PharmaSUG 2018 - Paper QT-08

A Macro to Create Program Inventory for Analysis Data Reviewer's Guide

Xianhua (Allen) Zeng, PAREXEL International, Shanghai, China

ABSTRACT

As per Analysis Data Reviewer's Guide (ADRG) released by the PhUSE CSS Working Group in January, 2015, a program inventory that contains all inputs (such as SDTM domains or other analysis datasets) and macros used should be listed in the section Submission of Programs. It can be laborious and time consuming to extract inputs and macros manually. This paper presents a SAS® macro ExtDsnMan, which programmatically extracts all input datasets and macros and produces an XLS report.

The code is currently hosted in my Github page:

https://raw.githubusercontent.com/XianhuaZeng/PharmaSUG/master/2018/extdsnman.sas

INTRODUCTION

When writing ADRG, we need to create the program inventory that contains all inputs (such as SDTM domains or other analysis datasets) and macros used in section Submission of Programs. To do this task manually might be cumbersome, so it seemed appropriate to create a macro to handle the work.

Note that the code for this paper was written in UNIX operating environment. Therefore, users would need to make some adjustments to the code based on their operating environment.

MACRO PARAMETERS

I call this macro as %ExtDsnMan and it has 4 input parameters as listed in Table 1 below:

Parameter	Description
P_PATH	Path to the parent directory where all SAS programs are located. This is a REQUIRED parameter and does not have a default value.
KWORD	Library name of analysis or SDTM datasets. This is a REQUIRED parameter and its default value is analysis|raw, with library names separated by a single pipe symbol.
M_PATH	Path to the directory where macros are located. This is a REQUIRED parameter and does not have a default value.
O_PATH	Path to the output directory where program inventory is located. This is not a REQUIRED parameter and does not have a default value.

Table 1. ExtDsnMan Parameters

METHODOLOGY

ExtDsnMan initially reads all programs into a dataset and then extracts input datasets and macros used using SAS Perl regular expression functions. Approaches to programmatically extract input datasets and macros used are demonstrated as follows:

READ ALL PROGRAMS INTO ONE DATASET

Read all programs in the program parent directory including all subdirectories to a dataset using FOPEN/FREAD/FGET functions. This method is more efficient than DO loop since the loop goes through all programs, one by one. The code used to do this task is shown below:

```
RC_FILE=filename('code', PATH, ,'LRECL=32767');
FID=fopen('code');
do while (fread(FID)=0);
    length CODELINE $32767;
    RC_READ=fget(FID, CODELINE, 32767);
end;
```

```
CLOSE=fclose(FID);
RC_FILE=filename('code');
```

IDENTIFY INPUTS AND MACROS CALL

As the rules for legal SAS names, a name can be 1 to 32 characters long, begin with a letter or underscore, and the rest of the characters can contain only letters, numerals, or underscores. SAS Perl regular expression function is an ideal method. A pattern like library.dataset in program is considered as input:

```
RE=prxparse("/\b(&kword)\.(?=\w{1,32})([a-zA-Z_][a-zA-Z0-9_]*)/");
```

A pattern like %maroname in program is considered as macro call:

```
RE=prxparse('/%(?=\w{1,32})[a-zA-Z_][a-zA-Z0-9_]*\(?/');
```

Note that a pattern like library.dataset or %maroname in a comment should not be identified as input or macro call. The following is regular expression to remove comments:

```
RE=prxparse('/^(\*|%\*|\/\*).+$/');
```

EXTRACT INPUTS AND MACROS CALL

In this section, the ultimate goal is to parse the code lines to extract the inputs and macros used. The regular expression used to extract the inputs is shown below:

```
RE=prxparse("s/.*?(?:(?:&kword)\.(\w+))?/\1 /i");
```

Regular expression visualization by Regexper:

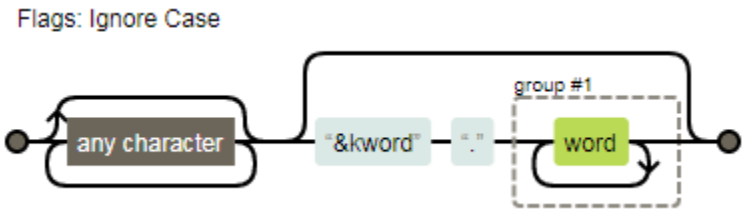

Figure 1. Regular Expression Visualization

The following piece of codes extracts the macros:

```
/*Create macros list within macro library */
proc sql noprint;
    select FNAME into :mlist separated by '|'
        from mlib
        ;
quit;
RE=prxparse("s/.*?(?:%(&mlist)\b)?/\1 /i");
```

Here is a brief explanation of the regular expression above. "." matches any single character except newline. "*?" is lazy repetition factor, matches 0 or more occurrences of the preceding character as few times as possible. "(?:…)" means non-capturing group. "%" matches percent sign. "\b" matches a word boundary. The second "(" and ")" characters matches a pattern and creates a capture buffer for the match. The last "?" is greedy repetition factor, matches the first capturing group zero or one time as many times as possible. The "\1" matches capture buffer 1.

MACRO SOURCE CODE

```
%macro extdsnman(p_path =,
```

```
                m_path =,
                kword  =analysis|raw,
                o_path =
                );

/*To mute "WARNING: The quoted string currently being processed has become
more than 262 characters long. You may have unbalanced quotation marks."*/
options NOQUOTELENMAX;

/*List all files within a directory including subdirectories*/
filename dir pipe "find &p_path -name '*.sas'";

data code;
    infile dir truncover lrecl=1024;
    length PATH $1024 FNAME $200;
    input PATH 1-1024;
    retain RE;
    if _N_=1 then RE=prxparse('s/(.+)\/(.+?)/\2/');
    FNAME=prxchange(RE, 1, PATH);
    /*Read all files within in a directory including subdirectories into a
dataset*/
    RC_FILE=filename('code', PATH, ,'LRECL=32767');
    FID=fopen('code');
    do while (fread(FID)=0);
        length CODELINE $32767;
        RC_READ=fget(FID, CODELINE, 32767);
        CODELINE=compress(CODELINE, , 'kw');
        if ^missing(compress(CODELINE)) then output;
    end;
    CLOSE=fclose(FID);
    RC_FILE=filename('code');
    keep PATH CODELINE FNAME;
run;

/*Close the pipe*/
filename dir clear;

%macro ftype(type=);
data temp01;
    set code;
    retain RE1 RE2;
    if _N_=1 then do;
        RE1=%if &type=INPUT %then prxparse("/\b(&kword)\.(?=\w{1,32})([a-zA-
Z_][a-zA-Z0-9_]*)/");
            %else prxparse('/%(?=\w{1,32})[a-zA-Z_][a-zA-Z0-9_]*\(?/');;
        RE2=prxparse('/^(\*|%\*|\/\*).+$/');
    end;
    if prxmatch(RE1, CODELINE) and ^ prxmatch(RE2, cats(CODELINE));
    proc sort nodupkey;
    by PATH FNAME CODELINE;
run;

data temp02;
    set temp01;
    length &type $32767;
    retain RE;
```

```
    if _N_=1 then RE=%if &type=INPUT %then
prxparse("s/.*?(?:(?:&kword)\.(\w+))?/\1 /i");
                    %else prxparse("s/.*?(?:%(&mlist)\b)?/\1 /i");;
    &type=compbl(prxchange(RE, -1, CODELINE));
    &type=prxchange('s/ /, /o', -1, cats(&type));
run;

data &type.S;
    set temp02;
    by PATH FNAME;
    length &type.S $32767;
    retain &type.S;
    if first.FNAME then &type.S=cats(&type);
    else &type.S=catx(", ", &type.S, &type);
    retain RE1 RE2;
    if _N_=1 then do;
        RE1=prxparse('s/(\b.+?\b)(,\s.*?)(\b\1+\b)/\2\3/i');
        RE2=prxparse('/(\b.+?\b)(,\s.*?)(\b\1+\b)/i');
   end;
    if last.FNAME;
    /*Remove repeated values*/
    do i=1 to 100;
        &type.S=prxchange(RE1, -1, cats(&type.S));
        if not prxmatch(RE2, cats(&type.S)) then leave;
    end;
    %if &type=INPUT %then &type.S=prxchange('s/'||cats(scan(FNAME, 1,
'.'))||'//i', -1, cats(&type.S));;
    &type.S=prxchange('s/(, )+/, /o', -1, cats(&type.S));
    &type.S=prxchange('s/^(, )+|(,\s?)+$//o', -1, cats(&type.S));
    %if &type=MACRO %then %do;
        if ^missing(&type.S) then do;
            &type.S=prxchange('s/,|$/.sas, /o', -1, cats(&type.S));
            &type.S=prxchange('s/,$//o', -1, cats(&type.S));
        end;
    %end;
    keep FNAME &type.S;
run;
%mend ftype;

/*Dataset name*/
%ftype(type=INPUT)

/*List all macros within macro library*/
filename macro pipe "find &m_path -name '*.sas'";

data mlib;
    infile macro truncover lrecl=1024;
    length PATH $1024 FNAME $200;
    input PATH 1-1024;
    FNAME=prxchange("s/(.+)\/(.+?).sas/\2/o", 1, PATH);
run;

/*Close the pipe*/
filename macro clear;

/*Create macros list within macro library */
proc sql noprint;
```

```
    select FNAME into :mlist separated by '|'
        from mlib
        ;
quit;

/*Macros name*/
%ftype(type=MACRO)

/*Combine all*/
proc sql;
    create table final as
        select a.FNAME 'Program name', INPUTS 'Inputs', MACROS 'Macros used'
        from code a
        left join
        inputs b
        on a.FNAME=b.FNAME
        left join
        macros c
        on a.FNAME=c.FNAME
        ;
quit;

/*Produce report*/
ods path work(update) sashelp.tmplmst(read);

ods listing close;
ods tagsets.excelxp file="&o_path.Program inventory.xls" style=htmlblue
                    options(frozen_headers        = '1'
                            autofilter            = 'all'
                            sheet_name            = 'Inputs and macros used'
                            absolute_column_width = '15, 50, 50'
                            );

proc print data=final label noobs;
run;

ods tagsets.excelxp close;
ods listing;

%mend extdsnman;
```

To include this macro in your SAS program, just run the following piece of code:

```
filename mac temp;
proc http
url="https://raw.githubusercontent.com/XianhuaZeng/PharmaSUG/master/2018/extd
snman.sas"
    method="GET"
    out=mac;
run;

filename code temp;
data _null_;
    file code;
    infile mac;
    input;
    put _infile_;
run;
```

```
%inc code;
```

OUTPUT

We use ODS TAGSETS.EXCELXP to produce output in Excel. A part of program inventory is shown below:

Program name	Inputs	Macros used
adsl.sas	zr, dm, ds, fa, is, pc, ex, suppex, cm, sv, vs	
adlb.sas	lb, adsl	ctcgrd.sas, attrib.sas
tefms.sas	adacr, adsl	waldci_pval.sas, page_optimize.sas
gefiga.sas	adbdc, adcm, adsl, adacr, admh	stratcifig.sas

Figure 2. Program Inventory

CONCLUSION

The macro described in this paper is a very useful tool which can greatly improve the efficiency of creating program inventory for ADRG. The usage of FOPEN/FREAD/FGET functions in this macro can also be used for other cases where retrieving multiple files, such as QC compare outputs and SAS log files.

REFERENCE

PhUSE CSS Working Group. "Analysis Data Reviewer's Guide (ADRG), v1.1". Available at http://www.phusewiki.org/wiki/images/d/df/ADRG_v1.1_2015-01-26.zip

Jeff Avallone. "Regexper". Available at http://www.regexper.com/.

CONTACT INFORMATION

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

Name: Xianhua Zeng
Enterprise: PAREXEL International
Address: 9F & Unit A/B/C 10F, No.506, Shangcheng Road, Pudong District, Shanghai, China, 200120
Work Phone: +86 21-51118305
Fax: +86 21 61609196
E-mail: Allen.Zeng@PAREXLE.com
Web: http://www.parexel.com/
http://www.xianhuazeng.com/