Paper 322-2013

What's in a SAS® Variable? Get Answers with a V!

William C. Murphy

Howard M. Proskin & Assoc., Inc.

ABSTRACT

If you need information on variable attributes, PROC CONTENTS will provide all of the specifics. You could also access the SQL DICTIONARY which contains tables filled with details on the variables in the active data sets. If you need just one piece of information on a single variable both of these methods could prove to be cumbersome. However, SAS has a whole series of functions that can produce the information on one attribute of one variable at one time. These functions are named with a V prefix followed by descriptive term for the attribute. They include VNAME, VLABEL, VTYPE, VFORMAT and several others. We will demonstrate how these V functions are useful not only in reporting but in standardizing the structure of a database

INTRODUCTION

In our small consulting company, we analyze and report on the results of drug studies. These studies involve dozens, if not hundreds, of SAS data sets, each containing scores of variables. These data sets are usually supplied with a full range of attributes assigned to the variables. When we are analyzing the data, however, we only think in terms of the values that are contained in the variables. But when it comes time for reading in the values from a raw file or outputting the value of variables into a report, all of the attributes of the variables come into play. Such characteristics as the variable name, format, informat, and label need to be known and used in accordance with the study mandates. Of course, to obtain this information a programmer can simply run the CONTENTS PROCEDURE. This procedure will supply you with ever possible detail of any and all variables in the data sets. Usually this data is more desirable in electronic form, so you run it with noprint and out= options. Alternately you could run PROC SQL on the data dictionary COLUMNS or use a DATA step to extract the content of sashelp.vcolumn.

These methods are all fine, but usually we want the information on one attribute of one variable at a time. We could whittle down the data set generated from PROC Contents or PROC SQL to contain only the variable and attribute in question. However, there is a far more eloquent solution that SAS offers through a series of variable functions that start with V. Instead of just listing these functions, we will illustrate a number of them and there possible use.

NAME

In our work, we often receive data that is gradually acquired from a study site over a period of time. After a certain amount of data has been gathered, the study site may ask us what subjects and variables have missing values. To create a data set containing the missing elements in, for example, the physical exam data, we could write

```
data Missing;
    set PhyscialExam;
    array Numbers _numeric_;
    do i=1 to dim(Numbers);
           if missing(number[i]) then output;
           end;
    keep SubjectID  i;
    run;
```

where we have create an array of all the numeric variables and preform a DO-loop to check and see which of these variables contain missing values. The output data set contains the subject information and the variable number of the missing information. However, unless you know what number goes with what variables, this is not going to be very useful. What we want is the name of the variable not its position in the array, so we alter the code to read

```
data Missing;
    set PhyscialExam;
    array Numbers _numeric_;
    do i=1 to dim(Numbers);
           VariableName=vname(numbers[i]);
           if missing(number[i]) then output;
           end;
```

```
        keep SubjectID VariableName;
        run;
```

where we now keep the subject identification and variable name in the output data set. We now have variable names like pulse, height, weight, and temperature, which are far more useful than the variable array positions.

We obtained the variable's name through use of the VNAME function. This function returns the name of a data set variable as a text string. This text string can be manipulate like any other text in the SAS DATA step

The VNAME function is the simplest of the V functions, which supplies information obvious to anyone viewing the data. However, it provides you with a simple route to get the name into the coding process. We have often used this function in the past (Murphy 2006, 2007)

VALUE

Just like we looked for missing data, we could also check for outrageous data. For example, we might check the ages of our subjects for possible issues with the birth date by outputting questionable information to the log:

```
    data Ages;
        set BirthDates;
        ExamDate='04NOV2012'd;
        Age=int((ExamDate-Birthdate)/365.25);
        if Age>100 or Age<1 then put SubjectID= Birthdate=;
        run;
```

Now hopefully, the data is clean and there will be no output in the log file.   However, you look and there is

```
        SubjectID=37        Birthdate=11111
```

You forgot to apply a format to the date and thus making it difficult to interpret. Now you could have put a format on the PUT statement line. What if you had no idea of what format to use? What if the variable you were outputting was some data set variable other than the date and you had no idea what the associated format was? However, this would be no problem if we replaced the if statement with the two lines

```
        textBirthDate=vvalue(Birthdate);
        if Age>100 or Age<1 then put SubjectID= textBirthdate=
```

The log would now contain

```
        SubjectID=37        textBirthdate=11/01/2012
```

We can now easily see the date problem and can take appropriate action.

The VVALUE function takes the value contained in a variable and provides it to another variable with the associated format applied. The VVALUE function provides the same results of a PUT function using the variable's format.

TYPE

One of my pet peeves is when we get a cut of data with different formats than previous cuts. This often occurs for date variables. Sometimes they are numeric variables and sometimes they are character variables. For coding, we must have consistency. We could write separate code to process either form of the data or we could write one to process both forms:

```
     data HewDates;
        set RawDates;

        length date_txt $10 date_num 8.;

        if vtype(StudyDate)='N' then do;
              date_num=StudyDate;
              FinalDate=put(date_num,mmddyy10.);
              end;
        else if vtype(StudyDate)='C' then do;
               date_txt=StudyDate;
               FinalDate=Date_Txt;

        drop date_txt date_num;
        run;
```

where we test to see if the input date (StudyDate) is numeric or text and do the appropriate steps to convert it to text(FinalDate). Here we have made use of the VTYPE function which tells me if a variable contains numeric ('N') or character ('C') variables. This code could be easily change if we desired numeric dates in the final data set.

FORMATS

In any given study, we usually receive data from many possible sources. Often the database provided has items like dates in different formats. For example, a lab might give us a specimen collection date as 11/4/2012; the medical record indicates a birthdate of 4MAY1950, but the clinic reports a physical exam date of 2012-09-01. These are all easily used by SAS, but in the reports, the client usually wants all the dates reported in the same format. When you ask what format should be used the response is often something like use the medical record format. As mentioned before, you could use PROC CONTENTS or the SAS Data Dictionary, or even the Viewtable to look up the format used by the medical records, and then apply this format throughout your report program. What happens if a future cut of the data uses a different format, but you are still instructed to use those records as your standard? You now must go back to your program and retype all of the formats you used. However, instead of going through all of this hassle, you could have written your report program with the lines

```
   CollDate=putn(CollectionDate,vformat(BirthDate));
```

where we have created a new text variable that contain the desired dates with the same format as the BirthDate variable. The VFORMAT function is a routine that supplies the program with the name of the format assigned to the variable contained in the argument.

MORE FORMATS

Usually in our reports, the client wants the same number of decimal points applied to all numbers. For example, we might want 2 decimals displayed but the data for pulse comes with none, the data for body temperature comes with one, and body mass index (BMI) is expressed with 2. We could easily set up a numeric format with 2 decimals and apply it to all the variables. However, if we want to preserve the variable width already applied to the variable but have the same number of decimals as the BMI variable, this might require more effort. However, using V functions, we could solve the problem by writing

```
   data FormatMacros;
       set PhysicalExam;
       Pulse_width=vformatw(Pulse);
       Temperature_width=vformatw(Temperature);
       BMI_decimal=vformatd(BMI);

       Pulse_fmt=cats(Pulse_width,'.',BMI_decimal);
       Temperature_fmt=cats(Temperature,'.',BMI_decimal);

       call symputx('Pulse_fmt', Pulse_fmt);
       call symputx('Temperature_fmt',Temperature_fmt);
       run;
```

where we have used the VFORMATW function to extract the format widths form the Temperature and Pulse variables. We then extract the number of decimals form the BMI variable with VFORMATD. Then we simply concatenate the results with a decimal between to give us formats, which we save in macro variables. These formats can then be applied via use of these macro variables in a later DATA step or with PROC DATASETS:

```
       proc datasets nolist library=work;
              modify PhysicalExam;
              format Pulse &Pulse_fmt  Temperature &Temperature_fmt;
              quit;
```

LABELS

Often we will receive data with some measurement contained in variables Visit1, Visit2, and Visit3. We prefer this data in a more vertical format. Therefore we write

```
proc transpose data=Horizontal out=Vertical;
    by Subjid;
    var visit1 vist2 visit3;
    run;
```

The new data set contains the identification variable Subjid, _name_ which contains the visit variable name, _label_ which contains the visit variable label, and col1 which contains the value of the visit variables. This works fine most of the time, but sometime we are confronted with dozens of identification variables. In addition to the subject identification, we might have to carry along gender, birth data, age, height, race, and many other variables. We may not even now the names of all of these extra variables. Using the BY or ID statements with the PROC TRANSPOSE under these conditions is difficult if not impossible. But we can transpose without PROC TRANSPOSE with V functions:

```
data Vertical;
    set Horizontal;
    array visit{*} v1 v2 v3;
    do i=1 to dim(visit);
           _name_=vname(visit[i]);
           _label_=vlabel(visit[i]);
           col1=visit[i];
           output;
           end;
    drop i v1 v2 v3;
    run;
```

where we have used the VNAME function to acquire the name of the variable and the VLABEL function to acquire the label of the function. The results of this code are identical to that of the PROC TRANSPOSE, except your output will keep not only SubjID but all of the other variables in the original data set.

THE X-FACTOR

It should be pointed out that most of the V functions come in two flavors. The one flavor we have illustrated in the above examples. The other flavor has the same function name with a X suffix. For example the VVALUE function has a corresponding VVALUEX. For the function without the X, the argument must be a variable or array reference. For the function with the X, the argument must resolve to a variable name. For example, we could write

```
    Parameter='Height';
    Height=188;

    NoX_Value=vvalue(Parameter);
    X_Value=vvaluex(Parameter);

    put NoX_Value=      X_Value=;
```

If this code was ran as part of a data step, it would write to the log

```
NoX_Value=Height  X_Value=188;
```

The VVALUE function returned the contents of Parameter (i.e. Height), whereas the VVALUEX function returned the contents of the variable contained in Parameter. Other V functions such as VNAMEX, VLABELX, and VTYPEX, work similarly.

CONCLUSIONS

The SAS variable information functions are a group of routines whose name starts with the letter V. These V functions can provide the programmer with direct access to the attributes of a given function without resorting to PROC CONTENTS or a search of the data dictionary. The V functions provide an easy direct way to provide clarity to outputs and the ability to check and modify the contents of a data step.

REFERENCES

Murphy, William C., 2006, “Squeezing Information out of Data”, *Proceedings of the Thirty-First Annual SAS Users Group International Conference*, SAS Institute Inc., NC. Available at http://www2.sas.com/proceedings/sugi31/028-31.pdf

Murphy, William C., 2007, “Changing Data Set Variables into Macro Variables”, *Proceedings of the SAS® Global Forum 2007 Conference*, SAS Institute Inc., NC. Available at http://www2.sas.com/proceedings/forum2007/050-2007.pdf

CONTACT INFORMATION

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

William C. Murphy
Howard M. Proskin & Associates, Inc.
300 Red Creek Dr., Suite 220
Rochester, NY 14623
Phone 585-359-2420
FAX 585-359-0465
Email wmurphy@hmproskin.com or wcmurphy@usa.net
Web www.hmproskin.com