

Using an Array to Examine Gastric and Colorectal Cancer Risk Factors

Verlin Joseph MPH, University of Florida

ABSTRACT

Colorectal and gastric cancers are two of the most prevalent forms of cancer worldwide. One of the key bacterial risk factors for gastrointestinal cancers is *Helicobacter Pylori* (H. Pylori). This study used an array to analyze the association H. Pylori in colorectal and gastric cancer. The analysis was conducted using 2002-2012 Florida Agency for Health Care Administration (AHCA) hospital discharge data. Each data set contained millions of records and up to thirty diagnoses codes. Arrays were utilized to quickly locate the variables of interest for this analysis. The purpose of this presentation is to illustrate how programmers may use arrays to analyze big data sets.

INTRODUCTION

The primary reason programmers use arrays is to simplify their code. Arrays are used to collect heterogeneous data and meta data including variables and observation names into a container variable. Generally, arrays are suitable for storing column-oriented or tabular data that are often stored as columns in a text file or in a spreadsheet, and can accommodate variables of different types, sizes, units, etc.

Data set arrays may contain different kinds of variables, including numeric, logical, character, categorical, and cell. However, a data set array is a different class than the variables that it contains. For example, even a data set array that contains only variables that are double arrays cannot be operated on as if it were a double array. However, using dot subscripting, you can operate on variable in a data set array as if it were a workspace variable.

ARRAY USES

For this analysis I was required to search over twelve million hospital records for a diagnosis of H. pylori. H. pylori is a common risk factor for both colorectal and gastric cancer. This diagnosis was located among 9 variables ranging from sec_dc1 – sec_dc9. One of the methods to locate this diagnosis would be using an IF-THEN statement. However, I would need to repeat the same calculation for all 9 secondary diagnosis variables:

```
data;
set;
h_pylori = 0;
if sec_dc1 = '04186' then h_pylori = 1;
if sec_dc2 = '04186' then h_pylori = 1;
...
if sec_dc9 = '04186' then h_pylori = 1;
run;
```

An alternative is to define arrays and use a loop to process the calculation for all variables:

```
data;
set;
h_pylori = 0;
array harray {9} sec_dc1-sec_dc9;
do i = 1 to 9;
if harray[i] = '04186' then h_pylori = 1;
end;
```

For this analysis only 9 elements for this array however, arrays may contain hundreds of elements. Furthermore, by defining arrays for the h_pylori values I could also have used them in an input statement to simplify the input process. The set statement was included because I used this array to create the variable 'h_pylori'.

ARRAY STATEMENT

The statement used to define an array is the ARRAY statement:

```
array array-name {n} <$> <length> array-elements <(initial-values)>;
```

The ARRAY statement is a compiler statement within the data step. However, the array elements cannot be used in compiler statements such as DROP or KEEP. For this analysis, I was required to conduct these tasks prior to the ARRAY statement. An array must be defined within the data step prior to being referenced and if an array is referenced without first being defined, an error will occur. Defining an array within one data step and referencing the array within another data step will also cause errors. Arrays only exist for the duration of the data step in which they are defined. Therefore, it is necessary to define an array within every data step where the array will be referenced.

The ARRAY statement provides the following information about arrays created using SAS® software:

- array-name – Any valid SAS name
- n – Number of elements within the array
- \$ - Indicates the elements within the array are character type variables
- length – A common length for the array elements
- elements – List of SAS variables to be part of the array
- initial values – Provides the initial values for each of the array elements

When an array is defined with the ARRAY statement SAS creates an array reference. The array reference is in the following form:

```
array-name {n}
```

The value of n will be the element's position within the array. For example, the aforementioned h_pylori array element has been assigned the 9th position within the array. Therefore, the array reference will be:

```
h_pylori {9}
```

The variable name and the array reference are interchangeable. When an array has been defined in a data step either the variable name or the array reference may be used.

Array Reference Variable Names:

```
hparray h_pylori{1}
hparray h_pylori{2}
hparray h_pylori{3}
hparray h_pylori{4}
hparray h_pylori{5}
```

An array reference may be used within the data step in almost any place other SAS variables may be used including as an argument to many SAS functions. If the data step does not have an ARRAY statement to define the array and create the array reference, errors will occur. When an array is referenced within a data step, it must be defined with an ARRAY statement in the same data step.

TEMPORARY ARRAY

Occasionally, you might need an array to hold values temporarily for subsequent calculations but have no need for the actual variables. In such cases, it is beneficial to create arrays in which specific variables are never created or associated with elements of an array. Arrays of this nature are referred to frequently as non-variable-based arrays. To define such an array, use the `_TEMPORARY_` keyword, as shown in this example:

```
array my_array[25] _temporary_;
```

Temporary arrays can be either numeric or character. In order to create a temporary array, you need to include the dollar sign (\$) after the array brackets.

```
array my_array[25] $ _temporary_;
```

If the array references a temporary array, the array name must always be used. Since there are no variables associated with the array (non-variable-based), a reference to `MY_ARRAY1` would not refer to the first element of the array. In addition, when defining a temporary array, you must always explicitly specify the number of elements within the array using a constant value within the array brackets. You cannot use an asterisk (*).

DO LOOP

So we have now defined our arrays, but now we have to use them to manipulate the data. We use a DO loop to perform the data manipulations on the arrays. Within a DATA step, a DO loop is used to specify a set of SAS statements or operations that are to be performed as a unit during an iteration of the loop. It is important to note that operations performed within a DO loop are performed within an observation. Another thing that you need to be aware of is that every DO loop has a corresponding END statement. If you don't END your DO loop, you will get a

SAS Error message in your log indicating that a corresponding END statement was not found for the DO statement. There are four different types of DO loops available in SAS:

1. DO index=, an iterative, or indexed, DO loop used to perform the operations in the DO loop at a specified start and ending index value for an array
2. DO OVER loop used to perform the operations in the DO loop over ALL elements in the array SAS Global Forum 2010 Hands-on Workshops 5
3. DO UNTIL (logical condition) loop used to perform the operations in the DO loop until the logical condition is satisfied
4. DO WHILE (logical condition) loop used to perform the operations in the DO loop while the logical condition is satisfied

Many times, DO loops are used in conjunction with a SAS array, with the basic idea being that the operations in the DO loop will be performed over all the elements in the array. It should be noted that within a single DO loop multiple arrays can be referenced and operations on different arrays can be performed.

CONCLUSION

No matter your skill level or statistical background arrays can be used to simplify several programs. While they may seem daunting, anyone can master them with practice.

REFERENCES

<http://www2.sas.com/proceedings/sugi30/242-30.pdf>

https://support.sas.com/resources/papers/97529_Using_Arrays_in_SAS_Programming.pdf

CONTACT INFORMATION

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

Name: Verlin Joseph
Enterprise: University of Florida
Address: 2004 Mowry Road
City, State ZIP: Gainesville, FL, 32611
E-mail: verlinjoseph@ufl.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.