

%X_Ray Macro to Explore Any SAS® Dataset

Kinwah Fung, Institute for Clinical Evaluative Sciences, Toronto, Ontario
Doug Zirbel, Pinnacle Solutions Inc, Indianapolis, IN

ABSTRACT

This paper presents a SAS macro which 1) shows Proc Contents-type information, and 2) prints examples of each variable's values alongside the Format and Length information. This makes it easy to see at a glance what an entire input dataset looks like – without having to read pages of Proc Print output. This will help you identify the distinct values of the variables before you proceed to further data management procedures. By showing outliers and missing values it can also help you to spot data problems in moments rather than hours.

INTRODUCTION

Situation: you get a new SAS file from your Data Management team, or from your own query of new database input. It is fairly large, let's say, 50 variables and 1 million observations. You are asked to write a SAS program to transpose, reformat, summarize, SQL, report on, and/or merge with other SAS datasets. To do so, you must of course thoroughly understand the dataset's data values and their ranges, as well as the file's dimensions and its metadata – in short, you must

"Know thy data!"

Question: So how do you "learn" this new data? Do you send it to Proc Print to see what comes out? Do you use Proc Univariate, or perhaps Proc Freq or Proc Contents... and then compare and contrast these various reports? Those are good first steps. But... would you like to have a simpler and faster way to do this? Read on.

Easier and faster solution (the short answer): Use the %X_RAYSAS macro. Just enter your SAS library name and the SAS file name and submit! The macro will produce a report like the one shown in Appendix B. For each variable in the dataset, it shows you the number of different (unique) values for each variable, as well as 10 actual values as examples, plus the datatype, its length, format, and label (if it has one).

This paper was presented in SUGI25 (paper 212) and has been modified since then.

DIRECTIONS FOR USE

ALL 4 PARAMETERS

The macro's parameters you can specify are

- 1) the library name,
- 2) the dataset name,
- 3) the maximum number of distinct values of a variable to be shown, and
- 4) the page length of the output report – adjust according to your local printer setup.

The distinct values maximum parameter is there so that if you know beforehand that at least one variable has a long list of distinct values, you can avoid printing a huge number of pages. For example, there are 1000 patients with 988 distinct dates of birth in a dataset. Setting the maximum number to, let's say 50, can keep the printout from including the long list of dates of birth.

A CONVENIENT FEATURE FOR USER-WRITTEN FORMATS

Values for variables with user-written formats will appear followed by a '=' sign, followed by their formatted values. For example a numeric variable called STARTDT uses a user-written format called "XYZ.". Its appearance in the Values column of the report will look like this: STARTDT 15810=TAX DAY!

If the variable is not formatted, or is formatted with a native SAS format (e.g. DATETIME16, DATE9, \$8., or 10.), the list of distinct

values for the variable will be printed using their formats only, without the equals sign.

INSTALLATION IN A MACRO LIBRARY

In the change-control-type comments header (line 4), be sure to enter the name of your macro library, as documentation. The macro also has two PUT statements (lines 67 and 311) which you should also modify. Again add the name of your macro library and optionally, its pathname. Then, whenever you run the macro, its beginning and end will be clearly marked in your SAS log.

REQUIREMENTS FOR RUNNING THE MACRO

Besides setting the values of the macro's parameters, you may wish to set the NOCENTER option. You should also set linesize=132 option. If you want to use a different line size, you might need to modify the positions of the PUT statements in the macro. as well.

Please note -- the code shown in this paper includes 2 lines of SAS code at the end which are not part of the macro. They do not belong in the macro library; they are just an illustration of how you might call the macro.

EXPLANATION OF THE MACRO

1) The macro begins with a change control header section, followed by a condensed illustration of an output report, followed by two sample calls to the macro .

2) Next, there is a %PUT statement which will put a line in the SAS log noting that the macro is beginning.

Ex: NOTE: NOW EXECUTING MACROS1(XRAYSAS) (AT C:\My Documents\Macros1)

3) The first data step in the macro checks to see whether the input file exists; if not, it prints an error message to the SAS log.

4) PROC SQL is then used to build a table with one row of metadata for each variable using the SAS DICTIONARY. This approach to getting metadata into a file is much simpler than the alternative of using PROC CONTENTS with a OUT= option, and then reading that dataset in turn to obtain the metadata.

5) The next data step puts each metadata variable from that table into a macro variable whose suffix is a number equal to the variable's position number in the input SAS dataset. Thus, for the dataset "TEMP" (in the sample output) you might picture a macro "table" like this:

```
&VARN1 &NAME1 &TYPE1 &LENG1 &LABLE1 &FORM1 (this
would be for the variable "age")
&VARN2 &NAME2 &TYPE2 &LENG2 &LABLE2 &FORM2 (this
would be for the variable "sex")
&VARN3 &NAME3 &TYPE3 &LENG3 &LABLE3 &FORM3 (this
would be for the variable "dead30")
etc.
```

6) A macro do loop is next, which writes a sequence of SAS code multiple times. The SAS code contains a PROC SQL and a data step – one set of these for each variable in the input dataset. Thus if there are 4 SAS input variables to begin with, this section writes 4 [SQL + Data step] code segments. Each segment is tailored to its own variable, and when it does execute later (at execution time -- after macro compilation time), the PROC SQL gets the number of distinct values in the input SAS dataset for the variable and puts that value into an additional macro variable for the macro "table" described in (5). The data step handles the formats.

7) A DATA _NULL_ step is used to write the control break output report.

8) A PROC DATASETS is used to delete the WORK table VARLIST.

9) Finally, another %PUT statement writes a note to the log indicating the end of the macro. This way, each time the macro runs, you will be able to trace its starting and stopping point in the SAS log.

CONCLUSION

This SAS macro shows a dataset's actual data alongside its metadata for quick analysis of a new or unknown SAS dataset. The report allows the user to browse the details of a data set and identify invalid data before using other procedures to describe the dataset. It simplifies the tasks of data managers and analysts.

REFERENCES

SAS Institute Inc., SAS Procedures Guide, Version 6, Third Edition, 1990. SAS Institute Inc., Cary, NC.
SAS Institute Inc., SAS Macro Language: Reference, First Edition, 1990. SAS Institute Inc., Cary, NC.
SAS Institute Inc., SAS Guide to the SQL Procedure: Usage and Reference Ver. 6, First Edition, 1989. SAS Institute Inc., Cary, NC.

SAS Institute Inc., SAS Sample Programs from Technical Support
http://ftp.sas.com/techsup/download/sample/unix/dstep/csv_labels.sassrc

CONTACT INFORMATION

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

Kinwah Fung
Programming and Biostatistics
Institute for Clinical Evaluative Sciences,
2075 Bayview Avenue
Toronto, Ontario M4N 3M5 Canada
kinwah.fung@ices.on.ca

Doug Zirbel
Pinnacle Solutions Inc.
One North Meridian Street, Suite 400
Indianapolis, IN 46204
(317) 423-0128
doug.zirbel@psiconsultants.com

This paper and the macro code are available at
<http://www.psiconsultants.com/about/publications.htm>

APPENDIX A: THE CODE

```

%MACRO XRAYSAS(LIBNAME=WORK,DSNAME=,MAX_NDV=10,PGLength=60);
*****
%*
%* PROGRAM NAME : <enter-your-SASmacro-libname-here>(XRAYSAS)
%* AUTHOR      : KINWAW FUNG (SUGI25 PAPER 212)
%* AUTHOR      : DOUG ZIRBEL (MODIFICATIONS)
%* DATE WRITTEN : 4/01/2002
%*
%* CHANGE HISTORY
%* *****
%*
%* CHANGED BY :
%* DATE      :
%* REVIEWER   :
%* DATE      :
%* TRACKING # :
%* REASON     :
%*
%* NARRATIVE: (A QUICK VIEW OF WHAT'S IN ANY SAS DATASET)
%* THIS IS LIKE A COMBINATION OF PROC CONTENTS AND PROC PRINT ON
%* ALL THE VARIABLES IN A SAS DATASET. IT PROVIDES A QUICK LOOK
%* AT WHAT IS CONTAINED IN A SAS DATASET. IT LISTS EACH VARIABLE
%* IN THE DATASET, THE NUMBER OF DISTINCT VALUES FOR THE VARIABLE,
%* THE DISTINCT VALUES THEMSELVES (UP TO A MAX_NDV LIMIT),
%* AS WELL AS FORMAT, LENGTH, LABEL, ETC.
%*
%* CONDENSED VIEW OF SAMPLE REPORT BELOW:
%*
%* ===== REPORT BEGIN =====
%*
%* **** SAS DATA XRAYSAS: DATASET = STUDYABC.PATINFO ****
%* NOTE: MAX NUMBER OF UNIQUE VALUES TO BE SHOWN HAS BEEN SET TO 3
%*
%* POS VARIABLE  UNIQ_CNT VALUES          TYPE LEN  FORMAT LABEL
%* -----
%* 1  DEPARTMT: 1      F3Z                  CHAR 3   $3.  DEPARTMENT
%* 2  LOC:        1      NY                  CHAR 2   $2.  LOCATION
%* 3  PROTOCOL:   1      STUDYABC             CHAR 8   $8.  PROTOCOL
%* 4  INV:        19     402                  CHAR 3   $3.  INVESTIGATOR
%*                    403
%*                    405
%* 5  PATIENT:    200    4032                 NUM 8    8.  PATIENT
%*                    4033
%*                    4034
%*                    (ETC ETC ETC)
%*
%* ===== REPORT END =====
%*
%* PARAMETERS
%* A SIMPLE DEFAULT CALL TO XRAYSAS WOULD LOOK SOMETHING LIKE THIS:
%*
%* %*XRAYSAS(DSNAME=PATFILE);
%*
%* HERE IS A CALL THAT USES ALL 4 PARAMETERS:
%*
%* %*XRAYSAS(LIBNAME=ORACLSAS,
%*           DSNAME=PATFILE,
%*           MAX_NDV=5,
%*           PGLength=43);
%*
%*****
%PUT NOTE: NOW EXECUTING <enter-your-SASmacro-libname-here>(XRAYSAS) ;

TITLE "**** SAS DATA XRAYSAS: DATASET = %UPCASE(&LIBNAME..&DSNAME) ****";
TITLE2 " NOTE: MAX NUMBER OF UNIQUE VALUES TO BE SHOWN HAS BEEN SET TO &MAX_NDV";
TITLE3; /* INSERTS EXTRA BLANK LINE AFTER 1ST 2 TITLES */

/***** FUTURE * ENHANCEMENTS *****/
/* - LIMIT 'VALUES' COL VALUES TO 42 IF IT'S */
/* - 1ST LINE FOR VARIABLE (DONT OVERWRITE)*/
/* - HANDLE VARIABLE-LENGTH (> 8 CHARS) VAR */
/* - NAMES IN THE REPORT */
/* - FIX CODE TO HANDLE VAR VALUES WHICH */
/* - HAVE QUOTATION MARKS IN THEM */
/* - ADD MEAN, RANGE, STD, %MISSING, ETC */
/* - CREATE A WHOLE-SASLIB VERSION */
/* - RIGHT-JUSTIFY FOR SOME COLS */
/* - CREATE TYPELEN AS ONE COL */
/* - ADD OTHER PROC CONTENTS INFO (DO CONT TO */
/* - A TEMP FILE, THEN UNSTRING FOR VARS */
/* - NOT AVAIL IN PROC SQL) */
/***** FUTURE * ENHANCEMENTS *****/

*****
*
* BE SURE DATASET EXISTS FIRST -- IF NOT, PUT MSG TO SAS LOG
*
*****
DATA _NULL_;
  IF NOT (EXIST("&LIBNAME..&DSNAME")) THEN DO;

```

```

PUT _PAGE_
/* *****
/* ERROR: ' &LIBNAME.&DSNAME' DOESN'T EXIST.
/* ERROR: ' &LIBNAME.&DSNAME' DOESN'T EXIST.
/* ERROR: ' &LIBNAME.&DSNAME' DOESN'T EXIST.
/* *****
/* *;
ABORT;
END;
RUN;

*****
*
* CREATE TABLE OF VARIABLES AND THEIR ATTRIBUTES (LIKE PROC CONTENTS)
*
*****;
PROC SQL NOPRINT;
  CREATE TABLE VARLIST AS
  SELECT VARNUM, NAME, TYPE, LENGTH, LABEL, FORMAT
  FROM DICTIONARY.COLUMNS
  WHERE LIBNAME=%UPCASE("&LIBNAME") AND
        MEMNAME=%UPCASE("&DSNAME") ORDER BY VARNUM;

  /* PUT TOTAL (MAX) NUMBER OF VARIABLES INTO A MACRO VAR */

  SELECT MAX(VARNUM) INTO: NO_VAR
  FROM VARLIST;
QUIT;

*****
*
* BUILD A 'TABLE' OF MACRO VARS FOR EACH SAS DATASET VAR & ITS ATTRS.
* EA ATTRIBUTE MACRO-VAR WILL BE SUFFIXED WITH A NUMBER (EG &NAME1...)
*
*****;
DATA _NULL_;
  SET VARLIST;
  CALL SYMPUT('VARN' || LEFT(PUT(_N_,4)), COMPRESS(VARNUM));
  CALL SYMPUT('NAME' || LEFT(PUT(_N_,4)), TRIM(NAME));
  CALL SYMPUT('TYPE' || LEFT(PUT(_N_,4)), COMPRESS(UPCASE(TYPE)));
  CALL SYMPUT('LENG' || LEFT(PUT(_N_,4)), COMPRESS(LENGTH));
  CALL SYMPUT('LABE' || LEFT(PUT(_N_,4)), TRIM(SUBSTR(LABEL,1,40)));
  CALL SYMPUT('FORM' || LEFT(PUT(_N_,4)), TRIM(FORMAT));
RUN;

*****
*
* FOR EACH VARIABLE IN THE SAS DATASET, STORE THE DISTINCT VALUES
* INTO MACRO VARIABLES.
* (BUILD MACRO VAR 'TABLE' OF THOSE DISTINCT *VALUES* FOR EACH VAR.)
*
*****;
%DO I=1 %TO &NO_VAR;
  /* *****
  /* SELECT ACTUAL DISTINCT VALUES FOR THAT VAR. ON 1ST MACRO-
  /* COMPILE PASS, THEY WILL RESOLVE TO &NAME1, &NAME1, &NAME1...
  /* ON 2ND PASS, RESOLVE TO THE VAR NAMES FROM DATA _NULL_ ABOVE
  /* *****
  PROC SQL NOPRINT;
    CREATE TABLE VAR&I AS
    SELECT DISTINCT &&NAME&I AS DV
    FROM &LIBNAME.&DSNAME;

    /* NOW GET THE COUNT OF DISTINCT VALUES FOR THAT VAR */
    SELECT COUNT(*) INTO: NO_DV&I
    FROM VAR&I;
  QUIT;

  /* MATCH THE ORIGINAL VALUE WITH
  THE FORMATTED VALUE IF THERE IS ANY */

  DATA _NULL_;
    LENGTH TEMP_FMT $ 30;
    SET VAR&I (OBS=&MAX_NDV);
    TEMP_FMT="&&FORM&I";
    TEMP_LG=LENGTH("&&FORM&I");

    %IF &&FORM&I NE %THEN %DO;
      IF SUBSTR(TEMP_FMT,TEMP_LG-1,1) IN
      ('1','2','3','4','5','6','7','8','9','0','.') THEN DO;
      %IF &&TYPE&I=NUM %THEN %DO;
        CALL SYMPUT("D&I_" || LEFT(PUT(_N_,3)),
          COMPRESS(PUT(DV,&&FORM&I...)));
      %END;
      %ELSE %DO;
        CALL SYMPUT("D&I_" || LEFT(PUT(_N_,3)),
          TRIM(PUT(DV,&&FORM&I...)));
      %END;
    END;
    ELSE DO;
      %IF &&TYPE&I=NUM %THEN %DO;
        CALL SYMPUT("D&I_"
          || LEFT(PUT(_N_,3)), COMPRESS(DV) || '='
          || TRIM(PUT(DV,&&FORM&I...)));

```

```

%END;
%ELSE %DO;
    CALL SYMPUT("D&I._"||LEFT(PUT(_N_,3.)),TRIM(DV)
    ||'|='||TRIM(PUT(DV,&&FORM&I.)));;
%END;
END;
%END;
%ELSE %IF &&FORM&I= %THEN %DO;
    %IF &&TYPE&I=NUM %THEN %DO;
        CALL SYMPUT("D&I._"||LEFT(PUT(_N_,3.)),COMPRESS(DV));
    %END;
    %ELSE %DO;
        CALL SYMPUT("D&I._"||LEFT(PUT(_N_,3.)),TRIM(DV));
    %END;
%END;
RUN;

PROC SQL;
    DROP TABLE VAR&I;
QUIT;
%END;

```

```

*****

```

```

* USE DATA _NULL_ STEP TO WRITE THIS TO A REPORT (PRINT) FILE.

```

```

*****

```

```

DATA _NULL_;
    FILE PRINT;

```

```

%DO I=1 %TO &NO_VAR;
    IF &I=1 OR &PGLLENGTH-LINE_NO LE 1 THEN DO;
        LINE_NO=2;
        PUT _PAGE_ #LINE_NO
            @1 'POS'
            @05 'VARIABLE'
            @20 'UNIQ_CNT'
            @29 'VALUES'
            @72 'TYPE'
            @77 'LEN'
            @82 'FORMAT'
            @92 'LABEL'
        / @1 '-----'
            @05 '-----'
            @20 '-----'
            @29 '-----'
            @72 '-----'
            @77 '-----'
            @82 '-----'
            @92 '-----';
        LINE_NO=LINE_NO+3;
    END;

```

```

%IF &&NO_DV&I > &MAX_NDV %THEN %LET K= &MAX_NDV;
%ELSE %LET K= &&NO_DV&I;

```

```

%DO J=1 %TO &K;
    /* IF YOU-VE REACHED END OF PAGE */
    IF &PGLLENGTH-LINE_NO LE 1 THEN DO;

```

```

        LINE_NO=1;
        PUT _PAGE_ #LINE_NO
            @1 'POS'
            @05 'VARIABLE'
            @20 'UNIQ_CNT'
            @29 'VALUES'
            @27 'TYPE'
            @77 'LEN'
            @82 'FORMAT'
            @92 'LABEL'
        / @1 '-----'
            @05 '-----'
            @20 '-----'
            @29 '-----'
            @72 '-----'
            @77 '-----'
            @82 '-----'
            @92 '-----';
        LINE_NO=LINE_NO+3;
    END;

```

```

/* IF 1ST (CONTROL BREAK) LINE FOR A VARIABLE */
%IF &J=1 %THEN %DO;
    PUT #LINE_NO

```

```

        @1 "&&VARN&I"
        @05 "&&NAME&I"
        @20 "%COMPRES(&&NO_DV&I)"
        @29 "&&D&I._&J"
        @72 "&&TYPE&I"
        @77 "&&LENG&I"
        @82 "&&FORM&I"
        @92 "&&LABE&I"
        ;
        LINE_NO=LINE_NO+1;

```

```

%END;
%ELSE %DO;
    /* IF 1ST DATA LINE ON 'NEXT' PAGE, PUT 'CONTINUED' NOTE */
    IF LINE_NO=4 THEN DO;

```

```

        PUT #LINE_NO
           @1 "&&VARN&I"
           @29 "&&D&I._&J"
           @72 "-CONTINUED-" ;
        LINE_NO=LINE_NO+1;
    END;
    /* IF UNIQUE-VALUE-ONLY LINE (AFTER 1ST FOR VARIABLE) */
    ELSE DO;
        PUT #LINE_NO
           @29 "&&D&I._&J" ;
        LINE_NO=LINE_NO+1;
    END;
%END;
%END;
PUT ' ' ;
LINE_NO=LINE_NO+1;
%END;
RUN;
PROC DATASETS LIB=WORK NOLIST;
    DELETE VARLIST;
QUIT;
%PUT NOTE: NOW ENDING <enter-your-SASmacro-libname-here>(XRAYASAS) ;
%MEND XRAYASAS;

```

SAMPLE CALL

```
options ls=132;
```

```
%XRAYASAS (LIBNAME=WORK,DSNAME=TEMP,MAX_NDV=10,PGLNGTH=60);
```

APPENDIX B: AN EXAMPLE

**** SAS DATA XRAYAS: DATASET = WORK.TEMP ****

NOTE: MAX NUMBER OF UNIQUE VALUES TO BE SHOWN HAS BEEN SET TO 10

16:55 Wednesday, July 10, 2002 1

POS	VARIABLE	UNIQ_CNT	VALUES	TYPE	LEN	FORMAT	LABEL
1	sex	2	F M	CHAR	1	\$1.	
2	age	58	2=15/Below 25=16-25 28=26-35 30=26-35 31=26-35 33=26-35 34=26-35 35=26-35 37=36-45 38=36-45	NUM	3	AGE.	Age at admdate
3	dead30	2	0=No 1=Yes	NUM	8	YES_NO.	Died within 30 days after 1st admission
4	readmit	2	0=No 1=Yes	NUM	8	YES_NO.	Re-Admit after 1st discharge
4	time_readmit	163	.= 0=0 1=1-7 days 2=1-7 days 3=1-7 days 4=1-7 days 5=1-7 days 6=1-7 days 7=1-7 days 8=8-14 days	NUM	8	DAYS.	Time to re-admit after 1st discharge

PRINTED OUTPUT

The SAS macro prints the following:

1. the title with the LIBNAME and the name of the data set
2. each variable's position number in the data set
3. the variable name
4. the count of unique (or distinct) values of the variable
5. the values of the variable (and the formatted values following a '=' sign if required)
6. the type of the variable, character or numeric.
7. the length of the variable.
8. the format
9. the label if any.