Paper CC-18

How SAS® Processes If-Then-Else Statements

Michael Leitson, Wellstar Research Institute, Marietta, GA
Elizabeth Leslie, Kaiser Permanente, Atlanta, GA

ABSTRACT

IF-THEN-ELSE statements are relatively simple and convenient to use in SAS. However, one must be careful to write correct SAS syntax in order to achieve the desired result from data step processing. A single misplaced or forgotten keyword can lead to erroneous results. This paper examines the proper way to produce IF-THEN-ELSE statements and also shows how to avoid common errors that might produce incorrect data sets. The intended audience is beginning SAS users who are learning about the joys and pitfalls of IF-THEN-ELSE statements.

INTRODUCTION

IF-THEN-ELSE statements are a very versatile piece of programming that can be used in many ways. Some common uses of IF-THEN-ELSE statements are transforming quantitative (numeric) to qualitative (categorical) and data cleaning. Learning how to correctly use IF-THEN-ELSE statements will strengthen a BASE SAS programmer's skills and help them become more efficient programmers.

SYNTAX

The general form of the syntax is:

IF *expression-1* **THEN** *statement-1*;
ELSE IF *expression-2* **THEN** *statement-2*;
.
.
.
ELSE *statement-n*;

Here's how SAS processes the IF-THEN-ELSE statements:

- First, SAS checks *expression-1*. What happens next depends on whether or not the condition in *expression-1* is met. This is why IF-THEN-ELSE statements are sometimes referred to Conditional Processing.
- If *expression-1* is **true**, SAS executes *statement-1* and does NOT check the remaining ELSE IF statements. SAS will then continue to process any additional code following the block of IF-THEN-ELSE statements.
- If *expression-1* is **false**, SAS will move to the next ELSE IF line and check *expression-2.*
- If *expression-2* is **true**, SAS executes *statement-2* and does NOT check the remaining ELSE IF statements. SAS will then continue to process any additional code following the block of IF-THEN-ELSE statements.
- If *expression-2* is **false**, SAS will check the next ELSE IF expression.
- The same process will continue until SAS gets to the last ELSE statement. This last ELSE statement does not have an expression to check, so SAS will execute the THEN statement whenever this line is read. IF-THEN-ELSE statements are processed sequentially, so the only way SAS will execute the last THEN statement is if none of the previous IF and ELSE IF conditions have been met. The final ELSE statement is a good catch-all to make sure all contingencies are accounted for.

A couple of objectives worth noting: first, a THEN statement must always follow a corresponding IF statement. Second, an ELSE statement is not always necessary. In fact, when an ELSE IF or ELSE statement is omitted, only the IF statement will be processed, conserving computing power when working with large datasets. A note of caution: make sure all contingencies are accounted for before omitting the else statement or you may get unexpected results.

Suppose that we have the following data set about students at a university:

ID	Name	GPA	City	Age	Loan?	Eye Color
00034111	Ann	3.7	Atlanta	19	No	Blue
00089999	Bart	2.9	Kennesaw	21	Yes	Blue
0008123e	Cecil	3.5	Marietta	39	Yes	Green
00035555	Denise	4.0		25	No	Brown
00045678	Emily	2.5	Kennesaw	22	No	Brown
00012929	Frank	3.6	Acworth	47	Yes	Brown

Table 1: Original Data Set

Example 1: Correcting Data Errors with IF-THEN-ELSE statements

After examining the ID column, we learn that the 'e' in the third row should actually be a 4. The following code (or a variation of the code) could be used to make the proper replacement:

```
IF ID = '0008123e' THEN ID = '00081234';
      ELSE ID = ID;
```

When SAS is processing the data set row by row, it will check to see if the IF expression is true for each row. If it is true, SAS will make the replacement. If it is false, SAS will set ID=ID and the values of ID will not change. In this way, ID will be replaced only when ID='0008123e'. Otherwise, ID remains the same.

Example 2: Categorizing Data with IF-THEN-ELSE statements

Now suppose that we want to group the students' GPAs into categories based on whether the student has an above average, average or below average GPA. The average GPA is 3.7. We could code:

```
IF GPA > 3.7 THEN GPA_CAT = 'Above Average';
      ELSE IF 3.0 >= GPA >= 3.7 THEN GPA_CAT = 'Average';
      ELSE GPA_CAT = 'Below Average';
```

Name	GPA	GPA_CAT
Ann	**3.7**	**Average**
Bart	2.9	Below Average
Cecil	3.5	Average
Denise	4.0	Above Average
Emily	2.5	Below Average
Frank	3.6	Average

Table 2: Creating GPA Category

Again, for every row of data, SAS checks the IF expression. IF true, SAS will execute the THEN statement, if false, SAS will move to the next ELSE IF statement. For the first student, Ann, SAS checks the first IF expression, `GPA > 3.7`. For Ann, with a GPA of 3.7, this expression is false, so SAS will not execute the THEN statement. SAS then continues on to check the next ELSE IF expression, `3.0 >= GPA >= 3.7`. This time, the expression is true, so SAS executes the THEN statement and assigns a value of "Average" to Ann's GPA_CAT. Since the second condition was met, SAS will not process the final ELSE statement for Ann. SAS repeats this process for all rows in the data set.

Because ELSE IF statements are only processed after IF statements, it is important to include the ELSE keyword when the conditions are not mutually exclusive. Otherwise, SAS will process every single IF statement, one after the other. When multiple conditions can be true at the same time, SAS could process multiple THEN statements leading to unexpected results.

For instance the following code is **incorrect** and will give unexpected results:

```
IF GPA > 3.7 THEN GPA_CAT = 'Above Average';
      IF 3.0 >= GPA >= 3.7 THEN GPA_CAT = 'Average';
      ELSE GPA_CAT = 'Below Average';
```

The results will appear as:

Name	GPA	GPA_CAT
Ann	3.7	Average
Bart	2.9	Below Average
Cecil	3.5	Average
Denise	**4.0**	**Below Average**
Emily	2.5	Below Average
Frank	3.6	Average

Table 3: Incorrect GPA Category

Surely a GPA of 4.0 is not Below Average! Hence, this explains why an ELSE statement is needed after the first IF statement. For Denise with a GPA of 4.0, SAS checks the first IF expression, `GPA > 3.7`. This is true, so SAS processes the THEN statement, assigning Denise's GPA_CAT to "Above Average". Since the next IF statement does not have ELSE in front of it, SAS will process the next IF statement, regardless of the previous condition being met. Since `3.0 >= GPA >= 3.7` is not true, SAS skips the THEN statement and moves on to process the ELSE statement. The ELSE statement does not have an expression, so SAS processes the THEN statement every time it reads this line, re-assigning Denise's GPA_CAT to "Below Average". This is why we have unexpected outcomes and an upset student. This code can be corrected by adding an ELSE keyword to the second line. Then the final ELSE statement will only be processed when the GPA is not greater than 3.0.

Example 3: Taking Advantage of Sequential Processing of ELSE IF statements

There is another way to code the GPAs into categories. The following code will produce the same results as the correct code in Example 2:

```
IF GPA > 3.7 THEN GPA_CAT = 'Above Average';
      ELSE IF GPA >= 3.0 THEN GPA_CAT = 'Average';
      ELSE GPA_CAT = 'Below Average';
```

The only difference between this code and the correct code in Example 2 lies in the second line; in the first code, we set the boundaries for the Average GPA, while in the second code, we set the Average GPA to be greater or equal to than 3.0. Why does this work? Because the ELSE keyword prevents SAS from processing the second line if the first condition was met. Therefore, SAS will only check `GPA >= 3.0` when `GPA > 3.7` is false. The final results should appear as below:

Name	GPA	GPA_CAT
Ann	3.7	Average
Bart	2.9	Below Average
Cecil	3.5	Average
Denise	4.0	Above Average
Emily	2.5	Below Average
Frank	3.6	Average

Table 4: Alternative Way of Creating GPA Category

Another way to take advantage of sequential processing of the ELSE IF statements is to check the most frequently occurring condition first. Remember, SAS will always check the first IF statement, but SAS will only check the ELSE IF statements when the proceeding IF expression is false. When the most common condition is in the first IF statement, computing resources are conserved because SAS will not have to check the other conditions most of the time.

Example 4: Recoding Values with IF-THEN-ELSE statements

Now let's suppose that we wanted to recategorize the loan values (which inform us if the student took out a loan) so that "No" appears as 0 and "Yes'' appears as 1, the standard method of binary classification. The following code will accomplish this (note that we are replacing the loan values instead of creating a new category):

```
IF LOAN_ = 'No' THEN LOAN_ = 0;
ELSE LOAN_ = 1;
```

CONCLUSION

IF-THEN-ELSE statements are a great way to use conditional processing, but it should also be noted that this method is not the only way for Base programmers to perform this task. PROC FORMAT or SELECT statements can also be used to perform the same operations. IF-THEN-ELSE statements are applicable in many other situations not covered in this paper. A key tool for the Base programmer is the utilization of IF-THEN-ELSE statements with correct syntax, which can be learned with relative ease. There are a variety of ways to avoid errors and knowledge of such methods will be beneficial to a new programmer.

REFERENCES

Cody, Ron. *Learning SAS® by Example: A Programmer's Guide,* Cary, NC: SAS Institute, Inc.

SAS Institute Inc. 2011. *SAS® Certification Prep Guide: Base Programming for SAS®9, Third Edition*. Cary, NC: SAS Institute, Inc.

ACKNOWLEDGEMENTS

We would like to thank the Kennesaw State University Department of Mathematics and Statistics for the depth of knowledge provided to us on this subject.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors by email at:
Michael.Leitson@wellstar.org or Elizabeth.Leslie@kp.org.

TRADEMARK CITATION