Using a SAS® Data Set to Expand the Program Data Vector
H. Ian Whitlock, Westat Inc.

Abstract

Temporary arrays are handy for storing information during a DATA step, but the number of variables is limited. Formats can store more information, but they must be created before the DATA step and they are not modifiable during the DATA step. SAS data sets can store still more information. In the past this information had to be created prior to the DATA step and was not modifiable during the DATA step. The MODIFY and REPLACE statements have changed this situation to provide a simple mechanism for storing large amounts of information which must be available and modifiable during the DATA step.

This principle is illustrated using code to solve the following problem. The records of two SAS data sets are to be matched, where match means that the corresponding key variable values are near each other. For each record in the first file, draw a random sample without replacement of SAMPSIZE such matches.

Introduction

Since I started using SAS, I have always been fascinated with ways to extend the Program Data Vector (PDV). Most standard SAS programming does not require such an extension, once one accepts the file passing method of communication between steps, that the SAS design uses. However, the exceptions provide interesting problems.

Recoding the values of a variable provides the most commonplace exception, where one would like an extension of the PDV. In this case formats (or informats) together with the PUT and INPUT functions provide an adequate extension, which has been covered in numerous SUGI papers. Here the formats (or informats) are used as an "associative array" of fixed information. One often considers this technique in the context of a look-up problem instead of a recoding one.

Using SET with the POINT option, one can treat a SAS data set as a collection of huge "arrays" of fixed information where each data set variable names an array and the point variable is the index into these arrays.

With the introduction of the MODIFY statement in SAS 6.07 these huge "arrays" can now hold modifiable information. Hence the same technique becomes applicable to a much wider class of problems.

A sampling problem, for which this technique is appropriate, was posed on SAS-L within the last year by <bxpltw@UABCVS.RCVSR.UAB.EDU>.

The Problem

Records from two data sets, SMALLER and LARGER are to be matched, but a large number of matches are possible for each record in SMALLER and only a random sample of matching records from LARGER is wanted. Various restrictions on the matching sample selection from LARGER are desired:

- with replacement
- unique within match group (weak form of without replacement)
- unique within the matching sample (strong form of without replacement)

A match is determined by the condition

\[
\text{abs (EF-\_EF) } \leq 5 \text{ and abs (PV02-\_PV02) } \leq 30
\]

where \_EF and \_PV02 are variables in SMALLER, and EF and PV02 are the corresponding variables in LARGER. Distinct keys are assumed in both files.

For example, I generated test data using:

```sas
data smaller;
   do _ef = 30 to 35;
      do _pv02 = 190 to 192;
         output;
      end;
   end;
run;
```
data larger;
  do ef=25 to 30;
    do pvo2=180 to 210;
      /* to randomly order set */
      md = ranuni ( 23423487 ) ;
      used = 0 ; /* to note selection */
      /* used in weak wo replacement */
      chk_ef = . ;
      /* used in weak wo replacement */
      chk_pvo2 = . ;
      output;
    end;
  end;
run;

The Code

/* number of matches per _EF _PVO2 */
%let sampsize = 2 ;
/* repltype determines method of selection */
***%let repltype = 1 ; /* no restriction */
/* use at most once strong wo replacement */
%let repltype = ( used = 0 ) ;
/* use at most once with record from smaller */
/* weak without replacement */
***%let repltype = ( used = 0 or
  ( chk_ef ^= _ef or chk_pvo2 ^= _pvo2 ) ) ;

proc sort data = larger out = larger ( drop = md ) ;
  by md ;
run ;

/* ----------------- select sample matches --------------- */
data samp ( drop = used chk_ef chk_pvo2 ) larger
  incmplet ( drop = used chk_ef
    chk_pvo2 ) ;
set smaller ;
cnt = 0 ;
do p = 1 to nobs until ( cnt = &sampsize ) ;
  modify larger point = p nobs = nobs ;
  if abs ( ef - _ef ) <= 5 and
    ( &repltype ) then
    do ;
      cnt + 1 ;
      used = 1 ;
      chk_ef = _ef ;
      chk_pvo2 = _pvo2 ;
      replace larger ;
      output samp ;
    end ;
  end ;
  if p > nobs then output incmplet ;
run ;

/* -------------- review sample ---------------- */
title "SAMPSIZE = &sampsize REPLTYPE = &repltype" ;
proc print data = samp ;
run ;

title2 "Incomplete selection, CNT is # selected" ;
proc print data = incmplet ;
run ;

Conclusion

The standard tools for extending the PDV have been significantly enhanced with the addition of the MODIFY statement to SAS.

The author can be contacted by mail or e-mail

H. Ian Whitlock
Westat
1650 Research Boulevard
Rockville, MD 20850-3129
Whitlo1@westat.com

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. © indicates USA registration.