PharmaSUG 2013 - Paper CC07

Tracking the Use of Standard Programs in Clinical Trials

Adel Salem, Novo Nordisk A/S, DK-2860 Søborg, Denmark

ABSTRACT

In pharmaceutical companies, hundreds of programs are used in each trial to generate the needed outputs (Tables, Listings and Graphs). Some of these programs are Standard Programs (programs that can be reused in other trials) and some of them are custom programs (programs that are special and cannot be reused in other trials).

As a trial programmer it can be helpful to know how many outputs are generated by each program, and you must also be sure that you are using the newest version of a given standard program. Some companies use Programming Plans (Excel sheet with a lot of information about each output in the trial, like Title, Program Name, Program Type, Output Name, ID , Programmer, Reviewer, … etc.) . The Programming Plan can of course give an overview of the outputs, but you have to do a lot of filtering and you might get wrong information if the Programming Plan is updated manually. Besides you will not be informed if you are using an old version of a Standard Program.

It could be nice if there was a tool that could give the programmer information about how many outputs are actually generated by each program, and whether it is a Standard or Custom program. This information can be used to enhance standardization and minimize custom programming. It can also be used to estimate the needed programming resources in the trial. If this tool also can alert the programmer, in case a newer version of a given Standard Program exists, then this tool will be perfect.

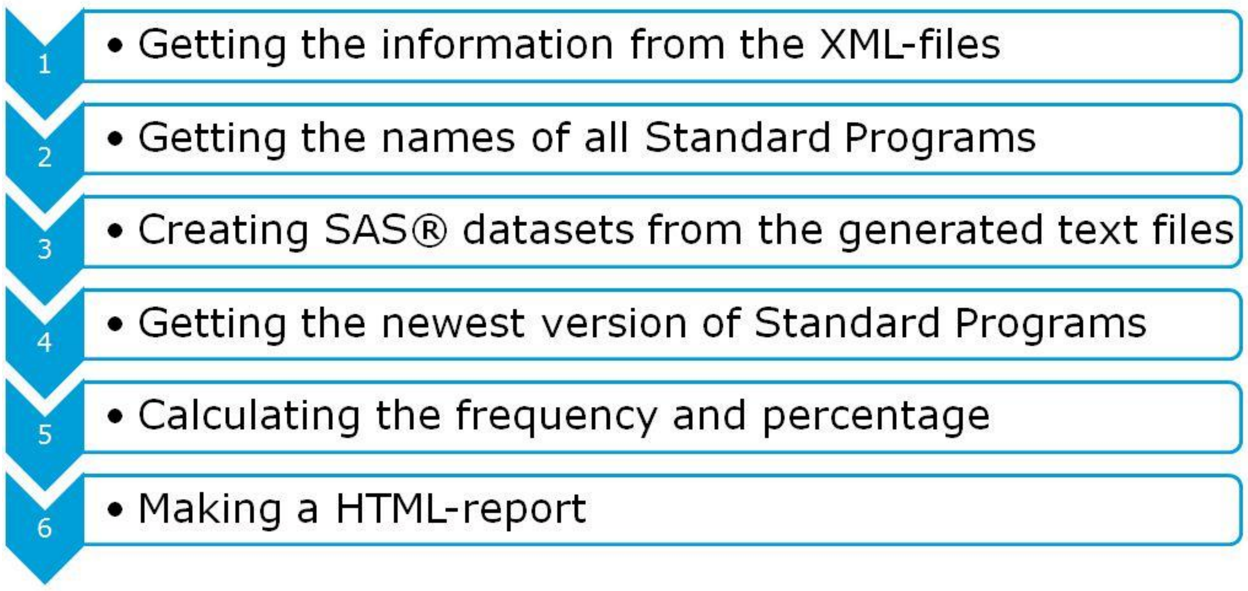
INTRODUCTION

For each created output, a XML-file containing information about the output is also created. The original purpose of the XML-files is to be used by a utility to put the outputs in the End Of Text (EOT). A macro called %usestat was developed. The %usestat is a SAS® macro that reads these XML-files to get the program name and timestamp. This is done by submitting a UNIX shell command that extract the program name and timestamp and pipe them to a text file. The macro submits also UNIX commands to extract the names of the Standard Programs from the Standard Program Library, and pipe them to text files.

The text files are read into SAS datasets that are merged to combine the information and give the overview.

The output from this macro is a HTML-report with information about the number, percentage and type of outputs created by each program, and a warning if you are using an old version of a Standard Program.

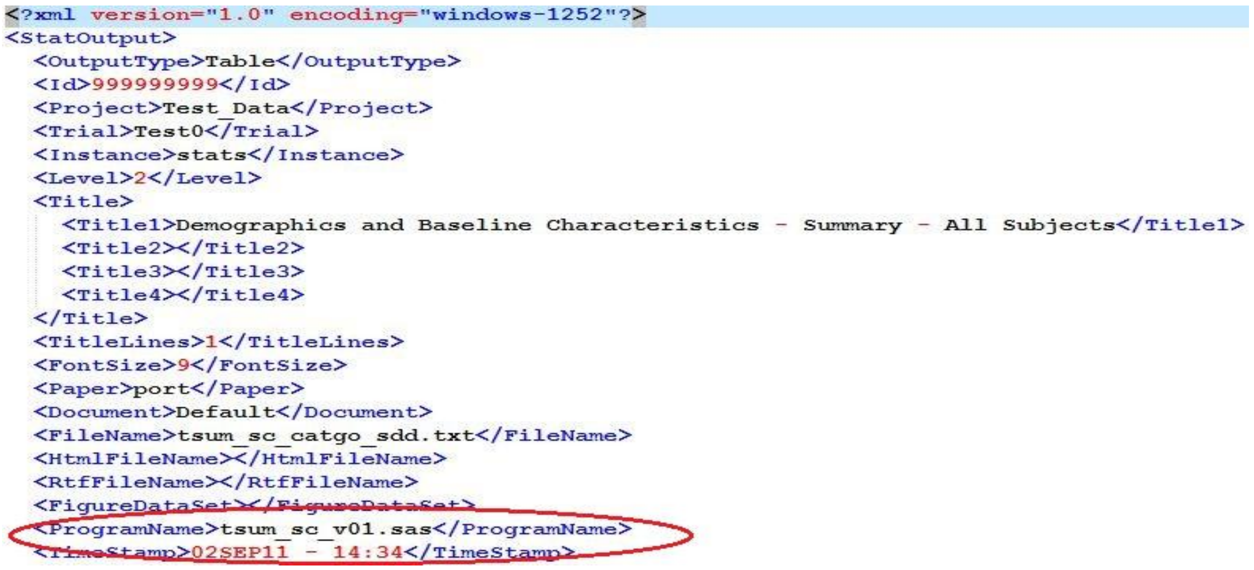
The macro also gives the programmer the ability to subset on Program Name, Type and Timestamp to get exact information about what was executed and when, by using the Where-clause parameter.


1
• Getting the information from the XML-files
2
• Getting the names of all Standard Programs
3
• Creating SAS® datasets from the generated text files
4
• Getting the newest version of Standard Programs
5
• Calculating the frequency and percentage
6
• Making a HTML-report

Display 1: The process flow

1 GETTING THE INFORMATION FROM THE XML-FILES

A UNIX command is used to get the program name and timestamp from all XML-files and pipe them to a text file.



```
<?xml version="1.0" encoding="windows-1252"?>
<StatOutput>
  <OutputType>Table</OutputType>
  <Id>999999999</Id>
  <Project>Test_Data</Project>
  <Trial>Test0</Trial>
  <Instance>stats</Instance>
  <Level>2</Level>
  <Title>
    <Title1>Demographics and Baseline Characteristics - Summary - All Subjects</Title1>
    <Title2></Title2>
    <Title3></Title3>
    <Title4></Title4>
  </Title>
  <TitleLines>1</TitleLines>
  <FontSize>9</FontSize>
  <Paper>port</Paper>
  <Document>Default</Document>
  <FileName>tsum_sc_catgo_sdd.txt</FileName>
  <HtmlFileName></HtmlFileName>
  <RtfFileName></RtfFileName>
  <FigureDataSet></FigureDataSet>
  <ProgramName>tsum_sc_v01.sas</ProgramName>
  <TimeStamp>02SEP11 - 14:34</TimeStamp>
```

Output 1: Capturing the program name and timestamp from the xml-file

```
%sysexec %str(find /projstat/&project./&trial./&instance./stats/output -name
'*.xml'
-exec egrep -i "<ProgramName>|<TimeStamp>" '{}' \; >
/projstat/&project./&trial./&instance./stats/output/my_search.txt);
```

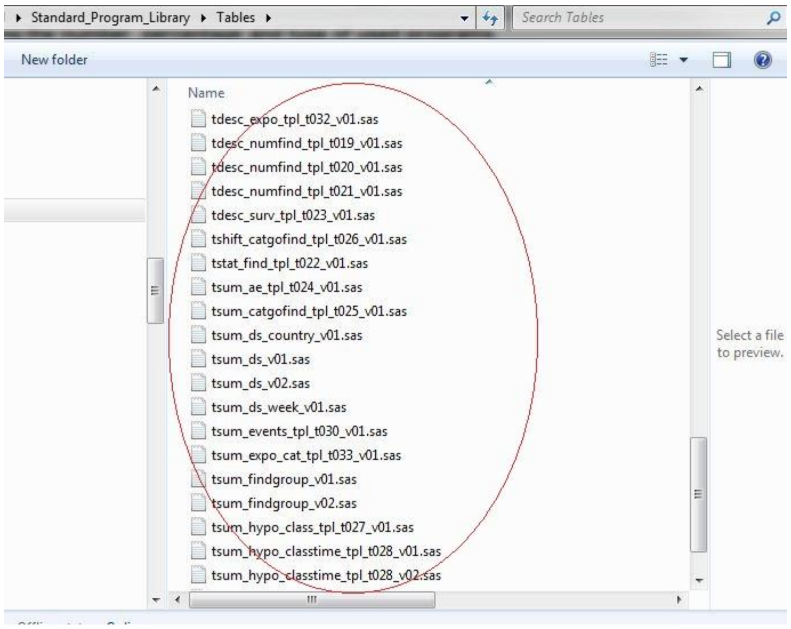
2 GETTING THE NAMES OF ALL STANDARD PROGRAMS

UNIX commands are used to get the names of the standard programs (Tables, Listings and Graphs) from the Standard Program Library, and pipe them to text files.

```
%sysexec find /projstat/general/Standard_Program_Library/Tables \( -name Doc -prune
\) -o -name '*.sas' > /projstat/&project./&trial./&instance./stats/output/tables.txt
;


%sysexec find /projstat/general/Standard_Program_Library/Listings \( -name Doc -
prune \) -o  -name '*.sas' >
/projstat/&project./&trial./&instance./stats/output/listings.txt ;


%sysexec find /projstat/general/Standard_Program_Library/Graphs \( -name Doc -prune
\) -o  -name '*.sas'  >
/projstat/&project./&trial./&instance./stats/output/graphs.txt ;
```


Standard_Program_Library ▸ Tables ▸
Search Tables
New folder
Name
Select a file to preview.

```
tdesc_expo_tpl_t032_v01.sas
tdesc_numfind_tpl_t019_v01.sas
tdesc_numfind_tpl_t020_v01.sas
tdesc_numfind_tpl_t021_v01.sas
tdesc_surv_tpl_t023_v01.sas
tshift_catgofind_tpl_t026_v01.sas
tstat_find_tpl_t022_v01.sas
tsum_ae_tpl_t024_v01.sas
tsum_catgofind_tpl_t025_v01.sas
tsum_ds_country_v01.sas
tsum_ds_v01.sas
tsum_ds_v02.sas
tsum_ds_week_v01.sas
tsum_events_tpl_t030_v01.sas
tsum_expo_cat_tpl_t033_v01.sas
tsum_findgroup_v01.sas
tsum_findgroup_v02.sas
tsum_hypo_class_tpl_t027_v01.sas
tsum_hypo_classtime_tpl_t028_v01.sas
tsum_hypo_classtime_tpl_t028_v02.sas
```

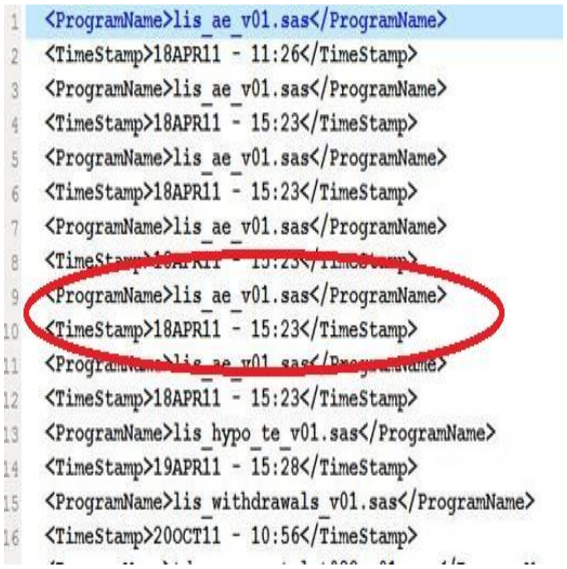
Display 2: Capturing the program names from the Standard Program Library

3 CREATING SAS DATASETS FROM THE GENERATED TEXTFILES

SAS datasets are created from the text files. Each output has 2 records in the text file. These must be transformed to one row in the SAS dataset.

```
filename infil "/projstat/&project./&trial./&instance./stats/output/my_search.txt";

data new;
infile infil delimiter='&';
length prognm $100 timestp $100 ;
input prognm /
timestp;
run;
```



```
<ProgramName>lis_ae_v01.sas</ProgramName>
<TimeStamp>18APR11 - 11:26</TimeStamp>
<ProgramName>lis_ae_v01.sas</ProgramName>
<TimeStamp>18APR11 - 15:23</TimeStamp>
<ProgramName>lis_ae_v01.sas</ProgramName>
<TimeStamp>18APR11 - 15:23</TimeStamp>
<ProgramName>lis_ae_v01.sas</ProgramName>
<TimeStamp>18APR11 - 15:23</TimeStamp>
<ProgramName>lis_ae_v01.sas</ProgramName>
<TimeStamp>18APR11 - 15:23</TimeStamp>
<ProgramName>lis_ae_v01.sas</ProgramName>
<TimeStamp>18APR11 - 15:23</TimeStamp>
<ProgramName>lis_hypo_te_v01.sas</ProgramName>
<TimeStamp>19APR11 - 15:28</TimeStamp>
<ProgramName>lis_withdrawals_v01.sas</ProgramName>
<TimeStamp>20OCT11 - 10:56</TimeStamp>
```

Display 3: Reading 2 records for each output from the text file.

4 GETTING THE NEWEST VERSION OF STANDARD PROGRAMS

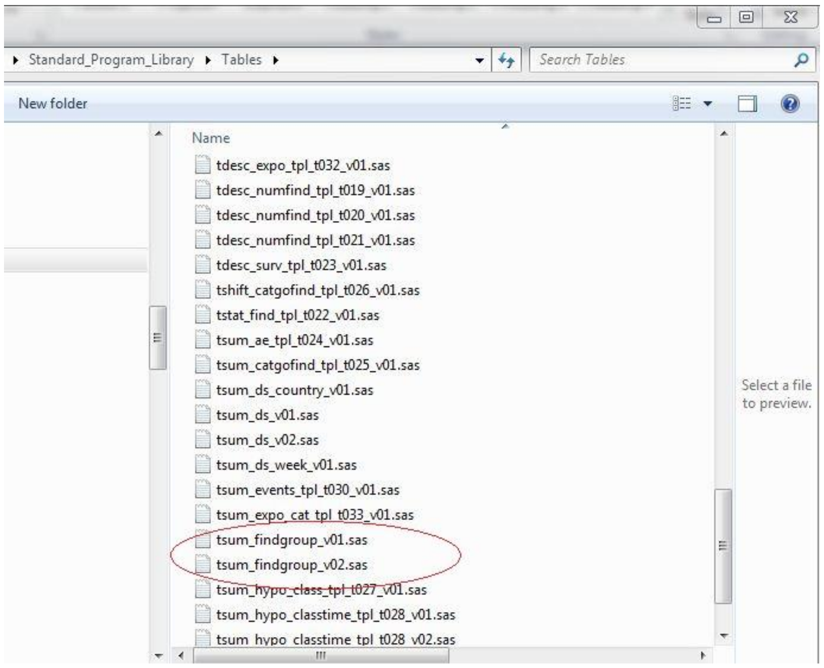
The dataset with program names and timestamp is merged with the dataset with the names of all standard programs to determine whether the program is a Standard program or a Custom program.

The newest version of each Standard Program is determined from the program name, as the version number I added as a suffix to the program name (e.g. tsum_findgroup_v02.sas) as it can be seen in display 4. See the entire code in the appendix.

```
data test2;
set standard2;
vers=input(reverse(substr(left(reverse(std_program)),5,2)),2.0);
pname=left(reverse(substr(left(reverse(std_program)),9)));
run;

proc sort data=test2 out=test3;
by pname vers;
run;

data test3;
set test3(where=(vers ne .)) ;
by pname vers;
if last.pname;
run;
```



tdesc_expo_tpl_t032_v01.sas
tdesc_numfind_tpl_t019_v01.sas
tdesc_numfind_tpl_t020_v01.sas
tdesc_numfind_tpl_t021_v01.sas
tdesc_surv_tpl_t023_v01.sas
tshift_catgofind_tpl_t026_v01.sas
tstat_find_tpl_t022_v01.sas
tsum_ae_tpl_t024_v01.sas
tsum_catgofind_tpl_t025_v01.sas
tsum_ds_country_v01.sas
tsum_ds_v01.sas
tsum_ds_v02.sas
tsum_ds_week_v01.sas
tsum_events_tpl_t030_v01.sas
tsum_expo_cat_tpl_t033_v01.sas
tsum_findgroup_v01.sas
tsum_findgroup_v02.sas
tsum_hypo_class_tpl_t027_v01.sas
tsum_hypo_classtime_tpl_t028_v01.sas
tsum_hypo_classtime_tpl_t028_v02.sas

Display 4: Getting the newest versions of standard programs.

5 CALCULATING THE FREQUENCY AND PERCENTAGE

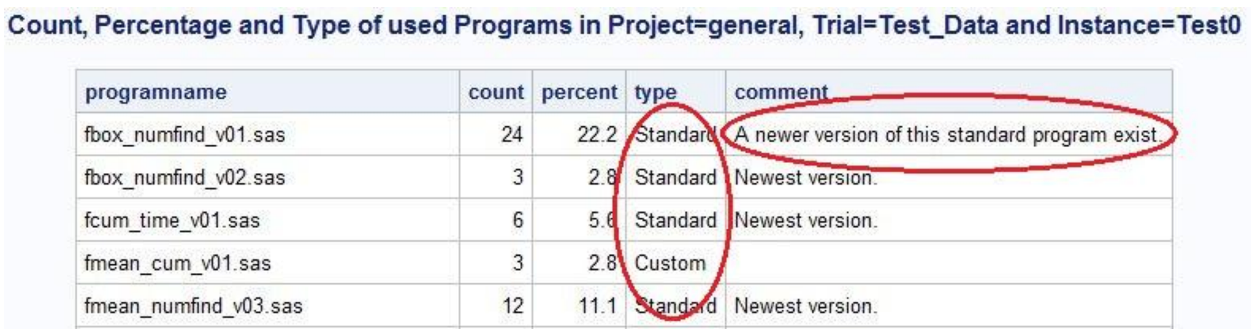
PROC FREQ is used to find the count, percentage and type of the used programs. See the entire code in the appendix.

```
proc freq data=finals noprint;
%if %superq(where_clause) ne %str() %then
%do;
   where &where_clause.;
%end;
table programname/nocum out=freqoutall;
run;

proc freq data=finals noprint;
%if %superq(where_clause) ne %str() %then
%do;
   where &where_clause.;
%end;
table type/nocum out=freqouttype;
run;
```

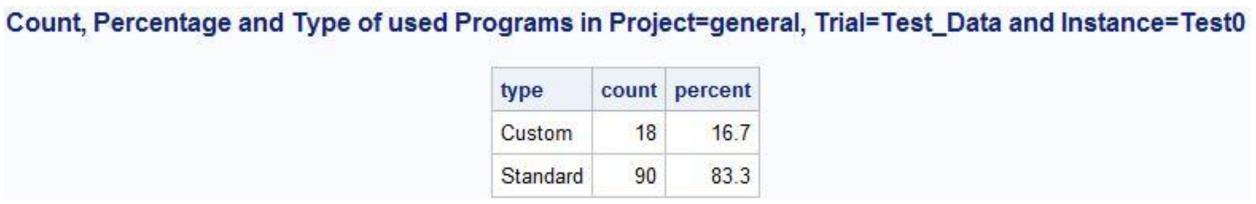
6 MAKING A HTML-REPORT

```
ods listing close;
ods html body="use_report.html" path=&OUTPATH.;
proc print data=freqoutall_final noobs;
run;
proc print data=freqouttypes noobs;
run;
title;
ods html close;
```



Count, Percentage and Type of used Programs in Project=general, Trial=Test_Data and Instance=Test0

programname	count	percent	type	comment
fbox_numfind_v01.sas	24	22.2	Standard	A newer version of this standard program exist.
fbox_numfind_v02.sas	3	2.8	Standard	Newest version.
fcum_time_v01.sas	6	5.6	Standard	Newest version.
fmean_cum_v01.sas	3	2.8	Custom	
fmean_numfind_v03.sas	12	11.1	Standard	Newest version.


Count, Percentage and Type of used Programs in Project=general, Trial=Test_Data and Instance=Test0

type	count	percent
Custom	18	16.7
Standard	90	83.3

Output 2: HTML-report showing the number, percentage and type of used programs.

EXAMPLES OF CALLING THE MACRO

```
•%usestat(project=general,trial=Test_Data,instance=Test0,where_clause=);
```

```
•%usestat(project=general,trial=Test_Data,instance=Test0,
where_clause=programname=:'tsum');
```

```
•%usestat(project=general,trial=Test_Data,instance=Test0,
where_clause= datepart(timestamp)>= '19sep2011'd);
```

```
•%usestat(project=general,trial=Test_Data,instance=Test0,
where_clause=upcase(type)='CUSTOM');
```

CONCLUSION

This %usestat macro is an easy to use utility that gives the programmer an overview of how many outputs were created by each program and whether it is a Standard or Custom program. Besides it alerts the programmer when he/she is using an old version of a Standard Program. The utility gives also the total number of outputs in the output folder and can therefore be used to check that the number of output is identical to what is planned in the Programming Plan.

The idea of this macro utility is to make use of UNIX shell commands in the SAS macro to extract information from many files (XML-files, log-files, programs… etc.) and transform them to knowledge. By using UNIX shell programming commands, the programmer can take advantage of these powerful commands to reduce the code size. Combining SAS macro programming and UNIX shell programming commands makes the code compact and gives excellent performance. See the entire code in the appendix.

CONTACT INFORMATION

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

Name: Adel Salem
Enterprise: Novo Nordisk A/S
Address: Vandtaarnsvej 114
City, State ZIP: 2860 Soeborg
Work Phone: +45 30752684
E-mail: aesa@novonordisk.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.

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

APPENDIX

```
%macro usestat(project=,trial=,instance=current,where_clause=);

*Find the standard listings.;
%sysexec find /projstat/general/Standard_Program_Library/Listings \( -name Doc -prune \) -o  -
name '*.sas'  >
             /projstat/&project./&trial./&instance./stats/output/listings.txt ;

filename inlist "/projstat/&project./&trial./&instance./stats/output/listings.txt";
%let OUTPATH="/projstat/&project./&trial./&instance./stats/output";

data listing;
   infile inlist;
   length path $100 ;
   input path;
run;

data listings(keep=std_program);
   set listing;
   std_program=reverse(substr(reverse(path),1,index(reverse(path),'/')-1));
run;

*Find the standard tables.;
%sysexec find /projstat/general/Standard_Program_Library/Tables \( -name Doc -prune \) -o  -name
'*.sas'  >
             /projstat/&project./&trial./&instance./stats/output/tables.txt ;

filename intable "/projstat/&project./&trial./&instance./stats/output/tables.txt";

data table;
   infile intable;
   length path $100 ;
   input path;
run;

data tables(keep=std_program);
   set table;
   std_program=reverse(substr(reverse(path),1,index(reverse(path),'/')-1));
run;

*Find the standard Graphs.;
%sysexec find /projstat/general/Standard_Program_Library/Graphs \( -name Doc -prune \) -o  -name
'*.sas'  >
             /projstat/&project./&trial./&instance./stats/output/graphs.txt ;

filename ingraph "/projstat/&project./&trial./&instance./stats/output/graphs.txt";

data graph;
   infile ingraph;
   length path $100 ;
   input path;
run;

data graphs(keep=std_program);
   set graph;
   std_program=reverse(substr(reverse(path),1,index(reverse(path),'/')-1));
run;

data standard;
   set listings tables graphs;
run;

proc sort data=standard out=standards;
   by std_program;
run;

*Find the executed programs.;
* find xml-files in the specified path and get the lines from these xml-files that contains
```

```
  <ProgramName> or <TimeStamp> and write them to the file my_search.txt.;

%sysexec %str(find /projstat/&project./&trial./&instance./stats/output -name '*.xml'
-exec egrep  -i "<ProgramName>|<TimeStamp>"  '{}' \; >
/projstat/&project./&trial./&instance./stats/output/my_search.txt);

filename infil "/projstat/&project./&trial./&instance./stats/output/my_search.txt";

data new;
   infile infil delimiter='&';
   length prognm $100 timestp $100 ;
   input prognm /
   timestp;
run;

data new1(keep=programname timestp2);
   set new;
   programname=substr(prognm,index(prognm,'<ProgramName>')+13,
   index(prognm,'</ProgramName>')-index(prognm,'<ProgramName>')-13);

   timestp2=substr(timestp,index(timestp,'<TimeStamp>')+11,
   index(timestp,'</TimeStamp>')-index(timestp,'<TimeStamp>')-11);
run;

data new2(drop=timestp2);
   format timestamp datetime.;
   set new1;
   timestamp=input(timestp2,datetime.);
run;

proc sort data=new2 out=new22;
   by programname;
run;

data finals;
   merge new22(in=a)
         standards(in=b rename=(std_program=programname));
   by programname;
   if a;
   if a and b then
   do;
      type='Standard';
   end;
   else
   do;
      type='Custom';
   end;
run;


* Count and Percentage of all Programs;

proc freq data=finals noprint;
   %if %superq(where_clause) ne %str() %then
   %do;
      where &where_clause.;
   %end;
   table programname/nocum out=freqoutall;
run;

data freqoutalls;
   length programname $100 count percent 8;
   format percent 5.1;
   set freqoutall;
run;

*Find version for standard program.;
proc sort data=standards out=standard2;
   by std_program;
run;
```

```
data test2;
   set standard2;
   vers=input(reverse(substr(left(reverse(std_program)),5,2)),2.0);
   pname=left(reverse(substr(left(reverse(std_program)),9)));
run;

proc sort data=test2 out=test3;
   by pname vers;
run;

data test3;
   set test3(where=(vers ne .)) ;
   by pname vers;
   if last.pname;
run;

*Find version for executed program;
proc sort data=freqoutalls out=freqoutall2;
   by programname;
run;

data test4;
   set freqoutall2;
   ver=input(reverse(substr(left(reverse(programname)),5,2)),2.0);
   pname=left(reverse(substr(left(reverse(programname)),9)));
run;

proc sort data=test4 out=test5;
   by programname pname ver;
run;

proc sort data=finals out=final nodupkey;
   by programname;
run;

data test6;
   merge test5(in=a)
         final(in=b keep=programname type);
   by programname;
   if a;
run;

proc sort data=test6;
   by pname;
run;

data freqoutall_final(drop=pname ver vers std_program);
   merge test6(in=a)
         test3(in=b);
   by pname;
   if a;
   if ver < vers and type='Standard' then
   do;
      comment='A newer version of this standard program exist.';
   end;
   else if ver >= vers and type='Standard' then
   do;
      comment='Newest version.';
   end;
   else
   do;
      comment='';
   end;
run;

* Count and Percentage of Program Types;

proc freq data=finals noprint;
   %if %superq(where_clause) ne %str() %then
   %do;
      where &where_clause.;
   %end;
   table type/nocum out=freqouttype;
run;

data freqouttypes;
   length type $30 count percent 8;
   format percent 5.1;
   set freqouttype;
run;

* make html-report with count, percentage and type of standard programs used in the trial.;
%let title1=Count, Percentage and Type of used Programs in Project=&project., Trial=&trial. and
Instance=&instance.;
%let title2=where &where_clause.;
%if %superq(where_clause) ne %str() %then
%do;
   title1 "&title1.";
   title2 "&title2.";
%end;
%else
%do;
   title "&title1.";
%end;

ods listing close;
ods html body="use_report.html" path=&OUTPATH.;
proc print data=freqoutall_final noobs;
run;

proc print data=freqouttypes noobs;
run;

title;
ods html close;

%sysexec rm /projstat/&project./&trial./&instance./stats/output/my_search.txt;
%sysexec rm /projstat/&project./&trial./&instance./stats/output/tables.txt;
%sysexec rm /projstat/&project./&trial./&instance./stats/output/graphs.txt;
%sysexec rm /projstat/&project./&trial./&instance./stats/output/listings.txt;

%mend;

%usestat(project=general,trial=Test_Data,instance=Test0,where_clause=);
```