%Addval: A SAS® Macro Which Completes the Cartesian Product of Dataset Observations for All Values of a Selected Set of Variables

Rich Schiefelbein, PRA International, Lenexa, KS

ABSTRACT

It is often useful in clinical programming to create additional observations to add to an existing dataset as "placeholders." These additional observations then complete the "Cartesian product" of the combinations of all values for a selected set of variables.

For example, the output dataset from the PROC FREQ in a SAS® analysis of count data does not include observations for combinations of variables that are not present in the input data, but may still be of interest (i.e. the result dataset does not include an observation with a count value of "0" for "Caucasian/Females" if that particular combination is not present in the input data).

This macro is a very flexible tool which allows one to quickly and easily create additional "combination" observations in a dataset. Not only will it create observations for all combinations of values for a selected group of existing variables, but entirely new variables, with predetermined values as needed, may also be created in the output dataset.

The final resulting dataset will then include the Cartesian product of all values for the set of variables specified as well as any additional placeholders needed.

INTRODUCTION

There are many instances in clinical trial programming where it is useful to form the cartestian product of the values of a set of variables in a dataset. One instance is in the "filling out" of an output dataset from PROC FREQ in SAS®.

The macro described in this paper provides a quick and flexible way to produce the desired result; a dataset containing observations for all combinations of all values of the specified variables. Additional values not present in the input dataset can be user specified prior to forming the Cartesian product.

MACRO INPUT DATA

The input to this macro can be any dataset containing more than one "key" variable (the Cartesian product of all values for the key variables will be formed as the output data).

For example, someone investigating any relationship between the gender of a subject, and the day of week or the month of the year of their birthday may produce an output dataset from PROC FREQ as:

Data=Temp.dummy1

Gender	Day	Month	Count
Male	Monday	January	4
Male	Monday	February	2
Male	Tuesday	March	1
Male	Tuesday	May	6
Male	Tuesday	June	4
Male	Wednesday	July	3
Male	Wednesday	July	5
Male	Thursday	July	1
Male	Thursday	September	3
Male	Saturday	September	2
Male	Saturday	October	2
Male	Saturday	November	3
Male	Saturday	November	5
Male	Sunday	November	4
Female	Monday	March	8
Female	Tuesday	March	2
Female	Tuesday	May	1
Female	Friday	July	3
Female	Friday	August	7
Female	Sunday	August	6
Female	Sunday	December	5

In this output from Proc Freq, many different combinations of the variables Sex, Day, and Month are not present (for example, the particular combination of Male/Monday/March is not present because no subject in the sample had that particular combination of values).

In a summary of this data, it would likely be of interest to explicitly show that for combinations not present in the database, no subjects met the criteria (i.e. show explicitly that zero subjects met the criteria of Male/Monday/March).

It may also be of interest to explicitly show that other values (not present in the input database AT ALL) for the "key" variables of Gender, Day, and Month had no subjects present. For example, the month of April is not present at all in the input dataset, and we may want to explicitly illustrate this in a summary.

Though in this case we would likely want to assign a value of zero to the variable "count" for any newly created observations, any numeric value can be specified using the %addval macro.

MACRO PARAMETERS

INDATA=	Specifies the name of the macro input dataset
OUTDATA=	Specifies the name of the macro output dataset
BYVAR=	-Specifies the "key" variables in the input dataset -String of variable names each separated by a space
SETVAR=	-Specifies the "non-key" variables to be assigned any user-specified numeric value (only newly created observations will be assigned this value) -String of variable names each separated by a space
SETVAL=	Specifies the numeric value to be assigned to each variable specified in the "SETVAR" variable (only newly created observations)
ADDVAL'X'=	-Variables which contains any values not present in the input dataset which are to be added to the dataset prior to forming the Cartesian product. -Addval1 should contain the values to be added to the first variable in the "BYVAR" string, Addval2 should contain the values to be added to the second variable in the "BYVAR" string, etc. -String of values each separated by a space

Note: Only the variables "INDATA", "OUTDATA", and "BYVAR" are required. All other variables are optional.

SAMPLE MACRO CALL #1

A basic call to complete the Cartesian product of the example dataset "temp.dummy1" (above) would appear as:

```
%addval (indata=dummy1, outdata=dummy2,
         byvar=Gender Day Month,
         setvar=count, setval=0);
```

This call will produce a dataset called "dummy2" which will contain all of the original observations in "dummy1" plus additional observations representing all possible combinations of the values of the "BYVAR" variables Gender, Day, and Month.

The "SETVAR" and "SETVAL" variables specify that for any newly created observations, the value of the variable "COUNT" should be set to a value of zero.

The dataset "dummy2" will contain a total of 154 observations (2 levels of Gender X 7 levels of Day X 11 levels of Month). Note that since the month of April does not appear at all in the input dataset, it does not appear at all in the output dataset.

SAMPLE MACRO CALL #2

In order to create observations containing values of any of the variables specified in the "BYVAR" string which are not present at all in the input dataset (such as the month of April), additional variables can be specified in the macro call.

```
%addval (indata=dummy1, outdata=dummy2,
         byvar=Gender Day Month,
         setvar=count, setval=0,
         addval3=April);
```

This call will produce a dataset called "dummy2" which will contain all of the original observations in "dummy1" plus additional observations representing all possible combinations of the values of the "BYVAR" variables Gender, Day, and Month.

Prior to forming the Cartesian product of the "BYVAR" values, however, any values specified in an "addval" variable are added to the dataset. Thus, the value of "April" is added to the variable "Month" (since addval3 is specified and "Month" is the third variable in the "BYVAR" string), and the resulting dataset will contain a total of 168 observations (2 levels of Gender X 7 levels of Day X 12 levels of Month).

Again, the "SETVAR" and "SETVAL" variables specify that for any newly created observations, the value of the variable "COUNT" should be set to a value of zero.

CONCLUSION

The macro presented in this paper is an effective tool in allowing clinical data programmers work more efficiently. The type of situation illustrated is a common one in preparing summary tables in the pharmaceutical industry. The macro can be used in any situation where it is useful to add additional observations to an existing dataset from combinations of existing dataset variable values and user specified values.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Rich Schiefelbein
PRA International
16400 College Blvd.
Lenexa, KS 66219
Work Phone: 913-577-2782
Fax: 913-599-0344
Email: Schiefelbeinrich@praintl.com
Web:

TRADEMARK INFORMATION

APPENDIX (SOURCE CODE)

```
%macro addval(indata=,outdata=,byvar=,setvar=,setval=,
          addval1=, addval2=, addval3=, addval4=, addval5=,
          addval6=, addval7=, addval8=, addval9=, addval10=,
          addval11=,addval12=,addval13=,addval14=,addval15=,
          addval16=,addval17=,addval18=,addval19=,addval20=);

%let miss=;

/* Create a macro variable containing the number of variables
present in the 'byvar' string */

data info1;
length bystring $200;
bystring="&byvar";
run;

/* value of 'x' in the resulting dataset 'info2' contains the number
of words in string */

data info2(keep=x bystring);
length y $200;
set info1;
x=1;
  do until(y=' ');
    y=scan(bystring,x,' ');
    x=x+1;
  end;
x=x-2;
run;

data _null_;
set info2;
call symput('numbyvar',x);
run;

/* Define macro variable references for each byvar  */
/* One macro variable string is defined for each variable in the
   'byvar' string  */

data bydata(keep=var2 var4);
length var1 var3 var4 $200;
do i=1 to &numbyvar;
  var1='Byvar';
  var2=i;
  var3=put(var2,3.0);
  var4=left(trim(var1))||left(trim(var3));
  output;
end;
run;

data _null_;
set bydata;
call symput(var4,' ');     /* output null values initially */
run;

data _null_;
length byvarc byvarc1 newvar $200;
set bydata;
byvarc="&byvar";
byvarc1=left(trim(byvarc));
newvar=scan(byvarc1,var2,' ');
call symput(var4,newvar);   /* output macro strings for byvars */
run;

/*************************************************************************/
/*   The following section of code will define the text                  */
/*   strings for the variables to be 'added' to the byvars               */
/*                                                                       */
/*  The form of the string is an output statement.  For example:         */
/*                                                                       */
/*         Day='Tuesday'; output; Month='April'; output;                 */
/*                                                                       */
/*    These strings are used to create a dataset containing the          */
/*    values for all byvars which need to be added to the original       */
/*    dataset.  This dataset is  then set into the original dataset      */
/*    prior to forming the 'Cartesian product'.                          */
/*************************************************************************/

data passadd1(keep=var3 var5 var6);
length var1 var2 var4 var5 var6 $200;
  do i=1 to &numbyvar;
    var1='Addval';
    var2='Byvar';
    var3=i;
    var4=put(var3,3.0);
    var5=left(trim(var1))||left(trim(var4));
    var6=left(trim(var2))||left(trim(var4));
    output;
  end;
run;

data passadd2;
length newvar1 newvar2 $200;
set passadd1;
newvar1=symget(var5);
newvar2=symget(var6);
run;

/* value of 'x' in the resulting dataset 'passadd3' contains the
number of words in string */

data passadd3;
length y $200;
set passadd2;
x=1;
  do until(y=' ');
    y=scan(newvar1,x,' ');
    x=x+1;
  end;
x=x-2;
run;
```

```
proc printto name=work.temp.addvalx.output new; run;
data _null_;
length wordval string1 $200;
set passadd3;
file print notitle;
  wordval=' ';
  string1=' ';
    do i=1 to x;
      wordval=scan(newvar1,i,' ');
      put newvar2"='"wordval"'; output;";    /* output text strings to
text file named 'addvalx' */
    end;
run;
proc printto; run;


/* Add the user specified values to the byvars ('key' variables) */

data addval(keep=&byvar);
set &indata;
if _n_=1;   /* use first obs to avoid creating missing values */
run;

filename addvalx1 catalog 'work.temp.addvalx.output';

data addvaln;
set addval end=eof;
if eof then
  %include addvalx1;
run;

data newdata;
set &indata addvaln;   /* set new obs with the original dataset */
run;

/* Build variable text strings to be used as sql statement  */
/*    -SQL statement used to create the Cartesian product for all
      'key' variables  */
/*    -One string is created (and 'put' to a text file).  This string
       contains the entire SQL statement  */

proc printto name=work.temp.sqlx.output new; run;
data _null_;
length newvar newvar1 save1 $200;

file print notitle;

newvar="&byvar";
newvar1=left(trim(newvar));
save1=newvar1;

put 'proc sql';
put %str(';');
put 'create table all as  select distinct';
mark=2;
w=0;
do until(mark = 0);
  w=w+1;

/* created dummy dataset name references for each byvar */

  if w=1  then letter = 'a';     if w=11  then letter = 'k';
  if w=2  then letter = 'b';     if w=12  then letter = 'l';
  if w=3  then letter = 'c';     if w=13  then letter = 'm';
  if w=4  then letter = 'd';     if w=14  then letter = 'n';
  if w=5  then letter = 'e';     if w=15  then letter = 'o';
  if w=6  then letter = 'f';     if w=16  then letter = 'p';
  if w=7  then letter = 'g';     if w=17  then letter = 'q';
  if w=8  then letter = 'h';     if w=18  then letter = 'r';
  if w=9  then letter = 'i';     if w=19  then letter = 's';
  if w=10 then letter = 'j';     if w=20  then letter = 't';

mark=index(left(trim(newvar1)),' ');
  if mark > 1 then do;
    subvar1=substr(newvar1,1,mark-1);
    put letter'.'subvar1',';
  end;                              /* define the 'select' SQL statement */
  if mark = 0 then do;
     put letter'.'newvar1;
  end;
  if mark > 1 then do;
     newvar1=substr(newvar1,mark+1);
  end;
end;
mark=2;
w=0;
put 'from';
newvar1=save1;

do until(mark = 0);
  w=w+1;

/* created dummy dataset name references for each byvar */

  if w=1  then letter = 'a';     if w=11  then letter = 'k';
  if w=2  then letter = 'b';     if w=12  then letter = 'l';
  if w=3  then letter = 'c';     if w=13  then letter = 'm';
  if w=4  then letter = 'd';     if w=14  then letter = 'n';
  if w=5  then letter = 'e';     if w=15  then letter = 'o';
  if w=6  then letter = 'f';     if w=16  then letter = 'p';
  if w=7  then letter = 'g';     if w=17  then letter = 'q';
  if w=8  then letter = 'h';     if w=18  then letter = 'r';
  if w=9  then letter = 'i';     if w=19  then letter = 's';
  if w=10 then letter = 'j';     if w=20  then letter = 't';

  mark=index(left(trim(newvar1)),' ');
  if mark > 1 then do;
     subvar2=substr(left(trim(newvar1)),1,mark-1);
      put 'newdata(keep='subvar2') as 'letter',';
  end;
  if mark = 0 then do;           /* define the 'from' SQL statement */
     put 'newdata(keep='newvar1') as 'letter;
  end;
  if mark > 1 then do;
     newvar1=substr(newvar1,mark+1);
  end;
end;

put ';quit;';
```

```
/*  Build strings for the 'set value' statements (stored in macro
variable 'setvar')  */
/*    -All variables specified in the 'setvar' string will be  set to the
       user specified numeric value ('setval')  */
/*    -This is only done for 'newly created' observations  */

%if &setvar ne &miss %then %do;
  length build1 $200;
  newvarz="&setvar";
  newvar1z=left(trim(newvarz));
  build1='';
  mark=2;
  do until(mark = 0);
    mark=index(left(trim(newvar1z)),' ');
    if mark > 1 then do;
      build1=left(trim(build1))||substr(left(trim(newvar1z)),1,mark-
1)||"=&setval;";
    end;
    if mark = 0 then do;
      build1=left(trim(build1))||trim(left(newvar1z))||"=&setval;";
    end;
    if mark > 1 then do;
      newvar1z=substr(newvar1z,mark+1);
    end;
  end;

  call symput('setvar',build1);

%end;

run;
proc printto; run;

/*  Form the Cartesian product (for 'key' variables) of the final
     dataset.    */
/*  The 'final' dataset contains all observation from the original
     dataset AND all 'newly created' observations. */

filename cartprod catalog 'work.temp.sqlx.output';

%include cartprod;   /*  -This is the complete sql statement used
                           to form the Cartesian product */
                     /*  - The dataset produced is named 'all'   */

/* Set all user specified 'non-key' variables to the user specified
numeric value */

data &outdata;
merge &indata(in=a) all(in=b);
by &byvar;

if not a then do;
  %if &setvar ne &miss %then %do;
    &setvar
  %end;
end;
run;

%mend addval;
```