is since then that I first seek a solution by using ARRAYs in all of my programs.

INTRODUCTION

The purpose of this paper is to show that the use of ARRAYs in SAS is relatively easy and transparent as compared to SAS macro language, PROC SQL and PROC TRANSPOSE. It is also a SAS tool that did not, to my own knowledge, undergo any changes throughout all the different SAS versions over the past decades.

Each section of this paper presents an occasion where the use of ARRAYs is appropriate. It will be explained by showing one or more examples of the actual SAS code that solved the problem.

The sections are:

1) From N variables to N records
2) Same operation on many variables
3) Creating N derived variables in one data step
4) Creating N derived variables from N original variables
5) Perform analysis on datasets with a lot of variables

Each example is based on experience as a SAS programmer in the Life Sciences industry, so their context will of course also briefly be explained, because not all readers may be familiar with the corresponding issues.

ARRAYS IN SAS

ARRAYs in SAS are very different from arrays in other programming languages. In most (3rd generation) programming languages, arrays can be seen as a 'box' or a collection of values within a single variable, referred to as `variable[1]` or `variable["number"]`. In SAS, an array refers to already existing or new variables (or columns) that are all unique variables of their own. The ARRAY is then used to refer to these variables in a way that, for example, allows you to run a programming loop (`if`, `while`, `do until`) over the variables in the array.

A restriction is that all the variables included in the SAS ARRAY must be of the same type; all numeric, or all character. Other properties such as formats and labels may vary.

ARRAYs are used within SAS data steps and have the following general syntax:

```
DATA y;
   SET x;

   ARRAY abc (3) a b c;

   <SAS statements>…
RUN;
```

In this data step example, the variables a, b and c are in the input dataset A. The SAS statements are used to achieve any of the five points mentioned above.

1. FROM N VARIABLES TO N REC

EXAMPLE 1.1: VISIT DEPENDENT POPULATIONS

In the research sections of Life Sciences, staff is mainly involved in conducting clinical trials. New medications are tested on healthy volunteers (Phase 1) or real patients who suffer from a certain disease that is supposed to be cured by the examined medication in the respective clinical study (Phase II and III).

In each clinical study the group of subjects who enter the study is being subdivided in subgroups. Roughly there are two groups:

- All subjects who follow the entire study without any violation of the study protocol (PP).
- All subjects who enter the study (ITT).

Recently statisticians and other involved staff found this subdivision too limited, because the subjects in the ITT may have violated the protocol only on one occasion during the entire study and still will be dropped from the PP. That is why 'Visit dependent populations' were constructed, so that a subject can be dropped from the PP at Visit 3, but may be included in the PP at all other visits.

The SAS code introduced in this example is an alternative to PROC TRANSPOSE, where the use of SAS ARRAYs give you more control over which records to output (and, if desired, to which dataset), which additional data to include using a MERGE statement, and provides room for other modifications to be made to the records.

This part of SAS code will achieve this:

```
DATA popul;
   KEEP subjid dvisitnum itt pp product;
   MERGE inp_fa.itt_pp
         ad_fa.adsl (Keep=subjid product);
   BY subjid;

   ARRAY pops (4) pprotbl pprotv2 pprotv3 pprotv4;

   DO dvisitnum = 1 TO 4;
      pp = pops(dvisitnum);
      OUTPUT;
   END;

RUN;
```

The dataset with population flags (INP_FA.ITT_PP) contains the visit dependent PP population flags for the four visits of this specific study: pprotbl pprotv2 pprotv3 pprotv4. This was not very practical for the remaining analyses to be done, because each subject should have one record for each visit with this population flag. The dataset AD_FA.ADSL is included in the MERGE, because this dataset contains the PRODUCT information for all subjects. The PRODUCT the subjects were randomised to is important for all further analyses.

Crucial in this program is the OUTPUT statement, indicating that a record will be created for each visit per subject with the new population variable PP.

Input dataset INP_FA.ITT_PP has the following general structure:

SUBJID	PRODUCT	ITT	PPROTBL	PPROTV2	PPROTV3	PPROTV4
1	A	YES	YES	YES	NO	YES

Corresponding output dataset POPUL has then the following general structure:

SUBJID	PRODUCT	DVISITNUM	ITT	PP
1	A	1	YES	YES
1	A	2	YES	YES
1	A	3	YES	NO
1	A	4	YES	YES

In the DO loop statement, you may choose to use the DIM function (e.g. `DO i = 1 to dim(pops)`) to determine the size of the array. This may allow for more generic code to be created and decreases the risk of variables being missed in case they are added to the ARRAY statement, where you may overlook that you have to update the DO statement as well.

2. SAME OPERATION ON MANY VARIABLES

EXAMPLE 2.1: REPLACING BLANKS

It sometimes occurs that a dataset contains completely missing values. That is not a problem for a SAS programmer if it correctly reflects the data as they are, but it may be for a researcher. They may want to replace the blanks by 'NA' (=Not Applicable) to give the reader of the report some explanation of why these values are missing. This is achieved by the following piece of SAS code:

```
DATA tab111;
   SET tab111_pval
       tab111_sumstat;

   ARRAY fillin (23) val4 – val26;

   DO i = 1 TO 23;
      IF (Missing(fillin(i))) THEN fillin(i) = 'NA';
   END;

RUN;
```

The twenty three variables VAL4 to VAL26 are elements of array FILLIN and are replaced by 'NA' in the DO loop only if they are originally *missing*.

EXAMPLE 2.2: RENAMING VARIABLES

If the output dataset of some standard macro gives only standard variable names as VAL1, VAL2 etc then it may be convenient to change these variable names to make them better interpretable. The next piece of SAS code achieves this:

```
DATA tab14_sumstat_wk26_2;
   LENGTH wk26_val1 - wk26_val6 $32;
   SET tab14_sumstat_wk26;

   ARRAY orgname (6) $ val1 - val6;
   ARRAY newname (6) $ wk26_val1 - wk26_val6;

   DO i = 1 TO 6;
      newname(i) = orgname(i);
   END;

   DROP val1 - val6;

RUN;
```

Variables VAL1 to VAL6 are renamed to WK26_VAL1 to WK26_VAL6. Of course, it is not an actual RENAME, but a replacement of existing variables. An advantage of this approach is, for example, that the properties of the newly created variables, such as length and format, can easily be redefined without touching the original variables.

3. CREATING N DERIVED VARIABLES IN ONE DATA STEP

EXAMPLE 3.1: CUMULATIVE INCIDENCE OF A CERTAIN ADVERSE EVENT

In a clinical study the analysis may be focused on the time-point when a certain event occurs after the first intake of the study medication e.g. when it is recorded on each day if the subject had a headache as a yes/no variable. In this case the analysis is to compare the percentage of subjects that had the incidence of a certain event before a certain time-point until the end of the study between the two treatment groups (the test product vs. the control product). This means that e.g. if a subject has the event after 6 days on study product, the indicator flag will remain 'Yes' from 6 days on until the end of the study for that subject. This is called cumulative incidence.

To achieve that goal the following SAS code was used:

```
PROC SORT Data=ae_2 Out=ae_3;
   BY subjid Descending AE_YN;
RUN;

PROC SORT Data=ae_3 Out=ae_4 Nodupkey;
   BY subjid;
RUN;

DATA cumul;
   KEEP subjid grp stday:;
   SET ae_4;

   FORMAT stday: yesnodv.;

   ARRAY startday (*) stday3 - stday129;

   DO i = 1 TO Dim(startday);
      IF (i >= dayonsp) AND (AE_YN = 1) THEN startday(i) = 1;
      ELSE startday(i) = 0;
   END;
```

```
RUN;
```

The PROC SORT in the first two steps is conducted to keep only the first occurrence of the event. Because 'No' is coded as 0 and 'Yes' as 1, the option 'Descending' is used.

The input dataset AE_2 has the following structure (for variable AE_YN: 0 = NO, 1 = YES):

SUBJID	GRP	DAYONSP	AE_YN
1	ACTIVE	3	0
1	ACTIVE	4	0
1	ACTIVE	5	0
1	ACTIVE	6	1
1	ACTIVE	7	0
.	ACTIVE	.	.
1	ACTIVE	129	0

Here the study duration was 129 days. Each day was a record for each subject in input dataset DSIN_2. The output dataset CUMUL has one record for each subject with 127 (starting at day 3) variables STDAY3 to STDAY129, each containing a 0 (= NO) or 1 (= YES).

The output dataset CUMUL will have only one record for each subject and then has the following structure, based on the input dataset AE_2 from above:

SUBJID	GRP	STDAY3	STDAY4	STDAY5	STDAY6	STDAY7	STDAY8	…	STDAY129
1	ACTIVE	0	0	0	1	1	1	1	1

4. CREATING N DERIVED VARIABLES FROM N ORIGINAL VARIABLES

EXAMPLE 4.1: CHANGE FROM BASELINE

In almost all clinical studies the main focus is on how the study medication affects a subject. For that purpose measurements are being done just before the first intake of the study medication (baseline) and then subsequent measurements thereafter. The difference between the values measured after baseline with the baseline values are the 'Change from baseline'.

In many cases the only interest is in one parameter, e.g. blood pressure. So then blood pressure is measured at baseline and a few times thereafter and as a next step the change from baseline is calculated at each time-point after baseline. In this clinical study instead of only one parameter 12 cholesterol parameter levels were measured in blood samples and for each of these parameters the change from baseline was determined.

```
DATA chol2;
   MERGE chol0
         chol1
         derived.infchar;
   BY identity;

   ARRAY blvalues  (12) ffa_bl tchol_bl tg_bl hdlc_bl ldlc_bl hdl_ldl_bl /* etc
*/;
   ARRAY values    (12) ffa tchol tg hdlc ldlc hdl_ldl /* etc */;
   ARRAY cblvalues (12) ffa_cbl tchol_cbl tg_cbl hdlc_cbl /* etc */;
   ARRAY perccbl (12) ffa_pcbl tchol_pcbl tg_pcbl hdlc_pcbl ldlc_pcbl
hdl_ldl_pcbl /* etc */;

  DO i = 1 TO 12;
          cblvalues(i) = values(i) - blvalues(i);
```

```
          perccbl(i) = Round((cblvalues(i)/blvalues(i))*100,0.1);
  END;

RUN;
```

The ARRAY BLVALUES contains all twelve baseline values, the ARRAY VALUES contains the values measured after baseline (in this example there is only one measurement after baseline). The ARRAYS CBLVALUES and PERCCBL contain new variables created in the DO loop, respectively the absolute change from baseline and the relative (percentage) change from baseline.

5. PERFORM ANALYSIS ON DATASETS WITH A LOT OF VARIABLES

EXAMPLE 5.1: MANY EXOTIC LAB PARAMETERS

Especially in the nutrition industry the search for an effect may lead to datasets with quite a huge number of laboratory blood or faecal sample test results. In the clinical study this example is based on there was a search for some effect on the immune system and therefore more than 100 parameters were measured. All of these test results had variable names that were really difficult to pronounce and in first instance not interpretable, so a way had to be found to create a dataset that contained all these variable names and use this dataset as input for further analyses, just to not have to type in any of those variable names.

This example provides an alternative to PROC CONTENTS, or PROC SQL with the INTO statement on DICTIONARY.TABLES.

```
DATA extra;
   LENGTH name $32;
   KEEP name j;
   SET pbmc;

   /* all numeric variables in PBMC */
   ARRAY def{*} _numeric_;

    DO j = 1 to Dim(def);
         /* get name of numeric variable */
      CALL VNAME(def{j}, name);
         /* write name to an observation */
      OUTPUT;
   END;
   STOP;
RUN;
```

The input dataset PBMC contains all the exotic lab test results. Because it was unknown how many, the asterisk * was used as a wild card and only the numeric variables were selected by using _NUMERIC_. The DO loop uses the DIM function to obtain the number of ARRAY elements. This DO LOOP creates a dataset with a record for each variable name as a character string of length $32. So the dataset EXTRA has two columns; j to indicate the variable ‘counter’ and NAME. The dataset contains as many records as there are numeric variables in the input dataset, each record containing a unique variable name with its counter.
The STOP statement is used to indicate that the DO loop should stop execution after one record. An alternative could be to use IF (_N_ = 1); at the start of the DO loop.

In a next data step CALL SYMPUT was used to obtain these variable names as a macro variable linked with the unique counter ‘j’. As a next step these were later used within a %DO loop. Within this %DO loop a SAS procedure as for example PROC MIXED may be used to conduct further analysis on each variable.

```
DATA _NULL_;
   SET extra END=LAST;
   CALL SYMPUT('var'||Left(j), name);
   IF LAST THEN CALL SYMPUT('N', Left(j));
RUN;

%MACRO multivar;

%DO j = 1 %TO &N;
   PROC MIXED..;
%END;
```

CONCLUSION

There are many programming challenges that may ask for the use of SAS ARRAYs as an alternative to commonly used procedures such as PROC CONTENTS, TRANSPOSE, or SQL. In this paper only some examples are presented that I came across with during my daily life practice in the Life Science industry. The number of possibilities and applications for ARRAYs are practically endless. Hopefully this paper has convinced you of the elegancy and transparency of SAS ARRAYs.

CONTACT DETAILS

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

Ruurd Bennink
OCS Consulting B.V.
P.O. Box 3434
5203 DK ‘s-Hertogenbosch
The Netherlands
+31 (0)73 523 6000
sasquestion@ocs-consulting.com
http://www.ocs-consulting.com/nl