SESUG2011 – Paper BB03

PIPE Dreams: Yet Another Tool for Dynamic Programming

Scott Burroughs, GlaxoSmithKline, Research Triangle Park, NC

ABSTRACT

Statisticians are often divided into Bayesians and Frequentists when it comes to study design and analysis beliefs. As a SAS programmer, you could put me in the Dynamic camp. This is my 5th presentation at a SUG, and all have had something to do with dynamic programming.

Dynamic programming is letting the ever-changing and often unknown data drive the results….no hardcoding! There is a dynamic tool that I've used when I needed to read in data sets from a certain directory where I didn't necessarily know the names of the data sets nor how many there were. The PIPE command in SAS is a tool to read in data sets from a directory when the names and quantity are unknown and changing.

INTRODUCTION

To many, PROC CONTENTS is simply a procedure to list the data sets, variables, formats, etc. for a library. Same with PROC FORMAT….to many, it's just a way to create formats. To me, they've always been much more. Both procedures have output data sets that can be very powerful in their use. For years, I've used the output data set from PROC CONTENTS to get the names of the data sets in a directory and create macro variables from them. Along with the number of data sets, I then use these data set name macro variables to loop through doing the same process for all data sets in a directory, vs calling each process separately for each data set and having to separately name (type) each data set.

Here would be my typical code for doing that:

```
libname daytuh '~/data';

**** procedure to print 20 obs of every data set ****;
%MACRO printt(sett);
proc print data=daytuh.&sett(obs=20);
  title "&ptid Data Set &sett";
run;
%MEND;

**** procedure to loop %printt for all data sets ****;
%MACRO doit;
%do i=1 %to &dimvar;
  %printt(&&v&i);
%end;
%MEND;

** macro I've used for years to convert data set names to macro variables **;
** could just as easily use PROC SQL code on the x2 data set **;
%MACRO mac_var(ds=);
data _null_;
  set &ds;
  if _n_=1 then do;
    array char{*} _character_;
    length name $ 8 ;
    do i=1 to dim(char);
      call vname(char{i},name);
      call symput('v'||compress(put(i,3.)),name );
      output;
    end;
    call symput('dimvar',i-1); ** &dimvar is no. of data sets in dir **;
  end;
```

```
%MEND;

**** final macro to put it all together ****;
%MACRO kontence(ptid=);
** do PROC CONTENTS for each data set, but also output data set names **;
proc contents
    data = daytuh._all_
    out  = x    (keep=memname)
    ;
  title "Contents of &ptid Data Sets";
run;

** get one obs for each data set name **;
proc sort
    data = x (where = (memname^=' '))
    out  = x2
    nodupkey
    ;
 by memname;
run;

** make horizontal data set for %mac_var macro **;
proc transpose
    data = x2
    out  = x3(drop=_name_ _label_)
    ;
  var memname;
  id memname;
run;

%mac_var(ds=x3) ;
run;

** loop through PROC PRINTs **;
%doit;
%MEND;

** run for NEWSTUDY **;
%kontence(ptid=newstudy);
```

Now, I'm just looping a PROC PRINT for each data set, but any type of code could be looped for all of the data sets. Additional code could also be built in to handle exceptions for certain data sets.

Need PROC CONTENTS-like Functionality

The limitation to PROC CONTENTS is that it only reads SAS data sets. What if I want to find the names of other types of files in a directory to use as variables (or macro variables)? What if there's a mix of file types in a directory and you only want one kind? That is the purpose of this paper.

METHODS

A couple years ago, I was on a project where I had to recreate analyses from scratch from the data sets provided, which were transport files. Not only that, but the .xpt files weren't the only files in that particular directory. With a lot of files to deal with, I wanted to do a similar PROC CONTENTS/PROC PRINT looping exercise as above, after converting the .xpt files to SAS data sets. But how do I dynamically read in just the .xpt files from this directory, without typing each one manually? This is where the PIPE command comes in.

Instead of using a LIBNAME statement, we will use the FILENAME statement, and the PIPE command to 'pipe' the results of a UNIX command (our platform) to the particular *fileref*. Since I'm able to use a UNIX command, I'm also able to use wildcards, which act similar to the DATA=*libref*._ALL_ in a PROC CONTENTS statement. Now, with the ability to pipe output to a filename using UNIX commands, the

options are fairly large. What I want is just a simple listing of files, one line at a time. In UNIX, ls -1 provides a listing of a directory, with one file per line. The final command looks as such:

```
filename xptfiles pipe "ls -1 ~/*.xpt" ;
```

where xptfiles is the *fileref* . Notice the UNIX command is surrounded by quotes. The *.xpt denotes listing all .xpt files in the given directory. The command output looks as such:

```
~/adverse.xpt
~/comments.xpt
~/demow.xpt
~/diaghist.xpt
~/ecg2.xpt
~/etrack.xpt
~/etrack2.xpt
~/global2.xpt
~/holter.xpt
~/holter2.xpt
~/holter3.xpt
~/labtest.xpt
~/medhist.xpt
~/neurexam.xpt
~/nsmed.xpt
~/othtrt.xpt
~/physexam.xpt
~/seisures.xpt
~/smedw2.xpt
~/term.xpt
~/vittest.xpt
```

And this output is piped to the filename 'xptfiles'. A simple data step can read this file and output the .xpt file name as the name for the SAS data set it will become later. Then I can loop through the data set names as in the program described in the Introduction.

```
*** read string 'fullname' from filename 'xptfiles' ***;
*** get .xpt file name as variable 'dsn' ***;
data z;
  length fullname $200 dsn $40;
  infile xptfiles;
  input fullname;
  dsn=reverse(scan(reverse(fullname),2,'./'));
run;
```

Data set z goes into the PROC TRANSPOSE above as in the %kontence macro:

```
** make horizontal data set for %mac_var macro **;
proc transpose
    data = z (keep = dsn)
    out  = x3(drop = _label_)
    ;
  var dsn;
  id dsn;
run;

%mac_var(ds=x3) ;
run;
```

Now, instead of macro %printt above to loop through for each data set, I will use the %lewp macro, which will contain code to first convert the .xpt files to SAS data sets, then do a PROC CONTENTS and PROC PRINT for each set.

```
%MACRO lewp(ptid,sett);
```

```
*** 'xport' needed to read as XPT file ***;
libname xp xport ~/&sett..xpt";

*** converts .xpt files to SAS data sets ***;
proc copy in=xp out=work;
run;

*** just the same as from %kontence, but one data set at a time ***;
proc contents
    data = work.&sett
    out  = &sett    (keep=memname memlabel name label format type)
    ;
  title "Contents of &ptid Data Sets";
run;

*** PROC PRINT of 20 observations ***;
proc print data=work.c&sett(obs=20);
  title "&ptid Transport Data Set &sett";
run;

%MEND;

%MACRO doit;
%do i=1 %to &dimvar;
  %lewp(NEWSTUDY,&&v&i);
%end;
%MEND;

%doit;
```

Building Exceptions In

As mentioned above, one could build exceptions into this functionality, if you don't want to do the same process for all of the data sets you're using (or .xpt files you've converted), just a subset of them.

A use for building in exceptions to the above process came along not long after the previous task, so my mind was still on using the PIPE command for doing such things. However, in the next task, I was going to be dealing with SAS data sets only, so I could have used PROC CONTENTS similar to what I'd done for years (in the Introduction) for reading in a variable and unknown set of data set names. But I must have been smokin' the PIPE at the time, so I used that. Instead of a PROC CONTENTS, a PROC SORT, and one more step to create macro variables of the data set names, I'm doing one DATA step w/the FILENAME statement.

The task at hand was converting old studies' data, which was in the form of a previous set of standards, to our new set of standards, so we can build an updated data repository for our HIV studies. I noticed that not all of the old studies' data had the exact same variables and/or formats, so doing the same code for all of the studies, looped in a macro by data set name, wasn't going to work straight up.

```
%MACRO allprots(dir    = ,  /* directory where data sets are located */
                mac_nm = ,  /* macro called for each data set        */
                except = %str() /* data set (study) exceptions not processed */
               );
filename lsinfo pipe "ls ~/&dir./*.sas7bdat"; * get SAS (V7 UNIX) data sets *;

*** take apart &except into separate variables ***;
data w;
  length x $ 9;
  txt="&except";
  i=0;
  do until(x='');
    i+1;
    x=scan(txt,i,',');
```

```
    call symput('EXC'||compress(i),"'"||compress(x)||"'");
  end;
  call symput('exccnt',compress(i-1));
run;

*** put together as macro variable string, with single quotes ***;
*** could have used SQL ***;
data _null_;
  length temp $ 200;
  temp="&exc1";
  %if &exccnt>1 %then %do;
    %do i=2 %to &exccnt;
      temp=compress(put(temp,$200.))||','||"&&exc&i";
    %end;
  %end;
  call symput('EXCEPT_',temp);
run;

*** read in PIPED file lsinfo ***;
data setz;
  retain cnt 0;
  infile lsinfo pad end=last;
  input string $85.;
  dsn=reverse(substr(reverse(scan(compress(string),8,'/')),10));
  if ^(dsn in(&except_)) then do; *** don't do for these exceptions ***;
    cnt+1;
    call symput('Z'||compress(cnt),compress(dsn));
    if last then call symput('ndsn',compress(cnt));
  end;
run;

*** loop through macro &mac_nm for all data sets except exceptions ***;
%do i=1 %to &ndsn;
  %&&mac_nm.(&&z&i)
%end;

%MEND;
```

Call %allprots on POP directory, looping through %convert macro, but not for studies abc123456 and def789012.

```
%allprots(dir=pop, mac_nm=convert,except=%str(abcl23456,def789012));
```

Other Uses

Many types of UNIX commands could be used with the PIPE command in the FILENAME statement to get other types of data such as document metadata (date, size, title, etc.).

CONCLUSION

In the end, I didn't need to know how many .xpt files there were in the directory, I didn't need to know the names of the files, just as I didn't need only .xpt files in the directory to do what I wanted. These names and quantities could change from day to day or from directory to directory. For a new study/directory, I would only need to change the directory in the libname or filename statements and the name of the study in the %lewp macro.

REFERENCES

CONTACT INFORMATION

BIOGRAPHY

Scott was a statistician for GlaxoSmithKline for almost 12 years until switching to a full-time programmer role in 2006. He has worked in Research Triangle Park, NC since 1994. He has programmed in SAS extensively since 1992 while at a previous pharmaceutical company. He has a B.S. and an M.S. in Statistics from Virginia Tech.

CONTACT

Scott Burroughs
GlaxoSmithKline
17.1243M
5 Moore Drive
Research Triangle Park, NC 27709
scott.m.burroughs@gsk.com