Paper 224-25

Everyone Wants Their Reports in a Spreadsheet: A SAS® Macro for Getting There

William C. Murphy
Rochester Community Individual Practice Association, Inc., Rochester, NY

ABSTRACT

Have you ever worked hard to produce a good printed report, only to find out that your client wanted to rework the numbers? When I produce *ad hoc* reports from a medical claims database, I am constantly confronted with this problem. Everyone has ideas of appropriate number groupings and report styles. Using the dynamic data exchange ability of a Microsoft Windows NT environment, I developed a SAS macro to solve this problem. The macro transfers a SAS data set into a Microsoft Excel spreadsheet. This method allows the total control of your data that the SAS system offers, but provides your client with a report that they can customize. The exact construction of this macro, along with its limitations and uses, will be detailed.

BACKGROUND

Rochester Community Individual Practice Association is a large organization composed of over 2,000 physicians and other healthcare providers. We act as a contracting agent for these doctors with a local HMO and we also offer our own health insurance products. We have about 100,000 covered lives for which we receive the claims records for each month. These data, combined with various supporting medical files and our physicians' list, are read into the SAS system to create our healthcare database. From this database, a variety of reports with a standard form and format are generated every month. Supplementing these is a large number of reports requested to fill a particular need. The form of these reports is variable and is often used only once.

INTRODUCTION

When an *ad hoc* report is requested from our claims database, I would diligently use PROC PRINT, PROC TABULATE, and/or PROC REPORT to produce the requested output. I would apply formats and titles and pay careful attention to the order of the variables. I would then contribute to the defoliation of the planet by printing hundreds of pages of these reports. When I presented the report to those requesting it, they would say it looked great and had everything that they wanted. "But could you put the data in a spreadsheet, so that we could play with it?"

Of course, I immediately transferred the data to a spreadsheet and my clients would proceed to rearrange the columns and rows of data. They would introduce their own formats and labels and then print out their own version of the report. Needless to say, my report was discarded. After this happened a hundred times or so, it finally dawned on me that I should forget printing out any report and just give my client a spreadsheet with minimal formatting to do with as they will.

INTO THE SPREADSHEET

Being a simple person, the first method that I used to get to a spreadsheet was to use PROC PRINT that was routed to a file with a PRINTTO. I would then run the spreadsheet program and load the file as a text files through the spreadsheets conversion utility. However, this was definitely a manual process and it sometime produces undesirable results. Spaces in text variables and missing variables posed a problem with the conversion process.

My next attempt was to use a DATA STEP with lots of PUT statements to output the data. Between each variable I would put a delimiter. This solved the problem with the blanks and the missing data, but it still wasn't automated.

Next I attempted to use ODBC. I found setting up the drives very tedious and anti-intuitive. I finally gave up on the effort.

SOLUTION

Finally, I attempted to use the Dynamic Data Exchange (DDE). I started up my spreadsheet, in this case Microsoft Excel and then I went to the SAS Editor. There I defined a FILENAME statement with the DDE option pointing to the spreadsheet. Then in a DATA STEP I would use several PUT statements to write my data to the spreadsheet. This method still takes a lot of coding, especially writing all of those put statements. I realized, however, that all of the data requests were for a simple table of the variables in my SAS data set. In short, my clients wanted an image of my SAS data set in a spreadsheet. I could add a header with some title statements and label each column with the variable name. This could be readily implemented in a SAS Macro.

MACRO CREATION

In order to dump a SAS data set into a spreadsheet, we must first determine the variables contained in the data set. Then we must decide on the structure and size of the spreadsheet. Following these tasks, we set up the DDE connection and then write our data to the spreadsheet. Lastly we do some macro housekeeping.

Variable List

We use PROC CONTENTS with a NOPRINT option to create a data set that contains a list of the variables that we wish to send to the spreadsheet:

```
proc contents data=&data
  noprint
  out=_______2(keep=name varnum nobs length);
```

where we have kept the length and nobs for use in determining the size of the spreadsheet. The input data set, &data, is chosen by the macro and the output data set name is chosen to minimize the chance of conflict with another data set. Then we sort the variables so that the columns we form are in the same order as in the SAS dataset:

```
proc sort data=_______2;
  by varnum;
```

SPREADSHEET STRUCTURE

Now we must write some code in order to store the variable names in macro variables and determine the dimensions of our spreadsheet:

```
data _null_;
  set _______2 end=over;
  sumlrecl+20+max(length,length(name));
  call symput(compress('vrbl' | | varnum),compress(name));
  if over then do;
    call symput('varcnt',varnum);
    call symput('obscnt',nobs+10);
    call symput('xlrecl',sumlrecl);
    end;
  run;
```

The first SYMPUT call places the names of each of our variables in the macro variables &vrbl1, &vrbl2,... We now estimate the logical record length, LRECL, by adding up the length of each variable (or the length of the variable name if it is larger), plus a little bit more for safety. It should be noted that, if you replace a small variable with a lengthy format description, you will need to adjust this value so that there is enough room for the format description. Finally we save the observation count (with a safety margin) and the variable count for use in determining the row and columns of the spreadsheet.

DDE CONNECTION

Next we write our FILENAME statement:

```
 filename spread dde
"Excel | [Book1]&sheet!r1c1:r%left(&obscnt.c%left(&varcnt))"
notab;
```

where 'dde' evokes the SAS dynamic data exchange engine. The statement in double quotes is the Excel dde triplet. It supplies the name of the book ('Book1) and the sheet within the book ('&sheet), followed by row and column specifications. We extend our spreadsheet from row '1' to row '&obscnt' and from column 1 to column '&varcnt'. The 'notabs' option prevents the SAS software from inserting tab delimiters. We will put these in ourselves to prevent the problems that occur with missing data and blanks in text strings.

WRITING DATA

Finally we feed the data from the SAS system into the spreadsheet:

```
data _null_;
  set &data;
  delim='09'x;
  file spread lrecl=&xlrecl;
  if _n_=1 then
    put "Rochester Community Individual Practice
Association" /
     %if "&title1" ne "NONE" %then "&title1" /;
     %if "&title2" ne "NONE" %then "&title2" /;
        /
     "&vrbl1"
     %do i=2 %to &varcnt;
       delim "&&vrbl&i"
       %end;
     ;
  put
    &vrbl1
    %do i=2 %to &varcnt;
      %str(delim) &&vrbl&i
      %end;
    ;
```

For the first pass through the data set (_n_=1), we print up to 3 title lines to the spreadsheet: the company's name which is always used and two optional user supplied lines. Next we send a blank line to the spreadsheet, followed by the variable name, which is placed in the appropriate column. The final put statement sends the variable values out to the spreadsheet separated by a delimiter. For us the delimiter is the TAB character, which is '09' in hexadecimal.

CLEAN UP

When we finish with the data writing, we dispose of our data set containing the variable list:

```
proc datasets library=work nolist;
  delete _______2 ;
  quit;
```

INVOCATION

To invoke this macro we write:

%wrtxcel(data= *optional*, sheet= *optional*, title1= *optional*, title2= *optional*);

All the arguments of the macro are optional. If you do not supply any of these variables, it will use the last data set to fill in Sheet1 of Book1 and it will have no title in the sheet other than the company's name.

IMPROVEMENTS

For our company's use, this macro is more than sufficient for almost all SAS data set to Excel tasks. Since we are always creating new spreadsheets, the Excel default of Book1 without a folder specification is ample for our needs. However, the macro could readily be extended to use any spreadsheet name in any folder by introducing 2 more macro parameters into the %MACRO statement. We could introduce &book for the name of the spreadsheet and assign to it a default of Book1. We could also introduce &path to specify the location of the spreadsheet Then we would make the appropriate changes in the FILENAME statement, thus enabling us to write to any spreadsheet in any

folder.

One other improvement that could readily be implemented is the use of variable LABELS for the column headers, rather than the variable names. When the PROC CONTENTS is preformed, you need to retain these LABELS. In the first null DATA STEP, you could store these LABELS into macro variables use the SYMPUT statement. In the output null DATA STEP you could then introduce the logic to put the LABELS rather then the names into the spreadsheet. You could even include another parameter in the %MACRO statement to act as a choice selector: LABELS or names.

CONCLUSIONS

We have developed a very simple and easy way to get from a SAS data set into a spreadsheet. Because we use a macro design, we avoid having to think about the various spreadsheet connection chores. Furthermore, since we are writing an image of our SAS data set to the spreadsheet, we can readily use the power of the SAS system to organize, format, and summarize our data before we send it to the spreadsheet. In fact other programmers' in my company use this macro regularly. Consequently, it was compiled into and used from our central SAS macro library (Murphy, 1998).

APPENDIX

The following is a listing of the macro that is presently in use.

```
%macro wrtxcel(data=_last_,sheet=Sheet1,title1=NONE,
title2=NONE);
%*
╔═══════════════════════════════════════════════════════╗
║ Project:     SAS TO EXCEL MACRO                       ║
║                                                       ║
║ Programmer:  WILLIAM C. MURPHY                        ║
║ Date:  FEBRUARY 2, 1999                               ║
║                                                       ║
║ Objectives:  WRITE A SAS DATA SET TO AN MS-EXCEL      ║
║              SPREADSHEET, WITH THE DATA IN THE        ║
║              SAME ORDER AS IN THE SASDATA SET,        ║
║              WITH VARIABLE NAMES AS THE COLUMN        ║
║              HEADINGS.                                ║
╚═══════════════════════════════════════════════════════╝;
 %local varnum obscnt xlrecl;

 proc contents data=&data
  noprint
  out=_______2(keep=name varnum nobs length);

 proc sort data=_______2;
  by varnum;

data _null_;
  set _______2 end=over;
  sumlrecl+20+max(length,length(name));
  call symput(compress('vrbl' | | varnum),compress(name));
  if over then do;
    call symput('varcnt',varnum);
    call symput('obscnt',nobs+10);
    call symput('xlrecl',sumlrecl);
    end;
  run;

filename spread dde
"Excel | [Book1]&sheet!r1c1:r%left(&obscnt.c%left(&varcnt))"
notab;

  data _null_;
     set &data;
     delim='09'x;
      file spread lrecl=&xlrecl;
   if _n_=1 then
     put "Rochester Community Individual Practice
Association" /
      %if "&title1" ne "NONE" %then "&title1" /;
      %if "&title2" ne "NONE" %then "&title2" /;
         /
      "&vrbl1"
      %do i=2 %to &varcnt;
        delim "&&vrbl&i"
        %end;
      ;
put
      &vrbl1
      %do i=2 %to &varcnt;
        %str(delim) &&vrbl&i
        %end;
      ;

  proc datasets library=work nolist;
   delete _______2 ;
   quit;
   run;

  %mend;
```

REFERENCES

Murphy, W. C. (1998), "Creating and Maintaining a Central SAS® Library for Health Care Management," *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*, 23, 1128-1130.

ACKNOWLEDGEMENTS

CONTACT INFORMATION

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

William C. Murphy
Howard M. Proskin & Associates, Inc.
2468 E. Henrietta Rd.
Rochester, NY 14623
Phone 716-359-2420
E-mail wcmurphy@usa.net