I Hate Typing Variable Names

Mark Kirkham, Merrick Bank, South Jordan, Utah

ABSTRACT

At the bank we have very large datasets most of these dataset have hundreds of variables. When I first started coding in SAS® I would create a macro variable at the top of my code that would contain a list of variables I am interested in. This macro variable is usually used in the keep = option. I've come up with three better techniques to build the macro variables with the important variable names. First, for frequently used datasets, I create the macro variable in the SAS autoexec file. Because these macro variables are built upon starting SAS, they are always available when I want to use them. Second, I use the output dataset from PROC Contents. I use this dataset and the into clause in PROC SQL to build a macro variable with variable names in it. The last technique is to use dictionary tables.

Keywords: autoexec, Proc Contents, Proc SQL, into, dictionary tables, macro variables.

INTRODUCTION

The datasets we use in banking have hundreds of variables. Typically we are interested in a subset of these variables. Either a keep statement or a keep = dataset option is used to include only those variables that are important for analysis. Many experienced SAS users will tell you that using a keep statement or a keep = dataset option is preferable to using the drop statement or the drop = dataset option because it documents what variables are in the newly created dataset. For me having long lists of variables in my code seems to clutter the code and typing out long lists of variable names can be time consuming. This paper reviews some techniques I use to avoid typing long lists of variable names in my code. This paper also demonstrates the flexibility built into SAS by using tables that are output from PROC steps and dictionary tables.

USING THE AUTOEXEC FILE

The SAS System searches for an autoexec file whenever it is invoked. The name of the default autoexec file and where the SAS System looks for the file vary by site and operating system.

A SAS autoexec file has SAS statements that are executed automatically whenever the SAS System is started. Any valid SAS statements can be used in an autoexec file. In my autoexec file for example, I have a list of commonly used libname statements. I also create several macro variables that have lists of commonly used variable names. I have one for the master file, one for our authorization file, etc. Several statements like the following are in my autoexec file.

```
%let VARLIST = <Variable name list> ;
```

In my code if I was doing analysis on authorizations, instead of a long list of variables I simply have

```
Data AUTHDATA;
  Set AUTHSOURCE (keep = &VARLIST);
.
.
.
Run;
```

USING PROC CONTENTS

The output to proc contents can be stored in SAS datasets. This output has some additional information about both variables and datasets that can be used. The following code creates an output dataset called cont_out from the dataset science_data.

```
proc contents
   data = science_data
   out = cont_out
   noprint;
run;
```

ADDITIONAL VARIABLE INFORMATION

One useful column in the output dataset is the column named *type* (this column is labeled 'variable type') values of 2 are character variables values of 1 are numeric. This is useful if you are interested only numeric or character variables. Another useful variable is *varnum*, which tells the order of the variables in the original dataset. This can be used to select the first or last *n* variables or to select variables by position. At the bank for example we have a table with customer indicative information at the beginning of the table and performance information at the end of the table. If we wanted just the performance variables we could just select the variables towards the end of the table.

	Library Name	Library Member Name	Data Set Label	Special Data Set Type (From TYPE=)	Variable Name	Variable Type	Variable Length	Variable Number
1	WORK	SCIENCE_DATA			ASTROSCI	2	21	5
2	WORK	SCIENCE_DATA			CONSCI	2	12	4
3	WORK	SCIENCE_DATA			EDUC	2	8	6
4	WORK	SCIENCE_DATA			EDUC2	1	8	9
5	WORK	SCIENCE_DATA			MARITAL	2	8	1
6	WORK	SCIENCE_DATA			POLVIEWS	2	20	2
7	WORK	SCIENCE_DATA			RELIG	2	23	3
8	WORK	SCIENCE_DATA			SEX	2	8	8
9	WORK	SCIENCE_DATA			SPEDUC	2	8	7

Figure 1: partial output from a SAS dataset produced by proc contents

ADDITIONAL DATASET INFORMATION: INDEXES AND SORTS

Proc contents can be used to determine if a dataset has an index or if it is sorted. There is a variable named IDXUSAGE which will have a value of *simple* if a simple index is used or a value of *composite* if a composite index is used. There is also a variable named SORTEDBY which is missing if the variable isn't used in a sort, or if the variable is used in a sort it will show its position in the sort. I have successfully used these variables as part of a macro loop to check to see if a table is indexed on the customer's account number. If it was indexed, the code would go ahead and merge it with another table, if not the macro code would do the additional step of sorting before merging. There are other variables that may be useful in this output dataset but are not listed here.

USING OUTPUT FROM PROC CONTENTS TO BUILD A VARIABLE LIST

Once we know which variables we want to include in our analysis, a macro variable is built using PROC SQL with the into clause. Here is some sample code using proc contents to build a variable list from of all the character variables in the dataset called science_data:

```
proc sql noprint;
  select name
  into : varlist
  separated by ' '
  from cont_out
  where type = 2;
quit;

%put &varlist;
```

Here is the log…

```
132  proc contents
133     data = science_data
134     out = cont_out
135     noprint;
136  run;

NOTE: The data set WORK.CONT_OUT has 9 observations and 40 variables.
NOTE: PROCEDURE CONTENTS used (Total process time):
```

```
      real time           0.01 seconds
      cpu time            0.03 seconds


137
138  proc sql noprint;
139    select name
140    into : varlist
141    separated by ' '
142    from cont_out
143    where type = 2;
144  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.03 seconds


145
146  %put &varlist;
ASTROSCI CONSCI EDUC MARITAL POLVIEWS RELIG SEX SPEDUC
```

Macro variable &varlist can now be used in a keep statement or keep = option in a dataset.

If you have a relatively short list of variables you don't want to keep, those variables can be specifically excluded. If for example we wanted to exclude ASTROSCI and CONSCI we can use the following code…

```
proc sql noprint;
  select name
  into : varlist
  separated by ' '
  from cont_out
  where type = 2 and name not in ('ASTROSCI', 'CONSCI');
quit;
```

DICTIONARY TABLES

I used the proc contents method above until I learned that much of the same information is available in dictionary tables. Using dictionary tables is much slicker than using PROC Contents. Using dictionary tables the above code can be simplified to one step. Dictionary tables are special read only PROC SQL views. They contain information about libraries, and dataset, in addition to other SAS information and external files from the current SAS session. When you use dictionary tables to get variable information you need to identify the libname and table name (memname) the variables are in. Data about variables is contained in a table called COLUMNS and for this example we are using the work library. To see what information is available about columns in a dataset the following PROC SQL code can be executed.

```
proc sql ;
  create table dict_col as
  select *
  from dictionary.columns
  where libname = 'WORK' and
        memname = 'SCIENCE_DATA' ;
quit;
```

It can be seen in the table below that we can tell if a variable is numeric or character, the variable position, if it is sorted or indexed and other information from the dictionary table.

	Library Name	Member Name	Member Type	Column Name	Column Type	Column Length	Column Position	Column Number in Table	Column Label	Column Format	Column Informat	Column Index Type	Order in Key Sequence
1	WORK	SCIENCE_DATA	DATA	MARITAL	char	8	8	1					1
2	WORK	SCIENCE_DATA	DATA	POLVIEWS	char	20	16	2					0
3	WORK	SCIENCE_DATA	DATA	RELIG	char	23	36	3					2
4	WORK	SCIENCE_DATA	DATA	CONSCI	char	12	59	4					0
5	WORK	SCIENCE_DATA	DATA	ASTROSCI	char	21	71	5					0
6	WORK	SCIENCE_DATA	DATA	EDUC	char	8	92	6					0
7	WORK	SCIENCE_DATA	DATA	SPEDUC	char	8	100	7					0
8	WORK	SCIENCE_DATA	DATA	SEX	char	8	108	8					0
9	WORK	SCIENCE_DATA	DATA	EDUC2	num	8	0	9					0

Figure 2: selected columns from table dict_col

The following code gives the same results as the first example code using PROC Contents. Instead of using the PROC Contents to create a table of variable names then PROC SQL to create a macro variable, it is all done in one PROC SQL step.

```
proc sql noprint;
  select name
  into : varlist
  separated by ' '
  from dictionary.columns
  where libname = 'WORK' and
        memname = 'SCIENCE_DATA' and
        where type = 'char';
quit;
```

CONCLUSION

This paper demonstrates some techniques to build macro variables of variable names. Techniques like using dictionary tables or the output from PROC Contents are flexible enough to allow these macro variables to be built on the fly (ie they are built while SAS is running). Using the autoexec file you can create macro variables that are always available in your SAS session.

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

Mark Kirkham
Merrick Bank
10705 South Jordan Gateway
South Jordan, Utah, 84095
Work Phone: 801-545-6683
E-mail: mark.kirkham@merrickbank.com