1. Problem Statement

This tutorial topic illustrates procedures for table lookup. Two major applications for table lookup are: 1) replacing a coded value with another value such as an alpha label; and 2) recoding, e.g., replacing some value or a range of values by a code, for example coding age in years into age groups (0-9, 10-19, etc.).

Various methods of performing table lookup in SAS are illustrated in the following sections. The first five examples operate on the data set used for the multiple output frequencies examples in the Transposing Data tutorial; for this topic, the variable REGION has been added to the data set. Methods for replacing REGION codes with names are presented. In the last section, several of the table lookup methods are combined to perform complex recoding.

2. Output Formats

In this example, the REGION code only needs to be replaced by its name on output (print). Therefore, it is not necessary to add a new variable to the data set. All that is required is the creation of a format library (lines 5-13, figure 1). A FORMAT statement (line 2), figure 1) is used to cause the REGION name to be printed instead of the REGION code itself.

3. IF Statements

IF statements can be used to add a new variable to the data set whose value is the REGION name. The IF statements used to create the variable REG_NAME (lines 4-13, figure 2) illustrate this technique.

There are two drawbacks associated with the use of IF statements for table lookup, especially for data sets with many distinct codes. First, the IF statements are tedious to code and tend to "clutter up" the program. Second, their use can be inefficient.

4. Merging Data with a Translation Table

The MERGE capability is frequently used for table lookup. It requires less coding than IF statements; however, it is also less efficient (in terms of machine time) and less flexible than most of the other methods presented here. Merging requires a separate sort and data step for each lookup, whereas the other methods allow for multiple table lookups in a single data step.

An example of table lookup using a MERGE is given in figures 3 and 4. A "translation" data set which contains one observation for each REGION code, with its name stored in the variable REG_NAME, is created (lines 6-9, figure 3). The data file is then sorted (lines 21-22, figure 3) and the data set is MERGED with the translation data set (lines 23-24, figure 3) using REGION as a key. If the original sort of the data must be retained, another sort is required (lines 25-27, figure 3).

5. The PUT Function

The PUT function provides an efficient method of table lookup (figure 5). First, a format must be created (lines 5-13). In the subsequent data step (lines 15-20), the REGION name is added to the data set using the PUT function (line 19) which "writes" the formatted value of the variable R into REGION. Note that REGION was renamed on the SET statement (line 17). This causes REGION to become a new variable in the data set. An example is given in section 5 of the Selecting Subsets of Data tutorial.

6. Using ARRAY Structures

Another method of table lookup involves storing the labels/names in an array with the coded value used as the subscript into the array (figures 6 and 7). This method is particularly applicable if the codes are sequential numbers.

The translation data set is read in (lines 6-9, figure 6). In the next data step (lines 22-34, figure 6), the lookup is done. The ARRAY is defined (line 23) and the names are read into the ARRAY using a loop which is executed on the first observation (lines 25-31) with the values of the ARRAY elements retained (line 24) across all observations (note that the names could be read from an external file here; a prior data step to read them in is unnecessary). The observations are read from the data set and the lookup is done (lines 32-33). The value of REGION on each observation is used as the subscript into the ARRAY to get the corresponding value for REG_NAME.

The position of the code to read the names into the ARRAY is important. If this code had appeared after the main SET statement (line 32), the value of REGION on the first observation would be overwritten before the lookup. The lookup on the first observation would use the value of REGION from the last observation in TRNSLATE, resulting in an error.

7. Recoding Using the PUT Function and Arrays

This example (figures 8-10) illustrates a more complex table lookup application using ARRAYS and the PUT function. JOB_CODE and INDUS-
TRY must be recoded (lines 5-17, figure 8). This recoding can be done by storing the recoded values in a two-dimensional ARRAY with JOB CODE and INDUSTRY used as subscripts into this ARRAY. This would require a large table, however. Instead, the job and industry codes can first be recoded, using the PUT function, to the row and column indices of the smaller 4x3 table. First, the formats to convert JOB CODE and INDUSTRY to the row and column indices are created (lines 22-30, figure 8). The two-dimensional table lookup is done in one data step (lines 52-75, figure 9). The ARRAYS are defined (lines 54-58, figure 9) and the table values are read into the two-dimensional array once (lines 60-67, figure 9), retaining (line 59) the values of the ARRAY elements across all observations. The data to be recoded is read in (line 70); the lookups for the JOB CODE and INDUSTRY indices are done (lines 72-73) using the PUT function; and finally, the two-dimensional recode/table lookup is done (line 74) using the ARRAYS.
THIS EXAMPLE ILLUSTRATES THE USE OF FORMATS FOR TABLE LOOKUP. THE FORMATTED VALUE IS NOT ADDED TO THE DATA SET. IT ONLY APPEARS AT PRINT TIME.

PROC FORMAT;
CREATE THE FORMATS;
VALUE REGFMT 1=NORTHEAST 2='MID-ATLANTIC' 3=SOUTH 4=MIDWEST 5=NO纬WEST 6=WEST 7=SOUTHWEST;
NOTE: THE PROCEDURE FORMAT USED 0.05 SECONDS AND 184K.
PROC FREQ DATA=SAVE.CNFRENCE TABLES REGION;
TITLE TABLE LOOKUP USING FORMATS;
TITLE2 FORMATTED VALUE ONLY APPEARS AT PRINT TIME;
TITLE3 WITHOUT THE FORMAT STATEMENT;
NOTE: THE PROCEDURE FREQ USED 0.16 SECONDS AND 172K AND PRINTED PAGE 1.
PROC FREQ DATA=SAVE.CNFRENCE TABLES REGION;
FORMAT REGION REGFMT;
TITLE3 WITH THE FORMAT STATEMENT;
NOTE: THE PROCEDURE FREQ USED 0.16 SECONDS AND 172K AND PRINTED PAGE 1.

REGION FREQUENCY CUM FREQ PERCENT CUM PERCENT
1 157 157 15.907 15.907
2 131 288 13.273 29.179
3 141 429 14.486 43.465
4 145 574 14.691 58.156
5 141 482 14.286 43.465
6 145 847 14.387 86.018
7 138 987 13.982 100.000

REGION FREQUENCY CUM FREQ PERCENT CUM PERCENT
1 157 157 15.907 15.907
2 131 288 13.273 29.179
3 141 429 14.486 43.465
4 145 574 14.691 58.156
5 141 482 14.286 43.465
6 145 847 14.387 86.018
7 138 987 13.982 100.000

THIS EXAMPLE ILLUSTRATES THE USE OF IF STATEMENTS FOR TABLE LOOKUP. A NEW VARIABLE IS ACTUALLY ADDED TO DATA SET.
DATA BUTNAME;
LENGTH REG_NAME $12;
IF REGION EQ 1 THEN REG_NAME = 'NORTHEAST';
ELSE IF REGION EQ 2 THEN REG_NAME = 'MID-ATLANTIC';
ELSE IF REGION EQ 3 THEN REG_NAME = 'SOUTH';
ELSE IF REGION EQ 4 THEN REG_NAME = 'MIDWEST';
ELSE IF REGION EQ 5 THEN REG_NAME = 'NORTHWEST';
ELSE IF REGION EQ 6 THEN REG_NAME = 'WEST';
ELSE IF REGION EQ 7 THEN REG_NAME = 'SOUTHWEST';
NOTE: DATA SET WORK.WITHNAME HAS 987 OBSERVATIONS AND 6 VARIABLES. 340 O DS/8K.
PROC PRINT DATA=WITHNAME OBS=20;
TITLE TABLE LOOKUP USING IF STATEMENTS;
TITLE2 VARIABLE REG_NAME IS ADDED TO THE DATA SET;
TITLE3 PRINT OF 20 OBSERVATIONS FROM DATA SET WITHNAME;
NOTE: THE PROCEDURE PRINT USED 0.10 SECONDS AND 172K AND PRINTED PAGE 1.
NOTE: SAS USED 240K MEMORY.
NOTE: SAS INSTITUTE INC.
SAS CIRCLE
BOX 8000
CARY, N.C. 27511

FILE NAME: WORK.WITHNAME
THE PROCEDURE PRINT USED 0.10 SECONDS AND 172K AND PRINTED PAGE 1.
NOTE: SAS INSTITUTE INC.
SAS CIRCLE
BOX 8000
CARY, N.C. 27511

TABLE LOOKUP USING FORMATS
REGION FREQUENCY CUM FREQ PERCENT CUM PERCENT
1 157 157 15.907 15.907
2 131 288 13.273 29.179
3 141 429 14.486 43.465
4 145 574 14.691 58.156
5 141 482 14.286 43.465
6 145 847 14.387 86.018
7 138 987 13.982 100.000

TABLE LOOKUP USING FORMATS
REGION FREQUENCY CUM FREQ PERCENT CUM PERCENT
1 157 157 15.907 15.907
2 131 288 13.273 29.179
3 141 429 14.486 43.465
4 145 574 14.691 58.156
5 141 482 14.286 43.465
6 145 847 14.387 86.018
7 138 987 13.982 100.000

FIGURE 1
FIGURE 2
**Figure 3**

TABLE LOOKUP USING THE MERGE CAPABILITY TO ADD THE VARIABLE REG_NAME TO THE DATA SET.

<table>
<thead>
<tr>
<th>OBS</th>
<th>REG_NAME</th>
<th>REGION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NORTHEAST</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>MID-ATLANTIC</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>SOUTH</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>MIDWEST</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>NORTHWEST</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>WEST</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>SOUTHWEST</td>
<td>7</td>
</tr>
</tbody>
</table>

**Figure 4**

TABLE LOOKUP USING THE MERGE CAPABILITY TO ADD THE VARIABLE REG_NAME TO THE DATA SET. FINAL DATASET - WITH REG_NAME ADDED.

<table>
<thead>
<tr>
<th>OBS</th>
<th>CBS</th>
<th>ID</th>
<th>PAPER</th>
<th>INVITED</th>
<th>ENJOY</th>
<th>REGION</th>
<th>REG_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>MIDWEST</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SOUTH</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>MIDWEST</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>MID-ATLANTIC</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>WEST</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NORTHEAST</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NORTHEAST</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>SOUTH</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>NORTHEAST</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>MID-ATLANTIC</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SOUTH</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>SOUTHWEST</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SOUTHWEST</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>MID-ATLANTIC</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>SOUTH</td>
</tr>
</tbody>
</table>

*NOTE: DATA SET WORK.TRNSLATE HAS 7 OBSERVATIONS AND 2 VARIABLES. 794 OBS/TK.
NOTE: THE DATA STATEMENT USED 0.04 SECONDS AND 172K.*
THIS EXAMPLE ILLUSTRATES THE USE OF FORMATS FOR TABLE LOOKUP. THE FORMATTED VALUE IS ADDED TO THE DATA SET AS THE VALUE OF THE VARIABLE REGION WHICH IS REDEFINED AS CHARACTER.

PROC FORMAT;
   1  note: the procedure format used 0.05 seconds and 184K.
   2  create the formats;
   3  value regfmt $ 1= "NORTHEAST"
   4  2= "MID-ATLANTIC"
   5  3= "SOUTH"
   6  4= "MIDWEST"
   7  5= "NORTHWEST"
   8  6= "WEST"
   9  7= "SOUTHWEST";
   10
   11  data redefine;
   12  note: the data set work. redefine has 987 observations and 5 variables. 3970 bytes.
   13  note: the data statement used 0.16 seconds and 180K.
   14  array redef (region) $ 12 ri-rg;
   15  retain rl-rg;
   16  if _n_ = 1 then
   17  do /read labels into the array/;
   18  do i=1 to 7;
   19  set translate point=i;
   20  regs=reg_name;
   21  end;
   22  end /read labels into the array/;
   23  set save.reference;
   24  region=regs;
   25  drop rl-rg;
   26
   27  title table lookup using formats and the put function to add the formatted value of region to the data set. the rename parameter is used so region can be defined as character.
   28  print of first 15 observations in data set redefine.
   29
   30  note: the procedure print used 0.10 seconds and 172K and printed page 1.

NOTE: DATA SET USEARRAY HAS 987 OBSERVATIONS AND 6 VARIABLES. 3600 BYTES.
NOTE: THE DATA STATEMENT USED 0.17 SECONDS AND 180K.

PROC PRINT DATA=USEARRAY(OBS=15);
  1  title data set usearray with reg_name added.
  2  title print of first 15 observations in data set redefine.
  3  note: the procedure print used 0.10 seconds and 172K and printed page 2.

NOTE: SAS used 168K memory.
NOTE: SAS INSTITUTE INC.
SAS CIRCLE
BOX 8000
CARY, N.C. 27511

FIGURE 5

FIGURE 6
TABLE LOOKUP BY STORING THE VALUES IN AN ARRAY. REGION IS USED AS A SUBSCRIPT INTO THE ARRAY. PRINT THE THE TRANSLATION DATA SET WHICH WILL BE READ INTO THE ARRAY.

<table>
<thead>
<tr>
<th>OBS</th>
<th>REG_NAME</th>
<th>REGION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NORTHEAST</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>MID-ATLANTIC</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>SOUTH</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>MIDWEST</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>NORTHWEST</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>WEST</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>SOUTHWEST</td>
<td>7</td>
</tr>
</tbody>
</table>

FIGURE 7

TABLE LOOKUP BY STORING THE VALUES IN AN ARRAY. REGION IS USED AS A SUBSCRIPT INTO THE ARRAY. DATA SET USEARRAY WITH REG_NAME ADDED.

<table>
<thead>
<tr>
<th>OBS</th>
<th>REGION</th>
<th>REG_NAME</th>
<th>ID</th>
<th>PAPER</th>
<th>INVITED</th>
<th>ENJOY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>MIDWEST</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>SOUTH</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>MIDWEST</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>MID-ATLANTIC</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>WEST</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>NORTHWEST</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>NORTHWEST</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>SOUTH</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>NORTHWEST</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>MID-ATLANTIC</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>7</td>
<td>SOUTHWEST</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>MID-ATLANTIC</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>SOUTHWEST</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>MID-ATLANTIC</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>SOUTH</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

NOTE: THE PROCEDURE FORMAT USED 0.07 SECONDS AND 164K.

FIGURE 8
DATA RECODE;
*SET UP THE ARRAYS FOR THE TABLE;
ARRAY ROW1 (C) X11-X13;
ARRAY ROW2 (C) X21-X23;
ARRAY ROW3 (C) X31-X33;
ARRAY ROW4 (C) X41-X43;
ARRAY TABLE (R) ROW1-ROW4;
RETAIN X11--X43;
IF _N_=1 THEN DO /* READ IN THE TABLE */
DO R = 1 TO 4; /* READ EACH ROW;*/
DO C = 1 TO 3; /* READ EACH COLUMN;*/
INPUT TABLE Q);
END;
END /* READ IN THE TABLE */;
SET JOBS;
*READ IN THE DATA TO BE RECODED;
R=PUT(JOB_CODE.ROW.)
C=PUT(INDUSTRY.COL.)
NEW_CODE=TABLE;
CARDS;

CHARACTER VALUES HAVE BEEN CONVERTED TO NUMERIC
VALUES AT THE PLACES GIVEN BY: (LINE):(COLUMN).
72:4 73:4
NOTE: SAS WENT TO A NEW LINE WHEN INPUT STATEMENT
REACHED PAST THE END OF A LINE.
NOTE: DATA SET WORK.RECODE HAS 10 OBSERVATIONS AND 17 VARIABLES. 136 OBS
*THE USE OF FORMATS, THE PUT FUNCTION AND ARRAYS TO
DO MORE COMPLICATED TABLE LOOKUP. THE TABLE IS
READ INTO A TWO DIMENSIONAL ARRAY.

THE TABLE
X11 X12 X13 X21 X22 X23 X31 X32 X33 X41 X42 X43
1 1 2 8 3 4 8 5 6 8 7 7 9

PROC PRINT DATA=RECODE(OBS=1));
VAR X11--X43;
TITLE4 THE TABLE;
NOTE: THE PROCEDURE PRINT USED 0.09 SECONDS AND 172K
AND PRINTED PAGE 2.

PROC PRINT;
DROP X11--X43;
TITLE4 DATA RECODE - WITH THE VARIABLE NEWCODE;
NOTE: THE PROCEDURE PRINT USED 0.09 SECONDS AND 172K
AND PRINTED PAGE 3.

NOTE: SAS INSTITUTE INC.
SAS CIRCLE
POX 8000
CARY, N.C. 27511

FIGURE 9

FIGURE 10