

Arrays and Scoring Tests Involving Multidimensional Criteria

Imelda C. Go, Richland County School District One, Columbia, SC

ABSTRACT

In educational applications, a step that is sometimes required after scoring responses to a test is looking up values of other measures that correspond to a test score. For example, scale scores can correspond to raw test scores. Scale scores can also depend on the test form that was used. There might even be the need to obtain several types of scale scores based on test sections. Unless you know how these scale scores are obtained computationally, these situations conjure images of multi-dimensional lookup tables. In either one-dimensional or multi-dimensional settings, temporary SAS® arrays provide an elegantly simple and efficient means for storing test answer keys and large lookup tables required for such scoring processes. Using SAS arrays has definite advantages over using other SAS statements (e.g., SET, RETAIN, IF-THEN/ELSE, SELECT, etc.) to accomplish the same task.

INTRODUCTION

Lookup tables printed on paper are convenient and do not require the user to know exactly how entries in the lookup table were computed. This paper illustrates the use of temporary arrays to obtain a score based on one or more dimensions. There are, of course, other ways to score a test, but this particular technique has certain advantages that will become apparent.

EXAMPLE WITH ONE DIMENSION

Consider a 3-item test with responses stored in the following SAS variables, and the answer key to the three items.

	Item 1	Item 2	Item 3
Response Variable	r1	r2	r3
Answer Key	B	D	C

```
data one;
input name $4. +1 (r1-r3) ($1.);
array answer {3} $1 _temporary_ ('B','D','C');
array response r1-r3;
score=0;
do _i=1 to 3;
  if answer(_i)=response(_i) then score+1;
end;
cards;
Dick BCD
Jane ADC
Spot BDC
;
proc print;
```

The following output is produced:

OBS	NAME	R1	R2	R3	SCORE
1	Dick	B	C	D	1
2	Jane	A	D	C	2
3	Spot	B	D	C	3

The SAS code involves a temporary array (answer) and the automatic temporary array index (_i_). Hence, you do not need to use a DROP or KEEP statement to eliminate superfluous variables that result from using non-temporary arrays or array indexes. Temporary array elements are

particularly useful when their values are mainly used for computations. If a temporary array element needs to be retained, it can be assigned to a variable.

EXAMPLE WITH TWO DIMENSIONS

Consider the situation where the aim is to obtain an adjusted score based on a student's raw score on a test and age in years.

Variable Name	Definition	Range of Values
score	raw score on a test	1, 2, 3, 4, 5
adjusted	adjusted raw score	2, 3, 4, 5
age	student's age in years	13, 14, 15

The following lookup table for the adjusted score (shaded cells) is associated with this situation:

Rule for Obtaining Adjusted Score	Age	Raw Score on a Test				
		1	2	3	4	5
raw score + 3 (not to exceed 5)	13	4	5	5	5	5
raw score + 2 (not to exceed 5)	14	3	4	5	5	5
raw score + 1 (not to exceed 5)	15	2	3	4	5	5

Note that the rules for obtaining the adjusted score depend on the age. Obviously, people who use lookup tables may not always know how the lookup table was created.

The following code represents the lookup table as a two-dimensional array representation that is visually intuitive. You may also specify the range of the array indexes. In this example, the first array dimension has 13, 14, and 15 as possible index values. The second array dimension has 1, 2, 3, 4, and 5 as possible index values.

```
data one;
input age score;
array answer {13:15, 1:5} _temporary_
  (4 5 5 5 5
   3 4 5 5 5
   2 3 4 5 5);
adjusted=answer(age,score);
```

Let's revise the above situation slightly and suppose the ages are in increments of half a year instead:

Age	Raw Score on a Test				
	1	2	3	4	5
12.5	4	5	5	5	5
13	3	4	5	5	5
13.5	2	3	4	5	5

SAS arrays can be indexed by nonnegative integers (0, 1, 2, 3, ...). Index values may start with 0. If you provide an index value that is not an integer, SAS will truncate it by applying the INT function that returns the argument's integer portion. If 0 is not in the range of your index values, .999 will produce an "Array subscript out of range" error because INT(.999)=0.

In order to deal with this new situation, one strategy is to assign the values of a variable Y as they correspond to the values of age as shown in the following table. You can use Y as the array index instead.

X = Age	Y = Index
12.5	1
13	2
13.5	3

In reality, $Y = 2X - 24$. A more formal way to arrive at this equation is to use a little algebra. The X and Y values in the preceding table have a linear relationship. Consider the following representations:

X_1 is the first X value
 X_{last} is the last X value
 Y_1 is the first array index value
 Y_{last} is the last array index value

When the correspondence between X and Y is linear, you can use the following equation to compute Y:

$$Y = mX + b \text{ where}$$

$$m = \frac{Y_{last} - Y_1}{X_{last} - X_1}$$

$$\text{and } b = Y_{last} - mX_{last}.$$

In the previous example, $X_1=12.5$, $X_{last}=13.5$, $Y_1=1$, and $Y_{last}=3$. Based on the previous equations, $Y = 2X - 24$. In general, you may also need to round off Y. For example, it is possible to compute .999 when the index value should be 1.

Now if the relationship between X and Y is not linear, you may have X and Y values such as the following:

X = Age	Y = Index
12.5	1
13.5	2
15	3

In this case, the linear equation above would be inappropriate. One way to deal with this is to transform X so that X and Y have a linear relationship. A more straightforward way is to use conditional statements IF-THEN/ELSE or SELECT to assign the Y index values based on the X values.

Another way is to create a data set with the X values and the corresponding Y values. You can merge this data set (by X) with the essential data set that contains X. The merged data set would have the X and Y values. Consequently, you may now use the Y values in the merged data set to access the array elements.

The idea is to be able to come up with an index that can be used to access the elements in the array. There are many ways to approach this task.

If the above methods do not appeal to you, a method that does not involve arrays uses conditional IF-THEN/ELSE or SELECT statements to obtain the adjusted score based on the raw score and age. Other solutions could involve PROC SQL, etc.

EXAMPLE WITH THREE DIMENSIONS

Building on the previous example, let's add another dimension: the student's sex. This results in two lookup tables for the adjusted score:

Female Student's Raw Score					
Age	1	2	3	4	5
12.5	4	4	5	5	5
13	3	4	4	5	5
13.5	2	3	4	4	5

Male Student's Raw Score					
Age	1	2	3	4	5
12.5	4	5	5	5	5
13	3	4	5	5	5
13.5	2	3	4	5	5

A single array representation of the lookup tables requires adding a third dimension to the array:

```
array answer (1:2,0:3,0:6) _temporary_
(1 1 2 3 4 5 .
1 4 4 5 5 5 12.5
1 3 4 4 5 5 13.0
1 2 3 4 4 5 13.5
2 1 2 3 4 5 .
2 4 5 5 5 5 12.5
2 3 4 5 5 5 13.0
2 2 3 4 5 5 13.5);
if sex='F' then sexindex=1; else sexindex=2;
adjusted=answer(sexindex,2*age-24,score);
```

The first array dimension is the student's sex and its index values are 1 (for female) and 2 (for male). Unfortunately, the variables in an array must be all numeric or be all character. The second dimension is the age and its index values go from 0 till 3. The third dimension is the raw score and its index values go from 0 till 6.

For this example, the array elements that have actual raw score and age values were included to make the array statement more readable and resemble the lookup table it represents. These extra elements may be omitted because they serve no computational purpose. One may also wish to deal with two two-dimensional arrays instead of one three-dimensional array.

CONCLUSION

Temporary SAS arrays provide an elegantly simple way of dealing with scoring applications. Using temporary SAS arrays have certain advantages. Temporary data elements do not require a DROP or KEEP statement to eliminate superfluous variables. There are various ways to write a SAS array statement, but it is possible to write them in such a way that they look similar to a lookup table. In contrast, this is not quite possible with other SAS statements such as conditional IF-THEN/ELSE, and SELECT statements.

The paper illustrates an application of SAS arrays. It does not claim that the illustrations provide the most ideal or only way to deal with scoring applications.

REFERENCES

SAS Institute Inc. (1990), *SAS Language Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

TRADEMARK NOTICE

SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Imelda C. Go
Office of Research and Evaluation
Richland County School District One
1616 Richland St.
Columbia, SC 29201
Tel.: (803) 733-6079
Fax: (803) 929-3873
icgo@juno.com