ABSTRACT

Hot-deck imputation is a means of imputing data, using the data from other observations in the sample at hand. This paper deals with a method of imputation we used for the Survey of Adults on Probation. In this method, we first attempt to impute race based on other variables in an observation. We impute the remaining missing race data, and age and gender, using the same variable in prior or latter observations in the data set.

This method uses macro variables to subset the data into groups within which imputation is done. It also uses arrays to keep track of values to use for imputation.

INTRODUCTION

The method of hot-deck imputation described in this paper is used for the Survey of Adults on Probation (SAP), conducted by the Bureau of the Census. To provide a basis for the imputation method, I will give a brief description of the survey sample design. The programming principles described can then be extended to other situations where imputation is needed.

The Survey of Adults on Probation was based on a sample of prisons from across the country. Inmates were selected using a roster filled out by a probation officer for the given prison. Each prison was considered a group (or ctrlnum), for imputation purposes. The probation officer filled out a questionnaire for each inmate. The questionnaire was later compared to one completed by the inmate. The imputation discussed here was for the data missing from the questionnaire filled out by the probation officer.

The imputation requirements for SAP are based on the data we need to impute, and the survey sample design.

1. We impute age, race and gender independently.
2. For race, we first try to base imputation on other data (e.g., ethnicity) for the same person.
3. We impute all remaining missing values using only "good" (unimputed) data for another person in the same group, or ctrlnum.
4. We use each "good" data value only once for imputation.
5. We work backwards over the ctrlnum; then, if necessary, we work forwards.
6. If no "good" data can be used from within the ctrlnum, we assign some type of "out of range" value.

THE METHOD

The imputation process is accomplished using four successive steps, taking into account the above requirements. First, we produce a data set containing the number of persons per ctrlnum.

Next, we create macro variables for the ctrlnums, and number of persons per ctrlnum, using the output from the first step. As the first step in the actual imputation, we attempt to resolve race based on other data for the same person if possible. We finally use hot-deck imputation to fill in the remaining data, where possible.

This is accomplished by first creating the necessary data and flag arrays for age, race and gender. For race, we create a new temporary flag array, so we can retain new race variable imputations. Finally, we work backwards, and then forwards through a given ctrlnum, to impute data, by evaluating the data for other persons in the ctrlnum. Further details on these principles, including imputation through datastep concepts, are given below.

To produce a data set containing the number of persons per ctrlnum, we "set" the original data set "by" ctrlnum. The data set is sorted by ctrlnum. Within each ctrlnum, we use a counter for the number of lines encountered (numlines), using first.ctrlnum to initialize numlines. We retain the previous value with each datastep iteration, and output to the new data set only for the last.ctrlnum. In this way, we produce a new data set (sap1ctl). This data set contains one observation for each ctrlnum, with two variables: ctrlnum, and total number of persons in the ctrlnum, numlines.

Next, we create macro variables for the above two variables and number of ctrlnums. Using a null datastep, we SET sap1ctl and use the call symput routine to create the macro variables &ctrlnum&i and &numline&i for the ith observation in sap1ctl. On the last iteration of this datastep we again call the symput routine to create &n, the macro variable for the total number of ctrlnums encountered.

The code for this section of the program is in Appendix A. This appendix also shows a PROC PRINT indicating the first several observations of sap1ctl, including the ctrlnum number, and number of lines within the ctrlnum. There are 167 ctrlnums within SAP. (It should be noted here that ctrlnums shown in this...
paper are for display purposes only, and not actual ctlnums used in the survey. This change in value is made for
The overall imputation process is done using a macro, "subseta", within which we work with each ctlnum, one at a time. This is done with a %do loop using the macro variable &n, defined above, for the number of ctlnums in the sample. The last step in this macro is a proc append, which accumulates the component data sets for ctlnums into a final data set, sap1imp. Because we've using a PROC APPEND, if it is necessary to rerun the program, sap1imp must be deleted beforehand, since it is the base data added to, during the procedure.

The first step in the imputation process is to attempt to resolve race based on other data for an observation. First, we subset the original data set, sap1e, BY ctlnum, to get sap1e&i. We do this so we can use the automatic iteration indicator, _n_, in the next datastep in the race resolve process. In the next DATA step, using sap1e&i, we produce data sets of imputed race only (sap1r&i), and ctlnum-level arrays of imputed race flags (sap1f&i). Each element in the array corresponds to a line where race has (or has not) been imputed. The final steps in the initial race resolve process are to MERGE sap1e&i with sap1r&i, and finally the resulting sap1e&i with sap1f&i. The code for this section of the program is shown in Appendix E. The code is shown for the variable AGE; that for the other two variables is similar.

AN EXAMPLE

An example of how the imputation process works is shown in Appendix F, for ctlnum "2001". Some of the AGE data in this example is fabricated, to more fully illustrate how the algorithm executes. I show the demographic variable of AGE only, for simplicity of observing the working process. This example contains basically four sets of imputations, three of which occur within lines 001-014. The last set occurs for lines 075-076.

The first imputation occurs for line 001. Since we cannot work backward, we must take the AGE value for the next line, 002. The AGE value for line 004 can be imputed using that for line 003, so we use it. We must impute for lines 006-010 using the backward-forward approach. Line 005 is the only one of the previous observations in the ctlnum which has a "good" value of AGE which has not already been used for imputation. Therefore, its value is the only one of the beforehand ones we can use. For the remainder of the lines in the sequence (007-010), we work successively forward, using the AGE values from lines 011-014, respectively.

The last set of imputations for ctlnum "2001" occurs for lines 075-076. These can be done using the simple "backward" approach. The AGE values for lines 074 and 073 are assigned to lines 075 and 076, respectively.

CONCLUSION

This method is one approach to hot-deck imputation. Other methods may be appropriate to different sample designs. Aspects of this method may be useful for other cases of hot-deck imputation and can be applied as necessary. The full text of the program described here is available on request from the author.

AUTHOR

Lawrence Altmayer
U.S. Bureau of the Census
ESMPD, Room 1200-4
Appendix A

data sap1ctl (keep=ctrlnum numlines);
set sap1e;
by ctrlnum;
if first.ctrlnum then numlines=0;
numlines+1;
if last.ctrlnum then output;
run;

data _null_; set sap1ctl end=e;
call symput('ctrl'||left(_n_),trim(ctrlnum));
call symput('line'||left(_n_),left(trim(numlines)));
if e then call symput('n',left(_n_));
run;

Appendix B

%macro subseta;
%do i=1 %to &n;
/* subset sap1e by ctrlnum, and first resolve race based */
/* on other data - create flag array for imputing race */
data sap1e&i;
set sap1e;
if ctrlnum="&&ctrl&i";
run;

data sap1r&i (keep=imprace)
   sap1a&i (keep=ctrlnum racfl1-racfl&&line&i);
set sap1e&i;
by ctrlnum;
if ctrlnum="&&ctrl&i";
retain racfl1-racfl&&line&i;
array arrfl{&&line&i} $ racfl1-racfl&&line&i;
if (b4='5' or b4='8' or b4='') and b5='1' then do;
imprace='1'; arrfl{_n_}='1'; end;
if ctrlnum='2601' and (b4='8' or b4='') then do;
imprace='1'; arrfl{_n_}='1'; end;
if last.ctrlnum then output sap1a&i;
output sap1r&i;
run;

data sap1e&i;
merge sap1e&i sap1r&i;
run;

data sap1e&i;
merge sap1e&i sap1a&i;
by ctrlnum;
run;

Appendix C
/* create arrays necessary for imputing age, sex and race using hot deck method */

data sap1b&i (keep=ctrlnum age1-age&&line&i sex1-sex&&line&i rac1-rac&&line&i);
set sap1e&i;
by ctrlnum;
retain age1-age&&line&i sex1-sex&&line&i rac1-rac&&line&i;
array arage{&&line&i} $ age1-age&&line&i;
array arsex{&&line&i} $ sex1-sex&&line&i;
array arrac{&&line&i} $ rac1-rac&&line&i;
array arrfl{&&line&i} $ racfl1-racfl&&line&i;
arage{_n_}=age; arsex{_n_}=b2; arrac{_n_}=b4;
if last.ctrlnum then output;
run;

data sap1e&i;
merge sap1e&i sap1b&i;
by ctrlnum;
run;

Appendix D

/* hot deck imputation for age, sex and race */

data sap1f&i (drop=i age1-age&&line&i sex1-sex&&line&i rac1-rac&&line&i agefl1-agefl&&line&i sexfl1-sexfl&&line&i racfl1-racfl&&line&i trcfl1-trcfl&&line&i);
attrib impage length=$3 impsex length=$1;
set sap1e&i;
retain agefl1-agefl&&line&i sexfl1-sexfl&&line&i trcfl1-trcfl&&line&i;
array arage{&&line&i} $ age1-age&&line&i;
array arsex{&&line&i} $ sex1-sex&&line&i;
array arrac{&&line&i} $ rac1-rac&&line&i;
array arrfl{&&line&i} $ racfl1-racfl&&line&i;
array arfl{&&line&i} $ racfl1-racfl&&line&i;
array arrfl{&&line&i} $ racfl1-racfl&&line&i;
array trcfl{&&line&i} $ trcfl1-trcfl&&line&i;

/* create temporary array for race flags, since previously created array cannot be retained */

if _n_=1 then do;
   %do j=1 %to &&line&i;
      trcfl&j=racfl&j;
   %end;
end;

Appendix E
/* hot deck imputation for age */
if age=" then do;
    if _n_>1 then do;
        do i=_n_-1 to 1 by -1;
            if arage{i}=" and arafl{i}='1' then do;
                arafl{i}='1'; impage=arage{i};
                goto nextsex;
                end;
            end;
    end;
    if impage=" then do;
        do i=_n_+1 to &&line&i;
            if arage{i}=" and arafl{i}='1' then do;
                arafl{i}='1'; impage=arage{i};
                goto nextsex;
                end;
            end;
        end;
    if impage=" then impage=999;
    end;
/* _n_=1 */
else do;
    do i=_n_+1 to &&line&i;
        if arage{i}=" and arafl{i}='1' then do;
            arafl{i}='1'; impage=arage{i};
            goto nextsex;
            end;
        end;
    if impage=" then impage=999;
end;
end:

Appendix F
<table>
<thead>
<tr>
<th>CTRLNUM</th>
<th>LINE</th>
<th>AGE</th>
<th>IMPAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>001</td>
<td></td>
<td>026</td>
</tr>
<tr>
<td>2001</td>
<td>002</td>
<td>026</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>003</td>
<td>046</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>004</td>
<td></td>
<td>046</td>
</tr>
<tr>
<td>2001</td>
<td>005</td>
<td>027</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>006</td>
<td></td>
<td>027</td>
</tr>
<tr>
<td>2001</td>
<td>007</td>
<td>034</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>008</td>
<td>040</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>009</td>
<td></td>
<td>023</td>
</tr>
<tr>
<td>2001</td>
<td>010</td>
<td></td>
<td>030</td>
</tr>
<tr>
<td>2001</td>
<td>011</td>
<td>034</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>012</td>
<td>040</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>013</td>
<td>023</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>014</td>
<td>030</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>015</td>
<td>025</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>072</td>
<td>027</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>073</td>
<td>023</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>074</td>
<td>022</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>075</td>
<td></td>
<td>022</td>
</tr>
<tr>
<td>2001</td>
<td>076</td>
<td></td>
<td>023</td>
</tr>
<tr>
<td>2001</td>
<td>077</td>
<td>030</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>078</td>
<td>021</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>079</td>
<td>039</td>
<td></td>
</tr>
</tbody>
</table>