DEMOXRPT: macros for writing Exception Reports: perform range and logic checks on a data set; write file of exceptions to edit and use for updates

Ronald Fehd, Centers for Disease Control and Prevention, Atlanta GA

ABSTRACT

Data review can be viewed as a two-step process:
1. Compare: review differences between two data sets.
2. Exception Report: review data consistency with range and logic checks.

A data set may be updated with a file containing sets of these three statements -- ID, assignment and closure:

```
*  if ID = 1 then do;
* <variable name> = <value>;
*  end;
```

This paper reviews common problems in writing exception reports for range and logic checking. The product is a file which contains update commands in an easily editable form. Summary information is written at the end of the update file.

The output file is in this form:

```
if ID = 1 then do;
* error: Var_A gt 24;
* VAR_A  = 28;
end;
```

The message indicates why the value has been printed.

INTRODUCTION

Data review is a necessary step in data management. Its purpose is to provide accurate and consistent data for data analysis. While a comparison report (see Fehd 1998, COMPARWS) may identify individual characters or numbers to be updated, the second half of data review consists of identifying values that are out of the range of expected values for a particular variable.

SAS® software provides procedures which may be used to identify outliers. Proc FREQ may be used for character variables; proc UNIVARIATE may be used for numerics. Handsfield (1998) provides a program to write an update file which recodes outliers to missing in order to more accurately calculate confidence intervals before using proc GLM.

This paper reports on a demonstration program and template which fulfills these criteria:
1. A concise report per observation, easy to read and edit.
2. The exception report is written to file for later editing and subsequent use as an %included update file.
3. A summary report provided for data processing review.

The primary macros are:
%ID: write ID line only once,
  write message to reviewer
%SHOW: write variable name and value,
  using either variable name or array reference

Other macros are provided which facilitate quick, clear and easy writing of code to perform range checks of either numeric or character data; data may be checked as individual variables or arrays.

Exception reports are unique to a data set and its variables. The test data set provided contains identifiers, individual variables and arrays in both character and numeric variables. The specifications may be tested by assigning either character or numeric variables to macro variables used as program parameters. A macro variable TESTING is provided in order to be able to test any new routines.

Specifications of a correction file to be used for updates

In order for a text file to be usable as an %included update file, it must be written in the following format:

* if ID = 1 then do;	*1 ID;
* <exception note>;	*2 message;
* variable = <value>;	*3 variable assignment;
* end;	*4 ID closure;

1. The ID line opens an if-do statement for an observation with exceptions. This line is to be printed only once. This example shows a single numeric ID. If the ID is character then the ID value must be enclosed in quotes. Similarly, if other secondary identifiers are character, they must likewise be quoted. The first specification is:

1.1. print ID line once per observation.
1.2. quote ID value(s), according to their type.

2. The exception message indicates why the variable is being reviewed. The second specification is:

2. print the exception message as a comment statement.

3. This program was written to eliminate the errors associated with retyping variable names and values. As with identifiers, variables listed must be quoted, or not, according to their type. The variable assignment specification is:

3. quote exception values, according to their type.

4. Every open control statement must have closure. The last specification is:

4. provide an end statement.

These specifications are implemented in a report writer in step 3.

This DEMOXRPT program contains four sections. The program may be run with the data provided or test data may be inserted into another data set in order to test other macros that call ID and SHOW.

1. Program parameters

TESTING: This macro variable is provided for three purposes: 1. When testing the report is written to file PRINT for quick programmer review. 2. The data set is subset by reducing the number of observations. 3. Invalid values may be inserted into the data by the TESTDATA macro.

ID1, ID2, ID3: Ids are the identifiers or keys of the data set. An exception report is unique to a data set. To add a third ID it is only necessary to enable the macro variable, one line in the ID macro and one line in the summary listings.

DATA_SET: The name of the data set is provided as a macro variable to be used in title and set statements; this ensures that the data evaluated is the correct version.

TITLEn is the number of the last title statement used. It is used to correctly number title lines in the summary.

PRNTFILE is the name of the output file. This file is expected to be edited, have corrections enabled, then used as an %include file for an update. See an example of usage below.

2. Macro definitions

ID prints two lines: 1. If the identifier line has not been printed, i.e., if this is the first exception line for an observation, then the line is printed. 2. The exception message is always printed.

SHOW prints a variable and its value, appropriately quoted. The parameter VAR may be either a variable name or an array reference. The variable type is returned by the macro function VARTYPE; the type of an array is supplied by the CHKARRAY macro. When out-of-range values are to be recoded to missing a second line is provided which sets the variable to missing.

Macros ID and SHOW are used as a pair in this sequence:

```
if <condition> then do;
  %ID(exception message);
  %SHOW(<variable(s) in condition>);
end;
```

When a condition refers to two or more variables each variable is to be shown.

The CHK* macros:

To further reduce the number of lines of code in step 3.3, the programmer-written section of the report writer, these examples using ID and SHOW are provided. They illustrate the use of different functions in conditions.

CHKARRAY illustrates reviewing an array. The macro declares an array of character or numeric variables. It provides the array type to SHOW by declaring a macro variable named with the array name.

CHKFRMT is used for variables with formats where a set of values described by <other='INVALID'> has been defined; invalid values can then be listed as exceptions. Refer to the formats before the Test Data for examples.

CHKHIGH may be used with either character or numeric variables to check if the value exceeds a high point. High points for numeric variables may be discovered by using proc UNIVARIATE. Proc FREQ can be used to identify cutoffs for character variables.

CHKVRIFY is used for individual character variables with a defined set of values. If the set of valid values can be written in a format, use CHKFRMT.

NOBSVARS returns the number of observations and number of variables of a data set. These variables are used in the brief summary. This macro is based on the macro OBSNVARS as described in *SAS® Macro Language Reference, First Edition*.

VARTYPE: Macros ID and SHOW are described in object-oriented programming conventions as methods. This macro returns the variable type which is used by the methods to quote character variables.

3. Report Writer:

3.1. Setup: The data set is saved for use by the summary listings in step 4. The retained variables are used in the brief summary in step 3.4.

3.2. %TESTDATA: changing values to invalid: Test data has been provided with this demonstration program. To facilitate testing this program as it is written and debugged for other data sets the TESTDATA macro: 1. writes output to file PRINT; 2. reduces the number of observations; 3. provides opportunity to insert dummy values.

3.3. list of exceptions: The simple part begins here: for each variable it is only necessary to provide the checking macro, variable- or array-name and check-values. Each of the checking macros generates a minimum of 16 statements. Compare the savings in time spent writing and reviewing one page of code with 50 lines for 50 variables or 50 * 16 = 800 lines: approximately 16 pages of code.

3.4. ID and data step closure, brief summary: This report provides a brief summary at the end of the print file: the number of observations, variables and cells in the data set, the number of identifiers of the exception data, the number of observations of the exception data set, and percentages of each. This data is intended as feedback to the data collection design process. The percentage of cells reported is the exception report rate.

Example of detail with summary data (from test data). Exception report rate is at lower right: Pcnt : Cells

```
*;if NID1 = 3 then do;
* ChkHigh: N4 ge 300;
* N4        =  341  ;
*;end;*Xrpts =1 ;
**smry :  Ids     Vars      Cells;
** Base:  4       8         32;
** Diff:  3                 4;
** Pcnt:  75%               12.5%;
```

4. Summary reports

When the report is written to a file, the following reports may be enabled to be printed: Summary of Ids: proc FREQ of Ids shows Ids in the Exception report. This is used by data review staff to pull data collection forms. Summary of Variables: proc FREQ of Name shows which variables have most exceptions. Detail, by Variable: prints the XRPT data set sorted by Variable Name. Each report is enabled by a macro variable. The program may be used with any set of reports enabled.

Usage:

In a production environment, an exception report program is used to write and print both a correction file and a paper report. The paper report is reviewed by colleagues responsible for data collection. The marked up copy is returned and the desired corrections are enabled in the file by removing the asterisks in front of the appropriate statements. These updates are applied to the data and another data set created. A comparison report (Fehd 1998, COMPARWS) is used to verify that all indicated corrections were made.

Program XRPT
filename XRPT<filespec>;
note: writes report to file, print output for review.

Process: review report, note corrections to be made.

Process: edit corrections file, enable corrections.

Program UPDATE4: apply corrections
```
DATA <DATA4>;
 set <DATA3>;
 %include XRPT;
```

Program COMPAR34: confirm corrections
```
%COMPARWS(LIBRARY.<DATA3>
,LIBRARY.<DATA4>
,<ID-list>);
```
note: print output for review and confirm update.

CONCLUSION

Many different procedures may be used to identify data that is not in the expected set of values for a particular variable. This procedure aids the data review process by writing exceptions to a file where corrections can be made by removing asterisks. This eliminates retyping of variables with their values and the associated edit and correct cycle.

REFERENCES

Fehd, Ronald (1998), “%COMPARWS: Compare with summary: a macro using proc COMPARE to write a file of differences to edit and use for updates,” *Proceedings of the Twenty-Third Annual SAS Users Group International Conference. 23.*

Handsfield, James (1998), “CHEKOUT: A SAS Program to Screen for Outliers” *Proceedings of the Twenty-Third Annual SAS Users Group International Conference. 23.*

SAS Institute Inc. (1997), *SAS® Macro Language Reference, First Edition,* Cary, NC: SAS Institute Inc.

Author: Ronald Fehd e-mail: RJF2@cdc.gov
Centers for Disease Control PHPPO/DLS MS-G25
Atlanta GA 30341-3724 voice: 770/488-4316
SAS-L archives: send e-mail
to: SAScontrib@SASserv.uga.edu
for %COMPARWS subject: cntb0034: download
for DEMOXRPT subject: cntb0039: download

ACKNOWLEDGMENTS

These routines were developed over a period of five years while I crunched the numbers of sample and questionnaire survey data collected by the Model Performance Evaluation Program (MPEP) of the Division of Laboratory Systems, Public Health Practice Program Office of the Centers for Disease Control, and Prevention, Atlanta, Georgia. John Hancock, chief, Information Services Activity, encouraged me to write up these routines. I wish to thank Sharon Blumer, David Cross, Thomas Hearn, and William Schalla of the MPEP group for their data collection.

```
;/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
* PROGRAM: DEMOXRPT: Exception Report demo, with macros ID and SHOW    *
*                                                                      *
* DESCRIPTION: data _NULL_ report writer reads data set                *
*              writes file of exceptions with brief summary            *
*              writes summary report(s) of exceptions                  *
*              file to be used as %included file to update data set    *
*              see also: %COMPARWS                                     *
*                                                                      *
* MACROS:  ID      : prints: 1: ID line once, 2: exception message     *
*          SHOW    : prints variable or array-reference                *
*          CHKARRAY: check each value in array                         *
*          CHKFRMT : check one variable against its format: is in range*
*          CHKHIGH : check one variable is less than high              *
*          CHKVRIFY: check one variable contains only specified chars  *
*          NOBSVARS: returns NOBS and NVARS of data set                *
*          VARTYPE : returns type of variable in ('C','N')             *
*                                                                      *
* PROCESS: 0. make test data from cards, including formats             *
*          1. set program parameters                                   *
*          2. macro definitions                                        *
*          3. data step: report writer                                 *
*          3.1. initialize vars                                        *
*          3.2. if testing, change values to invalid                   *
*            ***programmer-written section -------------------- begin *
*          3.3. list of exceptions                                     *
*            ***programmer-written section ...................... end *
*          3.4  data step closure, brief summary                       *
*          4. summary report(s)                                        *
*                                                                      *
* NOTES: ID and SHOW are methods for printing variables                *
*        variable type is made available to these methods              *
*        thru mac-function VARTYPE                                     *
*        or   mac-vars created by CHKARRAY                             *
*                                                                      *
* KEYWORDS: "data _NULL_ report writer" "exception report"             *
*           "object methods" "data checking" "data review"             *
*                                                                      *
* AUTHOR: Ronald Fehd, B.S. C.Sc                 e-mail: RJF2@cdc.gov *
*         Centers for Disease Control  PHPPO/DLS  MS-G25               *
*         Atlanta  GA  30341-3724                 voice: 770/488-4316 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*%*0. TEST DATA  disable after first run *-------------------------**/
%*recommend MISSING and INVALID be set in AUTOEXEC.SAS;
```

```
%LET MISSING=MISSING;%LET INVALID=INVALID;%*used by CHKFRMT;
proc FORMAT;%*note three sets of values (valid, blank/missing, invalid);
 value Nfour   100-400  ='OK'   . ="&MISSING." other="&INVALID.";
 value $Cthree '01','34'='OK'  '.'="&MISSING." other="&INVALID.";
data DEMOXRPT;  infile cards;
 input @1 CID  $char2.      @1 NID      2.
       @3 CID2 $char1.      @3 NID2     1.
       @4 C1   $char1.      @5 N1       1.
       @6 C21  $char1.      @7 C22  $char1.      @8 C23  $char1.
       @6 N21      1.       @7 N22      1.       @8 N23      1.
      @30 C3   $char2.     @30 N3       2.
      @30 C4   $char3.     @30 N4       3.                      ;cards;
011A1010101000              019
021A1010110101              129
031A1910101010              341
032A1010101010              341
;/*4567890123456789012345678901234567890123456789O
    .    1    .    2    .    3    .    4    .    5*******************/

%*1. set program parameters;

%LET TESTING  = 1;%*set output file to PRINT, see macro TESTDATA*;
%LET TESTING  = 0;

%LET ID1      = CID;%*id is character*;
%LET ID1      = NID;%*id is numeric  *;
%LET ID2      =;%*secondary id is blank*;
%LET ID2      = CID2;
%LET ID2      = NID2;
%LET ID3      =;%*tertiary  id is blank*;
*LET ID3      = CID3;
*LET ID3      = NID3;

%LET DATA_SET =    WORK.DEMOXRPT;%*testdata provided*, used by VARTYPE;
*LET DATA_SET = LIBRARY.XRPTDATA;

TITLE1 'DEMOXRPT: Exception Report: Data out of range or inconsistent';
TITLE2 "report from &DATA_SET.";
%LET TITLEN = 2;%*used for numbering titles in summary report(s)*;

%LET PRNTFILE = "C:\SAS\ZXRPT.SAS";
*LET PRNTFILE = PRINT;%*TESTING=1 sets PRNTFILE to PRINT in TESTDATA;

options linesize = 95   pagesize = 54;*SAS ms 10 portrait duplex:long;
options /*NO*/mprint nocenter pageno=1;

%*2. macro definitions;

%macro ID(MSG);%* print id line, msg, returns 7 statements -----------*;
if not ID then do;
 if LynzLeft le 4 then put _page_;
                       put "*;if &ID1 = " QID1 +B1 &ID1 +B1 QID1
 %IF &ID2 ne %THEN         " and &ID2 = " QID2 +B1 &ID2 +B1 QID2 ;
 %*F &ID3 ne %THEN         " and &ID3 = " QID3 +B1 &ID3 +B1 QID3 ;
                           " then do;";
                       XrptIds + 1; ID = 1;           %*if not ID*; end;
put "* &MSG.;"; %*........................................... ID*; %MEND;

%macro SHOW(VAR);%* print var or array-ref, returns 7 statements -----*
VAR: variable name  : TYPE is supplied by function %VARTYPE
                      var-name is direct reference
     array reference: array-name{I}
                      TYPE is supplied by mac-vars from MAKARRAY
                      var-name comes from vname ......................*;
%local C_BRACKT;%*curly-bracket: array-reference*;
%LET C_BRACKT = %index(&VAR,{);
%IF &C_BRACKT %THEN %DO;                    call vname(&VAR,Name); %END;
%ELSE               %DO;                    Name = "&VAR.";        %END;

if
 %IF &C_BRACKT %THEN "&&&%substr(&VAR,1,&C_BRACKT-1)";
 %ELSE                %VARTYPE(&VAR);
 = 'C' then do;                             Q = "'";
                                            ValueNum =  .   ;
                                            ValueChr = &VAR.;       end;
else        do;                             Q = '';
                                            ValueChr = '.'  ;
                                            ValueNum = &VAR.;       end;
put @1 "*" @3 Name @12 "= " Q +B1 &VAR. +B1 Q  ";";
/*put @1 "*" @3 Name @12 "= " Q +B1 '.'      Q  ";*set to missing?;";*/
XrptCels + 1;Xrpts + 1;output XRPT;
%IF &TESTING %THEN %PUT SHOW:VAR=<&VAR.>;%*............... SHOW*; %MEND;

%macro CHKARRAY(ARAYNAME,C_N,ELEMENTS
,VALUE=%NRSTR('0','1',' ','.'));%* returns 3+7+7+2=19 statements------*;
%local &ARAYNAME;%*local mac-var available to SHOW;
%LET   &ARAYNAME = &C_N.;
array &ARAYNAME {*} %IF "&C_N" = "C" %THEN $; &ELEMENTS;

do I = 1 to dim(&ARAYNAME.);
 if not(&ARAYNAME.{I} in (&VALUE.))
  then do;                          %ID(array &ARAYNAME not in &VALUE);
                                    %SHOW(&ARAYNAME.{I});           end;
                                                     %*do I=1:dim*; end;
%IF &TESTING %THEN %PUT TYPE(&ARAYNAME)=&&&ARAYNAME;%* CHKARRAY*; %MEND;

%macro CHKFRMT(VAR,FORMAT);%* returns 1+7+7+1=16 statements ----------*
INVALID is set in either AUTOEXEC or proc FORMAT program, see test data;
if put(&VAR.,&FORMAT.) eq "&INVALID."
  then do;                          %ID(ChkFrmt: &VAR. not in &FORMAT);
                                    %SHOW(&VAR.);                   end;
%*..................................................... CHKFRMT*; %MEND;

%macro CHKHIGH(VAR,TYPE,HI);%* returns 1+7+7+1=16 statements --------*;
if
 %IF       "&TYPE" = "C" %THEN (&VAR ne "") and (&VAR. ge "&HI");
 %ELSE %IF "&TYPE" = "N" %THEN (&VAR ne . ) and (&VAR. ge  &HI );
 then do;                           %ID(ChkHigh: &VAR. ge &HI);
                                    %SHOW(&VAR.);                   end;
%*..................................................... CHKHIGH*; %MEND;

%macro CHKVRIFY(VAR,RANGE);%* returns 1+7+7+1=16 statements ---------*;
if verify(&VAR.,"&RANGE.") then do; %ID(ChkVrify: other than <&RANGE.>);
                                    %SHOW(&VAR.);                   end;
%*..................................................... CHKVRIFY; %MEND;

%MACRO NOBSVARS(NAMENOBS,NAMENVAR,DATA=.);%*SAS Macro Ln ref 1e pg 242;
%IF &DATA = . %THEN %LET DSN = &SYSLAST.;%*resolve mac-vars in name;
%ELSE               %LET DSN = &DATA.;
%LET DSID = %sysfunc(open(&DSN));
%IF &DSID %THEN %DO;
 %LET &NAMENOBS = %sysfunc(attrn(&DSID,NOBS));
 %LET &NAMENVAR = %sysfunc(attrn(&DSID,NVARS));
 %LET RC        = %sysfunc(close(&DSID.));                          %END;
%ELSE %put Open for data set &DATA. failed - %sysfunc(sysmsg());
%put data=&DATA. &NAMENOBS=<&&&NAMENOBS> &NAMENVAR=<&&&NAMENVAR>; %MEND;
%*global;%LET DATANOBS=;%LET DATAVARS=;%*used in EndoFile summary;
%NOBSVARS(DATANOBS,DATAVARS,DATA=&DATA_SET.);

%MACRO VARTYPE(VARNAME);/*returns "C" or "N" ------------------------*/
%str(vartype(open(%upcase("&DATA_SET"),"I"),
     varnum (open(%upcase("&DATA_SET"),"I"),"&VARNAME"))) %MEND VARTYPE;

%*3.1. initialize report writer vars;
DATA XRPT(keep = &ID1 &ID2 &ID3 Name ValueChr ValueNum N);
 B1=-1;%*pointer control*;
 format _all_;%*turn off all formats;
 length Name $ 8 ValueChr $ 6 ValueNum 8 Xrpts 4 ID 3;
 retain XrptIds    /*incr by ID,   used in EndoFile summary  */
        XrptCels 0 /*incr by SHOW, used in EndoFile summary  */
        N        1 /*no change,    used in summary report(s) */
        QID1 QID2 QID3 '';
  if %VARTYPE(&ID1) = 'C' then QID1 = "'";%*used by ID;
  if %VARTYPE(&ID2) = 'C' then QID2 = "'";%*disable if unused;
%*if %VARTYPE(&ID3) = 'C' then QID3 = "'";%*disable if unused;

 file &PRNTFILE  Linesleft = Lynzleft;
 do until(EndoFile);%*------------------------------------------------*;
  set &DATA_SET end = EndoFile;
  ID    = 0; %*reset by ID *;
  Xrpts = 0; %*incr by SHOW*;

%*3.2. if testing, change values to invalid;
%macro TESTDATA;%*----------------------------------------------------*;
%IF &TESTING %THEN %DO;         file PRINT Linesleft = Lynzleft;
                               *where &ID1 le 2;
                               *C1 = 'Z';
                               *C22 = 'Z';
%*.................................... TESTDATA*; %END; %MEND;%TESTDATA;

%*programmer-written section************************************* begin;

%*3.3. list of exceptions;
%CHKVRIFY(C1,AB);
%CHKARRAY(C2,C,C2:);
%CHKARRAY(N2,N,N2:,VALUE=%NRSTR(0,1));
%CHKFRMT (C3,Cthree.);
%CHKFRMT (N4,Nfour.);
%CHKHIGH (C4,C,300);
%CHKHIGH (N4,N,300);
%CHKVRIFY(C4,012345678);
/****************************************/

%*programmer-written section*************************************** end;

%*3.4  data step closure, brief summary;
 if ID then  put "*;end;*" Xrpts= &ID1= &ID2= /*&ID3=*/ ";";
%*............................................ do until(EndoFile)*; end;

%LET DATACELS = %eval(&DATANOBS * &DATAVARS);
PcntNobs = 100 * XrptIds  / &DATANOBS;
PcntCels = 100 * XrptCels / &DATACELS;
format PcntNobs PcntCels 7.3;
retain          Z1 11             Z2 21            Z3 31;
put "**smry :" @Z1 "Ids"         @Z2 "Vars"       @Z3 "Cells;";
put "** data:" @Z1 "&DATANOBS"   @Z2 "&DATAVARS"  @Z3 "&DATACELS   ;";
put "** xrpt:" @Z1   XrptIds                      @Z3   XrptCels " ;";
put "** pcnt:" @Z1   PcntNobs "%"                 @Z3   PcntCels "%;";
stop;run;

%macro PRNTSMRY();%local SMRYIDS SMRYNAME SMRYVARS;%*----------------*;
%*5. print desired summary report(s);

%LET SMRYIDS =1;%*?print FREQ of IDS?;
%LET SMRYNAME=1;%*?print detail  of variables, by name?;
%LET SMRYVARS=1;%*?print FREQ of variables with exceptions?;

%IF "&PRNTFILE" ne "PRINT" %THEN %DO;options pageno=1;%*--------------;

%IF &SMRYIDS %THEN %DO;
 proc FREQ data = XRPT;
 %*tables &ID1             /list;
   tables &ID1 * &ID2       /list;
 %*tables &ID1 * &ID2 * &ID3/list;
  TITLE%eval(&TITLEN.+1) "summary: IDs listed";                    %END;

%IF &SMRYVARS %THEN %DO;
 proc FREQ data = XRPT;
  tables Name         ;
  TITLE%eval(&TITLEN.+1) "summary: Variables listed";              %END;

%IF &SMRYNAME %THEN %DO;
 proc SORT data = XRPT;
  by  Name ValueChr ValueNum &ID1. &ID2. &ID3.;

 proc PRINT data = XRPT;sum N;
  by Name; id Name;
  TITLE%eval(&TITLEN.+1) "detail: by Var-Name";                    %END;
%*..................................... %IF "&PRNTFILE" ne "PRINT"; %END;
%IF not &TESTING %THEN %DO;%*print exception report written to file*;
 data _NULL_; options pageno=1;
  TITLE%eval(&TITLEN.+1);
  file PRINT;
  do until(EndoFile);%local LRECL;%LET LRECL = 72;
   infile &PRNTFILE end = EndoFile pad lrecl = &LRECL.;
   input @1 Line $char&LRECL..;
   put   @1 Line $char&LRECL..;             %*do until(EndoFile)*; end;
 stop;%*................................*; %END; run; %MEND; %PRNTSMRY;
```