Many SAS MACRO users have found that they can use 'if-then' to replace '%if-%then' in their SAS MACRO programs. The SAS user manual 'SAS Guide to Macro Processing' did not show the difference clearly, so some people wonder when '%if-%then' should be used in their MACRO programs.

It is TRUE that in some cases, either 'if-then' or '%if-%then' works. But they are different.

The main differences between %if-%then and if-then are:

1. '%if-%then' can ONLY be used in the MACRO program (whether inside the DATA step, or outside) and 'if-then' is ONLY used inside the DATA step (whether MACRO or non-MACRO). Please see an example:

   %let z=3;
   data test;
   x=1;
   y=1;
   run;
   %macro diff (v)
     %if &z=&v %then %do
       proc print data=test
     run
     %end
   %mend diff
   %diff (3)

   Then %diff(3) will print the data and %diff(5) will not print. But you can not replace '%if-%then' by 'if-then' in this case. If you do, you will have compile errors because now 'if-then' is used outside the data step. This probably is the major difference between 'if-then' and '%if-%then'. If you understand this point, you probably will do fine with '%if-%then'.

2. In the DATA step, you can use 'if' alone to do logical comparison. But you can not use '%if' alone. You also can not use '%else' alone without '%do; %end' pair.

3. '%if-%then' uses MACRO variables in logical expressions and can not refer to DATA step variables in logical expressions (you can not write: %if x=1 %then %do; because x is not a MACRO variable), while 'if-then' can use MACRO variables or use DATA step variables in logical expressions. This is the further explanation of (1).

4. '%if-%then' in MACRO determines what text should be copied to the input stack. When you use 'if-then' inside a MACRO (or non-MACRO), it does logical comparison to determine what DATA step statements should be executed. This is also a further explanation of (1).

The following example shows if no % used, the output is not right:

   DATA test;
   infile cards delimiter=',';
   length codes state $2 vote 8;
   input codes state vote cards;
   Y, WA, 1235
   N, WA, 3210
   Y, CA, 18990
   N, CA, 21222
   ;
   data why;
run;
%macro show(param);
data why;
set test;
if &param=codes then do;
  if codes='Y' then do;
    vote=vote*100;
  end;
end;
else if &param=state then do;
  if state='CA' then vote=vote*100;
end;
else vote=.;
proc print data=why;
run;
%mend;
%show(codes)

<table>
<thead>
<tr>
<th>OBS</th>
<th>CODES</th>
<th>STATE</th>
<th>VOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>WA</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>WA</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>CA</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>CA</td>
<td>.</td>
</tr>
</tbody>
</table>

In the output, we have a new variable 'code' which was not we intended to do (you can change 'code' to 'codes' to see what will happen). The following is the output after using %show(codes)

<table>
<thead>
<tr>
<th>OBS</th>
<th>CODES</th>
<th>STATE</th>
<th>VOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>WA</td>
<td>123500</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>WA</td>
<td>3210</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>CA</td>
<td>189900</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>CA</td>
<td>21222</td>
</tr>
</tbody>
</table>

The following will generate what you really want, if '%' is added:

DATA test;
infile cards delimiter=',';
length codes state $2 vote 8;
input codes state vote;
cards;
  Y, WA, 1235
  N, WA, 3210
  Y, CA, 189900
  N, CA, 21222
;

In this case, '%if-%then' SEEMS WORKING the same way as 'if-then'. But if you use 'code' you will find the difference immediately. Also you can NOT use '%else vote=.;' You can only use '%else %do;
vote=.;%end;'

About Author:
Minghui Yang, a long time SAS user and developer, is a senior business analyst/statistician at Boeing. He went to Stanford in 1982 as a visiting scholar in the Department of Operations Research. In 1988, he received his Ph.D. in statistics from Oregon State University. He has worked for the State of Washington, West Fargo Bank, Providian Bancorp, and MCI in simulation, modeling, forecasting, SAS consulting and business analysis. His current research interest is in modeling, decision support and forecasting. He can be reached at minghui.yang@boeing.com.