Paper CT04

Efficient SAS programming using CHOOSEC/CHOOSEN and WHICHC/WHICHN over traditional IF-THEN-ELSE clause

Nimit Agrawal, Syneos Health®, Gurugram, India
Sumit Pratap Pradhan, Syneos Health®, Gurugram, India

ABSTRACT

While writing a SAS program for a conditional statement we always tend to use IF-THEN-ELSE clause as it is a straightforward and common approach but do we ever ponder over it whether it is always an efficient approach to use IF-THEN-ELSE clause in each scenario. CHOOSEC/CHOOSEN and WHICHC/WHICHN are set of two efficient SAS functions that come into picture here and can be used effectively in many scenarios over traditional IF-THEN-ELSE clause e.g.

- assignment of numeric value to different Overall Response values (CR, PR, etc.) for the derivation of BOR.
- assignment of CRF text to a numeric toxicity grading scale

This approach will help reduce the long time we spend for writing repetitive IF-THEN-ELSE clauses and as a result, will make our program more efficient, readable, precise and robust.

In this paper we will discuss in detail the syntax, usage and application of CHOOSEC/CHOOSEN and WHICHC/WHICHN set of functions in different programming scenarios.

INTRODUCTION

CHOOSEC/CHOOSEN and WHICHC/WHICHN are the new set of functions introduced in SAS 9.2, these set of functions provide handy and efficient alternative to traditional IF-THEN-ELSE statement for implementation of conditional logic, let's understand the functionality with an example as shown below:

Scenario: To derive AE toxicity grades text present on CRF from AETOXGR

```
if AETOXGR = "1" then ATOXGR = "Grade 1 - Mild";
else if AETOXGR = "2" then ATOXGR = "Grade 2 - Moderate";
else if AETOXGR = "3" then ATOXGR = "Grade 3 - Severe";
else if AETOXGR = "4" then ATOXGR = "Grade 4 - Very Severe";
else if AETOXGR = "5" then ATOXGR = "Grade 5 - Life-Threatning";
```

Alternative Scenario: Using CHOOSEC function

```
ATOXGR = choosec(input(AETOXGR,best.),"Grade 1 - Mild","Grade 2 - Moderate","Grade 3 - Severe",
                                      "Grade 4 - Very Severe","Grade 5 - Life-Threatning");
```

OUTPUT:

AETOXGR	ATOXGR
1	Grade 1 - Mild
2	Grade 2 - Moderate
3	Grade 3 - Severe
4	Grade 4 - Very Severe
5	Grade 5 - Life-Threatning

As we can see by using CHOOSEC function long 5 line repetitive code got converted to single line code which increases the readability, apart from this one other very useful advantage is we don't need to worry about the values getting truncated in newly created variable ATOXGR, CHOOSEC function automatically assigns the default length of 200 to ATOXGR while with traditional IF-THEN-ELSE clause we have to specifically assign the length of ATOXGR at the top of the data step otherwise value may get truncated.

Overview of CHOOSEC Function:

Returns a character value that represents the results of choosing from a list of arguments.

Syntax: CHOOSEC (index-expression, selection-1 <, selection-2, …selection-n>)

Length of Returned Variable: In a DATA step, if the CHOOSEC function returns a value to a variable that has not previously been assigned a length, that variable is given a length of 200 bytes.

Basics: The CHOOSEC function uses the value of index-expression to select from the arguments that follow. For example, if index-expression is 3, CHOOSEC returns the value of selection-3. If the first argument is negative, the function counts backward from the list of arguments and returns that value.

Example: In the example shown on previous page if the variable AETOXGR has a value "1" then CHOOSEC function will return the first value "Grade 1 – Mild" from the list of selection values Similarly, if the variable AETOXGR has a value "5" then CHOOSEC function will return the fifth value "Grade 5 – Life-Threatning" from the list of selection values.

Please note that first argument of a CHOOSEC function is an expression so if we are passing the variable then it expects the variable type to be numeric so here in this example since AETOXGR is a character variable we have extrinsically converted the variable into numeric with the use of INPUT function to avoid automatic conversion notes in the log.

Caveat: If the number of values in the selection list is less than number of distinct value present in the variable passed as first argument then CHOOSEC function will return the missing value.

Overview of CHOOSEN Function:

The CHOOSEN function is the numeric counterpart of the CHOOSEC function, returns a numeric value that represents the results of choosing from a list of arguments.

Syntax: CHOOSEN (index-expression, selection-1 <, selection-2, …selection-n>)

Now, let's see an example of the CHOOSEN function:

Scenario: The CHOOSEN function will be useful in the scenario where the sequential numeric values need to be decoded to the discrete numeric values.

```
if ORD = 1 then ORD_DE = 7;
else if ORD = 2 then ORD_DE = 13;
else if ORD = 3 then ORD_DE = 17;
```

Alternative Scenario: Using CHOOSEN function

```
ORD_DE = choosen(ORD,7,13,17);
```

OUTPUT:

ORD	ORD_DE
1	7
2	13
3	17

Here, ORD = 1 is re-assigned as 7 in the new variable ORD_DE. Similarly, ORD = 2 is re-assigned as 13 in the new variable ORD_DE and ORD = 3 is re-assigned as 17 in the new variable ORD_DE.

Overview of WHICHC Function:

Search for a character value that is equal to the first argument and returns the index of the first matching value.

Syntax: WHICHC (string, value-1 <, value-2, …>)

Basics: The WHICHC function searches the second and subsequent arguments for a value that is equal to the first argument and returns the index of the first matching value.

Now, let's see an example of WHICHC function:

Scenario: Assignment of numeric value to corresponding different Overall Response values

```
if OVERRESP = "CR" then ORESPN = 1;
else if OVERRESP = "PR" then ORESPN = 2;
else if OVERRESP = "SD" then ORESPN = 3;
else if OVERRESP = "PD" then ORESPN = 4;
else if OVERRESP = "NE" then ORESPN = 5;
```

Alternative Scenario: Using WHICHC function

```
ORESPN = whichc(OVERRESP,"CR","PR","SD","PD","NE");
```

OUTPUT:

OVERRESP	ORESPN
CR	1
PR	2
SD	3
PD	4
NE	5

If the variable OVERRESP has the value “CR”, then the WHICHC function will return a 1 since “CR” is the first argument in the list of arguments to be searched (that is, the list beginning with the second argument). Similarly, if OVERRESP has the value “PD”, the value returned by WHICHC will be 4 since “PD” is the fourth argument, again not counting the first one.

Caveat: If the string is missing, then WHICHC returns the missing value. Otherwise, WHICHC compares the value of string with value-1, value-2, and so on, in sequence. If an argument value-i equals string, then WHICHC returns the positive integer i. If string does not equal any subsequent argument, then WHICHC returns 0. It’s important that conditional logic be designed to accommodate this behavior.

Now, let’s see another application of WHICHC Function

Scenario: As part of Demographics Table creation, we represent count and percentage of different RACE values and often we need to display RACE values in a certain order generally in the order the text is captured on CRF and not necessarily in the alphabetical order

```
if RACE = "Asian" then RACEN = 1;
else if RACE = "Black" then RACEN = 2;
else if RACE = "White" then RACEN = 3;
else if RACE = "American Indian or Alaska Native" then RACEN = 4;
else if RACE = "Native Hawaiian or Other Pacific Islander" then RACEN = 5;
else if RACE = "Other" then RACEN = 6;
else if RACE = "Unknown" then RACEN = 7;
```

Alternative Scenario: Using WHICHC function

```
RACEN = whichc(RACE,"Asian","Black","White","American Indian or Alaska Native",
                    "Native Hawaiian or Other Pacific Islander","Other","Unknown");
```

OUTPUT:

RACE	RACEN
Asian	1
Black	2
White	3
American Indian or Alaska Native	4
Native Hawaiian or Other Pacific Islander	5
Other	6
Unknown	7

Overview of WHICHN Function:

The WHICHN function is the numeric counterpart of WHICHC function, searches for a numeric value that is equal to the first argument and returns the index of the first matching value.

Syntax: WHICHN (argument, value-1 <, value-2, …>)

Now, let's see an example of WHICHN function:

Scenario: Conversion of Questionnaire Scores into sequential numeric value as part of primary endpoint analysis

```
if QSSTRESN = 10 then AVAL = 1;
else if QSSTRESN = 20 then AVAL = 2;
else if QSSTRESN = 35 then AVAL = 3;
else if QSSTRESN = 43 then AVAL = 4;
else if QSSTRESN = 47 then AVAL = 5;
```

Alternative Scenario: Using WHICHN function

```
AVAL = whichn(QSSTRESN,10,20,35,43,47);
```

OUTPUT:

QSSTRESN	AVAL
10	1
20	2
35	3
43	4
47	5

If the variable QSSTRESN has the value 10, then the WHICHN function will return a 1 since 10 is the first argument in the list of arguments to be searched and so on.

Application of CHOOSEC and WHICHC Function in combination as a nested function:

Scenario: Assignment of full CRF text corresponding to decoded Overall Response values

```
if OVERRESP = 'PD' then PARAM = 'Progressive Disease';
else if OVERRESP = 'PR' then PARAM = 'Partial Response';
else if OVERRESP = 'CR' then PARAM = 'Complete Response';
else if OVERRESP = 'SD' then PARAM = 'Stable Disease';
else if OVERRESP = 'NE' then PARAM = 'Not Evaluable';
```

Alternative Scenario: Using CHOOSEC and WHICHC function

```
PARAM = choosec (whichc (OVERRESP,'PD','PR','CR','SD','NE'),
         'Progressive Disease','Partial Response','Complete Response', 'Stable Disease','Not Evaluable');
```

OUTPUT:

OVERRESP	PARAM
CR	Complete Response
PR	Partial Response
SD	Stable Disease
PD	Progressive Disease
NE	Not Evaluable

The call to WHICHC operates as explained in the previous example related to decoding of overall response values. WHICHC returns a number, in this case from 1 to 5, depending on the value of OVERRESP. This number is the first argument to CHOOSEC and serves to select the correct corresponding value that should be assigned to PARAM.

CONCLUSION

Conditional processing in SAS can be performed in multiple ways - one way is to follow traditional approach by using IF-THEN-ELSE statement. In this paper an alternative approach to perform conditional processing using CHOOSEC/CHOOSEN and WHICHC/WHICHN is illustrated with multiple examples and use cases. These two sets of SAS functions are still under the unexplored territory but they are very efficient and versatile and if used regularly in day-to-day programming activities could result in simplified programming, readability, reusability and robustness.

REFERENCES

SAS Documentation
https://documentation.sas.com/doc/en/workbenchcdc/v_001/vwblefunctionsref/p0zkcvlqdqdnu6n16vv32uojnznj.htm
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/fedsqlref/n0tftsutb9q97ln1gshs8tn4glty.htm
Joshua M. Horstman - "Beyond IF THEN ELSE: Techniques for Conditional Execution of SAS Code" - Paper SA10 MWSUG 2016
Balram Chauhan - "Alternative programming approach for Conditional processing in SAS" - PHUSE 2018

CONTACT INFORMATION

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

Nimit Agrawal
Syneos Health, Principal Statistical Programmer, Statistical Programming
DLF Downtown, DLF City Phase 3 Road, Sector 25A, Gurugram - 122002, Haryana, India
E-mail: nimit.agrawal@syneoshealth.com
LinkedIn: https://www.linkedin.com/in/nimit-agrawal-171bb333/

Sumit Pratap Pradhan
Syneos Health, Manager, Statistical Programming
DLF Downtown, DLF City Phase 3 Road, Sector 25A, Gurgaon - 122002, Haryana, India
E-mail: sumit.pradhan@syneoshealth.com
LinkedIn: https://www.linkedin.com/in/sumit-pradhan-71133345/