A Few Alternatives to IF-THEN-ELSE

Jonas V. Bilenas, JP Morgan Chase, New York, NY

ABSTRACT

IF-THEN-ELSE is a common logical syntax used in the SAS Datastep. We will examine alternatives to this logic that will streamline code. Examples include user defined formats and logical expressions. Efficiency comparisons will be provided for these alternatives as well.

INTRODUCTION

Most programming languages have IF-THEN-ELSE logic that is used to create new variables, reassign existing variables and to direct logic flow. The same logic is used in the SAS DATASTEP as well as in SAS MACRO language. In this paper we will look at some alternatives to IF-THEN-ELSE logic within the DATASTEP. Advantages and disadvantages of these techniques will be also evaluated.

Let us start with a simple example. In this code, we wish to select certain records for inclusion in a new SAS DATASET and also create new variables based on values of an identification variable.

```
data nesug02;
  set nesug;

  if id not in(9,11,20,21,19,10,13) then delete;

  if id in(9,13,21) then balance_transfer_apr=1.99;
    else balance_transfer_apr=0;

  if id in(9,11) then purchase_apr=1.99;
    else if id in(20,21) then purchase_apr=0;
    else purchase_apr=10.99;
  run;
```

ALTERNATIVE #1: USING PROC FORMAT CODE

PROC FORMAT can be viewed as a table lookup allowing 1-to-1 mapping or many-to-1 mapping.

User defined formats are created in the PROC FORMAT and are called in subsequent DATA STEPS or PROCS.

Here is code that sets up user formats and applies the formats in the DATASTEP.

```
proc format;
  value idkeep 9-11,13,19-21 = 'OK'
               other = 'NG'
  ;
  value b_apr 9,13,21 = '1.99'
              other   = '0.00'
  ;
  value p_apr 9,11  = ' 1.99'
              20,21 = ' 0.00'
              other = '10.99'
  ;
data nesug02;
  set nesug;

  if put(id,idkeep.)='NG' then delete;

  balance_transfer_apr=input(put(id,b_apr.),4.2)

  purchase_apr=input(put(id,p_apr.),5.2);
  run;
```

The FORMATS are created in the PROC FORMAT. Format names must be 8 characters or less for numeric formats and 7 characters or less for character formats. Character formats start with a $ prefix. Format names do not have to be the same as the variable names and are assigned to variables with put or input statements in DATASTEPS or with the FORMAT statement in PROCS.

This alternative DATASTEP code still includes the first IF to select records to retain in the DATSET, but the subsequent IF statements are replaced with input(put(variable,format),format) statements. PUT writes out the variable using the FORMAT specified and returns a character variable. The INPUT statement returns a numeric variable since we had a desire to define numeric variables.

This alternative code has about the same number of lines as the first example. However, FORMATS have certain features that make them more desirable than using IF-THEN-ELSE logic. These are:

1. FORMAT is independent of the DATASTEP and can be saved in a permanent library to be called any number of times later within the code.
2. FORMATs can not be created with one-to-many or many-to-many mappings. A warning message will be issued by SAS if you try to map one value to more than one label. IF-THEN-ELSE does not have this feature.
3. Format VALUES can be listed with commas and with a '-' to indicate a range of values. This can save time in specifying a large range of values and prevent potential errors.

FORMAT VALUE ranges can be specified in a number of ways and special keywords can be used to simplify the expression of the range. Examples are given below:

1. Ranges can be constant values or values separated by commas:
   - 'a', 'b', 'c'
   - 1,22,43
2. Ranges can include intervals such as:
   - lower – higher. Interval includes both endpoints.
   - lower <- higher. Interval includes higher endpoint.
   - lower - < higher. Interval includes lower endpoint.
   - lower <- < higher. Interval does not include either endpoint.
3. Ranges can be specified with special kewords:
   - LOW, HIGH, OTHER, ., ' '
4. The LOW keyword does not format missing values.
5. The OTHER keyword does include missing values unless accounted for with a '.' or ' '.

ALTERNATIVE #2: USING LOGICAL EXPRESSIONS

An alternative to IF-THEN-ELSE I use often is the use of logical expressions. A logical expression is specified within parentheses '()' and are evaluated as being true or false. If the expression is true, a 1 is returned. If the expression is false, a 0 is returned.

Here is the sample problem coded with logical expressions.

```
data nesug02;
  set nesug;

  if id not in(9,11,20,21,19,10,13) then delete;

  balance_transfer_apr = (id in(9,13,21))*1.99;

  purchase_apr= (id in(9,11))*1.99 +
     (id not in(9,11,20,21))*10.99;
  run;
```

Note that the code is compact. It may be a bit difficult to interpret the first time you encounter logical expressions. Remember that if the expression is true a 1 is returned and if false a 0 is returned. So, in evaluating the purchase_apr, here are some examples worked out logically:

- If id = 9:
  `purchase_apr= (id in(9,11))*1.99 +`
  `(id not in(9,11,20,21))*10.99=`
  (1)*1.99 + (0)*10.99 = 1.99
- If id=20:
  `purchase_apr= (id in(9,11))*1.99 +`
  `(id not in(9,11,20,21))*10.99=`
  (0)*1.99 + (0)*10.99 = 0
- If id=13
  `purchase_apr= (id in(9,11))*1.99 +`
  `(id not in(9,11,20,21))*10.99=`
  (0)*1.99 + (1)*10.99 = 10.99

Another example of logical expressions is applied to creating a weight variable to weight up non-event observations when reporting results for the full population. In a response model example, non-responders were sampled at a 2% rate. To report full statistics, a weight variable is created to weight up non-responders by 50 and to weight responders by 1. The code to create the variable is:

```
weight = respond + (respond=0)*50;
```

In this example, if an observation is a responder (respond=1) then weight = 1. When the observation is a non-responder, then weight = 0 +(1)*50 = 50.

ALTERNATIVE #3: USING THEN SELECT STATEMENT

The select statement is one that I don't use often, but it is another alternative to IF-THEN-ELSE code. Here is the weight variable creation problem worked out using the select statement.

```
select (respond);
  where (respond=1) weight=1;
  otherwise weight=50;
  end;
```

CPU CONSIDERATIONS

I was surprised to discover that the IF-THEN-ELSE logic took the least amount of CPU time in simulations. This was followed by SELECT statements, followed by logical expressions and FORMATS.

In simulations run for this paper, the format method took about twice as long to run than the IF-THEN-ELSE statements. However, using user defined FORMATS have the advantage that logical mistakes are avoided since PROC FORMAT does not allow one-to-many mapping of values-to-labels. Another benefit of FORMATS is that with the use of user defined FORMATS, a data step can be avoided altogether, making the FORMAT more efficient. Here is an example of by-passing the data step to recode a numeric risk score variable into pass/fail ranges:

```
proc format;
  value score low-600  = 'FAIL'
              601-high = 'PASS'
  ;
proc freq data=sample;
  table score;format score score.; run;
```

If we did not use the PROC FORMAT, a DATASTEP would've been required to set up the assignment of PASS and FAIL labels.

Logical expressions also take longer in terms of CPU than IF-THEN-ELSE. However, the statements are compact and have a clean look to them.

CONCLUSION

IF-THEN-ELSE code is common in many programming languages. Other alternatives are more compact and more error-safe but may take more in terms of CPU time.

CONTACT INFORMATION

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

Jonas V. Bilenas
JP Morgan Chase
Experimental Design
2 Chase Manhattan Plaza, 10th flr
New York NY 10081
Email: Jonas.Bilenas@Chase.com

ACKNOLEDGEMENTS:

SAS® and all other SAS Institute Inc. product or service namesare registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.