Table Look-Up and Recoding the Lazy Way with PROC FORMAT and CNTLIN Data Sets

Michael L. Davis
Bassett Consulting Services, Inc., North Haven, Connecticut

(Note: This material on the FORMAT procedure and CNTLIN data sets has been presented previously at SAS® User Conferences. Michael crusades against "spaghetti *if* ... *then* coding" whenever a soapbox is present. However, he has nothing against a steaming plate of spaghetti.)

While FORMAT Procedure can be a very machine-efficient method to accomplish table look-up, its ability to improve a programmer's coding efficiency is probably its most attractive feature. For example, this procedure catches duplicate key values. Also, once saved in a library, the formats you create can be called into other programs with a line of code. Finally, if you use the library concept, when you change the format, you update all programs that use it. Along with covering how to create the CNTLIN data sets to do this, recoding SAS variables using the FORMAT procedure and multiple arguments will be covered.

The following examples demonstrate how to create formats and informats from SAS data sets. Lets assume you work for Baystate Baby Buggies. You need to calculate salesperson commissions from a sales transaction data set for the New England region only. This involves three table-lookups. First, you have to identify non-New England transactions so you can drop them. Second, you have to lookup the commission rate for each transaction since they vary by model of baby buggy. Last, you have to look up the salesperson since the transaction record only has the salesperson code.

Instead of sorting and using by-merges, create the following data set as illustrated from the following example:

```
data buggyfmt;
    input
      @1  fmtname  $7.
      @8  start    $7.
      @15 label    $7.
      @22 hlo      $1.
      @24 type     $1.
      ;
cards;
region CT     1        I
region MA     1        I
region ME     1        I
region NH     1        I
region RI     1        I
region VT     1        I
region OTHER  0      O I
slsprn JB     BLACK    C
slsprn RW     WHITE    C
slsprn OTHER  *ERROR*O C
crate  SINGLE 0.050    I
crate  BIG    0.075    I
crate  SUPER  0.100    I
crate  OTHER  0.000  O I
;
run;

proc format cntlin=buggyfmt;
run;
```

When the above code is run, two informats and one format will be outputted by the FORMAT procedure. The REGION format will return 1 for each of the six New England states and zero for any other. The CRATE format will return a decimal commission rate from 5%-10% for each of the buggy models (SINGLE, BIG, or SUPER). The $SLSPRN sales format returns the last name of the salesperson for each of the salesperson codes (JB, RW).

Some additional explanation about the FORMAT procedure may be helpful. One can generate several formats from a single CNTLIN SAS data set as shown in the preceding example. To denote that a particular observation should assign a LABEL value to data ranges not listed, the HLO (high-low-other) variable is assigned the value "O" (oh). There is no need to prefix contents of the FMTNAME variable with a dollar sign ($) when creating the $SLSPRN format since the FORMAT procedure does this automatically when the CNTLIN= option is used.

To insure that we do not fail to match a lookup value due to case differences, all values for the START variable are inputted as upper-case. If one applies the UPCASE function to the SAS variable being used as the lookup key, the possibility of not matching due to case differences will be eliminated.

In the case of the CRATE and $SLSPRN formats, the "OTHER" data range is used to flag potential keying errors. If we see *ERROR* in a report for the salesperson name, we know that a value not in the salesperson table was keyed in for salesperson code. Similarly, we can trap commission amounts equal to zero for possible mis-coded buggy models.

To use these formats effectively, one would code a subsequent data step to look like the following:

```
data sales;
    set sales;
    if input(upcase(state),region.);
    slsprnm=put(upcase(slsprn),
      $slsprn.);
    comamt=
input(upcase(model),crate.)
      *salesamt;
run;
```

The INPUT expression in the IF statement is true (equal to 1) only for the six New England states and subsets the SALES data set accordingly. The variable SLSPRNM assumes the last name that corresponds to the salesperson code. The COMAMT variable contains the commission amount, which is the product of the commission rate times the sales amount.

In many applications, you have an existing SAS data set containing the values you need for the required CNTLIN data set. However the variable names need to be changed and you want to specify an OTHER data range. The following is a sample program that shows an efficient way to accomplish this when you have a data set containing the region assignments for the Baystate Baby Buggies:

```
data statefmt(drop= region) ;
    retain
      fmtname 'region'
      hlo ' '
      type 'I'
      ;
    set
statenm(rename=(state_nm=start)
      keep=state_nm region) nobs=eos;
    if region= 'NE' then label= 1;
    else label= 0;
    output;
    if _n_ = eos then do;
        start= 'OTHER';
        label= 0;
        hlo= "O";
        output;
    end; run;
```

As a final example, lets assume that Bay State Baby Buggies wants to recode the salesperson assignments based on both state and customer size. Here is how to create the salesperson recoding format (SLSRCD):

```
data recode;
  retain
    fmtname 'slsrcd'
    hlo     ' '
    type 'C'
    ;
  input start $ label $;
  if start= 'other' then hlo= 'O';
  cards;
CTLARGE     JB
MALARGE     JB
MELARGE     JB
NHLARGE     RW
RILARGE     RW
VTLARGE     RW
CTSMALL     RW
MASMALL     RW
MESMALL     RW
NHSMALL     JB
RISMALL     JB
VTSMALL     JB
OTHER       **
;
run;

proc format cntlin=recode;
run;

data sales;
  set sales;
  slsprn= put(state ||
cosize,$slsrcd.);
run;
```

In this example, the RETAIN statement is used to assign values to the variables used in the CNTLIN data set variables that remain constant. The lines following the CARDS statement create a "truth table" representing all possible combinations of states and company sizes, plus an OTHER range to trap coding errors. The last part of the example recodes SLSPRN based upon the combination of STATE and COSIZE (company size).

So when you have a table look-up or recoding application, give the FORMAT procedure a try!