

SAS[®] Macro Tool to Find Source Data Sets Used in Programs

Prasanna Murugesan¹, Sushant Thakare² – Quintiles Inc., Overland Park, KS

ABSTRACT

Often, as a SAS[®] programmer, we need to know the source data sets used in a program. Traditionally, this has been achieved by opening individual programs in an editor and manually searching for source data sets used in the program. When the search involves multiple programs, it can be time consuming and prone to errors. This paper discusses an automated process using a SAS macro that searches through multiple SAS programs. The search results are stored in Excel format and consist of all the source data sets used in the SAS programs located in the search directory specified.

Operating system: Windows XP
 SAS version used: SAS 9.2
 Other tools used: Microsoft Office[®] - Excel 2007
 Skill level: Intermediate

INTRODUCTION

In the pharmaceutical industry, derived data sets (SDTM³, ADaM⁴, etc.) need to be generated before executing tables/listings/figures (TLFs). Assuming a study has hundreds of TLFs and you want to run only a few of them for a particular delivery, it will be efficient to run only the derived data sets that are required for TLFs for that particular delivery. Also for documentation purposes, we would like to know the source data sets used in every program. The above mentioned steps are usually achieved by opening the individual TLF programs and manually searching for source data sets used in the program.

TECHNIQUE

INPUT

The input to the macro is an Excel spreadsheet which has the following information:

- i) Directory location where the SAS programs are stored.
- ii) Search terms (libnames). The libnames should include period at the end of the search term (e.g. cdm.).

A sample input Excel file is in Figure 1.

	A	B	C
1	Directory to be searched	D:\Programs	Instructions: 1. Enter directory to be searched in cell B1. 2. Enter search terms (e.g. - derived., sdtm., adam., etc.) in cells B2, B3 and so on. Not case sensitive. 3. Save the file in the same location as the program - Data_Set_Name_Search_V3.sas. (can be in any directory) 4. Run the program. 5. An Excel file with the search results will be created in the same folder.
2	Search term 1	cdm.	
3	Search term 2	derived.	
4	Search term 3	adam.	
5	Search term 4		
6	Search term 5		
7	Search term 6		
8	Search term 7		
9	Search term 8		
10	Search term 9		
11	Search term 10		
12	Search term 11		
13	Search term 12		

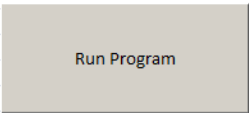


Figure 1. Sample Input Excel file

¹ – Responsible for the technical content of the paper and the SAS Code. ² – Responsible for the concept of this macro.

³ – SDTM stands for Study Data Tabulation Model. It is a standard structure for clinical study data tabulations which need to be submitted as part of product application to regulatory authority.

⁴ – ADaM stands for Analysis Data Model. ADaM data sets are analysis data sets generated from SDTM data sets. TLFs are generated using ADaM data sets.

PRE-PROCESSING

The SAS macro pre-processes the above input file to check for existence of the directory and creates an error message in the output Excel file if the search directory is not found.

Assuming the directory exists, the macro searches for SAS programs present in the folder and stores the program names in macro variables for later use. If there are no SAS programs present in the directory, the program stops executing and creates an error message in the output Excel file.

A sample output Excel file with error message for invalid directory specified as input is in Figure 2.

	A	B	C	D
1	<i>Data_Set_Name_Search (Not case sensitive)</i>			
2				
3	Search Term	Program	Search Term Findings Separated	Directory Searched
4	Check Directory Location			D:_Programs
5				
6	<i>Run from Data_Set_Name_Search.sas on 13AUG2012 18:22</i>			
7				

Figure 2. Sample output Excel file with error message

The macro also stores the search terms in macro variables. If no search terms are provided an error message is created in the output file.

PROCESSING

For every search term, the macro reads all SAS programs in the search folder into data sets. Every word in the program forms an observation in the data set as shown in Figure 3.

CONVERTING SAS PROGRAMS TO DATA SETS

```
FILENAME code "&dirname.\&&dsn&cnt";
```

```
DATA raw&mscnt&cnt (KEEP = derv);
  INFILE code MISSOVER LENGTH=reclen;
  LENGTH inline derv $800.;
  INPUT inline $varying800. reclen;
  line = TRANWRD(inline,'09'x,' ');
  cnt = 1;
  DO UNTIL(derv EQ ' ');
    derv=SCAN(inline,cnt,' ');
    IF derv NE ' ' THEN OUTPUT;
    cnt+1;
  END;
```

```
RUN;
```

1. `proc sort data=raw.cm;`

	derv
1	proc
2	sort
3	data=raw.cm;

2. `data raw.mh(keep = subjid);`

	derv
1	data
2	raw.mh(keep
3	=
4	subjid);

Figure 3. Illustration - SAS Program to SAS data set

The macro looks for a specific pattern in the data set which starts with the libname. In Figure 1, for the first search term “cdm.”, it looks for occurrences of the following patterns:

- i) cdm.dsn
- ii) cdm.dsn(keep=...)
- iii) cdm.dsn (keep=...)

The macro uses a set of SCANQ, FIND and SUBSTR functions to identify and extract the data set names. The macro stores the libname.dsn (e.g. adam.dsn) combination in a macro variable and ignores any characters following the data set name. The above process is continued for all the search terms. If a search term is not found in a program the macro variable is set to “not found”.

SEARCHING FOR SOURCE DATA SETS

```
DATA d_mod&mscnt&cnt (KEEP = new_word2);
  LENGTH derv1 $150;
  SET raw&mscnt&cnt;
    derv1 = UPCASE (COMPBL (LEFT (TRIM (derv))));
    len = LENGTH (derv1);
    a = 0;
    DO i = 1 TO len;
      X = LEFT (TRIM (SUBSTR (derv1, i, 1)));
      IF i < len AND x=" " THEN /* condition 1 */
        DO;
          a = a + 1;
          new_word = SCANQ (derv1, a, " ");
          der_find = FIND (new_word, %UPCASE ("%&look_for&mscnt"));
          IF der_find > 1 THEN
            new_word1 = SUBSTR (new_word, der_find);
          ELSE IF der_find = 1 THEN
            new_word1 = new_word;
            /* characters to be removed after search term */
            der_find1 = FINDC (new_word1, '%STR(,;)"');
            IF der_find1 > 1 THEN
              DO;
                new_word2 = SUBSTR (new_word1, 1, der_find1-1) ;
                IF new_word2 ^= " " THEN OUTPUT;
              END;
            ELSE IF der_find1 = 0 THEN
              DO;
                new_word2 = new_word1;
                IF new_word2 ^= " " THEN OUTPUT;
              END;
            END;
          END;
        END;
      END;
    END;
  END;
RUN;
```

After processing all the search terms, the macro variables are stored in a data set for final reporting.

EXECUTION AND OUTPUT

The macro is executed by clicking on the “Run Program” button in the input Excel file. The VBA macro that is assigned to “Run Program” button in the input Excel file is discussed later in the VBA Macro section.

ODS tagset is used to provide the output in Excel format with color coding and filtering options for easy readability. The output has the following four columns.

1. Search term name.
2. File name that is searched.
3. Data set names separated by commas.
4. Directory searched.

A sample output file is shown in Figure 4.

	A	B	C	D
1	Data_Set_Name_Search (Not case sensitive)			
2				
3	Search Term ▾	Program Searched ▾	Search Term Findings Separated By , ▾	Directory Searched ▾
4	adam.	Listing1.sas	adam. not found.	D:\Programs
5	sdtm.	Listing1.sas	sdtm. not found.	D:\Programs
6	derived.	Listing1.sas	DERIVED.ADAE	D:\Programs
7	adam.	Table1.sas	adam. not found.	D:\Programs
8	sdtm.	Table1.sas	sdtm. not found.	D:\Programs
9	derived.	Table1.sas	DERIVED.ADDS, DERIVED.ADDA, DERIVED.ADEX	D:\Programs

Figure 4. Sample Output Excel file

CONCLUSION

This automated search process provides a consolidated report of the search results in much less time when compared to a manual search process. As mentioned earlier, it can also be used as a tool to document source data sets used in programs.

REFERENCES

Consistency Check: QC across Outputs for Inconsistencies

John Morrill, Quintiles Inc., Kansas City, MO, - David J. Austin, PRA International, Charlottesville, VA

<http://www.lexjansen.com/pharmasug/2006/technicaltechniques/tt14.pdf>

<http://support.sas.com>

www.cdisc.org

SDTM Implementation Guide V 3.1.2

ACKNOWLEDGMENTS

We want to thank our colleagues at Quintiles and MWSUG section chair Richann Watson (Paraxel) who reviewed and provided their thoughts on this paper and presentation.

CONTACT INFORMATION

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

Name: Prasanna Murugesan
 Enterprise: Quintiles Inc.
 Address: 6700 W 115th Street
 City, State ZIP: Overland Park KS 66211
 Work Phone: 913-708-6683
 Fax: 913-708-6761
 E-mail: prasanna.murugesan@quintiles.com

Name: Sushant Thakare
 Enterprise: Quintiles Inc.
 Address: 6700 W 115th Street
 City, State ZIP: Overland Park KS 66211
 Work Phone: 913-708-6700
 Fax: 913-708-6761
 E-mail: sushant.thakare@quintiles.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. MS Office is a registered trademark of the Microsoft Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.

PROGRAM CODE

```
/* ***** */
/* Get input information */
/* ***** */

/* Current program path */
DATA prgpath (KEEP = xpath);
  SET sashelp.vextfl
    (WHERE=(UPCASE(xpath) LIKE '%.SAS'));
  IF _n_ = 1 THEN OUTPUT;
RUN;

DATA prgpath;
  SET prgpath;
  pathloc = FIND(xpath, '\', -500);
  outpath = SUBSTR(xpath, 1, (FIND(xpath, '\', -500) - 1));
  CALL SYMPUT('outpath', LEFT(TRIM(outpath)));
RUN;

/* output path */
%PUT &outpath;

/* search spec location */
%LET searchspecs = &outpath.\Search_input.xlsm;
%PUT &searchspecs;

PROC IMPORT FILE = "&searchspecs"
  OUT      = search_info
  DBMS     = excel2000 replace;
  SHEET    = "search_spec" ;
  GETNAMES = no;
  MIXED    = yes;
  SCANTEXT = yes;
RUN;

%GLOBAL dirname ;

PROC SQL NOPRINT;
  SELECT f2 INTO :study SEPARATED BY '***'
  FROM search_info;
  /* no. of words to be searched */
  %LET mstr_cnt = %EVAL(%SYSFUNC(COUNTW(&study, '***')) - 1);
  /* directory location to be searched */
  %LET dirname = %SUBSTR(&study, 1, (%INDEX(&study, **)) - 1);
QUIT;

%PUT "executed";
/* type of file to be searched */
FILENAME dirlist PIPE "dir /B &dirname\*.sas";

/* search begins */
OPTIONS MPRINT NOMLOGIC NOSYMBOLGEN;

%MACRO search;
  /* directory location not provided - QUIT execution */
  %IF "&dirname" = "" %THEN
    %DO;
      DATA final;
        LENGTH word $30. prg $50. srch_term $250. dir_name $250. ;
        word      = "Directory location missing";
        prg       = " ";
        srch_term = " ";
        dir_name  = "&dirname";
      RUN;
    %END;
  %ELSE
    %DO;
      /* search logic */
    %END;
  %END;
%END;
```

```

PROC FORMAT ;
VALUE $ r_color
  " " = "#FAF1B6"
;
RUN;

%END;
/* search terms not provided - QUIT execution */
%ELSE %IF &mstr_cnt = 0 %THEN
%DO;
DATA final;
  LENGTH word $30. prg $50. srch_term $250. dir_name $250. ;
  word      = "No search words provided";
  prg       = " " ;
  srch_term = " " ;
  dir_name  = " " ;
RUN;

PROC FORMAT ;
VALUE $ r_color
  " " = "#FAF1B6"
;
RUN;

%END;

%ELSE
%DO;
/* directory location incorrect - QUIT execution */
OPTIONS NOXWAIT;
%LOCAL rc fileref ;
%LET rc = %SYSFUNC(FILENAME(fileref,&dirname)) ;
%IF %SYSFUNC(FEXIST(&fileref)) = 0 %THEN
%DO;
DATA final;
  LENGTH word $30. prg $50. srch_term $250. dir_name $250. ;
  word      = "Check Directory Location";
  prg       = " " ;
  srch_term = " " ;
  dir_name  = "&dirname" ;
RUN;

PROC FORMAT ;
VALUE $ r_color
  " " = "#FAF1B6"
;
RUN;

%END;
%ELSE
%DO ;
  /* Store all search terms */
%DO i = 1 %TO &mstr_cnt;
  %GLOBAL look_for&i;
  %LET look_for&i = %scan(&study,%EVAL(&i+1),'**');
%END;

DATA dirlist ;
  LENGTH fname $256;
  INFILE dirlist LENGTH = reflen ;
  INPUT fname $varying256. reflen ;
  num = LEFT(TRIM(PUT(_n_,5.)));
  CALL SYMPUT ('num_files',num);
RUN;
/* No target files -.sas in directory location - QUIT execution */
PROC SQL NOPRINT;

```

```

        SELECT COUNT(*) INTO :filecnt FROM dirlist;
QUIT;

%IF &filecnt = 0 %THEN
%DO;
    DATA final;
        LENGTH word $30. prg $50. srch_term $250. dir_name $250. ;
        word      = "No .SAS files found";
        prg       = " " ;
        srch_term = " " ;
        dir_name  = "&dirname" ;
    RUN;

    PROC FORMAT ;
    VALUE $ r_color
        " " = "#FAF1B6"
    ;
    RUN;
%END;
%ELSE
%DO;
    DATA dirlist;
    SET dirlist;
        %DO i = 1 %TO &num_files;
            %GLOBAL dsn&i;
        %END;
        num = LEFT(TRIM(PUT(_n_,5.)));
        CALL SYMPUT ("dsn"||num,LEFT(TRIM(fname)));
    RUN;

/* loop 1 begins; search all files for first word & then go to next word */

%DO mscnt = 1 %TO &mstr_cnt;          /* no. of words */
%DO cnt = 1 %TO &num_files;          /* no. of files */
    FILENAME code "&dirname.\&&dsn&cnt";

    /* split every word in the file to an observation */
    DATA raw&mscnt&cnt (KEEP = derv);
    INFILE code MISSOVER LENGTH=reclen;
    LENGTH inline derv $750.;
    INPUT inline $varying750. reclen;
    inline = TRANWRD(line,'09'x,' ');
    cnt = 1;
    DO UNTIL(derv eq ' ');
        derv = SCAN(line,cnt,' ');
        IF derv ne ' ' THEN OUTPUT;
        cnt+1;
    END;
    RUN;

    /* scan previous step output for search term */
    DATA d_mod&mscnt&cnt (KEEP = new_word2);
    LENGTH derv1 $150;
    SET raw&mscnt&cnt;
        derv1 = UPCASE(COMPBL(LEFT(TRIM(derv))));
        len = LENGTH(derv1);
        a = 0;
        DO i = 1 TO len;
            x = LEFT(TRIM(SUBSTR(derv1,i,1)));
            IF i < len and x = " " THEN          /* condition 1 */
                DO;
                    a = a + 1;
                    new_word = SCANQ(derv1,a," ");
                    der_find = FIND(new_word,%UPCASE("&&look_for&mscnt"));
                    IF der_find > 1 THEN
                        new_word1 = SUBSTR(new_word,der_find);
                    ELSE IF der_find = 1 THEN

```

```

new_word1 = new_word;
/* characters to be removed after search term */
der_find1 = FINDC(new_word1, '%STR((;"))');
IF der_find1 > 1 THEN
DO;
new_word2 = SUBSTR(new_word1,1,der_find1-1) ;
IF new_word2 ^= " " THEN OUTPUT;
END;
ELSE IF der_find1 = 0 THEN
DO;
new_word2 = new_word1;
IF new_word2 ^= " " THEN OUTPUT;
END;
END;
IF i = len THEN /* condition 2 */
DO;
new_word = SCANQ(derv1,-1);
der_find = FIND(new_word,%UPCASE("&&look_for&mscnt"));
IF der_find > 1 THEN
new_word1 = SUBSTR(new_word,der_find);
ELSE IF der_find = 1 THEN
new_word1 = new_word;
/* characters to be removed after search term */
der_find1 = FINDC(new_word1, '%STR((;"))');
IF der_find1 > 1 THEN
DO;
new_word2 = SUBSTR(new_word1,1,der_find1-1) ;
IF new_word2 ^= " " THEN OUTPUT;
END;
ELSE IF der_find1 = 0 THEN
DO;
new_word2 = new_word1;
IF new_word2 ^= " " THEN OUTPUT;
END;
END;
END;
END;
RUN;

DATA d_mod&mscnt&cnt;
SET d_mod&mscnt&cnt;
IF new_word2 = UPCASE("&&look_for&mscnt") THEN DELETE;
IF SUBSTR(new_word2,LENGTH(new_word2),1) = "." THEN
new_word2=SUBSTR(new_word2,1,(LENGTH(new_word2)-1));
RUN;

/* concatenating all search findings in one file to a macro */
PROC SQL NOPRINT;
%GLOBAL search&mscnt&cnt;
SELECT DISTINCT(new_word2) INTO :search&mscnt&cnt
SEPARATED BY ", " FROM d_mod&mscnt&cnt
WHERE ((SELECT COUNT(new_word2) FROM d_mod&mscnt&cnt) > 0);
QUIT;

/* result data set for individual file */
DATA final&mscnt&cnt;
LENGTH target $200. word $30. prg $50. srch_term $250. dir_name $250. ;
word = "&&look_for&mscnt.";
prg = "&&dsn&cnt.";
target = RESOLVE('%NRBQUOTE(&&search&mscnt&cnt)');
IF target = "" THEN
srch_term = "&&look_for&mscnt." || " not found." ;
ELSE
srch_term = target;
dir_name = "&dirname.";
cnt = &cnt;
RUN;

```



```

%END;
/* go to next search word */
%END;
/* go to next file */

/* loop 1 ends */

/* set all individual findings to one master data set */
DATA final;
SET
  %DO msn = 1 %TO &mstr_cnt;
    %DO i = 1 %TO &num_files;
      Final&msn&i
    %END;
  %END;
;
RUN;

PROC SORT DATA = final ;
  BY cnt ;
RUN;

/* color coding as per each file name */
PROC FORMAT ;
VALUE $ r_color
  %DO c = 1 %TO &num_files %BY 2;
  "&&dsn&c" = '#FAF1B6'
%END;
  %DO d = 2 %TO &num_files %BY 2;
  "&&dsn&d" = '#C5F0C5'
%END;
;
RUN;;

/* Delete macro variables */
%DO i = 1 %TO &num_files;
  %SYMDEL dsn&i ;
%END;

%DO msn = 1 %TO &mstr_cnt;
  %DO i = 1 %TO &num_files;
    %SYMDEL search&msn&i;
  %END;
%END;
%END;
%END;

%MEND search;

%search;

DATA final;
SET final;
  LABEL
    word = "Search Term"
    prg = "Program Searched"
    srch_term = "Search Term Findings SEPARATED BY ,"
    dir_name = "Directory Searched"
;
RUN;

/* Report */

%* Use ODS to output to XML file *;
ODS ESCAPECHAR='~';

```

```

ODS LISTING CLOSE;
ODS TAGSETS.ExcelXP
FILE="%outpath.\Data_set_Name_Search_&sysdate9..xls";

%LET sline=borderrightwidth=2 borderbottomwidth=1 bordertopwidth=1;
%LET nline=borderrightwidth=1 borderbottomwidth=1 bordertopwidth=1;
%LET bline=bordertopwidth=1.5;

TITLE1 'Data_Set_Name_Search (Not case sensitive)';
FOOTNOTE1 "Run from Data_Set_Name_Search.sas on &sysdate9 &sysptime";

%MACRO listing;
ODS TAGSETS.EXCELXP OPTIONS(
SHEET_NAME           ='Data_Set_Name_Search'
AUTOFIT_HEIGHT       ='Yes'
ORIENTATION          ='landscape'
EMBEDDED_TITLES      ='yes'
EMBEDDED_FOOTNOTES   ='yes'
ZOOM                 ='80'
FROZEN_HEADERS       ='3'
AUTOFILTER           ='all'
FITTOPAGE            ='yes'
PAGES_FITWIDTH       ='1'
PAGES_FITHEIGHT      ='300'
ABSOLUTE_COLUMN_WIDTH='10,15,50,60');

PROC SQL NOPRINT;
  SELECT COUNT(*) INTO :rec_num FROM final;
QUIT;

%IF &rec_num = 0 %THEN %DO;
  * Create Empty Observation Data set;
  DATA empty;
    message = 'Program issues - Check log';
  RUN;

  PROC PRINT DATA = empty NOOBS LABEL;
  RUN;
%END;

%ELSE %DO;
  PROC REPORT DATA = final NOWD HEADLINE HEADSKIP MISSING SPACING=5 SPLIT='^';
    COLUMNS word prg srch_term dir_name ;
    DEFINE word          / DISPLAY;
    DEFINE prg           / DISPLAY;
    DEFINE srch_term     / DISPLAY;
    DEFINE dir_name      / DISPLAY;
    /* color coding */
    COLUMN row_color ;
    DEFINE row_color    / COMPUTED NOPRINT;
    COMPUTE row_color ;

    CALL DEFINE ( _ROW_ , 'STYLE',
      'STYLE=[BACKGROUND=' || PUT(prg, r_color.) || ']' );
    ENDCOMP;
  RUN;
%END;

%MEND listing;
%listing;

ODS TAGSETS.EXCELXP CLOSE;
ODS LISTING;
TITLE1;
FOOTNOTE1;
RUN;

```

VBA Macro

The steps mentioned below should be followed to run the SAS macro using the “Run Program” button in the input Excel spreadsheet.

- The following VBA code should be inserted in Excel. Click on “Developer” tab followed by “Visual Basic” icon. Insert the following code in the VB window and save the excel file. The file will be saved in “Excel Macro-Enabled Workbook” format.
- Click on “Developer” tab followed by “Insert” icon and select the “button” option. This will allow the user to place a button anywhere in the spreadsheet. After placing the button, rename the button as “Run Program”. Right click on the button and select “Assign Macro” to assign the macro created in step1 to this button.
- The variables “exepath” and “pgmpath” should be updated. “exepath” contains the path for the SAS execution file. “pgmpath” contains the location of the program and the input Excel spreadsheet.

`to run SAS macro from Excel:

```
Sub connect()  
Dim exepath As String  
Dim pgmpath As String  
Dim cm As Long  
  
`SAS execution .exe file  
exepath = "...\\9.2(32-bit)\\sas.exe"  
`SAS Macro location and name  
pgmpath = ".....\\program_name.sas"  
  
newHour = Hour(Now())  
newMinute = Minute(Now())  
newSecond = Second(Now()) + 5  
waitTime = TimeSerial(newHour, newMinute, newSecond)  
Application.Wait waitTime  
  
cm = Shell(exepath & " " & "&&" & " " & "-sysin" & " " & pgmpath & " ", vbNormalFocus)  
  
End Sub
```