Hot-deck Imputation with SAS® Arrays and Macros for Large Surveys

John Stiller and Donald R. Dalzell
Continuous Measurement Office, Demographic Statistical Methods Division, U.S. Census Bureau

ABSTRACT

SAS arrays and macros are used to implement a hot-deck imputation routine to fill in missing survey data for all items of a large demographic survey, the American Community Survey (ACS). This paper describes the use of a data step with multi-dimensional arrays and macros to perform the imputation on a set of variables simultaneously. This differs from other approaches which sort the data by classing variables and use data steps with RETAIN statements to impute for each variable, one at a time. The technique outlined here is more suitable when data files are large, and where many variables need to be imputed. In this scheme, variables which define classes for the imputed variable become associated with dimensions in the hot-deck array. Through the use of arrays, four donors are held in the hot-deck for each imputation cell. Thus, different donors are available when consecutive occurrences of missing data are encountered. The macros simplify the process of retrieving values from the hot-deck, circulating values within the hot-deck, and storing donor values to the hot-deck. Though cold-deck values are used to initialize the hot-deck, the macros warm the hot-deck to provide more realistic values for the imputation.

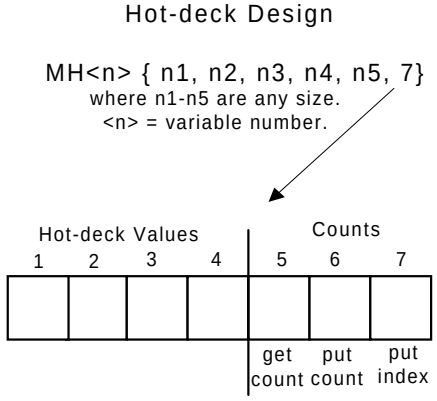
INTRODUCTION

Before survey data can be tabulated, the problem of missing values must be addressed. At the Census Bureau, hot-deck imputation has been used successfully in the decennial census to fill in the missing data items for a number of years. In the past, the primary language used was FORTRAN. Here we present SAS macros and code which apply the programming approach used in processing the census to the ACS, where SAS is the programming language. Variables can have missing values for one of two reasons: either the answer was not reported, or it was reported but logically inconsistent with other items (and thus blanked by consistency edits prior to imputation). The hot-deck imputation process fills in missing values as a function of other related variables in the observation and the position of the observation relative to others in the data. The approach we have developed is a reliable and efficient imputation procedure, and should be applicable to other large surveys.

HOT-DECK DESIGN

The hot-deck is a complex set of rules implemented as a computer program for manipulating data; it produces a "clean" data set containing no missing values (Schafer 1989). To create the hot-deck matrix for a variable, we define an array with six dimensions. We associate class variables which define cells for the imputation variable with different dimensions in the hot-deck matrix in this scheme. The class variables are typically a recode of one or more of the related input variables. As many as five separate combinations of class variables may be specified; however, two or three dimensions are used in most cases. The sixth dimension is reserved for storing hot-deck values and imputation counts for each cell (see Figure 1). In our system, the hot-deck has a depth of four values. Storing four donors per cell prevents reuse of the same donor in cases where consecutive occurrences of missing are present in the data. This is an issue for us in the ACS, since our item non-response rates vary between 1-20% depending on the question, and therefore some "runs" of missing data occur. Cold-deck values are used to initialize the hot-deck arrays when they are declared, but we utilize them only in those cases when there are insufficient donors for a particular cell. Cold-deck values are typically chosen to be the most likely legal value for the variable in question. Imputation cell counts are written to a data set for analysis at the end of the process.

Figure 1



IMPUTATION METHODS

File Order- We sort the input file so as to place observations with similar characteristics together in the file. For the housing items of the ACS, we sort the file in geographical order, by state, county, census tract, block, street name, house number, and apartment number. The rationale for this is that housing units will tend to have housing characteristics similar to those of their neighbors. When population items are imputed, the file could first be sorted by characteristics other than geography such as race, gender, etc.

Imputation Cells- For each matrix, class variables are defined which specify various cells for categorizing donors for the imputation variable. In other words, class variables indicate the related variables of which the imputation variable is a

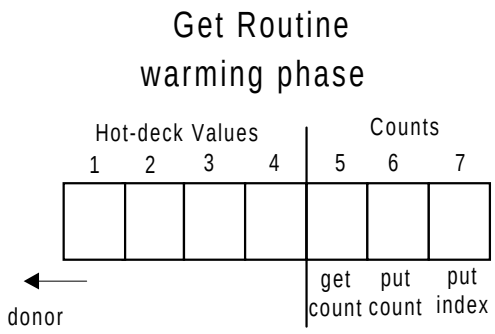
function. For example, for the ACS item Agricultural Sales (H6), the class variables have been defined as Type of Building (H1) and Lot Size (H5). Values of Type of Building such as "mobile home" or "single family", and of Lot Size equal to "1 to less than 10 acres" or "10+ acres" define categories into which donor values should be grouped. Class variables themselves need not be imputed prior to their use with other variables, though a category where the class variable is missing will need to be included. Creating imputation cells using arrays replaces the need for sorting the input data by class variables, as is done in some other imputation procedures.

Hot-deck Warming- Warming the hot-deck entails reading through the input file and loading donors into the matrix. This is done so that donors are available in the hot-deck prior to the actual imputation. Thus, in cases when recipients precede donors in the file for a given hot-deck cell, donors are already present in the matrix.

MACROS

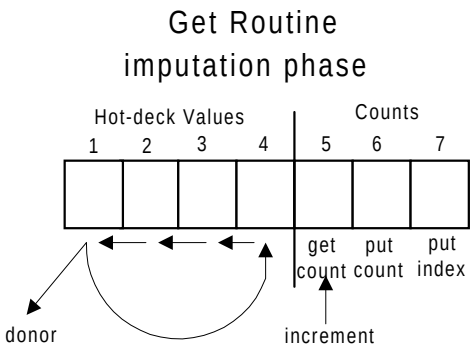
We wrote the mget and mput macros given below to implement these techniques. Note that they operate somewhat differently in the warming and imputation phases. They distinguish the two phases by testing the flag variable called "warm", which must be set prior to invoking the macros. It takes on values of either 1 (warming phase) or 0 (imputation phase). The macros also require that the indices pointing to a particular matrix cell, i, j, k, l, and m are assigned prior to their invocation. In addition, they make use of a temporary variable, vv, to hold the value stored into or returned from the matrix.

Figure 2



Mget macro - returns the last value loaded into the referenced hot-deck matrix. The only parameter to be specified is the imputation matrix name. In the warming phase, donor values are taken from the first element of the hot-deck with no circulation of values within the hot-deck. In the imputation phase, a donor is taken from the first element of the hot-deck, it is also placed at the end of the deck (element 4), and the other values are shifted back (left) one position. The get counter (element 5) is then incremented by one (See Figures 2 and 3).

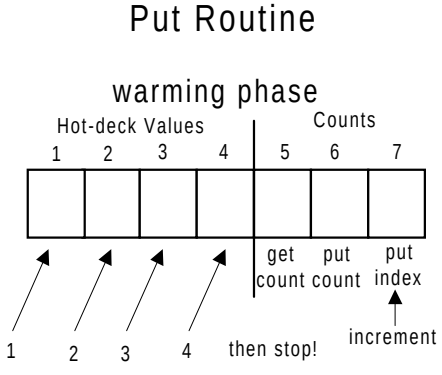
Figure 3




```
%macro mget( arr ) ;
%* Retrieve value from hot-deck matrix arr * ;
      vv = &arr{i,j,k,l,m,1} ;
      if ( not warm  ) then do ;
           %* cycle back prev. values - recirculate * ;
           &arr{i,j,k,l,m,1} = &arr{i,j,k,l,m,2} ;
           &arr{i,j,k,l,m,2} = &arr{i,j,k,l,m,3} ;
           &arr{i,j,k,l,m,3} = &arr{i,j,k,l,m,4} ;
           %* place used value at end of deck * ;
           &arr{i,j,k,l,m,4} = vv ;
           %* increment get counter * ;
           &arr{i,j,k,l,m,5} + 1 ;
      end ;
%mend mget ;
```

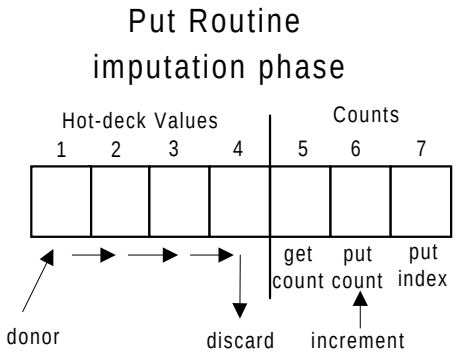
Mput macro - loads donor values into the referenced hot-deck matrix. It also requires the imputation matrix name as the sole parameter. In the warming phase, donors are loaded into the hot-deck left to right until four values are collected. During warming, a put index (element 7) is used to keep track of where the next donor should be stored. In the imputation phase, donors are loaded into element 1, the other values are shifted forward (right) one position, and the last value (the oldest) is discarded. The put counter (element 6) is also incremented by one (see Figures 4 and 5).

Figure 4



```
%macro mput( arr ) ;
%* Store value into hot-deck matrix arr * ;
      if ( warm ) then do ;
            %* warming - warm deck left to right & stop at 4 *;
            if ( &arr{i,j,k,l,m,7} <= 3 ) then do ;
               %* increment put index * ;
               &arr{i,j,k,l,m,7} + 1 ;
               &arr{i,j,k,l,m, &arr{i,j,k,l,m,7} } = vv ;
            end ;
            end ;
      else do ;
            %* imput. phase-cycle values forw. in deck * ;
            &arr{i,j,k,l,m,4} = &arr{i,j,k,l,m,3} ;
            &arr{i,j,k,l,m,3} = &arr{i,j,k,l,m,2} ;
            &arr{i,j,k,l,m,2} = &arr{i,j,k,l,m,1} ;
            &arr{i,j,k,l,m,1} = vv ;
            %* increment put counter * ;
            &arr{i,j,k,l,m,6} + 1 ;
      end ;
%mend mput ;
```

Figure 5



EXAMPLE

The Agricultural Sales (H6) question on the ACS survey asks "In the past 12 months, were the sales of all agricultural products from this property $1,000 or more?". Valid responses are either 1=yes, or 2=no. The universe for the question is all occupied mobile and single family homes (H1=1-3) with a lot size one acre or larger (H5 = 2 or 3). H6 should be missing (out of universe) for all other units.

The imputation matrix for this variable has been defined in our specifications as having four cells, using Type of Building (H1) and Lot Size (H5) as the class variables. H1 is associated with the row dimension in the matrix, and H5 in the column dimension (see Table 1). The starting values, 2=No for this case, indicate the cold-deck values used at array declaration. Note that H1 and H5 precede H6 in our imputation program, so any missing values in these class variables would be filled in by this point.

Table 1
MH6 - Imputation Matrix and Starting Values for Agricultural Sales (H6)

		Lot Size (H5)	
		1-10 acres (H5=2)	10+ acres (H5=3)
Type of Building (H1)	Mobile home (H1=1)	2	2
	Single family (H1=2,3)	2	2

Imputation code for variable H6:

We present code fragments that are part of a larger SAS data step program for performing the imputation for this variable.

First we declare the imputation matrix for H6 using a temporary array called MH6. Two elements each for the row and column dimensions are used in this case (as specified in Table 1), and a size of seven for the hot-deck dimension, as follows:

```
array MH6{ 2, 2, 1, 1, 1,  7 }  _temporary_

      ( 2 2 2 2 0 0 0      /* for cell (1,1) */
        2 2 2 2 0 0 0      /* for cell (1,2) */
        2 2 2 2 0 0 0      /* for cell (2,1) */
        2 2 2 2 0 0 0      /* for cell (2,2) */
      ) ;
```

The list of initial values includes seven values for each cell for a total of 28, the first four of each set are the starting values (cold-deck values), and the last three are zeros for the cells holding imputation counts.

Next, we initialize the imputation flag variable to "not allocated" (FH6=0), and test to see if the universe condition for H6 is met. If false, H6 should not be answered and is set to missing and we are done. If true, then H6 should have a response and will go through the imputation code. Then it is necessary to set the indices into the hot-deck matrix that will be used by the mget and mput macros. The row dimension (i subscript) corresponds to the H1 class variable, and the column dimension (j subscript) to the H5 class variable. The other indices required by the macros are not used for this variable, and are therefore assigned a default value of one. If H6 is missing, we will assign it a value from the matrix using mget. We then set the related imputation flag variable, FH6, to one to indicate an imputed value. Alternatively, if H6 has a value, we load this into the matrix as a donor using mput.

```
* Initialize imputation flag to not imputed * ;
FH6 = 0 ;
* Test universe condition * ;
if  ( H1 in (1,2,3) and H5 in (2,3)  ) then do ;
        * Set indices into hot-deck matrix * ;
        if ( H1 eq 1 ) then i = 1 ; else i = 2 ;
        if ( H5 eq 2 ) then j = 1 ; else j = 2 ;
```

```
        k = 1 ; l = 1 ; m= 1 ; * unused * ;

        * Perform the Imputation * ;
        if ( H6 eq .) then do ;
                %mget( MH6 )      /* retrieve value from matrix */
                H6 = vv ;         /* impute H6 */
                FH6 = 1 ;         /* set flag for H6 to imputed */
                end ;
        else do ;
                vv = H6 ;         /* place donor in temp. var */
                %mput( MH6 )      /* store H6 to the matrix */
        end ;
else
        H6 = . ;
```

JOINT IMPUTATION

Imputing two or more variables jointly is not difficult, even though only one number at a time can reside in a given element of the hot-deck array. The values are first combined into one number and then stored in the matrix. In the ACS, for example, we impute Property Value (H28) and Property Taxes (H29) jointly using matrix MH28. Taxes (H29) has a maximum value of $99,999 so a multiplier of 100,000 is used to combine the two without losing data. When a value is retrieved from the matrix, it must be split into two numbers again using the INT and MOD functions. The code is as follows:

```
if ( H28 ne . and H29 ne . ) then do;
        vv = H28 * 100000 + H29 ;
        %mput( MH28 )
        end ;

else if ( H28 eq . and H29 eq . ) then do;
        %mget( MH28 )
        H28 = int( vv / 100000 ) ;
        H29 = mod( vv, 100000 ) ;
        FH28 = 1 ;
        FH29 = 1 ;
        end;

else . . .
```

IMPUTATION DRIVER PROGRAM

The following data step code gives a driver program which can be used with the above macros to implement a flexible and efficient hot-deck imputation scheme. The input data is sorted once in a particular sequence for a group of variables (e.g. geographical for the ACS housing items). The program then passes the input data twice, once for warming and again for the imputation. Imputation counts are accumulated in the matrices and written to a data set, mcnts, at the end of the program.

```
* Driver Program for Hot-Deck Imputation * ;
proc sort data = lib.in ;
        by <sort keys>  ;
run;

data    lib.out         /* imputed data */
        lib.mcnts  ;    /* matrix counts */

        [array/matrix declarations]

        if ( _n_ eq 1 ) then do ;
                * Set flag to indicate warming phase * ;
                warm = 1 ;
                do until( wend ) ;
                   set lib.in  end=wend ;
                   link  impcode ;        /* call imputation code */
                end ;
        end ;

        set lib.in  end=iend ;

        * Set flag to indicate imputation phase * ;
        warm = 0 ;
        link impcode ;                    /* call imputation code */

        if ( iend ) then do;
                <dump imputation matrices to data set > ;
        end;
return ;

impcode:
        * Imputation code for each variable * ;
        %include 'imph1.sas' ;
        %include 'imph2.sas' ;
                .
                .
                .
return ;

run;
```

The program files imph<n>.sas contain code like that given previously for H6.

IMPUTATION ANALYSIS REPORTS

We produce a report for each matrix in the imputation which gives counts and imputation rates for every hot-deck cell. The report is used for analysis purposes only. The code to dump the matrix counts is straightforward, and consists of nested do loops with the matrix reference and an OUTPUT statement. The resulting data set is transposed to group all the data for each cell on one observation, and then the report is generated from the result using Proc TABULATE.

In the following example, the report for H6 is given for one county in our survey. The table displays the last four values remaining in the hot-deck at the end of the imputation, along with the number of imputed values (gets), number of donors stored (puts), and the cell imputation rate (pct) (see Table 2). It is apparent from the report that imputation rates for the four cells vary between 0 and 19%, which is in acceptable limits. Notice also that the largest number of donors (puts), 1307, is found in cell (2,1) for single-family homes on medium lots (1 to less than 10 acres), as one might expect.

Another report we generate for analysis is the "Editals" report. It is a standard Census report for analyzing the results of the imputation (and of the consistency edit). We programmed it in SAS using a combination of Proc FREQ and Proc COMPUTAB. The report gives counts and percentages of reported and imputed values for each of the variables in the imputation. Table 3 gives the results for H6 for Brevard Co., Florida. The imputation rate for each value of a variable, along with the overall imputation rate, are of particular interest. For H6, the table shows that of the 396 who answered yes, 367 were reported and 29 were imputed. The imputation rate for the Yes responses was 7.3% (the Across-% column), while the overall imputation rate for H6 was 6.8% (Across-% of In Universe row), well within acceptable bounds.

Table 2
Brevard Co., FL - Imputation Rates By Hot-deck Cell
American Community Survey - 1996

Matrix MH6

					Depth1	Depth2	Depth3	Depth4	Gets	Puts	Pct
I	J	K	L	M							
1	1	1	1	1	2	2	2	2	8	89	9.0
	2	1	1	1	1	1	2	2	0	15	0.0
2	1	1	1	1	2	2	2	2	101	1307	7.7
	2	1	1	1	1	1	2	2	7	37	18.9

Table 3
AMERICAN COMMUNITY SURVEY - 1996 EDITALS REPORT
BREVARD CO., FL

VARIABLE=H6 Agricultural sales of $1000 or more

VALUES	NUMBER	----%	CUM-%	REPORTED	----%	IMPUTED	----%	ACROSS-%
TOTAL	206,313	100.0	0.0	205,477	100.0	836	100.0	0.4
NOT IN UNIVERSE	193,984	94.0	0.0	193,984	94.4	0	0.0	0.0
IN UNIVERSE	12,329	6.0	0.0	11,493	5.6	836	100.0	6.8
1= Yes	396	3.2	3.2	367	3.2	29	3.5	7.3
2= No	11,933	96.8	100.0	11,126	96.8	807	96.5	6.8

CONCLUSION

Using some simple SAS macros and array processing, we have implemented a robust and efficient hot-deck imputation procedure suitable for large surveys. Sorting passes are reduced by placing imputation cells into multidimensional arrays, while macros simplify the task of managing the hot-deck.

REFERENCES

Schafer, Joseph L., 1989 "Evaluating Imputation in the Decennial Census", Joint Statistical Agreement 88-02, U.S. Bureau of the Census and Harvard University, 16 pages

Carlson, Cox and Bandeh 1995. "SAS Macros Useful in Imputing Missing Survey Data", Proceedings of the Twentieth Annual SAS Users Group International Conference, SAS Institute Inc., Cary, NC. pp 1089-1094.

Wright, P.M. 1993 "Filling in the Blanks: Multiple Imputation for Replacing Missing Values in Survey Data", Proceedings of the Eighteenth Annual SAS Users Group Conference. SAS Institute, Inc., Cary, NC pp 1123-1125

Euller, Roald 1995 "Imputing Missing Survey Data: Three SAS Algorithms", Proceedings of the 8th Annual NESUG Conference, NorthEast SAS Users Group, pp 269-270

Burns, Olsen 1995 "HOTROD: An Intelligent Hotdeck Imputation System", Proceedings of the 8th Annual NESUG Conference, NorthEast SAS Users Group, pp 764-771

ACKNOWLEDGMENTS

The authors would like to express their appreciation to Barbara Diskin, DP Team Leader, Continuous Measurement Office, for her support, encouragement and editing help. We would also like to thank Peggy Goldsmith for editing assistance. Thanks

also to Linda Ball who wrote the original matrix counts program which we modified, and to Gregg Diffendal for sharing his knowledge of the statistical basis of hot-deck imputation.

CONTACT INFORMATION

The authors can be contacted at Continuous Measurement Office, Demographic Statistical Methods Division, Rm SHEP-2A, Bureau of the Census, Washington, DC 20233, or by telephone at (301)763-8332. John's internet address is jstiller@census.gov