PharmaSUG 2021 – Paper QT-027

What Kind of WHICH Do You CHOOSE to be?

Richann Watson, DataRich Consulting; Louise Hadden, Abt Associates Inc.

ABSTRACT

A typical task for a SAS® practitioner is the creation of a new variable that is based on the value of another variable or string. This task is frequently accomplished by the use of IF-THEN-ELSE statements. However, manually typing a series of IF-THEN-ELSE statements can be time-consuming and tedious, as well as prone to typos or cut and paste errors. Serendipitously, SAS has provided us with an easier way to assign values to a new variable. The WHICH and CHOOSE functions provide a convenient and efficient method for data-driven variable creation.

INTRODUCTION

SAS functions are powerful tools that transform variables, create variables, or provide valuable information regarding a variable. There are four basic categories that functions fall into: arithmetic (i.e., MIN, MEAN, MAX); date / time (i.e., TODAY, DATETIME); truncation (i.e., FUZZ, ROUND, TRUNC); and last but not least, string or character functions, such as CAT, TRIM, and FIND. WHICH and CHOOSE functions are valuable members of the string function category, which are generally utilized to clean and analyze string variables. WHICH and CHOOSE functions are relatively new and are available for both character and numeric processing: WHICHC, WHICHN, CHOOSEC, and CHOOSEN.

WHICH and CHOOSE functions are frequently used in conjunction with other character functions in order to streamline verbose coding. Functions can greatly reduce the amount of programming required to achieve desired results compared to the data step, formats, and other techniques. We will explore several time and effort saving applications for the WHICH and CHOOSE functions below.

Note that data used in this paper are fictitious and do not represent any subject level data associated with a particular study. This paper and presentation are intended for all proficiency levels and all industries. Code samples were run on the Windows operating system using SAS 9.4 Maintenance Release 5.

WHICH FUNCTIONS

The WHICH functions return the index number from the first value in the list of values that matches the string. Table 1 provides the syntax and description of the two WHICH functions.

Which Functions*	Description
WHICHC(*string, value-1 <, … value-n*>)	Returns the index of the first item in the ***character*** *value* list that matches the *string*
WHICHN(*string, value-1 <, … value-n*>)	Returns the index of the first item in the ***numeric*** *value* list that matches the *string*

Table 1: Syntax and Description of WHICH Functions

Both WHICHC and WHICHN require at a minimum two arguments: string and value-1.

String is a constant, variable or an expression that evaluates to a value that will be searched for in the *value* list.

Value-n is a constant, variable or an expression that evaluates to a value to be searched. There should be a value for each item that is to be searched with the values separated by commas.

The WHICH functions return the positive integer *i* that corresponds to the i^{th} value in the list that matches the string. Note that *i* corresponds to the argument number – 1. Recall that the first argument is the string, so the list of values does not start till the second argument.

The following section goes through an example to illustrate the use of these functions.

WHICHC

You have subject level data that contains gender and birth year and month. In addition, it contains some key dates for the study. The dates are time-to-event (TTEDT), end of study (EOSDT), death (DTHDT), and disease progression (PRGDT). Data Display 1 is a sample of the subject level data that is used.

Row	USUBJID	BRTHYR	BRTHMO	SEX	TTEDT	EOSDT	DTHDT	PRGDT
1	ABC-001	1972	JUL	M	04APR2017	30MAY2017	03AUG2017	04APR2017
2	ABC-002	1976	NOV	M	23AUG2017	24AUG2017	23AUG2017	

Data Display 1: Subject Level Data with Birth Information, Gender and Key Dates

If the age is not provided, then the age may need to be calculated. Because the full date of birth is not provided, then imputation rules need to be applied so that an age can be calculated. In order to do so, you need the birth month in a numeric format. In addition, you may need to create a numeric code for gender. The traditional way is to use a series of IF-THEN-ELSE as shown in SAS Program 1:

```
data outdsn;
   set indsn;
   if SEX = 'M' then SEXN = 2;
   else if SEX = 'F' then SEXN = 1;
   if BRTHMO = 'JAN' then BRTHMO_N = 1;
   else if BRTHMO = 'FEB' then BRTHMO_N = 2;
   …
   else if BRTHMO = 'DEC' then BRTHMO_N = 12;
run;
```

SAS Program 1: Assign Numeric Codes for Gender and Birth Month Using IF-THEN-ELSE

However, an alternative method for assigning numeric values is to use WHICHC function as demonstrated in SAS Program 2:

```
data outdsn;
  set indsn;
  SEXN = whichc(SEX, 'F', 'M');
  BRTHMO_N = whichc(BRTHMO, 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
                            'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC');
run;
```

SAS Program 2: Assign Numeric Codes for Gender and Birth Month Using WHICHC

In SAS Program 2, SEXN is assigned a value of 1 or 2 based on which argument in the list of values, the search string, SEX, matches. For the birth month, the numeric code is assigned a value from 1 – 12 depending on the what the i^{th} index is for the value that matches BRTHMO.

After execution we end up with the appropriate numeric codes for SEX and BRTHMO (Data Display 2).

Row	USUBJID	BRTHYR	BRTHMO	SEX	SEXN	BRTHMO_N
1	ABC-001	1972	JUL	M	2	7
2	ABC-002	1976	NOV	M	2	11

Data Display 2: Subject Level Data with Numeric Codes for Birth Month and Gender

WHICHN

Similar to WHICHC, WHICHN returns the i^{th} index for a match. The difference between the two is that the search string for WHICHN is a numeric value. Continuing with the example data in Data Display 1, you can determine if the time-to-event date was censored or not by comparing it to the other dates in the data set. SAS Program 3 shows the traditional approach to setting the censor variable:

```
data outdsn;
   set indsn;
   /* SAS CODE FROM PREVIOUS SAS PROGRAM */
   if TTEDT = PRGDT then CENSOR = 0;
   else if TTEDT = EOSDT then CENSOR = 1;
   else if TTEDT = DTHDT then CENSOR = 2;
run;
```

SAS Program 3: Assign Censor Variable Based on Time-to-Event Date Using IF-THEN-ELSE

With the use of WHICHN, you can streamline your code as shown in SAS Program 4:

```
data outdsn;
   set indsn;
   /* SAS CODE FROM PREVIOUS SAS PROGRAM */
   CENSOR = whichn(TTEDT, PRGDT, EOSDT, DTHDT) - 1;
run;
```

SAS Program 4: Assign Censor Variable Based on Time-to-Event Date Using WHICHN

In SAS Program 4, the first argument, TTEDT, is what is being searched for in the remaining three arguments. The values for CENSOR are 0 – 2, where 0 indicates that the data is not censored, and the event did occur. The values of 1 and 2 represent different levels of censoring. Because having an event takes higher precedence of being censored, then the date that represents the event (PRGDT) is the first item in the list of values to search. Since it is the first item, if a match is found (i.e., TTEDT = PRGDT), then it returns a value of 1, but that is not what is needed. A value of 0 represents that the event happened. In order to get the appropriate values for the censoring variable, you need to subtract one after the function has returned a value.

Once a match is found the function returns the position of that first match. Thus, if it is possible for more than one variable to have the same date, the order in which they are listed is important. The values for CENSOR are shown in Data Display 3 after the execution of SAS Program 4.

Row	USUBJID	TTEDT	EOSDT	DTHDT	PRGDT	CENSOR
1	ABC-001	04APR2017	30MAY2017	03AUG2017	04APR2017	0
2	ABC-002	23AUG2017	24AUG2017	23AUG2017		2

Data Display 3: Subject Level Data with Censoring

Note that for both WHICHC and WHICHN if there is no match in the list of searched items, then the functions will return a 0.

CHOOSE FUNCTIONS

The CHOOSE functions return either a character or numeric value based on the item selected from the selection list. Table 2 provides the syntax and description of the CHOOSE functions.

CHOOSE Functions*	Description
CHOOSEC(*index-expression, selection-1 <, … selection-n>*)	Returns the ***character*** value from the *selection* list that is associated with the *index-expression*
CHOOSEN(*index-expression, selection-1 <, … selection-n>*)	Returns the ***numeric*** value from the *selection* list that is associated with the *index-expression*

Table 2: Syntax and Description of CHOOSE Functions

Both functions require at least two arguments.

The first argument is the *index-expression* which is used to determine which item to select from the selection list. The *index-expression* can be a numeric constant, variable that represents a numeric value or an expression that evaluates to a numeric value.

The second argument is the first time in the selection list. You can add additional arguments for each item in the selection list separated by commas. The items in the selection list are a constant, variable or an expression that evaluates to a value.

The CHOOSE functions returns one of the values in the comma separated list that corresponds to the value in the *index-expression*.

The best way to understand these functions is through the use of examples.

CHOOSEC

Assume we have subject level data that contains the treatment and race values as numeric value (Data Display 4), but we want it to show the decoded values in the data set as well.

Row	USUBJID	BRTHYR	BRTHMO	SEX	RACEN	TRT01PN
1	ABC-001	1972	JUL	M	5	1
2	ABC-002	1976	NOV	M	5	4

Data Display 4: Subject Level Data with Numeric Values for Race and Treatment

Before discovering the CHOOSE functions, we may have used IF-THEN-ELSE statements in order to assign the values, as shown in SAS Program 5:

```
data outdsn;
   set indsn;
   if TRT01PN = 1 then TRTP = 'TRT A';
   else if TRT01PN = 2 then TRTP = 'TRT B';
   else if TRT01PN = 3 then TRTP = 'TRT C';
   else if TRT01PN = 4 then TRTP = 'TRT D';

   length RACE $41;
   if RACEN = 5 then RACE = 'White';
   else if RACEN = 6 then RACE = 'Asian';
   else if RACEN = 7 then RACE = 'Black';
   else if RACEN = 9 then RACE = 'Other';
   else call missing(RACE);
run;
```

SAS Program 5: Create Decoded Variables for Race and Treatment with IF-THEN-ELSE

An alternative approach is to use the CHOOSEC function to return a value from a list using the numeric version of the race and treatment variables, as shown in SAS Program 6:

```
data outdsn;
   set indsn;
   length RACE $41;
   TRTP = choosec(TRT01PN, 'TRT A', 'TRT B', 'TRT C', 'TRT D');
   RACE = choosec(RACEN - 4, 'White', 'Asian', 'Black', '', 'Other');
run;
```

SAS Program 6: Create Decoded Variables for Race and Treatment with CHOOSEC

In SAS Program 6, the values for TRT01PN are 1 – 4, so we were able to use the variable as the index to select the correct decoded value for treatment. However, the values for RACEN are 5 – 7 and 9. In order to use the list we could either subtract 4 from RACEN in the *index-expression* argument of the function, as we did in the example, or we could have added blank values in the list so that the selection list comprised of 4 blanks followed by the actual list. Notice that for RACEN there is no value 8, which converts to item 4 in the list; therefore, item 4 in the list is denoted with a blank value. It could be denoted with any value since it would never be used. It is mainly a placeholder so that we have a continuous sequence.

After execution of the program, we end up with character values for race and treatment (Data Display 5).

Row	USUBJID	BRTHYR	BRTHMO	SEX	RACEN	TRT01PN	RACE	TRTP
1	ABC-001	1972	JUL	M	5	1	White	TRTA
2	ABC-002	1976	NOV	M	5	4	White	TRTD

Data Display 5: Subject Level Data with Character Values for Race and Treatment Added

By default, if the length of a variable being populated with the CHOOSEC function is not specified it will default to 200 characters. Thus, in the example, TRTP has a length of 200 but RACE has a length of 41. To avoid overly large variable lengths, use a length statement to set an appropriate length.

CHOOSEN

CHOOSEN is similar to CHOOSEC with the exception being that it returns a numeric value. Going back to our data from Data Display 5, you notice that the numeric codes for treatment are reversed. In other words, TRTP = “TRTD” should correspond to TRTPN = 1 and not TRTPN = 4. Again, you could use the tried-and-true approach of IF-THEN-ELSE as illustrated in SAS Program 7.

```
data outdsn;
   set indsn;
   /* SAS CODE FROM PREVIOUS SAS PROGRAM */
   if TRT01PN = 1 then TRTPN = 4;
   else if TRT01PN = 2 then TRTPN = 3;
   else if TRT01PN = 3 then TRTPN = 2;
   else if TRT01PN = 4 then TRTPN = 1;
run;
```

SAS Program 7: Reverse Order of Treatment Numeric Codes Using IF-THEN-ELSE

The approach in SAS Program 7 is perfectly fine and there are only four lines of code. It is short enough to maintain but what if more values are added which would mean more IF-THEN-ELSE statements. An easier approach would be to use CHOOSEN as done in SAS Program 8.

```
data outdsn;
   set indsn;
   /* SAS CODE FROM PREVIOUS SAS PROGRAM */
   /* reverse the order of the treatments */
   TRTN = choosen(TRT01PN, 4, 3, 2, 1);
run;
```

SAS Program 8: Reverse Order of Treatment Numeric Codes Using CHOOSEN

Row	USUBJID	BRTHYR	BRTHMO	SEX	RACEN	TRT01PN	RACE	TRTP	TRTPN
1	ABC-001	1972	JUL	M	5	1	White	TRTA	4
2	ABC-002	1976	NOV	M	5	4	White	TRTD	1

Data Display 6: Subject Level Data with Treatment Numeric Codes Reversed

In SAS Program 8, TRT01PN has values of 1 – 4, which is used to select from the list of arguments to assign the value of TRTPN. Data Display 6 shows that TRT01PN = 1 was assigned TRTPN = 4.

For both CHOOSEC and CHOOSEN if the first argument is a negative value, then function starts from the end of the list and returns the value counting from the right.

CONCLUSION

Which WHICH do you CHOOSE? The WHICH and CHOOSE functions can greatly reduce the amount of IF-THEN-ELSE style coding and open the door to conditional processing. It can also replace PROC FORMAT coding for recoding variables. We encourage you to try out these efficient functions.

REFERENCES

SAS Institute Inc. (n.d.). Dictionary of Functions and CALL Routines. Retrieved from https://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=p1q8bq2v0o11n6n1gpij335fqpph.htm&docsetVersion=9.4&locale=en

RECOMMENDED READING

Chauhan, Balram. “Alternative programming approach for Conditional processing in SAS”. *Proceedings of the PhUSE 2018 Conference.* Raleigh, NC: PhUSE. https://www.lexjansen.com/phuse-us/2018/ct/CT11_ppt.pdf

Horstman, J. “Beyond IF THEN ELSE: Techniques for Conditional Execution of SAS® Code”. *Proceedings of the SAS Global Forum 2017 Conference.* Orlando, FL: SAS Global Forum. https://support.sas.com/resources/papers/proceedings17/0326-2017.pdf

Horstman, J. “Fifteen Functions to Supercharge Your SAS® Code”. *Proceedings of the PharmaSUG 2018 Conference.* Seattle, WA: PharmaSUG. https://www.pharmasug.org/proceedings/2018/BB/PharmaSUG-2018-BB17.pdf

Su, Jason J. “A Game Plan for Beating the IF-THEN-ELSE Overhead in DATA Steps”. *Proceedings of the SESUG 2020 Conference.* Virtual: SESUG. https://www.lexjansen.com/sesug/2020/SESUG2020_Paper_152_Final_PDF.pdf

CONTACT INFORMATION

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

Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com

Louise Hadden
Abt Associates, Inc.
louise_hadden@abtassoc.com