A Consolidated Macro for Iterative Hot Deck Imputation

Bruce Ellis, Battelle Memorial Institute, Arlington, VA

ABSTRACT

A commonly accepted method to deal with item nonresponse is hot deck imputation, in which missing values are imputed from other records in the database that share attributes related to the incomplete variable. Sequential hot deck imputation uses related sort variables to impute nearest neighbor values from adjacent records (“nearest neighbors”), and is particularly useful for variables missing 5% of values or less. Weighted sequential hot deck imputation retains the weighted distribution of data, and should be considered with missing data rates of 10% or more. A series of SAS macros was presented at SUGI (Iannacchione, 1982, Carlson, 1995) to implement these procedures. However, they involve separate execution of programs for specialized sorting, routines to insure initial values are not missing, generally are designed to impute one variable at a time, and require a degree of subsequent programming to include imputation flags in a final output file. This poster presents software that consolidates the programming to execute all the imputation steps with a macro to iteratively execute the programming for a series of variables, each with their own set of missing values, sorting variables and imputation classes.

INTRODUCTION

Item nonresponse is a continuing problem in survey research. Discarding cases with missing values for survey items can lead to imprecise estimates to the extent that nonrespondents differ from respondents. Imputation techniques such as imputing the mean or imputing regression estimates can lead to underestimation of true item variance because they can flatten out the variation in survey responses. Hot deck imputation techniques can also use information from related survey items to impute data to nonrespondents from respondents with similar attributes. Hot deck techniques may increase, rather than deflate, variance of survey estimates, but offsets increased variance by reducing bias in estimation. Sequential or unweighted hot deck imputation uses class and sort variables to identify nearest neighbor candidates for imputation, while weighted hot deck techniques retain the weighted distribution of the survey and avoids imputing from the same respondent more than once.

This poster describes adaptations of two programs for weighted and unweighted hot deck imputation for more general and convenient use. A SAS macro for unweighted hot deck imputation was presented by Barbara Lepidus Carlson, Brenda Cox and Linda S. Bandeh at the Twentieth Annual SAS User's Group International Conference on April 2-5, 1995, in “SAS® Macros Useful in Imputing Missing Survey Data.” A SAS macro for weighted hot deck imputation was provided by Vincent G. Iannacchione in “Weighted Sequential Hot Deck Imputation Macros” at the Seventh Annual SAS® Users Group International Conference. Both programs were modified to support iterative operations to impute a series of variables, support multiple missing values or the use of a response variable to identify nonrespondents, and to automatically add imputation flags to a single output file.

PROGRAMMING AND EXECUTION

Figure 1 lists a program entitled UNWGTED.SAS for sequential, unweighted hot deck imputation, and Figure 2 presents WEIGHTED.SAS for weighted imputation. User input for both programs are placed in two blocks of macro variables at the beginning of each program and at the very end. Both programs use the same block of macro variables at the beginning to identify the folder in which input, output and format files are located; the name of the input and output file; and the name of an ID variable uniquely identifying subjects in the file. This block also contains a macro variable to identify an optional response variable, which identifies survey respondents with 1 and nonrespondents with 0. To identify nonresponses with missing codes variable by variable, “none” should be placed here.

The user input block at the end of each program is used to specify detail such as variables to be imputed, class or sorting variables to use, the missing values that identify nonresponse, and additional items specific to each program. More than one variable can be specified for imputation, class variables or sorting variables. These can be entered as lists, but must be separated by a space, not a comma. Also, more than one missing value can be listed, but these also must be separated by a space and not a comma. In both
UNWGTED.SAS and WEIGHTED.SAS, the user input block at the end of each program can be repeated with additional sets of variables to be imputed and the appropriate classes and missing values.

Multiple imputation variables can be placed in a single iteration block so that responses from the same contributing respondent can be taken for several related variables. This avoids logical incongruities that might arise if related variables were imputed independently from different respondents. This presents the possibility, however, that responses will be missing for some but not all of the related variables, resulting in an incongruity or in replacing a valid nonmissing value with an imputed one. You can choose which strategy to use in this relatively rare occurrence with the last macro variable, in which a 1 is placed to impute all of the variables or a 0 to impute only those with missing values. If only one imputation variable is identified for a single iteration block, this last macro variable is irrelevant.

With each execution, accumulating imputation results are placed in the output files, along with an imputation flag variable for each imputation variable. These variables are named by adding “_if” to the name of the imputed variable, and contain the ID of the case from which data were imputed.

PROGRAM EXAMPLES

Figure 3 lists a series of variables taken from the CDC's Behavioral Risk Factor Surveillance System (BRFSS) survey data for 2000. Core data for the State of Texas were used for program examples, using variables asking whether the respondent engaged in any exercise or activity (EXERANY) as well as age and gender to impute variables asking about specific activities (EXERACT1) and the hours spent in them (EXEROFT1). A second example imputes the number of days in the past month that a person’s physical health was not good (PHYSHLTH) from their reported general health (GENHLTH), as well as age and gender. These examples were not chosen because they are necessarily good candidates for imputation, but because they illustrate issues specific to the software and how it works.

Figures 4 through 6 show the user input blocks used for these examples.

In both programs, the purpose of class variables is to impute from the most likely group of similar respondents and in each program, imputation cannot occur across class lines. When skip patterns are in effect for variables to be imputed, it is therefore important that the ‘gateway’ variables are included among the class variables to prevent inappropriate imputations. A response of ‘no’ to EXERANY, for example, logically disallows a response about specific activities (EXERACT1) or the time spent in them (EXEROFT1). However, both programs require that at least one respondent (with valid response) must be present in each unique combination of the class variables. In addition to imposing a practical limit on the level of detail possible in the class variables, the programs will not run properly if the ‘N/A’ option for the gateway variable is coded with a SAS missing value. All cases in which a SAS missing value was present for EXERANY because the subject answered ‘NO’ (code 2) for EXERANY were changed to a value not itself eligible for imputation, in this case, 9999.

UNWGTED.SAS uses sort variables to determine nearest neighbors for imputation, while sort variables are not used by WEIGHTED.SAS. Age and gender were used as class variables to guide imputation in the WEIGHTED.SAS example. However, without some sort of recoding the requirement for at least one respondent in each class group would necessarily apply to each single year of age. Therefore in the example for WEIGHTED.SAS, age was recoded (AGEGRP) into six categories.

While the SAS missing value causes problems when it is used to identify skip patterns, it can be included as a missing value subject to imputation. The first example is imputing two refusals for EXERACT1 (code 99), and three cases (one refusal coded 999 and two with the SAS missing value) for EXEROFT1. Although these variables are related, the last macro variable requests that only the missing values be imputed (code 0).

The second example uses general health (GENHLTH) to impute days lost due to poor health (PHYSHLTH). As can be seen in Figure 3, the class variable GENHLTH itself has 14 cases in which coded values indicate no response. The example is set up only to impute the value of 99, and the programs assume 7 and 9 as valid classes. The user should carefully consider all values occurring in the data.
PROGRAM OUTPUT

Figure 7 shows output from the first example, displaying frequencies of the flag variables. The ID numbers from two cases used to impute EXERACT1 are saved in EXERACT1_IF and the three cases used for EXEROFT1 are saved in EXEROFT1_IF.

ACKNOWLEDGEMENT

These programs are adaptations of programs developed by Barbara Lepidus Carlson and Vincent G. Iannacchione. I’d like to express my gratitude for their assistance and support.

REFERENCES


CONTACT INFORMATION

Bruce Ellis
Battelle Memorial Institute
2101 Wilson Boulevard, Suite 800
Arlington, Virginia 22201-3008
Work Phone: 703-875-2965
Fax: 703-527-5640
E-Mail ellis@battelle.org

SAS and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
APPENDIX

Figure 1

/************************************************************/
/*                       UNWGTED.SAS                        */
/* Adapted from "Unweighted Sequential Hot Deck Imputation" */
/* Developed by Barbara Lepidus Carlson and Linda S. Bandeh */
/* and documented in "SAS Macros Useful in Imputing Missing */
/* Survey Data", Carlson, Cox & Bandeh, SUGI 20, April 1995 */
/* */
/* by Bruce Ellis, April 2007, Battelle Memorial Institute, */
/* to 1) consolidate Initial Value Routine, Serpentine */
/* sorting and imputation into a single stand-alone */
/* program, */
/* 2) support iterative execution for a series of */
/* imputations, */
/* 3) automatically save imputation flag variables, */
/* 4) support multiple missing values or a variable */
/* identifying cases to be imputed, */
/* 5) automatically generate some macro variables and */
/* simplify user input, */
/* 6) iteratively merge imputed variables and imputation */
/* flags to a single output file. */
/* 7) handles series of variables correctly. */
/* */
/************************************************************/

/*********************************************************************/
/*                           USER INPUT                              */
/*********************************************************************/
plugin folder with your input file==>*/ LIBNAME LIB 'c:\path';/*also need this if there are formats==>*/ LIBNAME LIBRARY 'c:\path';/*input filename========================>*/ %LET INFILE = infile;/*output filename=======================>*/ %LET OUTFILE = outfile;/*variable uniquely identifies records==>*/ %LET ID = ID_var;/*IF YOU HAVE A VARIABLE IDENTIFYING RESPONDENTS (CODED 1) AND NON--*//*optional response var (1=R 0=NR)======>*/ %LET RESPONSE = NONE;/*********************************************************************/
/*                SKIP TO BOTTOM FOR MORE USER INPUT                 */
/*********************************************************************/
option ps=55 ls=80 formchar=' ' macrogen symbolgen mprint mlogic; /* MACROS to generate code if RESPONSE variable in use */ %MACRO RESPCDEA; %IF %UPCASE(&RESPONSE)=NONE %THEN %DO;MISSVAR ~in(&MVALUE) %END; %ELSE %DO;&RESPONSE=1 %END; %MEND RESPCDEA; %MACRO RESPCDEB; %IF %UPCASE(&RESPONSE)=NONE %THEN %DO;MISSVAR IN(&MVALUE) %END; %ELSE %DO;&RESPONSE=0 %END; %MEND RESPCDEB; data lib.&OUTFILE;set lib.&INFILE; %MACRO HOTDECK(MISSVAR,CLASS,SORTVAR,DIRECTN,MVALUE,IMPALL); /* find LSTCLASS from macro variable */ %LET LSTCLASS = &CLASS; %DO %WHILE (%INDEX(&LSTCLASS, %STR( ) ) GT 1); %LET POS = %INDEX(&LSTCLASS, %STR( ) ); %LET LSTCLASS = %SUBSTR(&LSTCLASS, &POS ); %END;
/* establish initial values */
PROC SORT DATA=LIB.&OUTFILE;
   BY &CLASS &SORTVAR;
DATA SOURCE2;SET LIB.&OUTFILE;
   BY &CLASS &SORTVAR;
RESIND=9;RETAIN RESPFLAG;
IF FIRST.&LSTCLASS THEN RESPFLAG=0;
ARRAY MISSVAR &MISSVAR;
DO OVER MISSVAR;
   IF %RESPCDEA & RESPFLAG=0 THEN DO;
      RESPIND=1;RESPFLAG=1;
   END;
END;
RUN;
PROC SORT DATA=SOURCE2;
   BY &CLASS RESPIND &SORTVAR;
/* serpentine sorting */
PROC SURVEYSELECT DATA=SOURCE2 OUT=SOURCE3 METHOD=SYS RATE=1.0 NOPRINT;
   STRATA &CLASS;
   CONTROL &SORTVAR;
/* generate macro variable with imputation flags */
%LET COUNT = 1;
%LET FLAGVAR = ;
%DO %UNTIL (&PART = %STR( ) );
   %LET PART = %SCAN(&MISSVAR, &COUNT );
   %LET COUNT = %EVAL( &COUNT + 1);
   %IF &PART NE %STR( ) %THEN %LET FLAGVAR = %STR(&FLAGVAR &PART._IF);
%END;
%IF %UPCASE(&DIRECTN)=BOTH OR %UPCASE(&DIRECTN)=PREV %THEN %LET START=1;
%ELSE %LET START=2;
%IF %UPCASE(&DIRECTN)=BOTH OR %UPCASE(&DIRECTN)=NEXT %THEN %LET XEND=2;
%ELSE %LET XEND=1;
%LOCAL J;
%DO J=&START %TO &XEND;
   PROC SORT
      %IF (%UPCASE(&DIRECTN)=BOTH AND &J=2) %THEN %DO;
         DATA=HOTDECK1
      %END;
      %ELSE %DO;
         DATA=SOURCE3 (KEEP=&ID &CLASS &MISSVAR &SORTVAR
            %IF &RESPONSE NE NONE %THEN %DO; &RESPONSE %END; &RESPONSE %END; )
      %END;
      OUT=SORTFILE;
      BY &CLASS %IF &J=2 %THEN %DO; DESCENDING %END; &SORTVAR;
   %END;
/****************************
/* The following steps will allow you to find the first or last */
/* respondent within a sorted imputation class */
/****************************/
DATA RESPIND&J (DROP=RESPFLAG XABORT);
   SET SORTFILE END=EOF;
   BY &CLASS;
   RETAIN RESPFLAG XABORT 0;
   RESPIND&J=9;
   ARRAY MISSVAR &MISSVAR;
      CALL SYMPUT('NMISSV',LEFT(TRIM(DIM(MISSVAR))));
   IF FIRST.&LSTCLASS THEN RESPFLAG=0;
   IF RESPFLAG EQ 0 THEN DO;
RESPIND&J=1;
    DO OVER MISSVAR; IF %RESPCDEB THEN RESPIND&J=9; END;
END;
IF RESPIND&J EQ 1 THEN RESPFLAG=1;
IF LAST.&LSTCLASS AND RESPFLAG EQ 0 THEN DO;
    ERROR 'ERROR: AT LEAST ONE IMPUTATION CLASS WITH NO DONOR ' &LSTCLASS=;
    XABORT+1;
END;
IF EOF & XABORT THEN ABORT;
RUN;

PROC SORT DATA=RESPIND&J;
    BY &CLASS RESPIND&J %IF &J=2 %THEN %DO; DESCENDING %END; &SORTVAR;
RUN;

DATA HOTDECK&J (DROP=PRVR&J.R1-PRVR&J.R&NMISSV DONORID);
    SET RESPIND&J;
    BY &CLASS;
    RETAIN PRVR&J.R1-PRVR&J.R&NMISSV DONORID;
    ARRAY MISSVAR &MISSVAR;
    ARRAY IMPVAR&J IMPV&J.R1-IMPV&J.R&NMISSV;
    ARRAY PREVAR&J PRVR&J.R1-PRVR&J.R&NMISSV;
    IF FIRST.&LSTCLASS THEN DO;
        DO OVER PREVAR&J; PREVAR&J=.; END;
        DONORID=.;
    END;
    NONMISS=1;
    DO OVER MISSVAR;
        IF %RESPCDEB THEN NONMISS=0;
    END;
    IF NONMISS EQ 1 THEN DO;
        DO OVER MISSVAR;
            IMPVAR&J=MISSVAR; PREVAR&J=MISSVAR;
        END;
        DONORID=&ID;
        IRFLAG=0;
        DONOR&J=.;
    END;
    ELSE IF NONMISS EQ 0 THEN DO;
        DO OVER MISSVAR;
            IMPVAR&J=PREVAR&J;
        END;
        IRFLAG=1;
        DONOR&J=DONORID;
    END;
    RUN;
%END;

/********************************************
/* The next section randomly selects either */
/* the previous or next value imputed. */
********************************************/
%IF %UPCASE(&DIRECTN)=BOTH %THEN %DO;
DATA IMPUTED; SET HOTDECK2;
    ARRAY IMPVAR1R IMPV1R1-IMPV1R&NMISSV;
    ARRAY IMPVAR2R IMPV2R1-IMPV2R&NMISSV;
    ARRAY DONOR DONOR1-DONOR&NMISSV;
    ARRAY MISSVAR &MISSVAR;
    RAN=MOD(INT(RANUNI(0)*10),2)+1;
    IF RAN EQ 1 THEN DO;
        DO OVER MISSVAR;
            IF &IMPALL EQ 1 OR (&IMPALL=0 AND %RESPCDEB ) THEN DO;
                MISSVAR=IMPVAR1R;
                DONOR=DONOR1;
            END;
        END;
    END;
    CHOSE='PREV';
END;
ELSE IF RAN EQ 2 THEN DO;
   DO OVER MISSVAR;
      MISSVAR=IMPVAR2R;
      IF &IMPALL EQ 1 OR (&IMPALL=0 AND %RESPCDEB ) THEN DO;
         MISSVAR=IMPVAR2R;
         DONOR=DONOR2;
      END;
   END;
   CHOOSE='NEXT';
END;
RUN;
%END;

/*********************************************************/
/* This section is for when only one direction is used. */
/*********************************************************/
%ELSE %IF %UPCASE(&DIRECTN)=PREV OR %UPCASE(&DIRECTN)=NEXT %THEN %DO;
   DATA IMPUTED; SET HOTDECK&XEND;
      ARRAY IMPVAR1 IMPV1R1-IMPV1R&NMISSV;
      ARRAY IMPVAR2 IMPV2R1-IMPV2R&NMISSV;
      ARRAY DONOR DONOR1-DONOR&NMISSV;
      ARRAY MISSVAR &MISSVAR;
      %IF %UPCASE(&DIRECTN)=PREV %THEN %DO;
         DO OVER MISSVAR;
            IF &IMPALL EQ 1 OR (&IMPALL=0 AND %RESPCDEB ) THEN DO;
               MISSVAR=IMPVAR1;
               DONOR=DONOR1;
            END;
         END;
      END;
      %ELSE %IF %UPCASE(&DIRECTN)=NEXT %THEN %DO;
         DO OVER MISSVAR;
            IF &IMPALL EQ 1 OR (&IMPALL=0 AND %RESPCDEB ) THEN DO;
               MISSVAR=IMPVAR2;
               DONOR=DONOR2;
            END;
         END;
      END;
      CHOOSE='NEXT';
%END;
RUN;
/* generate imputation flags and eliminate working variables */
DATA IMPUTED(KEEP=&ID CHOSE &MISSVAR &FLAGVAR);
   SET IMPUTED;
      ARRAY FLAGVAR &FLAGVAR;
      ARRAY DONOR DONOR1-DONOR&NMISSV;
      DO OVER FLAGVAR;
         FLAGVAR = DONOR;
      END;
RUN;
/* merge results back to original file */
PROC SORT DATA=LIB.&OUTFILE OUT=SOURCE1;BY &ID;
PROC SORT DATA=IMPUTED;BY &ID;
DATA LIB.&OUTFILE;
   MERGE SOURCE1(DROP=&MISSVAR) IMPUTED;BY &ID;
RUN;
PROC FREQ;TABLES &MISSVAR &FLAGVAR / MISSING;
%END;

/*********************************************************/
/* The next section identifies donors used more than three times. */
/*********************************************************/
PROC FREQ DATA=LIB.&OUTFILE;
%LET DONOR = %SCAN(&FLAGVAR, 1 );
WHERE ( %SCAN(&FLAGVAR, 1 ) > .);
TABLES &DONOR/NOPRINT OUT=CHECKCNT;

PROC FREQ DATA=CHECKCNT;
TABLES COUNT;
TITLE 'NUMBER OF TIMES DONORS WERE USED';RUN;

DATA CHECKCNT;SET CHECKCNT;
DONOR = %SCAN(&FLAGVAR, 1 );

DATA TEMPFILE; SET LIB.&OUTFILE (KEEP=&ID &CLASS &SORTVAR &MISSVAR);
DONOR=&ID;

PROC SORT DATA=TEMPFILE; BY DONOR;
DATA MORTHAN3; MERGE CHECKCNT (IN=A) TEMPFILE; BY DONOR; IF A;
IF DONOR NE . AND COUNT > 3;

PROC PRINT DATA=MORTHAN3;
VAR DONOR COUNT &MISSVAR &CLASS &SORTVAR;
TITLE 'DONORS USED MORE THAN 3 TIMES';RUN;
%MEND HOTDECK;

/******************************************************
/* INSERT APPROPRIATE VALUES FOR ITEMS 1 THROUGH 6 BELOW FOR EACH SET*/
/* OF VARIABLES TO BE IMPUTED. COPY AND INSERT VALUES AS APPROPRIATE */
/* FOR EACH ADDITIONAL VARIABLE OR SET OF VARIABLES TO BE IMPUTED. */
/* */
/* FOR ITEMS 1 THRU 3, INSERT ONE VARIABLE FOLLOWED BY A COMMA, OR A */
/* LIST OF VARIABLES SEPARATED BY A SPACE, WITH A COMMA AFTER THE */
/* LAST VARIABLE. */
/* FOR ITEM 4 PLACE 'PREV', 'NEXT' OR 'BOTH', FOLLOWED BY A COMMA, */
/* TO CHOOSE THE METHOD TO SELECT THE DONOR. */
/* FOR ITEM 5 INSERT A NUMERIC VALUE OR A LIST OF NUMERIC VALUES */
/* (SEPARATED BY A SPACE) AND FOLLOWED BY A COMMA. A PERIOD CAN BE */
/* FOR THE SAS MISSING VALUE. */
/* FOR ITEM 6, INSERT 1 TO IMPUTE ALL VARIABLES IN THE ITEM 1 LIST */
/* WHETHER ALL ARE MISSING OR NOT. DO NOT FOLLOW ITEM 6 WITH A COMMA*/
/****************************************/

%HOTDECK(
/*1)place variable(s) to be imputed here===>/ impute1 impute2 ,
/*2)CLASSING or POST-STRATA variable(s)===>*/ classvar1 classvar2 ,
/*3)place sorting variable(s) here=========>*/ sortvar1 sortvar2 ,
/*4)select method: PREV, NEXT, or BOTH===>*/ prev,
/*5)list value(s) to treat as missing here==>*/ . 97 98 99 ,
/*6)put 1 to impute all, 0 just the missing>*/ 0
);
Figure 2

Weathered SAS

Adapted from "Weighted Sequential Hot Deck Imputation Macros" developed by Vincent G. Iannacchione, Research Triangle Institute, & documented in 'Weighted Sequential Hot Deck Imputation Macros', SUGI 7, February 1982.

by Bruce Ellis, April 2007, Battelle Memorial Institute,

to 1) support iterative execution for a series of imputations,
2) automatically save imputation flag variables,
3) support multiple missing values or a variable identifying cases to be imputed,
4) automatically generate some macro variables and simplify user input,
5) iteratively merge imputed variables and imputation flags to a single output file.
6) handles series of variables correctly.

User Input

LIBNAME LIB 'c:\path';
LIBRARY 'c:\path';
%LET INFILE = infile ;
%LET OUTFILE = outfile;
%LET _ID = ID_var ;
%LET RESPONSE = NONE;

DATA LIB.&OUTFILE;SET LIB.&INFILE;
%MACRO _HOTDECK(_VARS,_STRATA,_WEIGHT,_SEED,MVALUE,IMPALL);

PROC SORT DATA=LIB.&OUTFILE OUT=SOURCE;BY &_STRATA;
/* find _LASTRAT from macro variable */
%LET _LASTRAT = &_STRATA;
%DO %WHILE (%INDEX(&_LASTRAT, %STR( ) ) GT 1);
%LET POS = %INDEX(&_LASTRAT, %STR( ) );
%LET _LASTRAT = %SUBSTR(&_LASTRAT, &POS );
%END;

/* generate macro variable with imputation flags and _IVARn */
%LET COUNT = 1;
%LET FLAGVAR = ;
%DO %UNTIL (&PART = %STR( ) );
%LET PART = %SCAN(_VARS, &COUNT );
%IF &PART NE %STR( ) %THEN %DO;
%LET FLAGVAR = %STR(&FLAGVAR &PART._IF);
%LET _IVAR = %STR(_IVAR&COUNT) ;
%END;
%LET COUNT = %EVAL( &COUNT + 1);
%END;
**STEP 1**************************************************************************
/* a) Split INFILFILE into respondent & nonrespondent data sets. */
/* b) Compute -RATIO of respondents, to nonrespondents' sum of */
/* weights for each poststratum. */
/* c) Check for nonpositive weights & respondent only poststrata*/
/* d) Abort if there are no respondents-in a poststratum. */
**************************************************************************/

DATA _RESP1 (KEEP = &_STRATA &_WEIGHT &_ID &_VARS _SUMSI)
  _NRESP1 (KEEP = &_STRATA &_WEIGHT &_ID &_VARS)
  _STRATIO(KEEP = &_STRATA _RATIO)
  _ALLRESP(KEEP = &_STRATA _SUMSI);
SET SOURCE END = EOF;  BY &_STRATA;
LENGTH _EQWGT 3;
RETAIN _EQWGT 1 _NORESP 0;
LABEL  _EQWGT = 'Default weight variable';
ARRAY _ST &_STRATA;
/* Initialize at the start of each poststratum. */
IF FIRST.&_LASTRAT THEN DO;
  _SUMSI = 0; _SUMWJ = 0;
END;
/* Identify respondents or missing/non-missing based on macro */
%IF %UPCASE(&RESPONSE)=NONE %THEN %DO;
  ARRAY INPUTED &_VARS;
  RESPOND=0;
  DO OVER INPUTED;
    IF INPUTED IN(&MVALUE) THEN RESPOND=1;
  END;
%END;
%ELSE %DO;
  RESPOND=1 - &RESPONSE; /*Note:reversed fro rig prg*/
%END;
/* Exclude nonpositive weights. */
IF &_WEIGHT > 0
/* Split respondents and nonrespondents. */
THEN IF RESPOND=0 THEN DO;
  _SUMSI + &_WEIGHT;
  OUTPUT _RESP1;
END;
ELSE DO;
  _SUMWJ + &_WEIGHT;
  OUTPUT _NRESP1;
END;
/* Count the number of nonpositive weights. */
ELSE _NONPOS+1;
/* At the end of each poststratum: */
/* If there is at least one respondent with a positive weight: */
IF LAST.&_LASTRAT THEN IF _SUMSI > 0
/* Output poststrata with both respondents & nonrespondents */
THEN IF _SUMWJ > 0 THEN DO;
  _RATIO = _SUMSI / _SUMWJ;
  OUTPUT _STRATIO;
END;
/* Output poststrata with only respondents */
ELSE DO;
  _SUMSI = 0;
  OUTPUT _ALLRESP;
END;
/* If there are no respondents with positive weights: */
/* Print that poststratum and set Abort flag */
ELSE DO;
PUT // 'ERROR: THERE ARE NO RESPONDENTS IN THE FOLLOWING POSTRATUM:';
DO OVER _ST;
   PUT +10 _ST= ;
END;
NORESP = 1;
END;
/* At the end of the input data set: */
/* Print warning if there are any nonpositive weights. */
IF EOF THEN DO;
   IF _NONPOS > 0 THEN
      PUT // 'WARNING: THERE ARE ' _NONPOS 'OBSERVATIONS WITH NONPOSITIVE WEIGHTS ON THE INPUT ' 'DATA SET.' / 'THEY WILL NOT BE INCLUDED ON ' 'THE OUTPUT DATA SET(s),' +2;
   /* Abort if a poststratum has no respondents. */
   IF NORESP THEN ABORT;
END;
PROC PRINT DATA = _ALLRESP;
VAR &_STRATA;
TITLE 'Imputation Poststrata That Contain Respondents Only';
RUN;
TITLE;
/** STEP 2 ***************************************************/
/* a) Multiply each nonrespondent's weight by _RATIO. */
/* b) Lag each nonrespondent's scaled weight. */
/* c) Add an extra observation to each poststratum so that */
/* the first zone length is known. */
/*************************************************************/
DATA _NRESP2 (KEEP = &_STRATA _SUMSI _VJ) ;
MERGE _NRESP1 _STRATIO ;
BY &_STRATA;
RETAIN _VJ ;
/* Initialize at the start of each poststratum. */
IF FIRST.&_LASTRAT THEN DO;
   _SUMSI = 0; _VJ = 0;
END;
/* Name the sum _SUMSI for the interleave in Step 3. */
 _SUMSI+_VJ;
/* Scale each nonrespondent's weight. */
 _VJ = _RATIO* &_WEIGHT;
OUTPUT _NRESP2;
/* Create an extra observation for each poststratum. */
IF LAST.&_LASTRAT THEN DO;
   _SUMSI + _VJ;
   _VJ = 1;
OUTPUT _NRESP2;
END;
RUN;
/** STEP 3 ***********************************************************/
/* a) Interleave respondents & nonrespondents to form selection zones*/
/* b) Use weighted sequential sample selection to select a respondent*/
/* from each zone. */
/* c) Put the selected respondent's data onto an output data set of */
/* nonrespondents. */
/* d) Put each respondent's expected & actual number of selections */
/* onto an output data set of respondents. */
*********************************************************************/
DATA _NRDSET (LABEL = Output Data Set of Nonrespondents)
  KEEP = &_STRATA &_ID &_WEIGHT &_VARS _IMPID _IVAR1-&_IVAR)
_RDSET (LABEL = Output Data Set of Respondents)
  KEEP = &_STRATA &_ID &_WEIGHT &_VARS _EXPN _N);

SET _ALLRESP(IN = _INALLR)
   _NRESP2 (IN = _INNR)
   _RESP1 (IN = _INR);
BY &_STRATA _SUMSI;

ARRAY _IVAR _IVAR1-&_IVAR;
ARRAY _VAR &_VARS;
RETAIN _LAGI _LAGF _I _SUMVJ _VIPLUS1 _ALLR;

/* At the start of each postratum, check for all respondents. */
IF FIRST.&_LASTR THEN IF _INALLR THEN DO;
   _ALLR = 1;
   RETURN;
END;
/* Otherwise, reinitialize sample selection variables. */
ELSE DO;
   _ALLR = 0; _I = -1; _LAGI = 0; _LAGF = 0; _LAGSUMN = 0;
END;

/* Set the expected & actual number of selections to zero
   for each all respondent poststratum. */
IF _ALLR THEN DO;
   _N = 0;
   _EXPN = 0;
OUTPUT _RDSET;
RETURN;
END;
/* Increment _I, retain sum of weights and next zone length
   for each nonrespondent. */
IF _INNR THEN DO;
   _I + 1;
   _SUMVJ = _SUMSI;
   _VIPLUS1 = _VJ;
END;
/* Execute The Weighted Sequential Sample Selection Procedure
   for each respondent. */
IF _INR;
/* Compute _F using the most recent nonrepondent's
   sum of weights and next zone length. */
   _F = ROUND((SUMSI - _SUMVJ) / _VIPLUS1, 1.0E-12);
/* Determine _P by checking the previous respondent's selection. */
IF _LAGSUMN = _LAGI THEN IF 0 <= _LAGF < _F THEN _P = (_F - _LAGF) / (1 - _LAGF);
ELSE _P = 0;
ELSE IF 0 <= _LAGF < _F THEN _P = 1;
ELSE IF 0 < _F <= _LAGF THEN _P = _F / _LAGF;
ELSE _P = 0;
/* Decide if the first i respondents are used I or I+1 times. */
IF _P > UNIFORM(&_SEED) THEN _SUMN = _I + 1;
ELSE _SUMN = _I;
/* Output each respondent's expected & actual number of selections. */
N = _SUMN - _LAGSUMN;
_EXPN = (_I + _F) - (_LAGI + _LAGF);
OUTPUT _RDSET;
/* Assign the respondent's ID & data to the imputation variables. */
  _IMPID = &_ID;
DO OVER _VAR;
  _IVAR = _VAR;
END;

/* Output the imputation variables to N nonrespondents. */
DO WHILE (_LAGSUMN < _SUMN);
  SET _NRESP1;
  OUTPUT _NRDSET;
  _LAGSUMN + 1;
END;

/* Retain for next respondent's sample selection. */
  _LAGI = _I;
  _LAGF = _F;
LABEL _N = Actual number of selections
  _EXPN = Expected number of selections
  _IMPID = ID of selected respondent;
RUN;

DATA _N RDSET;SET _N RDSET;
  ARRAY _IVAR _IVAR1-&_IVAR;
  ARRAY _VAR &_VARS;
  ARRAY FLAGS &FLAGVAR;
DO OVER _IVAR;
  %IF %UPCASE(&RESPONSE) = NONE %THEN %DO;
    IF &IMPALL EQ 1 OR _VAR IN(&MVALUE) THEN DO;
      _VAR = _IVAR; FLAGS = _IMPID;
    END;
  %END;
  %ELSE %DO;
    _VAR = _IVAR; FLAGS = _IMPID;
  %END;
END;

/* Combine imputation results with output file */
PROC SORT DATA=_N RDSET;BY &_ID;
PROC SORT DATA=_RDSET;BY &_ID;
PROC SORT DATA=SOURCE;BY &_ID;
DATA LIB.&OUTFILE;MERGE
  SOURCE(DROP= &_VARS )
  _N RDSET(KEEP= &_ID &_VARS &FLAGVAR )
  _RDSET(KEEP= &_ID &_VARS _EXPN _N);
BY &_ID;RUN;
/* Delete all temporary data sets. */
PROC DELETE DATA = _RESP1 _NRESP1 _NRESP2 _STRATIO _ALLRESP
  _N RDSET _RDSET;RUN;
PROC FREQ DATA=LIB.&OUTFILE;
  TABLES &VARS &FLAGVAR / MISSING;RUN;

/* End of the macro HOTDECK. */
%mend _hotdeck;
Figure 3: Imputed, Class, Sorting, and Weight Variables Used in Two Examples

<table>
<thead>
<tr>
<th>Variable Name and Associated BRFSS Survey Question</th>
<th>Coding and Distribution for Texas, 2000, BRFSS Data Code-Value (Count)</th>
<th>How Used in:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNWGT</td>
<td>WGTD</td>
</tr>
<tr>
<td><strong>Variables for Example 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXERANY – “During the past month did you participate in any physical activities or exercises such as running…, or walking for exercise?”</td>
<td>1-Yes (3618), 2-No (1398), None Missing</td>
<td>Class Variable</td>
</tr>
<tr>
<td>EXERACT1 – “What type of physical activity or exercise did you spend the most time doing during the past month?”</td>
<td>1-56 for specific activities, with 1795 responses for walking, 20 activities with less than 10 responses and 9 with 1 respondent. 9999-N/A (1398), 99-Refused (2)</td>
<td>Impute Refused</td>
</tr>
<tr>
<td>EXEROFT1 – “How many times per week or per month did you take part in this activity, during the past month?”</td>
<td>101-199 times per week (2640), 201-299 times per month (931), 777-Don’t Know (44), 9999-N/A (1398), 999-Refused (1), SAS Missing-Not answered (2)</td>
<td>Impute Refused &amp;Missing</td>
</tr>
<tr>
<td><strong>Variables for Example 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENHLTH – “Would you say that your general health is:”</td>
<td>1-Excellent (949), 2-Very Good (1460), 3-Good (1649), 4-Fair (717), 5-Poor (227), 7-Don’t Know (13), 9-Refused (1)</td>
<td>Class Variable</td>
</tr>
<tr>
<td>PHYSHLTH – “For how many days during the past 30 days was your physical health not good?”</td>
<td>01-30 Days (1759), 77-Don’t Know (110), 88-None (3145), 99-Refused (2)</td>
<td>Impute Refused</td>
</tr>
<tr>
<td><strong>Used in Both Examples</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_IMPAGE – Single year of age, BRFSS imputed</td>
<td>18-95 years, No Missing</td>
<td>Sort Var</td>
</tr>
<tr>
<td>AGEGRP – Six 10-year or greater categories</td>
<td>_IMPAGE Recode, &lt;30, -39, -49, -59, -69, 70+, No Missing</td>
<td>-</td>
</tr>
<tr>
<td>SEX – Gender, coded numerically</td>
<td>1-Male, 2-Female, No Missing</td>
<td>Sort Var</td>
</tr>
<tr>
<td>FINAL_WT – Final Survey Weight</td>
<td>415.9 – 36582.5, No Missing</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 4: First Block of User Input for both WEIGHTED.sas and UNWEIGHTED.sas

/*******************************************************************************/
/*                           USER INPUT                              */
/*******************************************************************************/
/*plug in folder with your input file==>*/  LIBNAME LIB 'c:\brfss2000';
/*also need this if there are formats==>*/  LIBNAME LIBRARY 'c:\brfss2000';
/*input filename============================================================*/  %LET INFILE = TXbrfs00;
/*output filename============================================================*/  %LET OUTFILE = OUT_UW;
/*variable uniquely identifies records==>*/  %LET ID = ID_var;
/*IF YOU HAVE A VARIABLE IDENTIFYING RESPONDENTS (CODED 1) AND NON-* /
/*RESPONDENTS (CODED 0) REPLACE 'NONE' WITH THE VARIABLE NAME BELOW*/
/*optional response var (1=R 0=NR)==>*/  %LET RESPONSE = NONE;
/*******************************************************************************/
/*                SKIP TO BOTTOM FOR MORE USER INPUT                 */
/*******************************************************************************/

Figure 5. Second Block of User Input for UNWEIGHTED.sas

%HOTDECK(
/*1)place variable(s) to be imputed here==>*/  exeract1 exeroft1,
/*2)CLASSING or POST-STRATA variable(s)===>*/  exerany,
/*3)place sorting variable(s) here===>*/  sex _impage,
/*4)select method: PREV, NEXT, or BOTH==>*/  next,
/*5)list value(s) to treat as missing here==>*/  . 99 999,
/*6)put 1 to impute all, 0 just the missing==>*/  0
);
%HOTDECK(
/*1)place variable(s) to be imputed here==>*/  physhlth,
/*2)CLASSING or POST-STRATA variable(s)===>*/  genhlth,
/*3)place sorting variable(s) here===>*/  sex _impage,
/*4)select method: PREV, NEXT, or BOTH==>*/  next,
/*5)list value(s) to treat as missing here==>*/  99,
/*6)put 1 to impute all, 0 just the missing==>*/  0
);

Figure 6: Second Block of User Input for WEIGHTED.sas

%HOTDECK(
/*1)place variable(s) to be imputed here==>*/  exeract1 exeroft1,
/*2)CLASSING or POST-STRATA variable(s)===>*/  exerany sex agegrp ,
/*3)Numeric WEIGHT variable ==============>*/  _finalwt ,
/*4)Numeric SEED value ==================>*/  2 ,
/*5)list value(s) to treat as missing here==>*/  99 ,
/*6)put 1 to impute all, 0 just the missing==>*/  0
);

Figure 7: Example Output – Imputation Flag Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>exeract1_IF</td>
<td>5014</td>
<td>99.96</td>
<td>5014</td>
<td>99.96</td>
</tr>
<tr>
<td>861</td>
<td>1</td>
<td>0.02</td>
<td>5015</td>
<td>99.98</td>
</tr>
<tr>
<td>4994</td>
<td>1</td>
<td>0.02</td>
<td>5016</td>
<td>100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>exeroft1_IF</td>
<td>5013</td>
<td>99.94</td>
<td>5013</td>
<td>99.94</td>
</tr>
<tr>
<td>861</td>
<td>1</td>
<td>0.02</td>
<td>5014</td>
<td>99.96</td>
</tr>
<tr>
<td>4422</td>
<td>1</td>
<td>0.02</td>
<td>5015</td>
<td>99.98</td>
</tr>
<tr>
<td>4994</td>
<td>1</td>
<td>0.02</td>
<td>5016</td>
<td>100.00</td>
</tr>
</tbody>
</table>