Leveraging Metadata with the SAS Macro Language

Keith Curtis

Senior Modeling Analyst

USAA

Abstract

Do you spend too much time typing variable names in your programs? Do you want SAS to write this code for you? By leveraging metadata from Proc Contents in tandem with Call Symputx, the need to manually enter variables is eliminated.

Introduction

- Two SAS programs that produce the same output table will be compared.
  - Program 1 requires manual entry of variable names.
  - Program 2 utilizes Proc Contents and Call Symputx.
    - No manual entry of variable names is required.
- The output table contains cardinality counts for 50 variables.
  - Cardinality: The number of unique values within a list of observations.
    - Example: The five observations 27, 31, 31, 21 and 34 have a cardinality of four, since 31 is repeated twice.
- The SAS output table is tabulated in the next slide.

SAS Output Table

Row	Name	Cardinality
1	ACC_FREE	3
2	AFA_AccSizeMax	7
3	ANTI_THFT_IND	2
4	BAS_LOC_CD	48
5	CRMCYLARC	168
6	CRMCYMURD	158
7	CRMCYMVEH	149
8	CRMCYPERC	166
9	CRMCYROBB	134
10	CRMCYTOTC	180
11	DMCTCNTL21	9
12	DMCTDST100K	7
13	DMHWCNT21	50
14	DMHWDST	7
15	DMSCDST	7
16	DMTRAVG21	253
17	DMUNDST	7
18	DPTP_KT	191
19	DPTP_SK	131
20	FAMGRW0010	250
21	HTDG_SK	246
22	TAX_EXEMPT_AMT	253
23	M19049a_I	53
24	POPDENS_CY	253
25	PRCP1_SD	146

Row	Name	Cardinality
26	PRCP2_SK	83
27	PRCPANY_SD	227
28	PRCPANY_SK	85
29	SNOWANY_SK	97
30	SNOW_KT	206
31	SNWD_SD	199
32	SROUGH01KM	253
33	TMAX90_MX	175
34	TOTPOP00	253
35	WNDS10_MN	69
36	WNDS15_KT	192
37	WNDS_KT	111
38	X14001_I	186
39	CLOSEST_BASE	7
40	BASE_IND	2
41	pp_B01001e26	253
42	pp_B08007e8	253
43	pp_B08302e5	253
44	pp_B11004e10	253
45	pp_B25034e9	253
46	pp_B25040e2	252
47	pp_B25042e8	253
48	pp_B25081e5	253
49	pp_FEM18UP_CY	253
50	pp_FEM60C10	253

Program 1: Manual Entry Code

Initialize a dataset named cardinality. It will contain the final output.

```
data cardinality;
   format name $32.;
run;
```

Initialize the compute_card macro. It receives the three parameters below:

1. *i*: The row number in the cardinality dataset. It is also used to name the individual cardinality dataset records.
2. *var*: The variable within the example dataset on which cardinality is computed.
3. *var_char*: A character label corresponding to the variable above.

```
%macro compute_card (i,var,var_char);
```

Proc sql calculates the number of unique observations, or cardinality, for the &var. variable within the example dataset.

```
proc sql;
create table cardinality&i. as
select &i. as row,
        count(distinct &var.) as cardinality
from example;
quit;
```

Append the single observation computed in Proc SQL to the bottom of the cardinality dataset. For this appended row, the label contained in the &var_char. macro variable is assigned to the name variable.

```
data cardinality;
set cardinality cardinality&i. (in=a);
   if a then name = &var_char.;
run;
```

Delete the cardinality&i. dataset, since its information now resides in the cumulative cardinality dataset.

```
proc datasets library=work;
   delete cardinality&i.;
run;
```

Specify the end of the compute_card macro with the %mend statement.

```
%mend compute_card;
```

Program 1: Manual Entry Code (Cont.)

Call the compute_card macro 50 times, or once for each variable residing in the example dataset.

```
%compute_card( 1,ACC_FREE,"ACC_FREE");
%compute_card( 2,AFA_AccSizeMax,"AFA_AccSizeMax");
%compute_card( 3,ANTI_THFT_IND,"ANTI_THFT_IND");
%compute_card( 4,BAS_LOC_CD,"BAS_LOC_CD");
%compute_card( 5,CRMCYLARC,"CRMCYLARC");
%compute_card( 6,CRMCYMURD,"CRMCYMURD");
%compute_card( 7,CRMCYMVEH,"CRMCYMVEH");
%compute_card( 8,CRMCYPERC,"CRMCYPERC");
%compute_card( 9,CRMCYROBB,"CRMCYROBB");
%compute_card(10,CRMCYTOTC,"CRMCYTOTC");
%compute_card(11,DMCTCNTL21,"DMCTCNTL21");
%compute_card(12,DMCTDST100K,"DMCTDST100K");
%compute_card(13,DMHWCNT21,"DMHWCNT21");
%compute_card(14,DMHWDST,"DMHWDST");
%compute_card(15,DMSCDST,"DMSCDST");
%compute_card(16,DMTRAVG21,"DMTRAVG21");
%compute_card(17,DMUNDST,"DMUNDST");
%compute_card(18,DPTP_KT,"DPTP_KT");
%compute_card(19,DPTP_SK,"DPTP_SK");
%compute_card(20,FAMGRW0010,"FAMGRW0010");
%compute_card(21,HTDG_SK,"HTDG_SK");
%compute_card(22,TAX_EXEMPT_AMT,"TAX_EXEMPT_AMT");
%compute_card(23,M19049a_I,"M19049a_I");
%compute_card(24,POPDENS_CY,"POPDENS_CY");
%compute_card(25,PRCP1_SD,"PRCP1_SD");
%compute_card(26,PRCP2_SK,"PRCP2_SK");
%compute_card(27,PRCPANY_SD,"PRCPANY_SD");
%compute_card(28,PRCPANY_SK,"PRCPANY_SK");
%compute_card(29,SNOWANY_SK,"SNOWANY_SK");
%compute_card(30,SNOW_KT,"SNOW_KT");
%compute_card(31,SNWD_SD,"SNWD_SD");
%compute_card(32,SROUGH01KM,"SROUGH01KM");
%compute_card(33,TMAX90_MX,"TMAX90_MX");
%compute_card(34,TOTPOP00,"TOTPOP00");
%compute_card(35,WNDS10_MN,"WNDS10_MN");
%compute_card(36,WNDS15_KT,"WNDS15_KT");
%compute_card(37,WNDS_KT,"WNDS_KT");
%compute_card(38,X14001_I,"X14001_I");
%compute_card(39,CLOSEST_BASE,"CLOSEST_BASE");
%compute_card(40,BASE_IND,"BASE_IND");
%compute_card(41,pp_B01001e26,"pp_B01001e26");
%compute_card(42,pp_B08007e8,"pp_B08007e8");
%compute_card(43,pp_B08302e5,"pp_B08302e5");
%compute_card(44,pp_B11004e10,"pp_B11004e10");
%compute_card(45,pp_B25034e9,"pp_B25034e9");
%compute_card(46,pp_B25040e2,"pp_B25040e2");
%compute_card(47,pp_B25042e8,"pp_B25042e8");
%compute_card(48,pp_B25081e5,"pp_B25081e5");
%compute_card(49,pp_FEM18UP_CY,"pp_FEM18UP_CY");
%compute_card(50,pp_FEM60C10,"pp_FEM60C10");
```

Print the final output.

```
proc print data=cardinality noobs;
   var row name cardinality;
run;
```

Program 1: 50 Macro Calls

- Entering 50 variable names is time consuming.
  - Spelling errors are also likely.
- Using Proc Contents with Call Symputx statements eliminates the above macro calls.
  - Analysis may be applied to new datasets without knowing variable names, or even the number of variables, beforehand.

Proc Contents

- Proc Contents produces metadata for a SAS Dataset.
- Metadata regarding the overall dataset includes the following:
  - Row (observations) and column (variables) counts
  - The file path location of the dataset
- Variable specific metadata includes variable names, lengths and formats.
- The basic syntax is shown below:

```
proc contents data=<SAS Dataset Name>;
run;
```

- Default results window output sorts variables in alphabetical order.
  - Adding 'varnum' will sort variables in the order they appear in the dataset.
- Results may also be saved into a SAS dataset.
  - Variables are sorted in alphabetical order, even when 'varnum' is present.

Proc Contents Output (Exported to Excel)

The CONTENTS Procedure

Data Set Name	WORK.EXAMPLE	**Observations**	117599819
Member Type	DATA	**Variables**	50
Engine	V9	**Indexes**	0
Created	Wed, Oct 02, 2013 06:12:42 PM	**Observation Length**	386
Last Modified	Wed, Oct 02, 2013 06:12:42 PM	**Deleted Observations**	0
Protection		**Compressed**	CHAR
Data Set Type		**Reuse Space**	NO
Label		**Point to Observations**	YES
Data Representation	LINUX_32, INTEL_ABI	**Sorted**	NO
Encoding	latin1 Western (ISO)		

Engine/Host Dependent Information

Data Set Page Size	24576
Number of Data Set Pages	458106
Number of Data Set Repairs	0
Filename	/sasworkg5/SAS_workA6EB00007F1B_prodsasem3l/SAS_workAA7C00007F1B_prodsasem3l/example.sas7bdat
Release Created	9.0202M3
Host Created	Linux
Inode Number	327708
Access Permission	rw-rw-r--
Owner Name	plg4305
File Size (bytes)	11258421248

Proc Contents Output (Cont.)
(First 25 Variables Exported to Excel)

#	Variable	Type	Len	Format	Informat	Label
5	ACC_FREE	Char	4	$4.00	$4.00	ACC_FREE
6	AFA_AccSizeMax	Char	17	$17.00	$17.00	AFA_AccSizeMax
49	ANTI_THFT_DSC_IND	Char	1	$1.00	$1.00	ANTI_THFT_DSC_IND
2	BAS_LOC_CD	Char	3	$3.00	$3.00	BAS_LOC_CD
46	BG_DMCTCNTL21	Num	8			BG_DMCTCNTL21
48	BG_DMHWCNT21	Num	8			BG_DMHWCNT21
47	BG_DMHWDST	Num	8			BG_DMHWDST
9	BI_EE	Num	8	8.4	8.4	BI_EE
13	BI_IncClmCnt	Num	8	6	6	BI_IncClmCnt
12	BI_IncNetSS_CAPPED	Num	8	20.2	20.2	BI_IncNetSS_CAPPED
1	BinRandom	Num	8	20	20	BinRandom
7	CENSUS_BLKGRP_2000	Char	12	$12.00	$12.00	CENSUS_BLKGRP_2000
8	CENSUS_BLKGRP_2010	Char	12	$12.00	$12.00	CENSUS_BLKGRP_2010
50	CLN_DRV_DSC_IND	Char	1	$1.00	$1.00	CLN_DRV_DSC_IND
10	CL_EE	Num	8	8.4	8.4	CL_EE
15	CL_IncClmCnt	Num	8	6	6	CL_IncClmCnt
14	CL_IncNetSS_CAPPED	Num	8	20.2	20.2	CL_IncNetSS_CAPPED
3	CO_SHA	Char	4	$4.00	$4.00	CO_SHA
4	CO_SHA_NO_USCM	Char	4	$4.00	$4.00	CO_SHA_NO_USCM
31	CP_CAT_ANIMAL_ClmCnt	Num	8	6	6	CP_CAT_ANIMAL_ClmCnt
32	CP_CAT_ANIMAL_IncNetSS	Num	8	20.2	20.2	CP_CAT_ANIMAL_IncNetSS
33	CP_CAT_ANIMAL_IncNetSS_CAPPED	Num	8	20.2	20.2	CP_CAT_ANIMAL_IncNetSS_CAPPED
34	CP_CAT_GLASS_ClmCnt	Num	8	6	6	CP_CAT_GLASS_ClmCnt
35	CP_CAT_GLASS_IncNetSS	Num	8	20.2	20.2	CP_CAT_GLASS_IncNetSS
36	CP_CAT_GLASS_IncNetSS_CAPPED	Num	8	20.2	20.2	CP_CAT_GLASS_IncNetSS_CAPPED

Call Symputx

- Call Symputx is a statement within the SAS Macro Language.
- It is executed within a data step.
- Call Symputx converts information within a SAS dataset into a macro variable:
  - Column names from a SAS dataset
  - Values from a column
  - Automatic variables generated by data steps (ex. _n_)
- It is also an alternative to % let for assigning constant values to a macro variable.
  - % let is used in open code, while Call Symputx is used in data steps.
- Syntax: call symputx ('macro-variable', value)

Program 2: Proc Contents/ Call Symputx Code

Assign the SAS dataset name to a macro variable named dataset.

```
%let dataset = example;
```

Run proc contents on the example data, and output the results to the variable_list_pre dataset. Variable_list_pre will contain 50 Rows, or one row for each variable residing in the example data.

```
proc contents data=&dataset. varnum out=variable_list_pre;
run;
```

The data step below that creates the variable_list dataset performs three tasks:

1. The SAS generated row number variable (_n_) is assigned to a variable named row.
2. Assignment of the 50 variable names to macro variables &V1. - &V50.
3. Assignment of the number of variables (50) to the &N. macro variable.

```
data variable_list;
set variable_list_pre end=last;
   format row 8.;
   row = left(_n_);
   call symputx('V'||left(row), name);
   if last then call symputx('N', row);
run;
```

Initialize a dataset named cardinality.

```
data cardinality;
   set _null_;
run;
```

Initialize the compute_card macro. This macro is needed since the do loop in the next line depends on the macro variable N, and therefore cannot be executed within open code. This do loop is iterated 50 times, or once for each variable.

```
%macro compute_card;
%do i = 1 %to &N.;
```

Proc sql calculates the number of unique observations, or cardinality, for the &&V&i. variable within the example dataset.

```
proc sql;
create table cardinality&i. as
select &i. as row,
        count(distinct  &&V&i.) as cardinality
from &dataset.;
quit;
```

Program 2: Proc Contents/ Call Symputx Code (Cont.)

Append the single observation computed in Proc SQL to the bottom of the cardinality dataset.

```
data cardinality;
   set cardinality cardinality&i. ;
run;
```

Delete the cardinality&i. dataset, since its information now resides in the cumulative cardinality dataset.

```
proc datasets library=work;
   delete cardinality&i.;
run;
```

Specify the end of the do loop and compute_card macro with the %end and %mend statements, respectively.

```
%end;
%mend compute_card;
```

Call the compute_card macro.

```
%compute_card;
```

Create the cardinality_final dataset by merging variable names with their corresponding cardinality values.

```
proc sql;
create table cardinality_final as
select b.name, a.*
from cardinality a left join variable_list b
on a.row = b.row;
quit;
```

Print the final output.

```
proc print data=cardinality_final noobs;
   var row name cardinality;
run;
```

Conclusions

- Leveraging metadata with the SAS Macro Language provides the following benefits:
  - The need to manually enter variables is eliminated.
    - This will both save time and reduce errors.
  - Flexible code is written that can easily accommodate new input datasets.
    - The user can apply identical analysis to new datasets without knowing variable names, or even the number of variables.

Contact Information

Name: Keith Curtis

Phone: 210.498.0820

Email: keith.curtis@usaa.com