Paper 093-2011

To FREQ, Perchance to MEANS

Christopher J. Bost, MDRC, New York, NY

ABSTRACT

Should you run PROC FREQ or PROC MEANS on a variable? That is the question. This paper shows how to use the PROC FREQ option NLEVELS to determine the number of levels of each variable and then, based on a user-supplied cutoff, run either PROC FREQ or PROC MEANS. The Output Delivery System (ODS) and macro variables created with PROC SQL are used to automate the process.

INTRODUCTION

A SAS® data set might contain hundreds or even thousands of variables. Programmers typically run PROC FREQ on categorical variables and PROC MEANS on continuous variables as part of checking data quality. Determining which variables are categorical and which variables are continuous can be a challenge. Typing long lists of variable names is tedious and error prone.

The PROC FREQ statement option NLEVELS can be used to determine the number of levels of each variable. The Output Delivery System (ODS) can be used to save this information to a SAS data set. PROC SQL can then be used to create two macro variables based on a user-supplied cutoff for the number of levels: one to store names of variables on which to run PROC FREQ, and one to store names of variables on which to run PROC MEANS.

SAMPLE DATA

The sample SAS data set SASHELP.CLASS is used in this paper. It contains five variables and 19 observations. NAME and SEX are character variables; AGE, HEIGHT, and WEIGHT are numeric variables:

PRINT of data set SASHELP.CLASS

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

SEX is a categorical variable with two levels. HEIGHT and WEIGHT are continuous variables with multiple levels, including fractional values. AGE has six levels and could be either a categorical variable or a continuous variable. NAME is a unique identifier with 19 levels; it is, arguably, neither a categorical variable nor a continuous variable.

A programmer would typically run PROC FREQ on SEX and AGE, and PROC MEANS on HEIGHT and WEIGHT.

DETERMINING THE NUMBER OF VARIABLE LEVELS

The number of levels of each variable and, thus, whether it should be treated as categorical or continuous, was determined above by visual inspection. This becomes impractical, however, as the number of observations and the number of variable levels increases.

Use the NLEVELS option on the PROC FREQ statement to determine the number of levels of each variable:

```
proc freq data=sashelp.class nlevels;
tables _all_/noprint;
run;
```

The PROC FREQ statement starts the procedure. The NLEVELS option counts the number of variable levels.

The TABLES statement requests frequency tables for all variables. It is unlikely that all variables are categorical; table output is suppressed with the NOPRINT option. The number of levels of each variable, however, is printed:

```
The FREQ Procedure

Number of Variable Levels

Variable      Levels
--------------------
Name              19
Sex                2
Age                6
Height            17
Weight            15
```

It is reasonable to run PROC FREQ on variables with 10 or fewer levels, and to run PROC MEANS on variables with more than 10 levels. (Other cutoffs are possible.) In this case, we would run PROC FREQ on SEX and AGE (2 and 6 levels, respectively) and PROC MEANS on HEIGHT and WEIGHT (17 and 15 levels, respectively).

NAME becomes an issue. Applying the above cutoff, we would run PROC MEANS on NAME given its 19 levels. NAME, however, is a character variable. PROC MEANS can only process numeric variables. We need to know each variable type as well as the number of levels to determine which procedure to run.

COMPILING METADATA (“DATA ABOUT DATA”)

Use the Output Delivery System (ODS) to save the NLEVELS information to a SAS data set. Then add each variable’s type from DICTIONARY.COLUMNS, which stores information about variables in all SAS data sets.

Use ODS TRACE ON to display the name of output objects (i.e., printed output):

```
ods trace on/listing;
proc freq data=sashelp.class nlevels;
tables _all_/noprint;
run;
ods trace off;
```

ODS TRACE ON tells SAS to print the names of output objects. This information is printed in the log by default. The LISTING option tells SAS to print the names of output objects in the listing, preceding the respective output:

```
Output Added:
-------------
Name:        NLevels
Template:    Base.Freq.NLevels
Path:        Freq.NLevels
-------------
```

Note: Use ODS TRACE OFF to turn off printing the names of output objects after running the FREQ procedure.

The name of the output object produced by the NLEVELS option is NLevels. Use the Output Delivery System to save it to a SAS data set:

```
ods output nlevels=nlevelsds;
proc freq data=sashelp.class nlevels;
tables _all_/noprint;
run;
```

The ODS OUTPUT statement saves the NLEVELS output object to a SAS data set named NLEVELSDS. This data set has two variables and five observations:

```
CONTENTS of data set NLEVELSDS

#    Variable    Type    Len    Format    Label

1    TableVar    Char      6              Table Variable
2    NLevels     Num       8    BEST8.    Number of Levels
```

TABLEVAR values are the names of the variables. NLEVELS values are the number of levels of each variable:

```
PRINT of data set NLEVELSDS

       Table
Obs    Var         NLevels

 1     Name             19
 2     Sex               2
 3     Age               6
 4     Height           17
 5     Weight           15
```

Next, retrieve the type of each variable from the SQL DICTIONARY table DICTIONARY.COLUMNS:

```
proc sql;
select name,type
from dictionary.columns
where libname='SASHELP' and memname='CLASS';
quit;
```

The PROC SQL statement starts the procedure.

The SELECT clause retrieves values of NAME (variable name) and TYPE (variable type).

The FROM clause reads rows from DICTIONARY.COLUMNS.

The WHERE clause subsets rows where the value of LIBNAME equals 'SASHELP' and the value of MEMNAME equals 'CLASS' (i.e., rows for variables in SAS data set SASHELP.CLASS):

```
                                  Column
Column Name                       Type
----------------------------------------
Name                              char
Sex                               char
Age                               num
Height                            num
Weight                            num
```

Note that values of LIBNAME and MEMNAME are stored in DICTIONARY.COLUMNS in uppercase.

Variable name (TABLEVAR) and the number of levels (NLEVELS) are in SAS data set NLEVELSDS. Variable name (NAME, labeled 'Column Name') and variable type (TYPE, labeled 'Column Type') are in DICTIONARY.COLUMNS. Rows with matching values of NAME and TABLEVAR can be output to a single data set with an inner join:

```
proc sql;
create table meta as
select name,type,nlevels
from dictionary.columns,nlevelsds
where libname='SASHELP' and memname='CLASS' and name=tablevar;
quit;
```

The PROC SQL statement starts the procedure.

The CREATE TABLE clause creates SAS data set META to store the query results.

The SELECT clause retrieves values of NAME, TYPE, and NLEVELS.

The FROM clause combines all rows in DICTIONARY.COLUMNS with all rows in NLEVELSDS.

The WHERE clause subsets the resulting rows where LIBNAME equals 'SASHELP', MEMNAME equals 'CLASS', and the value of NAME in DICTIONARY.COLUMNS equals the value of TABLEVAR in NLEVELSDS.

Data set META can be printed with PROC PRINT for inspection:

```
PRINT of data set META

Obs    name      type     NLevels

 1     Name      char          19
 2     Sex       char           2
 3     Age       num            6
 4     Height    num           17
 5     Weight    num           15
```

Data set META stores the metadata needed to determine whether to run PROC FREQ or PROC MEANS on each variable.

STORING VARIABLE LISTS IN MACRO VARIABLES

Use PROC SQL to check the number of levels and type of each variable. This will determine which variables should be processed with PROC FREQ and which with PROC MEANS:

```
proc sql;
title 'Variables to process with PROC FREQ';
select name
from meta
where nlevels <= 10;
title 'Variables to process with PROC MEANS';
select name
from meta
where nlevels > 10 and type='num';
quit;
```

The PROC SQL statement starts the procedure. Two queries are run (i.e., two SELECT statements follow).

The first SELECT clause retrieves values of NAME (variable name).

The FROM clause reads rows from SAS data set META.

The WHERE clause subsets rows where the value of NLEVELS (number of variable levels) is less than or equal to 10.

The second SELECT statement subsets rows where the value of NLEVELS is greater than 10 and the value of TYPE (variable type) equals 'num' (numeric).

Note that values of TYPE are stored in SAS data set META (and DICTIONARY.COLUMNS) in lowercase.

The query results are:

```
Variables to process with PROC FREQ

Column Name
--------------------------------
Sex
Age
```

```
Variables to process with PROC MEANS

Column Name
--------------------------------
Height
Weight
```

SQL can store query results in a macro variable. One macro variable can contain the list of categorical variables to process with PROC FREQ, another the list of continuous variables to process with PROC MEANS:

```
proc sql;
select name into :FREQvars separated by ' '
from meta
where nlevels <= 10;
select name into :MEANSvars separated by ' '
from meta
where nlevels > 10 and type='num';
quit;
```

The PROC SQL statement starts the procedure. Two queries are run (i.e., two SELECT statements follow).

The first SELECT clause retrieves values of NAME (variable name). The INTO clause stores the values in a macro variable named FREQVARS, separated by blanks.

The FROM clause reads rows from SAS data set META.

The WHERE clause subsets rows where the value of NLEVELS (number of variable levels) is less than or equal to 10.

The second SELECT statement subsets rows where the value of NLEVELS is greater than 10 and the value of TYPE (variable type) equals 'num' (numeric), and stores values in a macro variable named MEANSVARS.

%PUT statements can be used to display the values of the resulting macro variables:

```
%put FREQvars=&FREQvars;
%put MEANSvars=&MEANSvars;
```

SAS writes the following to the log:

```
FREQvars=Sex Age
MEANSvars=Height Weight
```

Macro variables FREQVARS and MEANSVARS can now be referenced in PROC FREQ and PROC MEANS.

USING VARIABLE LISTS STORED IN MACRO VARIABLES

The variables on which to run PROC FREQ and the variables on which to run PROC MEANS are now stored in macro variables FREQVARS and MEANSVARS, respectively. Reference FREQVARS on the TABLES statement in PROC FREQ and MEANSVARS on the VAR statement in PROC MEANS:

```
proc freq data=sashelp.class;
tables &FREQvars;
run;

proc means data=sashelp.class;
var &MEANSvars;
run;
```

The SYMBOLGEN option can be used to display the values of resolved macro variables in the log:

```
108        options symbolgen;
109        proc freq data=sashelp.class;
SYMBOLGEN:  Macro variable FREQVARS resolves to Sex Age
110        tables &FREQvars;
111        run;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The PROCEDURE FREQ printed page 1.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           0.12 seconds
      cpu time            0.01 seconds


112
113        proc means data=sashelp.class;
SYMBOLGEN:  Macro variable MEANSVARS resolves to Height Weight
114        var &MEANSvars;
115        run;
```

SAS prints the following in the listing:

The FREQ Procedure

Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	9	47.37	9	47.37
M	10	52.63	19	100.00

Age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11	2	10.53	2	10.53
12	5	26.32	7	36.84
13	3	15.79	10	52.63
14	4	21.05	14	73.68
15	4	21.05	18	94.74
16	1	5.26	19	100.00

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

PROCESSING CHARACTER VARIABLES WITH MANY LEVELS

The above approach ran PROC FREQ on SEX and AGE, and PROC MEANS on HEIGHT and WEIGHT. The variable NAME, however, was not processed.

The following table summarizes the procedure to run based on variable type and the number of variable levels:

	Number of Variable Levels	
Variable Type	10 or fewer	More than 10
Character	PROC FREQ	*PROC PRINT*
Numeric	PROC FREQ	PROC MEANS

Table 1. Variable Type, Number of Variable Levels, and PROC

When the variable has 10 or fewer levels, PROC FREQ is run (i.e., whether the variable is character or numeric). When the variable has more than 10 levels and is numeric, PROC MEANS is run. No rule was created, however, for when the variable has more than 10 levels and is character.

It is reasonable to run PROC PRINT on character variables with more than 10 levels provided that the number of observations is limited. The previous approach can be adapted to this situation:

```
proc sql;
select name into :PRINTvars separated by ' '
from meta
where nlevels > 10 and type='char';
quit;

proc print data=sashelp.class(obs=5);
var &PRINTvars;
run;
```

The PROC SQL statement starts the procedure.

The SELECT clause retrieves values of NAME (variable name). The INTO clause stores the values in a macro variable named PRINTVARS, separated by blanks.

The FROM clause reads rows from SAS data set META.

The WHERE clause subsets rows where the value of NLEVELS (number of variable levels) is greater than 10 and the value of TYPE (variable type) equals 'char' (character).

The PROC PRINT statement prints observations in SAS data set SASHELP.CLASS. Note the data set option OBS=5 in parentheses following SASHELP.CLASS. This tells SAS to stop processing after the fifth observation.

The VAR statement references macro variable PRINTVARS which stores the names of variables on which to run PROC PRINT.

SAS prints the following in the listing:

```
Obs     Name

  1     Alfred
  2     Alice
  3     Barbara
  4     Carol
  5     Henry
```

Adjust the number of observations to print based on the data.

FINAL PROGRAM

The above code is streamlined and incorporated into a single program below. A macro is included in the Appendix.

```
ods output nlevels=nlevelsds;
proc freq data=sashelp.class nlevels;
tables _all_/noprint;
run;

proc sql noprint;

create table meta as
select name,type,nlevels
from dictionary.columns,nlevelsds
where libname='SASHELP' and memname='CLASS' and name=tablevar;

*store names of all variables with NLEVELS <= 10 in macro variable FREQvars;
select name into :FREQvars separated by ' '
from meta
where nlevels <= 10;

*store names of numeric variables with NLEVELS > 10 in macro variable MEANSvars;
select name into :MEANSvars separated by ' '
from meta
where nlevels > 10 and type='num';

*store names of character variables with NLEVELS > 10 in macro variable PRINTvars;
select name into :PRINTvars separated by ' '
from meta
where nlevels > 10 and type='char';

quit;

proc freq data=sashelp.class;
tables &FREQvars;
run;

proc means data=sashelp.class;
var &MEANSvars;
run;

proc print data=sashelp.class(obs=5);
var &PRINTvars;
run;
```

CONCLUSION

The PROC FREQ option NLEVELS counts the number of levels of each variable. The Output Delivery System can save this information to a SAS data set. PROC SQL can check the number of levels and variable type and create macro variables that store respective lists of variables on which to run PROC FREQ and PROC MEANS.

CONTACT INFORMATION

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

Christopher J. Bost
MDRC
16 East 34th Street
New York, NY 10016
(212) 340-8613
christopher.bost@mdrc.org

APPENDIX

The following macro runs PROC FREQ, PROC MEANS, and (optionally) PROC PRINT based on the number of variable levels and the variable type:

```
%macro FreqMeans(lib=work,       /* libref for input data set            */
                 dsn=,           /* name of input data set               */
                 cutoff=10,      /* maximum number of levels for FREQ    */
                 metalib=work,   /* libref for meta data set             */
                 metadsn=meta,   /* name of meta data set                */
                 printcases=no,  /* print char vars with > cutoff levels */
                 ncases=20);     /* number of cases to print             */

*1. Create NLEVELS output data set;
ods listing close; *turn off printing;
ods output nlevels=nlevelsds;
proc freq data=&lib..&dsn nlevels;
tables _all_/noprint;
run;
ods listing; *turn on printing;

*2. Create META data set;
proc sql noprint;
create table &metalib..&metadsn as
select name,type,nlevels
from dictionary.columns,nlevelsds
where libname=upcase("&lib") and memname=upcase("&dsn") and name=tablevar;

*3A. Store names of all variables with NLEVELS <= cutoff
     in macro variable FREQvars;
select name into :FREQvars separated by ' '
from meta
where nlevels <= &cutoff;

*3B. Store names of numeric variables with NLEVELS > cutoff
     in macro variable MEANSvars;
select name into :MEANSvars separated by ' '
from meta
where nlevels > &cutoff and type="num";

*3C. Conditionally store names of character variables with NLEVELS > cutoff
     in macro variable PRINTvars;
%let PRINTvars=; *initialize macro variable;
%if %upcase(&printcases)=YES %then %do;
  select name into :PRINTvars separated by ' '
  from meta
  where nlevels > &cutoff and type="char";
%end;

quit;

*4A. Run PROC FREQ on all variables with NLEVELS <= cutoff;
%if &FREQvars ne %then %do;
  proc freq data=&lib..&dsn;
  tables &FREQvars;
  title "PROC FREQ of all variables with NLEVELS <= &cutoff";
  run;
%end;
```

```
*4B. Run PROC MEANS on numeric variables with NLEVELS > cutoff;
%if &MEANSvars ne %then %do;
  proc means data=&lib..&dsn;
  var &MEANSvars;
  title "PROC MEANS of numeric variables with NLEVELS > &cutoff";
  run;
%end;

*4C. Upon request, run PROC PRINT on character variables with NLEVELS > cutoff;
%if %upcase(&printcases)=YES and &PRINTvars ne %then %do;
  proc print data=&lib..&dsn(obs=&ncases);
  var &PRINTvars;
  title "PROC PRINT of character variables with NLEVELS > &cutoff";
  run;
%end;

%mend FreqMeans;
```

SAMPLE MACRO CALLS

```
%FreqMeans(lib=sashelp,             /* run macro on SASHELP.CLASS          */
           dsn=class)

%FreqMeans(lib=sashelp,
           dsn=class,
           printcases=yes,          /* print char vars with nlevels>cutoff */
           ncases=40)               /* print 40 observations               */

libname sasdata 'c:\metadata';
%FreqMeans(lib=sashelp,
           dsn=class,
           metalib=sasdata,         /* save metadata in permanent          */
           metadsn=classmeta)       /* SAS data set SASDATA.CLASSMETA      */
```

LIB and METALIB default to WORK (i.e., temporary SAS data sets).

DSN is the only required parameter.

CUTOFF defaults to 10.

METADSN defaults to META. To save the metadata to a permanent SAS data set, specify a previously defined libref for METALIB and, optionally, a data set name for METADSN.

PRINTCASES defaults to NO (i.e., character variables with more levels than the specified cutoff are not printed). Specify YES (in uppercase, lowercase, or mixed case) to print these variables.

NCASES defaults to 20. Specify PRINTCASES=YES to print cases.

Note: This macro is not guaranteed to work in every situation.