

Creating "Variable Name" Data Sets With CALL VNAME and PROC TRANSPOSE

David Swift, The Psychological Corporation

Introduction

When working with SAS data sets, one might want to create a separate data set containing a variable with values equal to the variable names of another data set. This poster discusses how such "variable name" data sets can be created using the routine CALL VNAME and PROC TRANSPOSE. In addition, the advantages of using each method are discussed. Finally, some important characteristics and uses of "variable name" data sets are illustrated and discussed.

Using CALL VNAME

The most obvious method of creating a "variable name" data set is to use the CALL VNAME routine. Like other SAS routines, CALL VNAME is used within a data step and has the following form:

```
CALL VNAME(variable1,variable2);
```

Where VARIABLE1 is the name of a variable from a source data set in which a variable name is "retrieved". VARIABLE2 is the name of the variable that will contain the variable names of the source data set as its values. Because SAS variable names can have a maximum length of eight characters, VARIABLE2's length is limited to eight characters as well. See SAS Language, Version 6 (SAS Institute, 1990) for documentation on CALL VNAME.

To illustrate the use of CALL VNAME, a source data set (i.e., ONE) is created. As shown in the INPUT statement, data set ONE contains five variables--two numeric and three character.

```
filename datain 'fileref';

data one;
  infile datain;
  input lastname $ firstname $ ssnumber gender $ age;
run;
```

The following statements use CALL VNAME to create a data set containing a variable (i.e., NAME) with two (2) variable names-- LASTNAME and FRSTNAME -- from data set ONE as its values.

```
data varname;
  set one;
  length name $8.;
  call vname (lastname,name);
  output;
  call vname (frstname,name);
  output;
  stop;
  keep name;
run;
```

When outputting several variable names from a source data set, the above method can become quite tedious and is shown here to illustrate CALL VNAME's syntax. For efficiency, CALL VNAME can be used within arrays as the following code displays (See SAS Language, Version 6 (SAS Institute, 1990) for documentation on arrays).

```
data varname;
  set one;
  length name $8.;
  array a (*) _character_;
  array b (*) _numeric_;
  do i=1 to dim(a);
    call vname(a(i),name);
    output;
  end;
  do j=1 to dim(b);
    call vname(b(j),name);
    output;
  end;
  stop;
  keep name;
run;
```

In this example, the variable to be created using CALL VNAME (i.e., NAME) is first specified in a LENGTH statement. The SAS keywords _NUMERIC_ and _CHARACTER_ are used to put all the numeric and character variables into separate arrays. CALL VNAME is placed inside the DO loop and the variable names in the source data set are output to the data set VARNAME as values of the variable NAME. A STOP statement must be used to stop the DATA step after

the first iteration. Without a STOP statement, the data step would loop through once for each observation, creating one list of variable names for each observation in the source data set ONE. Output 1 shows the result of printing the data set VARNAME.

```
proc print noobs data=varname;  
title1 'The VARNAME Data Set';  
run;
```

The VARNAME Data Set

```
NAME  
  
LASTNAME  
FRSTNAME  
GENDER  
SSNUMBER  
AGE
```

Output 1

Using PROC TRANSPOSE

What has been accomplished using CALL VNAME can be replicated using PROC TRANSPOSE. The following code produces a "variable name" data set identical to the VARNAME data set created using the source data set ONE with CALL VNAME.

```
proc transpose data=one(obs=1) name=name  
out=tvarname(keep=name);  
var _character_ _numeric_;  
run;
```

One important element of the transpose method is the use the OBS=1 option. This causes only the first observation to be transposed. Also used is the NAME option, which names the transposed variable names to NAME instead of the default SAS variable name, _NAME_. To maintain the same order of variable names as in the VARNAME data set, the SAS keywords _CHARACTER_ and _NUMERIC_ are used after the VAR statement. Output 2 shows the result of printing the data set TVARNAME.

The TVARNAME Data Set

NAME

LASTNAME

FRSTNAME

GENDER

SSNUMBER

AGE

Output 2

Advantages of PROC TRANSPOSE and CALL VNAME

One advantage of using PROC TRANSPOSE is that it is easier to preserve the order of variable names as they appear in a source data set. This can be done by using the SAS keyword `_ALL_` after the VAR statement. Because arrays can only group variables of the same type (i.e., numeric or character), using CALL VNAME within arrays will mean all numeric and character variables will be reordered and grouped together by variable type. The following code displays how PROC TRANSPOSE can be used to preserve the order of variable names from the source data set ONE. The result of printing the data set TVARNAM2 is displayed in output 3.

```
proc transpose data=one(obs=1) name=name  
  out=tvarnam2(keep=name);  
  var _all_ ;  
run;
```

The TVARNAM2 Data Set

NAME

LASTNAME

FRSTNAME

SSNUMBER

GENDER

AGE

Output 3

Another advantage of PROC TRANSPOSE is efficiency. To compare the two methods, a source data set with fifteen variables and two thousand observations was compared using the same syntax from the two methods shown previously. The comparison showed that PROC TRANSPOSE required .04 CPU seconds to create the "variable name" data set while CALL VNAME required .08 CPU seconds. It should be noted that the number of observations in the source data set does not significantly effect CPU time since both methods only process the first observation of the source data set.

The main advantage of using CALL VNAME is the added flexibility that results from creating a data set in a DATA step. For example, suppose a variable indicating variable type, besides variable name, was wanted. The following code shows how this can be done. Output 4 shows the result of printing the data set VARNAME2.

```
data varname2;
  set one;
  length name $8. type $9.;
  array a (*) _character_;
  array b (*) _numeric_;
  do i=1 to dim(a);
    call vname(a(i),name);
    type='Character';      /*add variable TYPE equal to character*/
  output;
  end;
  do j=1 to dim(b);
    call vname(b(j),name);
    type='Numeric';        /*add variable TYPE equal to Numeric*/
  output;
  end;
stop;
keep name type;           /*keep NAME and TYPE*/
run;
```

The VARNAME2 Data Set

NAME	TYPE
LASTNAME	CHARACTER
FRSTNAME	CHARACTER
GENDER	CHARACTER
SSNUMBER	NUMERIC
AGE	NUMERIC

Output 4

Using "Variable Name" Data Sets

One direct application of a "variable name" data set is to merge it with output in which the variable names were lost or never existed so that the original variable labels appear in the output. "Variable name" data sets can also be used less directly as these data sets have many important characteristics in common with their source data sets. For example, if all variables from a source data set are included in a "variable name" data set, the number of observations will be equal to the number of variables in the source data set. This allows one to have access to the number of variables in the source data set via the automatic variable, `_n_`. The following code displays how to make use of this characteristic of "variable name" data sets by accessing the data set VARNAME previously created.

```
data _null_;  
    set varname end=last;  
    if last then call symput ('varnum',left(_n_));  
run;
```

In the following DATA step, the END=LAST option is used with CALL SYMPUT to create the macro variable VARNUM. In this example VARNUM's value is equal to the number of variable names (i.e., 5) in the data set VARNAME or, what is more important, the number of variables in the source data set ONE. (Refer to SAS Guide to Macro Processing, Version 6, Second Addition (SAS Institute, 1990) for documentation on CALL SYMPUT and macro variables).

The following %LET statement is equivalent to the using CALL SYMPUT in the DATA step presented above.

```
%let varnum=5;
```

Another characteristic of a "variable name" data set is displayed by an additional CALL SYMPUT added to the previous DATA step. This additional CALL SYMPUT creates a macro variable (i.e., NAME1, NAME2, NAME3,...etc.) for each variable name in the source data set.

```
data _null_;  
    set varname end=last;  
    if last then call symput ('varnum',left(_n_));  
    call symput ('name' || left(_n_),name);
```

This additional CALL VNAME is equivalent to using the following %LET statements.

```
%let name1=LASTNAME;  
%let name2=FRSTNAME;  
%let name3=GENDER;  
%let name4=SSNUMBER;  
%let name5=AGE;
```

The advantage of declaring the variable names as macro variables is that they can be used as global variables referencing source data set variables. In addition, because the macro variables have numeric extensions, their values can be "retrieved" in a %DO loop. The following code is an example of how the variables names from a source data set can be "retrieved" in %DO loop.

```
%macro names;  
  %do j=1 %to &varnum;  
    proc print noobs data=ONE;  
      var &&name&j;  
      title1 "The Variable &&name&j From The Data Set ONE";  
    %end;  
  %mend names;  
%names;
```

In the first iteration of the %DO loop &&name&J will first resolve to &name1 and lastly to the variable LASTNAME. In this example, PROC PRINT is being used within the macro to print all variables from the source data set ONE separately. As previously shown, the macro variable VARNUM will resolve to the number of variables in the source data set ONE. Output 5 displays five (5) observations from the first variable (i.e., LASTNAME) being printed within the %DO loop.

The Variable LASTNAME From the Data Set ONE

```
SMITH  
JONES  
LOPEZ  
MOORE  
SIMMONS
```

Output 5

Conclusion

In this poster an emphasis was made on the creation and usefulness of "variable name" data sets. Two methods of creating "variable name" data sets were presented: CALL VNAME and PROC TRANSPOSE. The strengths of each of these methods were discussed. In addition, an example of using "variable name" data sets to reference source data sets and produce output was given. It is hoped that users will have learned how to use CALL VNAME and PROC TRANSPOSE in the creation of "variable name" data sets, and how "variable name" data sets can be used to reference source data sets.

References

SAS Institute, Inc. (1990). SAS Language: Reference, Version 6. Cary, NC: Author.

SAS Institute, Inc. (1990). SAS Guide to Macro Processing: Version 6, Second Addition. Cary, NC: Author.