Paper PP08

Which programs use that dataset?

Helen Nicholson, Cmed, Horsham, England

ABSTRACT

How many times do you want to quickly know just which programs use a certain dataset or variable, or you know you have done something similar in one of your SAS® programs before but which one is it? Or maybe you would like to do one final check on the program log files to check there is nothing in them that should not be there. You already have SAS and are familiar with it so how can you use it to answer these questions?

This shows how it can all be done in one macro, using the file handling functions that are part of base SAS. You specify the directory and which file type you want to search. There are some default search values or you can enter the values you require. A choice of report allows either a long or a short summary.

Introduction

You have just been told that the data has changed in one dataset and you need to know which programs need to be run again. Or you would like to do a final check on your programs to check there is nothing in there that should not be, warnings, errors etc. You need to write some code that you know you have done before but are not exactly sure where. These are all circumstances that I am sure you have met and are all solved by doing the same thing – searching through the directory until you find out what you want to know. You may have some sort of search facility to do this for you quickly; if not then you have to do it by hand. You are already in SAS so why not use a program to do the searching for you rather than opening each file by hand.

This following will look at some SAS code that has been developed to allow the programmers to quickly search through all the files in a directory, selecting those of a given extension and then within each file for selected text.

The code is broken down into different stages, with an explanation of what each section does and how it can be adapted. It is written as macro with the directory and the file extension passed n as macro variables. The user can choose the type of report wanted at the end (long, short or both) and either say which text they want to search for or use some default ones.

The code was developed in SAS Enterprise Guide, on a windows PC submitting to SAS on a Solaris server. To use it I simply call in the program, fill out the parameters in the macro call at the bottom and hit run.

Check the path name

The first thing to do is check that the directory really exists. There is no point running the whole program if there has been in a typo in the directory path name,

The code uses the *fileexist* function to check that the directory is there. If it is not then a message is written to say that the directory does not exist and the macro aborts.

```
%if %sysfunc(fileexist(&dir))=0  %then %do ;
   %put ;
   %put ERROR: The directory &dir does not exist.;
   %put ;
   %abort ;
%end ;
```

When you know the directory exists then you can carry on.

Read in the directory

The directory is read in using the file handling functions. First of all assign a *fileref* to the directory you want to open using the filename function. Check the return code from the function; if it has opened successfully the return code will be 0.

If the *fileref* has been assigned successfully then open the directory using dopen, which will assign a directory id to a variable which can be used in subsequent code. Use the dnum function to count the number of elements in the directory.

For every element in the directory use dread to return the directory member name, scan through the name and pick up the extension, using -1 in the scan function to pick up the last word rather than the first.

Check that the filename is not equal to the extension itself - this is just in case you have a filename that does not have an extension.

Check to see if the file is of the right type and if so then output the filename. These are the files that will be searched for the required text. You need to allow for the fact that you may have special characters in the filenames.

You may need to close the directory at the end of the step.

```
 data list_dir ;
   rc=filename("fileref","&dir");   * assign the fileref ;
   if rc =0 then do ;
     did=dopen("fileref") ;         * open the directory ;
     memcnt=dnum(did) ;             * count the number of the directory;
     do;i = 1 to memcnt;            * loop though each member ;
       name=scan(dread(did,i),-1,'.');* get the extension of the file ;
       if upcase(dread(did,i)) ne upcase("&ext") then do;                                  * checks to see that the file contains an extension ;
       if ("&superq&ext" ne ""  and upcase(name)=upcase("&ext")) or ("&superq&ext" = "" ) then do; * checks to see if the extension matches the ext parameter ;
         file_n = dread(did,i); * puts the filename into file_n ;
         output ;                 * outputs the record ;
       end;
     end;
   end ;
   else do ; sysmsg=sysmsg() ; put sysmsg= ; end ;

   rc=dclose(did);                * close the directory ;
 run ;
```

Read in the file

The next step uses the filenames that have been output and opens each file in turn, reading in every record from that file.

In the next step you read in every line of all the file names you output in the above step. Set up a character variable (with length of 200 or however long your lines may be), and a variable that will contain the full path to the file.

Attach the directory name to the filename, making sure that you remove any spaces. Assign a fileref to the file you want to open. If this is successful then open the file.

Read each line in the file into SAS. Use fopen to open the file, this will return a file id that can then be used in subsequent code. If the file id is 0 then the file did not open.
Use fread on the opened file (identified by the file id) to read in the file one line at a time, check the return code to make sure that the line read in OK.

Use fget to write the observation from the file data buffer into a variable. You need to specify which file id to read in the line from, which variable to put the line into and the length of the variable. This needs to be output to a dataset.

You may need to close the file at the end of the step.

```
data read_file ;
   set list_dir ;
   length file_n2 $200 ;
   file_n2=compress("&dir./"||file_n) ;
   length f_line $200. ;
   rc_fn=filename("f_ref",file_n2) ;            * apply the fileref ;
   if rc_fn=0 then do ;
     rc_f=fopen("f_ref") ;                       * open the file ;
     if rc_o gt 0 then do while (fread(rc_o)=0) ; * while there are lines in the file read it ;
       rc_g=fget(rc_o, f_line,200) ;              * put the line into an observation ;
       output ;
     end ;
   end ;
   rc_c=fclose(rc_o) ;                          * close the file ;
 run ;
```

The dataset now contains every line of every file that is going to be searched so this can be a large dataset.

Searching the file

The code then scans each line of code to see if it contains one of the words it is looking for, if it does then the line of code and the filename are output.

The code is written so that you can either pass in a search string or you can set some default search strings. You could pass in what you want to search for when you want t o look for a dataset or variable, where as the default searches could be used for example if you want to search for errors or warnings.

All the text that is going to be searched for is entered as separate elements in an array. The code then loops through each element of the array to see if that text is contained in the observation that is being looked at.

Before this can be done the code has to decide is it is using the default search text or you have passed in some text. It does this by looking to see what the length of the parameter that would contain the text to be searched for is. If this is greater than one then it assumes that some text has been passed in and .

Initialise a variable that will contain the list of text to be searched for. Replace the space between the quotes in the a comma using tranwrd, put this into a different variable. If you then compress this variable to remove all the spaces then count the number of separate bits of text you are going to search for. The reason that you compress the text to start with is in case there are any spaces in the text you want to search for.

Once you have worked out how many bits of text you are going to search for then put this into a macro variable so that you can use it in the array statement below.

```
  %if %length(&srch_for)> 1 %then %do ;
    * scan through and see how many words there are in the &srch_for ;
    data _null_;
      wrd_lst="initial" ;                       * give wrd_lst something to start with ;
    i=0 ;                                     * initialise i ;
    repl=tranwrd("&srch_for","+ ","+") ;      * replace ' ' with ',' ;
      do while (length(wrd_lst)>0) ;            * do while there is a word to find ;
        i=i+1 ;                                 * increase i ;
        wrd_lst=scan(compress(repl),i,',') ;    * scan for the next word ;
      output ;
      end ;
    call symputx('numwrd',i-1) ;              * if finished, output the number of searches into a macro variable ;
    run ;
  %end ;
```

If you are not passing in any search string then the srch_for parameter will be empty. In this case set up a temporary character array with the what will be your default search strings.

If you are passing in some text to search for then put the srch_for parameter in place of the default text and replace the number of elements of the array with the macro variable created above.

The idea is then to look in each observation to see if any of the elements if the array are in there. This is done by do a loop form 1 to the size of the array (using the dim function). Set a variable to be the array element that you are looking for (so in any report you know which text you have found) then use the scan function to look along the line of file read in to see if the text is in it. I upper case both the line being searched and the array element so that I do not worry about the case of the string I am looking for.

If the array element is found then output that observation. At the end of this your dataset will only contain the lines of text that contain one or more of the text that you are searching for.

```
"" want to search the statements for certain words and output the filename if the word is found ;
data search ;
  set read_file ;
  * put in the words that you want to check for in an array - if add/delete words, then remember to change the number of array elements ;
  %if %length(&srch_for)=0 %then %do ;
    array srch (13) $50  _temporary_ ('error' 'warning' 'trunc' 'has 0 obs' 'uninitialized' 'repeats of by' 'no by statement'
                         'does not exist' 'outside the axis' 'Missing value' 'converted' 'overwritten' 'then delete' ) ;
  end ;
  %else %do ;
    array srch (&numwrd) $50  _temporary_ (&srch_for.) ;
  %end ;
  do i=1 to dim(srch) ;                                                  * go through each line ;
    found=srch[i] ;                                                      * puts the word looking for into a variable ;
    if index(upcase(f_line), strip(upcase(srch[i])))> 0 then output ;    * if word found in record then output ;
  end ;
run ;
```

Producing a report

The macro is written so that two types of report are produced, a long one that lists every file and line that contains at least one of the search results, and a shorter one that lists just the text that was found and the file it was found in. By default both reports are produced. if you only want one produced then you pass in long or short as the report macro variable depending on which report is wanted.

If the long report is wanted sort the data by the file name, the file line and the text found, removing any duplicates.

Cont up the number of observations this results in. This is so if nothing was found then you can create a blank observation can be created.

Use PROC REPORT to create a report putting out the file name the text was found in, the line of the file where the text was found and the text that was found. If an order option is put on the file name then it is easy to see the separate files.

If none of the text strings were found use a compute block to put a message out saying nothing was found.

```
%if &report ne short %then %do ;
* remove any duplicates ;
proc sort data=search out=found nodupkey ;
  by file_n f_line found ;
run ;

* count the number of observations found ;
  data _null_;
    if 0 then do;
      set found nobs=count;
    end;
    call symputx("nobs",count);
    stop;
  run;

%if nobs=0 %then %do;
  * if no data then create a blank record ;
  data found ;
    file_n=' ' ; found=' ' ; f_line=' ' ;
  run ;
%end ;

  * create the report ;
  title2 "The following have been found in &dir " ;
  Proc report data=found nowd split='|' missing headline;
    column file_n found f_line ;
    define file_n / order left width=20 flow 'file name' ;
    define found / display left width=15 flow 'Word found' ;
  define f_line / display left width=60 flow 'On line' ;
%if nobs=0 %then %do ;
    compute after file_n ;
      line @25 ' ';
      line @25 ' ';
      line @25 '*********************************';
      line @25 '*       Nothing found           *';
      line @25 '*********************************';
      line @25 ' ';
      line @25 ' ';
      line @25 ' ';
    endcomp;
  %end ;
  run ;

%end ;
```

If the shorter report is wanted then sort the data just on the file name and the text found using the nodupkey options to remove duplicates.

```
proc sort data=search out=found2 nodupkey ;
  by file_n  found ;
run ;
```

The PROC REPORT is very similar to the one for the long report expect that it does not contain the line the text was found on. only the file name and the text found.

CONCLUSION

To use the macro pass in the directory , the extension, what you want to search for and the type of report. For example to look for which programs use the dataset SDTM.AE then use the following (where lsit_dir is the name given to the macro)

```
%list_dir(dir=aaaa/bbbbb/cccc/programs/sdtm, ext=sas, srch_for="sdtm.ae", report=short)
```

Using the above code you can easily check which programs use a specific dataset or variable, or you can quickly check the log files for warnings and/or errors. It is easy to use, allowing you to either specify what you are looking for or to use the default, with a choice of report.

This program was developed in SAS Enterprise Guide submitting the code to a Solaris server.

ACKNOWLEDGMENTS

I would like to thank Paul Janvier for giving me the idea for this code.

Contact Information

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

Helen Nicholson
Cmed Clinical Services
Holmwood,
Broadlands Business Campus,
Langhurstwood Road
Horsham,
West Sussex,
RH12 4QP
United Kingdom

Work Phone: +44 (0)1403 755640
Email: hnicholson@cmedresearch.com

Brand and product names are trademarks of their respective companies.