Paper 241-27

“Haven’t We Met Somewhere Before?” and Other Useful Lines (of code) to Get to Know Their Data

David Rucker, Covance Periapproval Services, Radnor PA.

ABSTRACT

“Know Thy Data”. Every programmer will tell you that this should be our motto. As programmers, we see all types of data, in all forms, on all media, from different operating systems, from different customers, and possibly different languages. We may never agree on the approach taken in our code. We may never agree on programming standards. We may never agree on data presentation. Nonetheless, any user of the SAS® System will tell you this is what he appreciates most about the product. Surely the one thing we can all agree on (excluding the fact that continuous learning is the foundation of growth) is that you MUST know your data. The first step in this new relationship is to introduce yourself.

In this paper, I will discuss several fundamental and possibly some new methods to learn about your data. Most examples provided will be relevant to the pharmaceutical industry in theory; however, these methods are applicable to all types of data. Although this paper is intended for the novice programmer it may provide the more seasoned programmer with new ideas and tools. (Base SAS®, Microsoft Windows NT® operating system).

INTRODUCTION

“Know Thy Data”, was how I was introduced to SAS software two years ago as part of Covance’s partnership with Philadelphia University (formerly Philadelphia College of Textiles). I was preparing to change careers and enter the yet unexplored world of the pharmaceutical industry. This intimidated me because I was not only learning a new programming language, I was also learning a new industry’s language. Having spent 12 years and some odd months of my career in the utility industry as an analyst, I worked with all types of data ranging from engineering, call center, budgeting, to the all encompassing bean counting. Did I “Know My Data”? Sure I did, but 90% of the time I worked with the same data, day-in day-out, just a different month, year, or project. I never thought about the importance that knowing the data had for me. “Knowing My Data” made me successful in my position. It made my work easier, and I knew when things were right and when they appeared wrong. “Knowing Your Data” will also make you successful!

“Know Thy Data”, now seemed like a monumental task to me, and make no mistake, it was a monumental task. My career changed when I accepted a position with Covance Periapproval Services, a Contract Research Organization (CRO). For those of you not in the pharmaceutical industry, a CRO typically functions as a contractor to pharmaceutical companies performing clinical trails.

That’s correct, you read right, that would be plural for company. CROs are the professional jugglers of the pharmaceutical industry, never holding the same flaming torch too long, but always keeping an interest in the torches that are not in hand. And as no two snowflakes are alike, be assured that no two companies, and their data, are alike. Feel free to substitute in place of “company”, other words such as client, manager, internal customers or whatever other word is suitable to your situation. This is what makes the job fun and ever challenging, but it also changes the axiom to know the clients data or ”Know Their Data”.

“Know Their Data”, is now my mantra. I think it is appropriate to say, that as programmers, the data are usually not ours. It may be semantics, but it is more accurate. This paper will present specific examples of methods that I use every day to introduce myself to the data, learn about what I have, and grow comfortable working with this data. I will attempt to provide examples that you can use *as is*, by virtue of the data sets that are already included with Base SAS software. The examples provided are written using SAS software version 8.2 for the Windows NT operating system, but can be easily modified if needed for other versions and platforms. This paper will assume that your data are already in a SAS data set.

Gaining Confidence (or doing your homework)

Depending in what industry you work, your company standards, or your clients requirements, you may have different methods for your initial approach. The following is a short list of accessories you may want to consider before you get started programming. Again this list is primarily related to the pharmaceutical industry; however, translate the description to fit you specific needs:

- Physical Data Model (PDM) - How and where the data are stored, organized, and related to other data?
- Data Entry Conventions - What should be entered into the database?
- Annotated Case Report Forms (CRFs) - Where did the data originate and how was it collected?
- Passwords or other security tools - How can I get access to the data?
- Contact List - Who can I ask questions listed above?

Your SAS Software toolbox

SAS Software Dictionary Tables

I would like to make mention of these tables, and the significant role they will play in getting to know your data. You will be able to learn more about these tables from a SAS paper titled “Data About Data: An introduction to Dictionary Tables” which is always in my toolbox. If you have never taken the time to look at these data sets, I would encourage you to do so. These tables could possibly open up a whole new world of information that you can use in your everyday programming.

Macros

When the chefs at your favorite restaurant makes a meal, they don’t reinvent the recipe each time. No, they pull out the recipe, add the necessary ingredients, follow the procedure, and *voila*! the perfect dish. Every now and then the chefs will get a customer for whom they need to modify the procedure to meet the client’s needs - that’s OK. It’s what the customer wants. Remember, you will be serving a lot of clients; try to generalize your code that will be used often.

Once you have gathered this information you are ready to start looking at the data. We should begin by looking at the contents of the data and its attributes.

PROC CONTENTS® and PROC PRINT® (and alternatives)

PROC CONTENTS and PROC PRINT may be the very first SAS System procedures you will ever learn, and they will be the ones that you will probably always use whenever you need to learn about your data. These procedures can be considered the foundation of getting to “Know Their Data”.

The PROC CONTENTS procedure is like the Bill of Material (a list of parts) for your data sets. The output of this procedure will list the variables in the order of their position in the data set. It is often difficult to find the variable you are looking for in larger data sets. The SAS system option POSITION provides a list of variables in alphabetical order, making it easier to locate a variable by name; however, you will get two lists: one sorted by position and the other in alphabetical order (Reference Example #1).

Example #1 - PROC CONTENTS with POSITION OPTION

```
proc contents data=sashelp._all_  position;
run;
quit;
```

As an alternative, I prefer using PROC DATASETS®, which will automatically give you output similar to PROC CONTENTS® sorted alphabetically by variable name (Reference Example #2).

Example #2 - PROC DATASETS - with or without varnum

```
proc datasets;
  contents data= sashelp._all_  /*varnum*/ ***add varnum to sort
by position;
run;
quit;
```

A third option combines DATA _NULL_® , CALL EXECUTE®, and the SAS software dictionary table VSTABLE to generate a PROC CONTENTS, and in this example, a PROC PRINT®. This code is useful to store in a macro library since this may be a procedure that you would use often. This code can easily be modified to not take advantage of the SAS software dictionary table VSTABLE (Reference Example #3).

Example #3 - Using SASHELP.VSTABLE and CALL EXECUTE

```
%macro pppc(libnamex,numbobs);
data _null_ ;
  set  sashelp.vstable ( where = ( libname = "&libnamex" ) ) ;
  call execute
      ('proc datasets'||';'||
         'contents data = &libnamex..' || memname ||';'||
       'run;'||
       'proc print' || ' data = &libnamex..' || memname || '
(obs=&numbobs);'||
       'run;') ;
run ;
%mend pppc;
%pppc (SASHELP,50);
```

Duplicate Observations (pet peeves of your data)

When getting to know you client's data, you may learn things as a programmer that you don't particularly like. More often than not, duplicate records may fall into this unpopular part of your relationship with their data. By duplicate records, I specifically mean duplicate primary keys, such that each record in the data set can maintain its own unique identity. It is true that many database designs will prevent such data from ever occurring; however, other data sources such as MS Excel spreadsheets will not prevent this possibility.

Getting to know your data in this manner may help later in the process when you analyze their data. It may also help identify data entry problems or other issues that can be caught early in the data collection process.

One of the tools that I use to identify duplicate records is the macro presented in Example #4. In this example, I use the SAS software dictionary table VTABLE to illustrate the point. The VTABLE data set contains information concerning the libraries and data sets. You would expect to see many data sets within each library; you may even see data sets located in different libraries to be named similarly, but you would never expect to see a duplicate data set within the same library.

Example #4 - Finding Duplicate Primary Keys in your data

```
%macro dups(lib,dset,keys);
proc freq data=&lib..&&dset;
  table &keys / list noprint out=&dset.A  (where=(count>=2));
run;

data _null_;
  set sashelp.vtable;
  where libname="WORK" and memname="&dset.A";
  call symput ('obsnum',trim(left(nobs)));
run;

%if &obsnum eq 0 %then %do;
data _null_;
  file print;
  put @1 "There are 0 duplicate records in Library: &lib, Data set:
&dset for keys: &keys ";
run;
%end;

%else %do;
proc print data=&dset.A;
  title1"The following records have Duplicate Keys in Library: &lib,
Data set: &dset";
run;
%end;
%mend dups;
%dups(SASHELP,VTABLE,LIBNAME);
%dups(SASHELP,VTABLE,MEMNAME);
%dups(SASHELP,VTABLE,LIBNAME*MEMNAME);
```

Variable Matrix

Up to this point, we have looked at our data from a single data set view, but most data sets are not used in isolation - they are used with other data sets. Example #5 presents an example of looking at the relationship that the data from one data set will have with the data from another data set. In this example, we are looking at this data within the same LIBRARY. For the purpose of illustration, I wanted to look at only the data sets that start with a 'V' and are in the SASHELP Library. This code will produce a matrix of variables, itemizing the data type, length, and flag the data set which contains the variable.

The matrix is quite useful later when you need to merge data sets. It provides information on how you may need to handle variables, which ones need to be renamed, kept, or dropped; when you need to change the variable type from character to numeric or numeric to character; and when the lengths of like named variables are different. Example #5, when executed, will allow you to notice that there are several variable names such as LEVEL, OBJNAME, TEXT, and TYPE in which one or more of their attributes differ; these features are printed in the second part of this code. Refer to Figure #1 for a sample of the output.

CAUTION should be exercised when using this method in that the assumption is not made that simply because a variable name is the same in two different data sets that the data are comparable and visa versa.

Example #5

```
data vtable (keep = libname name memname type length
counter);
  set sashelp.vcolumn;
  if libname='SASHELP' and substr(memname,1,1) eq 'V' ;
  counter="X";
  name = upcase(name);
run;

proc sort data=vtable;
  by libname name type length memname;
run;
```

```
proc transpose data=vtable out=final (rename=(name=variable)
drop = libname _name_ );
  by libname name type length;
  id memname;
  var counter;
run;

proc print;
  title1 'Variable to Table Matrix - All Variables';
run;

proc freq data=final noprint;
    tables variable / list out=varcount (drop =  percent
where=(count>=2));
run;

data final2 (rename=(name=variable) drop = _name_);
merge varcount (in=ina)
      final    (in=inb);
  by variable;
  if ina;
run;

proc print data=final2;
  title1 "Variable to Table Matrix for Exact Variable names with
different attributes";
run;
```

Figure #1 (sample of output)

```
                                 v   v   v   v   v   v
                                 c   i   o   s   m   t
                                 o   d   p   v   a   i
                                 l   m   t   i   c   t
                                 u   s   i   e   r   l
                                 m   g   o   w   o   e
variable      type     length    n       n
...
LENGTH        num           8    X
LEVEL         char          1        X
LEVEL         char          8            X
LIBNAME       char          8    X
MEMNAME       char         32    X           X
TEXT          char        200        X
TEXT          char        256                    X
TYPE          char          1                        X
TYPE          char          4    X
```

Alternative Example #5, modifies the code such that the variable attributes now become the values listed under the table name and only one record per variable exists. Although not illustrated in this example, one could create an array to identify instances where the values for any given variables are not consistent across tables. Reference Figure #2 for a sample of the output.

Alternative Example #5

```
data vtable (keep = libname name memname type length
counter);
  set sashelp.vcolumn;
  if libname='SASHELP' and substr(memname,1,1) eq 'V' ;
  name = upcase(name);
  if type = 'char' then counter = compress('$'||trim(left(length))||'.');
    else if type = 'num' then counter = compress(length||'.');
run;

proc sort data=vtable;
  by libname name memname;
run;

proc transpose data=vtable out=finala (rename=(name=variable)
drop = libname _name_ );
  by libname name;
  id memname;
  var counter;
run;

proc print;
  title1 'Variable to Table Matrix - All Variables';
run;
```

Figure #2 (sample of output)

```
                v       v       v       v       v       v
                c       i       o       s       m       t
                o       d       p       v       a       i
                l       m       t       i       c       t
                u       s       i       e       r       l
                m       g       o       w       o       e
variable        n               n
...
LENGTH          8
LEVEL                  $1
LEVEL                          $8
LIBNAME        $8
MEMNAME        $32                     $32
TEXT                   $200
TEXT                                           $256
TYPE                                                   $1
TYPE           $4
```

Patient Matrix

Now that we have reviewed the data as a single data set, and variable names across data sets, the next logical step is to look at specific variables across multiple data sets.

In the pharmaceutical industry, we typically use investigators (doctors) and patients as identifiers of a unit of observation from which we will analyze the data. This may not always be the case. Other industries will have a similar identifiers such as customer account or units of widgets.

The proper functioning of the code presented in Example #6 will depend on how your data are collected and stored. The pharmaceutical industry collects data for clinical trials on what are commonly known as Case Report Forms (CRFs), which are comparable to a survey questionnaire used to collect data. This information can be collected and stored many different ways; however, the SAS programmer/analyst will often be presented with the data in table form (observations and variables).

Some of the questions you will want to know about your data might be, “How many patients are in the database?” or “Do we have all the information in the database for a sample of patients?”. To be able to answer these questions, the patient matrix code in Example #6 will be helpful.

The purpose of the matrix is to identify all patients across all the tables of interest. The following macro will again take advantage of the SAS software dictionary tables to help us build this matrix. This macro will provide you with a cross sectional view of your data. In this example, a patient number is used as a primary key to join multiple tables. This macro will provide the user with a count of the number of observations each patient has in the selected data sets. This information is helpful during the data cleaning process or analysis programming, especially in cases where your data are not available all at once, therefore, providing you with some insight as to the results you should expect when you start working with your data.

At the end of this code, I export a file to a Microsoft Access® database, which can then be used for review by others in your organization or clients who may not have direct access to the SAS System. You will need to modify this code to meet your needs, but the essential information is contained.

Example #6

```
*************************************************************************;
*** MACRO PATIENTS (PATIENT MATRIX)
*** DIRECTRY  => Default is work (temporary) alter to create
***                      permanent directory.
*** PERMFILE  => This will be the data set name whether
***                      temporary or permanent.
*** TABLEVAR  => This will be the variable(s) which is common
***                       to all tables
***    TABLES    => List of tables / data sets to be included
*************************************************************************;
%macro patients(directry=work,
                              permfile=patlist,
                              tablevar=,
                              tabfreq=,
                              tables=,
                              conditon=);

*****************************************************;
*** set a number higher than the # of tables
*****************************************************;
%do I= 1 %to 999;
  %let dsn = %scan(&tables,&i);

*****************************************************;
*** When all data sets have been processed
*** the value of &dsn will be null and the IF
*** statement false, which will stopping loop
*****************************************************;
           %if &dsn ne %then
           %do;

*****************************************************;
*** n will equal the total number of tables.
*****************************************************;
                 %let n = &i;

*****************************************************;
*** sort each table by common variable(s)
*****************************************************;
                 proc sort data=sasdata.&&dsn
                      (keep=&tablevar /*verified*/) out=table&i;
                    by &tablevar;
                 run;

*****************************************************;
*** The where step is added to eliminate
*** records which you may want to exclude
*****************************************************;
                 proc freq data=table&i;
                 &conditon;
                 table &tabfreq / out=table&i noprint nocum nopercent;
                 run;
      %end;
      %else %let I=999;
%end;

*****************************************************;
*** Merge all data sets together. Rename
*** count variables to a table# - to assist with
*** counting tables later.
*** end=last finishes the merge statement
*** Establish variables to count total number
*** of tables a patient exists AND if the
*** patient is in all tables.
*** Rename all variables table# to the proper
*** table name.
*****************************************************;
data &directry..&&permfile;
merge
  %do i=1 %to &n;
     table&i (drop=percent rename=(count=table&i))
  %end;
  end=last;
  by &tablevar;
  tables = n(of table1-table&n);
  if tables = &n then all='Y';
  else all='N';
  %do i=1 %to &n;
       label table&i = "Number of obs for patient in "
       %upcase(%scan(&tables,&i));
  %end;
  %do i=1 %to &n;
       rename table&i=%scan(&tables,&i);
  %end;
        label tables = "Number of Tables Patient exists"
        all     = "Is Patient in all tables?";
run;
%mend patients;

***********************************************************************;
*** Prior to calling the macro, ensure that all data sets to be
*** include have their key variables named the same and have
*** the same attributes.
***********************************************************************;
data sasdata.saelog;
 set sasdata.saelog;
 kitno = site;
 ptno = subject;
run;

*************************************************************************;
*** ACTUAL MACRO call.
*************************************************************************;
%patients(directry=supp,
                 permfile=patlist,
                 tablevar=kitno ptno,
                 tabfreq= kitno*ptno,
                 tables=demomast
                              enroll
                              critmast
                              illness
                              vitmast,
         conditon=);

*************************************************************************;
*** Export data to MS Access for DM review
*************************************************************************;
PROC EXPORT DATA= SUPP.PATLIST
          OUTTABLE= "tblPat_matrix"
          DBMS=ACCESS97 replace;
          DATABASE="c:\temp\tables.mdb";
RUN;
```

Figure #3 (sample of output)

Invetigator	PatientID	demog	visits	lab	aes	conmeds	vitals	consent	tables
0001	001	X	X	X	X		X	X	6
0001	002	X		X		X	X	X	5
0001	003	X		X		X	X	X	5
0002	001	X		X			X	X	4
0003	001		X	X	X		X		4
0003	002	X		X			X	X	4
0004	001	X					X	X	3
0004x	001		X						1
0005	000	X							1
0005	001		X	X	X	X	X	X	6
0005	002	X	X	X	X	X	X	X	7

CONCLUSION

You have been assigned a new project. You gather all the documentation you can about your data. You prepare to meet their data for the first time. You ask questions to gain insight into your new endeavor. You're getting butterflies, but you're finally ready to introduce yourself. You turn on your computer, access their data and look deep into the pixels of your monitor, asking your data "Haven't we met somewhere before?". But no, you don't need this line anymore because you now have other lines, lines of code which will be much more effective towards the goal of getting to know your data.

As is always the case with the SAS System, there are many ways to accomplish the same goal. The examples presented here are only the beginning. Other simple steps to getting to know their data can also be accomplished using PROC FREQ®, PROC UNIVIARIATE®, and PROC TRANSPOSE®.

By the way, if you do hear their data answering when you ask the question "Haven't we met somewhere before?", you may want to seriously consider taking some time off and taking a nice relaxing vacation!

REFERENCES

SAS Institute Inc. SAS Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990.

SAS Institute Inc. SAS Procedures Guide: Reference, Version 6, Third Edition, Cary, NC: SAS Institute Inc., 1990.

The output for this paper was generated using BASE SAS software, Version 8.2 of the SAS System for Windows NT. Copyright © 1999 - 2001 SAS Institute Inc. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, USA

SAS Institute Inc., Proceedings of the Twenty-First Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc., 1996. "Data About Data: An introduction to Dictionary Tables"; Frank C. Dilorio, ASG, Inc., Cary, NC Nancy J. Michal, SAS Institute Inc., Cary NC

ACKNOWLEDGMENTS

I would first like to thank my family and friends for their support during my transition to my new career. I would also like to thank my past co-workers Lloyd Yates, Steven Needham, and Maryrose Sheppard for giving me opportunities to explore my interests and inspire me to follow them. I would also like to thank all my co-workers at Covance Radnor who have shared their knowledge and experience, with great thanks to James Copp and Jim Grudzinski for giving me the opportunity to pursue my interests. I would like to thank my clients for making my work challenging and ultimately rewarding. I need to also thank my office mate Paul Stafiniack for showing me the ropes and being there to bounce ideas around. I would like to thank Mary Whitman for her assistance with writting technique, I owe you a lemon cake. And lastly, I have to give a special thanks to the two people who made everything I have accomplished in my new career a reality, Ellen Brookstein and Jim Johnson. You two have changed many lives with the program that they coordinate with the Philadelphia University and SAS Institute, and I am forever thankful that I was one of those you changed. Thank You!

CONTACT INFORMATION

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

David Rucker
Covance Periapproval Services
One Radnor Corporate Center
Radnor, PA. 19087
Phone: 610-971-3485
Fax: 610-971-9755
email: david.rucker@covance.com