Alternative programming approach for Conditional processing in SAS

Balram Chauhan, Cytel, Mumbai, India

ABSTRACT

I would like to demonstrate few simple and powerful alternative programming approaches which will be very useful to carry out day to day programming activities more efficiently. We will see few novel approaches in comparison with commonly used IF… THEN… ELSE, WHERE and SELECT statement for conditional processing.

In this paper, we will explore various ways to construct conditional SAS logic, including some that may provide advantages over the standard conditional statements. Topics will include the IFC and IFN functions, the CHOOSE and WHICH families of functions, as well as few more alternative methods.

INTRODUCTION

We all use traditional IF THEN ELSE statements when working with conditional processing. Assigning one value for when the condition is true, and another value for when the condition is false. However there are other options available over IF THEN ELSE statements, we will see few of them in demonstration having advantages over the standard approach.

In this paper we will talk about task using standard programming approach followed by alternative approach. This alternative provides are not necessarily the best ways to handle specific programming situations however:

1. To make code more compact
2. To make code more readable
3. To make code more reusable.

/* **Scenario #1 :** Combine PCTESTCD and PCSPECS to get unique PARAMCD */

```
if PCTESTCD = 'RADICCON' AND PCSPEC = 'STOOL' THEN PARAMCD = 'RCON_FE';
if PCTESTCD = 'RADICCON' AND PCSPEC = 'URINE' THEN PARAMCD = 'RCON_UR';
if PCTESTCD = 'RADICCON' AND PCSPEC = 'STOOL/URINE' THEN PARAMCD = 'RCON_TOT';

if PCTESTCD = 'RADIOAMT' AND PCSPEC = 'STOOL' THEN PARAMCD = 'RAMT_FE';
if PCTESTCD = 'RADIOAMT' AND PCSPEC = 'URINE' THEN PARAMCD = 'RAMT_UR';
if PCTESTCD = 'RADIOAMT' AND PCSPEC = 'STOOL/URINE' THEN PARAMCD = 'RAMT_TOT';

if PCTESTCD = 'RADIOPCT' AND PCSPEC = 'STOOL' THEN PARAMCD = 'RPCT_FE';
if PCTESTCD = 'RADIOPCT' AND PCSPEC = 'URINE' THEN PARAMCD = 'RPCT_UR';
if PCTESTCD = 'RADIOPCT' AND PCSPEC = 'STOOL/URINE' THEN PARAMCD = 'RPCT_TOT';
```

/* **Alternative for Scenario #1 :** Using IFN/IFC Function */

```
PARAMCD1 = catx ('_',
       ifc (PCTESTCD = 'RADIOAMT','RAMT',
       ifc (PCTESTCD = 'RADIOPCT','RPCT',
       ifc (PCTESTCD = 'RADICCON','RCON',''))),
       ifc (PCSPEC = 'STOOL','FE',
       ifc (PCSPEC = 'URINE','UR',
       ifc (PCSPEC = 'STOOL/URINE','TOT',''))));
```

OUTPUT :

PCSPEC	PCTESTCD	PARAMCD	PARAMCD1
STOOL	RADICCON	RCON_FE	RCON_FE
STOOL/URINE	RADICCON	RCON_TOT	RCON_TOT
URINE	RADICCON	RCON_UR	RCON_UR
STOOL	RADIOAMT	RAMT_FE	RAMT_FE
STOOL/URINE	RADIOAMT	RAMT_TOT	RAMT_TOT
URINE	RADIOAMT	RAMT_UR	RAMT_UR
STOOL	RADIOPCT	RPCT_FE	RPCT_FE
STOOL/URINE	RADIOPCT	RPCT_TOT	RPCT_TOT
URINE	RADIOPCT	RPCT_UR	RPCT_UR

IFN/IFC Function:

In SAS 9, new function available IFC/IFN function which is similar to IF THEN ELSE statement which can compact the standard conditional processing.

Syntax : IFC(logical-expression, value-returned-when-true, value-returned-when-false)

The IFN function is similar to the IFC function except that IFN returns a numeric value while IFC returns a character value.

/* **Scenario #2 :** To create AE grade from severity variable AESEV */

```
if AESEV = "MILD" then  AGRD = 'GRADE I';
  else if AESEV = "MODERATE" then  AGRD = 'GRADE II';
  else if AESEV = "SEVERE" then AGRD = 'GRADE III';
  else if AESEV = "VERY SEVERE" then AGRD = 'GRADE IV';
  else if AESEV = "DEATH" then AGRD = 'GRADE V';
```

/* **Alternative for Scenario #2 :** Using WHICHC/WHICHN Function & CHOOSEC/CHOOSEN Function */

```
AGRD1 = choosec(whichc(AESEV,'MILD','MODERATE','SEVERE','VERY SEVERE','DEATH'),
                        'GRADE I','GRADE II','GRADE III', 'GRADE IV','GRADE V');
```

OUTPUT :

AESEV	AGRD	AGRD1
MODERATE	GRADE II	GRADE II
MILD	GRADE I	GRADE I
MODERATE	GRADE II	GRADE II
MODERATE	GRADE II	GRADE II
MILD	GRADE I	GRADE I
MILD	GRADE I	GRADE I
MILD	GRADE I	GRADE I
VERY SEVERE	GRADE IV	GRADE IV

WHICHC/WHICHN Function:

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

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.

If string is missing, then WHICHC returns a missing value and If string does not equal any subsequent argument, then WHICHC returns 0.

WHICHN is similar to WHICHC, Searches for a numeric value that is equal to the first argument, and returns the index of the first matching value.

CHOOSEC/CHOOSEN Function:

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

The CHOOSEN function uses the value of index-expression to select from the arguments that follow.

The CHOOSEN function is similar to the CHOOSEC function except that CHOOSEN returns a numeric value while CHOOSEC returns a character value.

/* **Scenario #3:** Filtering dataset using single/multiple population flags */

DATA :

	USUBJID	SAFFL	PKFL	COMPLFL
1	103-01-01	Y	Y	Y
2	103-01-02	Y	Y	Y
3	103-01-03	Y	Y	Y
4	103-01-04	N	N	N
5	103-01-05	Y	Y	Y
6	103-01-06	N	Y	Y
7	103-01-07	Y	Y	N
8	103-01-08	N	Y	Y
9	103-01-09	Y	N	Y
10	103-01-10	Y	N	Y

```
data TEST;
set ADSL;
where SAFFL = 'Y'; /*Filtering with one Population variable */
run;

data TEST;
set ADSL;
where SAFFL = 'Y' and PKFL = 'Y'; /*Filtering with two Population variables */
run;

data TEST;
set ADSL;
where SAFFL = 'Y' and PKFL = 'Y' and complfl = 'Y'; /*Filtering with three
Population variables */
run;
```

/* **Alternative for Scenario #3:** Shorten expressions using user defined functions */

```
data TEST;
set ADSL;
where pop1(SAFFL) ; /*Filtering with one Population variable */
run;
```

SAS lOG :

```
NOTE: There were 7 observations read from the data set WORK.ADSL.
      WHERE POP1(SAFFL);
NOTE: The data set WORK.TEST has 7 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
data TEST;
set ADSL;
where pop2(SAFFL,PKFL) ; /*Filtering with one Population variable */
run;
```

SAS lOG :

```
NOTE: There were 5 observations read from the data set WORK.ADSL.
      WHERE POP2(SAFFL, PKFL);
NOTE: The data set WORK.TEST has 5 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
data TEST;
set ADSL;
where pop3(SAFFL,PKFL,COMPLFL) ; /*Filtering with one Population variable */
run;
```

SAS lOG :

```
NOTE: There were 4 observations read from the data set WORK.ADSL.
      WHERE POP3(SAFFL, PKFL, COMPLFL);
NOTE: The data set WORK.TEST has 4 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

User define Functions:

POP1, POP2, POP3 are user define functions which can be used across all programs and studies/projects once created and stored in permanent/temporary library.

It works not only with population variables but also with any other variables having values as ‘Y’ and ‘N’ Purpose of these functions is to check text “Y” inside the variables.

FCMP procedure allows you to create new Functions or Call routines. Functions always return an explicit result. In other words, you can take the result of a function and assign it to any variable.

Functions can return either a numeric or a character type of result. Input variables can be numeric or character type.

User define functions can be customized with different combinations/options as per need/requirement.

You can create your own functions for most commonly used tasks like Study day calculation, BMI using height and weight, date imputation, calculating age … etc.

```
/*To create user define functions */
options cmplib = work.functions;

proc fcmp outlib =  work.functions.conf;
/*User define function : for POP1 */
      function pop1(str1 $);
      if upcase(strip(str1)) eq 'Y' then return(1);
      else return(0);
      endsub;
/*User define function : for POP2 */
      function pop2(str1 $, str2 $);
      if upcase(strip(str1))  eq 'Y' and upcase(strip(str2)) eq 'Y' then
return(1);
      else return(0);
      endsub;
/*User define function : for POP3 */
      function pop3(str1 $, str2 $, str3 $);
      if upcase(strip(str1)) eq 'Y' and upcase(strip(str2)) eq 'Y' and
upcase(strip(str3)) eq 'Y' then return(1);
      else return(0);
      endsub;
run;
```

CONCLUSION

There are varieties of ways to do same task in SAS, however each way has its own advantages and disadvantages. This paper shows some of the alternative approaches for commonly used conditional processing step in SAS. Using of alternative may result in simplified programming, readability & reusability.

REFERENCE:

IFN/IFC statements - SAS Support:
http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p07za51a2k0sxkn1m0tulx2cy464.htm

WHICHC/WHICHC statement - SAS Support:
http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0jfvenvsqk24vn1q2ypospoq9ij.htm

CHOOSEC/CHOOSEN statement - SAS Support:
http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0et55hpbrn2vln14mmqrhzffs64.htm

The FCMP Procedure - SAS Support:
https://support.sas.com/documentation/onlinedoc/base/91/fcmp.pdf

CONTACT INFORMATION

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

Balram Chauhan
Cytel Statistical Software & Services Pvt. Ltd.
Unit No. 03 & 04, Ground Floor, Reliable Plaza, Plot No. K10,
Thane-Belapur Road, Airoli, Navi-Mumbai - 400 708
Balram.Chauhan@cytel.com | www.cytel.com

TRADEMARK INFORMATION