PharmaSUG 2019 - Paper AA-158

Six Useful Data Tool Macros

Ting Sa, Cincinnati Children's Hospital Medical Center

ABSTRACT

In this paper, six macros are introduced that can work as helpful tools for some common data tasks. The macro "HelpConsistency" can detect those fields that have same name but different data lengths or data types among the SAS data sets and fix the data length inconsistencies automatically. The macro "SelectVars" can select any variables in batches from a SAS data set, for e.g, the variables have similar naming patterns like the same suffix or the middle parts. The macro "ExportExcelWithFormat" can export formatted SAS data to excel files without losing the formats. The macro "FindFiles" can help users find and access their folders and files very easily. The macro "SearchReplace" can search and replace any string in the SAS programs. The macro "checkCharVarType" can provide more information for your character variables, like if the variable only contains numeric values, or contains no values or contains date and datetime values. The paper includes the SAS codes for all these six macros.

INTRODUCTION

In this paper, six macros are presented that can work as useful tools for some common data tasks. Below are the summarizations for the usage of these six macros.

1. The "HelpConsistency" macro[1]: Common tasks that we need to perform are merging or appending SAS® data sets. During this process, we sometimes get error or warning messages saying that the same fields in different SAS data sets have different lengths or different types. If the problems involve a lot of fields and data sets, we need to spend a lot of time to identify those fields and write extra SAS codes to solve the issues. However, if you use the macro "HelpConsistency" in this paper, it can help you identify the fields that have inconsistent data type or length issues. It also solves the length issues automatically by finding the maximum field length among the current data sets and assigning that length to the field. An html report is generated after running the macro that includes the information about which fields' lengths have been changed and which fields have inconsistent data type issues.

2. The "SelectVars" macro[2]: Often we need to select a subset of variables from a dataset. SAS® software has provided the "SAS variable list" from which we can select the variables without typing their names individually. However, the "SAS variable list" cannot be used in all SAS procedures, such as the SELECT statement in the SQL procedure. Also, there is no "SAS variable list" available to select variables that share a common suffix or middle part in their names. In this paper, the ""SelectVars" macro is introduced that not only incorporates the "SAS variable list" to select a subset of variables, but also can be used to select variables that share common patterns in their names. Additionally, the results from the macro can be applied to all SAS procedures.

3. The "ExportExcelWithFormat" macro[3]: When using SAS to export data to excel, the formats got lost. In this paper, we introduce the macro ""ExportExcelWithFormat" that can help you to export formatted SAS data to excel files without losing the formats. The advantage of this macro is that it only requires you to provide the input data set and the location and the name of the output excel file, it will then create the excel file that preserves all the formats in the SAS data set for you.

4. The "FindFiles" macro[4]: The "FindFiles" macro can help users find and access their folders and files very easily. By providing a path to the macro and letting the macro know which folders and files you are looking for under this path, the macro creates an HTML report that lists the matched folders and files. The best part of this HTML report is that it also creates a hyperlink for each folder and file so that when a user clicks the hyperlink, it directly opens the folder or file. Users can also ask the macro to find certain folders or files by providing part of the folder or file name as the search criterion. The results shown in the report can be sorted in different ways so that it can further help users quickly find and access their folders and files.

5. The “SearchReplace” macro[5]: The “SearchReplace” macro can search and replace any string in the SAS programs. To use the macro, the user only needs to pass the folder name, the search string to it. If the user wants to use the replacement function, the user also needs to pass the replacement string to the macro. The macro will check all the SAS programs in this folder and the subfolders to find out which files contain the search string. The macro will generate new SAS files for the replacement so that the old files will not be affected. An html report will be generated by the macro to include the original file locations, the line number of the SAS codes that contain the search string and the SAS codes with search string highlighted in yellow. If you use the replacement string function, the html report will also include the location information for the new SAS files. The location information in the html report is created with hyperlinks, so the user can directly open the files from the report.

6. The “checkCharVarType” macro[6]: The “checkCharVarType” macro can provide more information about a character variable, like if the variable only contains numeric values, or contains no values or contains date and datetime values. Based on the information we get, we can do things in batches, like convert character values to numeric values, delete those character variables with no values, make date and datetime variables be consistent across the data sets.

THE HELP_CONSISTENCY MACRO

Presented below are the SAS codes for the Help_Consistency macro:

```
%macro Help_Consistency(libnm=,datasets=%str());
   %let datasets=%sysfunc(upcase(&datasets.));
   proc sql noprint;
   *select all the variables from the data sets and save them in data set
   tmp1;
   %if &datasets.= %then %do;
   create table tmp1 as
   select libname,name,type,length,memname
   from dictionary.columns
   where libname=upcase("&libnm.") and memtype="DATA";
   %end;
   %else %do;
   create table tmp1 as
   select libname,name,type,length,memname
   from dictionary.columns
   where libname=upcase("&libnm.") and upcase(memname) in (&datasets.) and
   memtype="DATA";
   %end;
   *select all the variables that have different lengths for the same type;
   create table tmp2 as
   select * from tmp1 where name in
  (select distinct name from
  (select name,type,count(distinct length) as length_ct from tmp1
   group by name,type having calculated length_ct >1))
   and type ^="num" order by name,type,length,memname;
```

```
 *find the maximum length of a variable and save the result to tmp3;
 create table tmp3 as
 select t.*,max_length from tmp2 as t,(select name,type,max(length) as
 max_length from tmp2 group by name,type) as m
 where t.name=m.name and t.type=m.type and t.length ^=m.max_length;
quit;
*prepare the SAS codes to change the length to the maximum length;
data tmp3;
set tmp3;
length codes $500.;
codes=catx(" ","proc sql;alter table",cats(libname,".",memname),
"modify",name,cats("char(",max_length,") format=$",max_length,".;quit;"));
run;
*execute the SAS codes to make the lengh consistent;
data _null_;
set tmp3;
call execute(codes);
run;
*save the length change info to the data set tmp4;
data tmp4;
set tmp3;
length comments $200.;
comments=catx(" ","the length of data field '",name,"'
 in",libname,".",memname," has been changed from ",length,"to ",max_length);
keep comments;
run;
proc sql;
*find the varibles with different types;
create table tmp5 as
select * from tmp1 where name in
(select distinct name from
(select name,count(distinct type) as type_ct
from tmp1 group by name
having calculated type_ct >1))
 order by name,type,memname;
 quit;
 data tmp5;
 set tmp5;
```

```
  dataset=cats(libname,".",memname);
  field_name=name;
  field_type=type;
  keep dataset--field_type;
  run;
  proc sort data=tmp5;by field_name dataset field_type;
  ods html;
  title "Data Fields that have different data types";
  proc print data=tmp5;run;
  title "Data Fields whose length have been modified";
  proc print data=tmp4;run;
  ods html close;
  ods listing;
  proc datasets;delete tmp1-tmp5;run;quit;
%mend;
```

To call the macro, you just need to pass the library name of the input datasets to the macro parameter "libname" and the input SAS datasets' names to the "datasets" macro parameter. The input data sets must be saved in the same library. For example, by running the following SAS codes, you will get 3 SAS data sets "sample1","sample2", "sample3" which are saved in the working library.

```
  data sample1;a="abc";b=1;run;
  data sample2;a="abcde";run;
  data sample3;b="one";run;
```

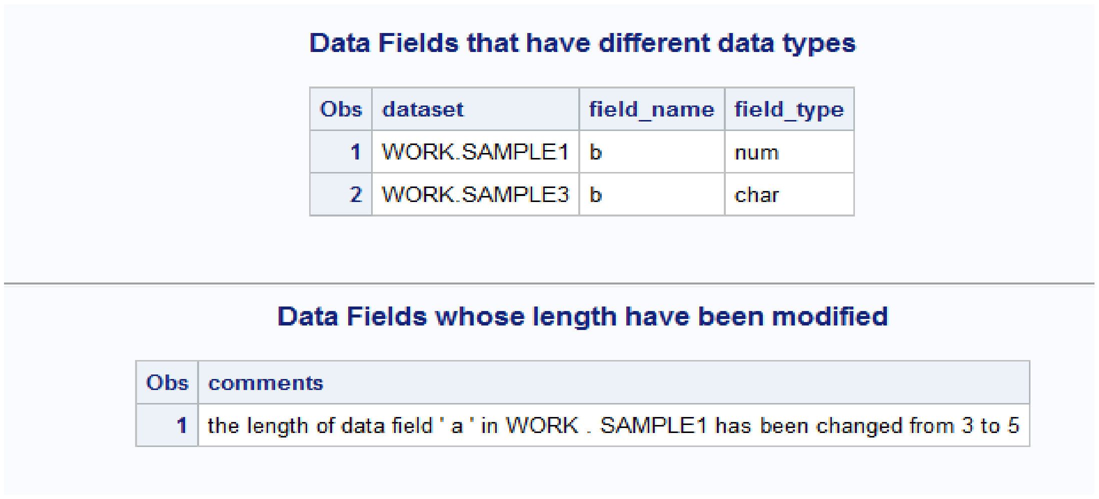
You can use the following SAS codes to check the data length and data type inconsistencies among the 3 data sets:

```
%help_consistency(libnm=work,datasets=%str('sample1','sample2','sample3'));
```

If you want to check all the data sets in a library, you don't need to pass the values to the "datesets" macro variable. For e.g, if you want to check all the data sets in the "work" library, you just need to call the macro in this way:

```
%help_consistency(libnm=work);
```

After running the macro, the macro will create an html report to summarize the field lengths that have been changed and report the data type issues if there are any. See figure1 for a sample html report.



Data Fields that have different data types

Obs	dataset	field_name	field_type
1	WORK.SAMPLE1	b	num
2	WORK.SAMPLE3	b	char

Data Fields whose length have been modified

Obs	comments
1	the length of data field ' a ' in WORK . SAMPLE1 has been changed from 3 to 5

Figure 1. The Screenshot for the HTML Report

THE SELECTVARS MACRO

Presented below are the SAS codes for the “SelectVars” macro. After calling the macro, the selected variables will be saved into a macro variable “lstVars,” and you can use this macro variable in any SAS procedures to select the variables you want.

```
%macro SelectVars(libname=,datanm=,range=_ALL_,pattern=%,separateby=%str(
));
data tmp; set &libname..&datanm.; run;
%if "&range." ^=%str() %then %do;
data tmp; set tmp; keep &range.; run;
%end;
%global lstVars;
%let lstVars=%str();*reset the macro variable lstVars;
proc sql noprint;
select name into :lstVars separated by "&separateby."
from dictionary.columns where libname="WORK" and
memname = upcase("tmp") and name like "&pattern." escape '#';
drop table tmp;
quit;
%mend;
```

- The macro parameter “libname” is used to identify the library name for the dataset from which you want to select the variables.

- The macro parameter “datanm” is used to identify the dataset name from which you want to select the variables.

- The macro parameter “range” is used to identify the range in the dataset from which you want to select the variables. The values that are accepted by the “range” parameter are those that you can use for the “SAS variable list”.

  Examples of acceptable “range” parameter values:
  1. To select the variables a1, a2, a3, simply pass “a1-a3” to the parameter “range”
  2. To select the variables from a1 to b2 based on the variable order in the dataset, simply pass “a1--b2” to the parameter “range”

3. To select the numerical variables from a1 to b2, simply pass a1-numeric-b2 to the parameter “range”
4. To select variables whose name is prefixed with “a,” simply pass “a:” to the parameter “range”
5. Likewise, the keywords “_CHARACTER_”, “_NUMERICAL_”, “_ALL_” could also be passed to the parameter “range”.
By default, the value for the “range” parameter is _ALL_, if you want to define the variable range as all the variables, you don’t need to pass a value to the “range” parameter.

- The macro parameter “patterns” is used to identify the naming patterns of the variables you want to select. In this paper, we use the “like” condition and the wildcard characters “%” and “_” in PROC SQL to help us identify the naming patterns. The wildcard character “_” could be used as a substitute for a single character, and the wildcard character “%” could be used as a substitute for zero or more characters.

  The sample code below illustrates how to use the wildcard characters. In this example, we want to select those variables whose names have the suffix “_abc” from the dataset “test” saved in the “work” library.

  ```
  proc sql;
  select name from dictionary.columns
  where libname= upcase("work") and memname = upcase("test") and
  name like "%#_abc" escape '#';
  quit;
  ```

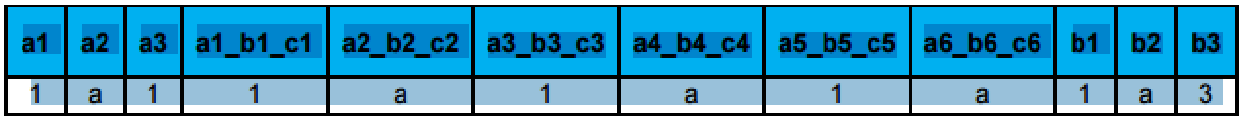
  The “escape” clause is used to search for literal instances of the percent (%) and underscore (_) characters, which are usually used for pattern matching. SAS® allows for variable names beginning with “_,” and it happens that “_” is a wildcard character. Thus, to select variables that contain the “_” in their names, we could use an ESCAPE character and add this ESCAPE character before the “_” to tell SAS® the “_” is not a wildcard character. In the above syntax, the “#” is used as the ESCAPE character.

  Suppose you want to select those variables whose names start with an “a,” followed by a character, then followed by “_123” in the middle, and end with “b.” To accomplish this, you could pass “a_#_123%b” to the parameter “pattern.”

  The default value for the “pattern” parameter is %, which means to select all the patterns. If you want to select all the patterns, you don’t need to pass a value to the “pattern” parameter.

- The macro parameter “separateby” instructs SAS® to separate the selected variables by a space or by a comma. By default, the variables are separated by a space. If you want to separate by comma, you could pass “%str(,)” to the “separateby.”

Below are some examples of calling the “SelectVars” macro to get different variables you want, figure2 is a sample SAS data set called “testdata”:



a1	a2	a3	a1_b1_c1	a2_b2_c2	a3_b3_c3	a4_b4_c4	a5_b5_c5	a6_b6_c6	b1	b2	b3
1	a	1	1	a	1	a	1	a	1	a	3

Figure 2. The “testdata” SAS Data Set

Example1: Select the variables a1, a2, a3 and separate the variables by comma:

```
%selectVars(libname=work,datanm=testdata,range=a1-a3,separateby=%str(,));
%put &lstVars.;
```

The result is: a1, a2, a3

Example2: Select the numerical variables from a1_b1_c1 to a6_b6_c6 and separate the selected variables by space:

```
%selectVars(libname=work,datanm=testdata,range=a1_b1_c1-numeric-a6_b6_c6);
%put &lstVars.;
```

The result is: a1_b1_c1 a3_b3_c3 a5_b5_c5

Example3: Select the variables whose names have prefix “a” and separate the variables by comma:

```
%selectVars(libname=work,datanm=testdata,pattern=a%,separateby=%str(,));
%put &lstVars.;
```

The result is: a1, a2, a3, a1_b1_c1, a2_b2_c2, a3_b3_c3, a4_b4_c4, a5_b5_c5, a6_b6_c6

Example4: Select the variables whose names’ middle parts are “_b”, followed by a character, and followed by “_c”, separate the variables by space:

```
%selectVars(libname=work,datanm=testdata,pattern=%#_b_#_c%);
%put &lstVars.; “#_” means treating “_” as underscore instead of the wild
card.
```

The result is: a1_b1_c1 a2_b2_c2 a3_b3_c3 a4_b4_c4 a5_b5_c5 a6_b6_c6

Example5: Select the variables whose names end with “_c”, followed by a character, separate the variables by comma:

```
%selectVars(libname=work,datanm=testdata,pattern=%#_c_,
separateby=%str(,)); %put &lstVars.;
```

The result is: a1_b1_c1, a2_b2_c2, a3_b3_c3, a4_b4_c4, a5_b5_c5, a6_b6_c6

Example6: Select the character variables from a1 to a6_b6_c6 whose names’ suffix is 2; separate the selected variables by comma:

```
%selectVars(libname=work,datanm=testdata,range=a1-character-a6_b6_c6,
pattern=%2,separateby=%str(,));
%put &lstVars.;
```

The result is: a2, a2_b2_c2

Below is an example about how to use the macro variable lstVars in the SELECT statement in the SQL procedure, make sure you have used the comma to separate the variables saved in the lstvars:

```
proc sql;
select &lstVars. from testdata;
quit;
```

THE EXPORTEXCELWITHFORMAT MACRO

Presented below are the SAS codes for the exportExcelWithFormat macro.

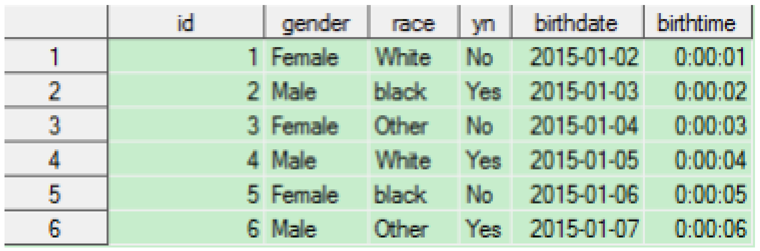
```
%macro ExportExcelWithFormat(libname=,dataname=,outputname=,sheetname=);
proc sql noprint;
create table tmp_vars as select name,format
```

```
from dictionary.columns where libname=upcase("&libname.") and
memname=upcase("&dataname.");
quit;
data tmp_vars;
set tmp_vars end=last;
length formatcode $400.;
if format ^="" then formatcode=catx("
",cats("put","(",name,",",format,")"), "as",name,",");
else formatcode=cats(name,",");
if last then formatcode=substr(formatcode,1,length(formatcode)-1);
run;
%let formatcodes=;
data _null_;
set tmp_vars;
call symput('formatcodes', trim(resolve('&formatcodes.')||' '||trim
(formatcode)));
run;
proc sql;
create view tmp_view as select &formatcodes. from &libname..&dataname.;
quit;
%let formatcodes=%str();
PROC EXPORT DATA= tmp_view OUTFILE= "&outputname." DBMS=EXCEL REPLACE;
SHEET="&sheetname.";
RUN;
proc sql;
drop table tmp_vars;
drop view tmp_view;
quit;
%mend;
```

- The “libname” is used to indicate the library name for the input dataset.
- The “dataname” is used to indicate the input SAS dataset name.
- The “outputname” is used to indicate the location and the name of the output excel file.
- The “sheetname” is used to indicate the sheet name that contains the outputs in the excel file.

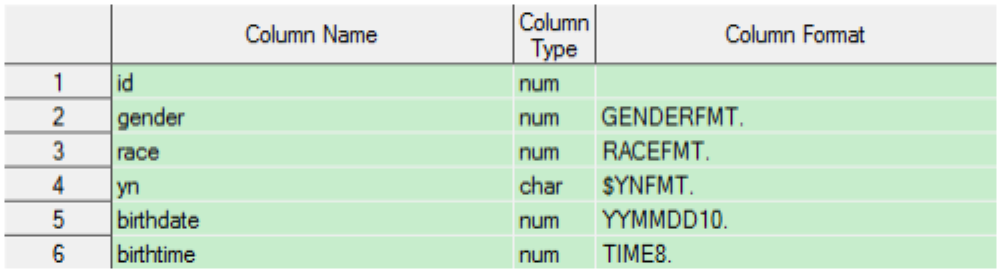
Suppose you have a SAS data set “test” saved in the working library, below are the SAS codes to generate this “test” data set.

```
proc format;
value genderfmt 0="Male" 1="Female";
value racefmt 1="White" 2="black" 0="Other";
value $YNfmt "Y"="Yes" "N"="No";
run;
data test;
do id=1 to 6;
gender=mod(id,2);
race=mod(id,3);
if gender=0 then yn="Y";else yn="N";
birthdate="01Jan2015"d+id;
birthtime="00:00:00"t+id;
format gender genderfmt. race racefmt. yn $YNfmt. birthdate yymmdd10.
birthtime time8.; output;
end;
run;
```



	id	gender	race	yn	birthdate	birthtime
1	1	Female	White	No	2015-01-02	0:00:01
2	2	Male	black	Yes	2015-01-03	0:00:02
3	3	Female	Other	No	2015-01-04	0:00:03
4	4	Male	White	Yes	2015-01-05	0:00:04
5	5	Female	black	No	2015-01-06	0:00:05
6	6	Male	Other	Yes	2015-01-07	0:00:06

Figure 3. A Sample Input Data Set “test” Saved in the “work” Libaray



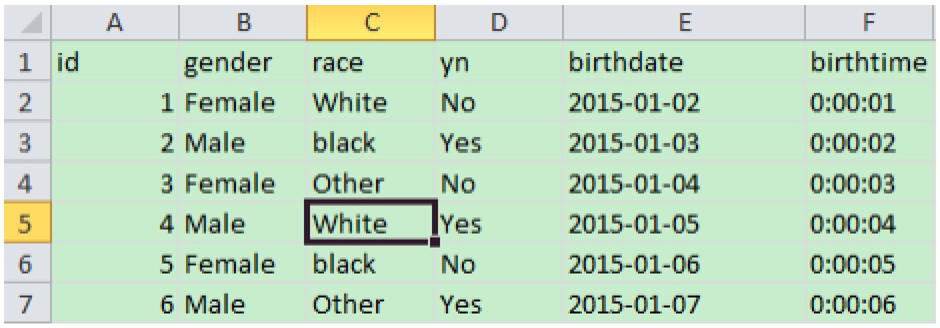
	Column Name	Column Type	Column Format
1	id	num	
2	gender	num	GENDERFMT.
3	race	num	RACEFMT.
4	yn	char	$YNFMT.
5	birthdate	num	YYMMDD10.
6	birthtime	num	TIME8.

Figure 4. Formats in the “test” SAS Data Set

You can call the macro like this:

```
%exportExcelWithFormat(libname=work,dataname=test,outputname=%str(C:\test.x
lsx), sheetname=sheet1);
```

This will create an excel file “test.xlsx” that is saved on the C: drive and the sheet name that contains the result is called “sheet1”.



	A	B	C	D	E	F
1	id	gender	race	yn	birthdate	birthtime
2	1	Female	White	No	2015-01-02	0:00:01
3	2	Male	black	Yes	2015-01-03	0:00:02
4	3	Female	Other	No	2015-01-04	0:00:03
5	4	Male	White	Yes	2015-01-05	0:00:04
6	5	Female	black	No	2015-01-06	0:00:05
7	6	Male	Other	Yes	2015-01-07	0:00:06

Figure 5. The “test.xlsx” File with the Formats

THE FINDFILES MACRO

Below are the macro codes:

```
%macro FindFiles(dirnm=,outhtml=,filetype=%str(),sortvars=%str(),
browser_type=iexplore);
filename DIRLIST pipe "dir /-c /q /s /t:w ""&dirnm""" ;
data tmp1;
length path filename $255 line $1024 owner $17 temp $16;
retain path;
```

```
infile DIRLIST length=reclen;
input line $varying1024. reclen;
if reclen = 0 then delete ;
if scan(line,1," ")='Volume' or scan(line,1," ")='Total' or
scan(line,2," ")='File(s)' or scan(line,2," ")='Dir(s)' then delete ;
dir_rec=upcase(scan(line,1," "))='DIRECTORY';
if dir_rec then path=left(substr(line,length("Directory of")+2));
else do;
date=input(scan(line,1," "),mmddyy10.);
time=input(scan(line,2," "),time5.);
post_meridian=(scan(line,3," ")='PM');
if post_meridian then time=time+'12:00:00'T ;
temp = scan(line,4," ");
if temp='<DIR>' then size=0; else size=input(temp,best.) ;
owner=scan(line,5," ");
filename=scan(line,6," ");
if filename in ('.' '..') then delete;
ndx=index(line,scan(filename,1));
filename=substr(line,ndx);
end;
run;
data tmp2;
set tmp1;
length Type $20.;
if index(filename,".")=0 then Type="Folder";
else Type=propcase(scan(filename,2,"."));
if filename ^="" then src=cats(path,"\",filename);
else src=path;
location=cats("<a href='",src,"'>",src,"</a><br>");
date_modified=catx(" ",put(date,yymmdd10.),put(time,time5.));
%if &filetype NE %str() %then %do;
if index(upcase(filename),upcase("&filetype."))>0 or
index(upcase(type),upcase("&filetype."))>0;
%end;
keep Type location date_modified;
label location="Location" date_modified="Date Modified";
run;
data tmp2;
set tmp2;
if index(lowcase(location),"service")>0 then delete;
run;
proc sort data=tmp2 nodupkey;by Location;run;
%if &sortvars NE %str() %then %do;
proc sort data=tmp2;by &sortvars.;run;
%end;
ods html file="&outhtml.";
proc print data=tmp2 noobs label;run;
ods html close;
options NOXWAIT NOXSYNC;
%if &browser_type.=iexplore %then %do;
x "start iexplore &outhtml.";
%end;
%else %if &browser_type.=chrome %then %do;
%let newhtml=%sysfunc(tranwrd(&outhtml.,\,//));
x  "start &browser_type. file://&newhtml.";
%end;
%mend;
```

- The “dirnm” is used to indicate the directory you want the macro to search for files and folders.
- The “outhtml” is used to indicate the location and the filename for the output html report.
- The “filetype” is used to indicate the type of the files you want to find, you can pass any values that are available for the column “Type” on the output html report. For e.g, if you want to find the folders, you can pass “folder” to the “filetype” parameter. The parameter “filetype” value is case-insensitive. By default, if you don’t pass any value to it, it will list all the folders and files.
- The “Sortvars” is used to indicate how you want to sort your results. To sort the results by column “Type”, pass “type” value to the macro variable “Sortvars”. To sort the results by column “Location”, pass “location” value to the macro variable “Sortvars”. To sort the results by column “Date Modified”, pass “date_modified” value to the macro variable “Sortvars”. For example, if you want to sort the results by type and descending modified date, you can pass “type descending modified_date” to the “sortvars” macro parameter. By default, if you don’t pass any value to it, it will sort the results by location.
- The “browser_type” is used to indicate which browser you want to use to open your html report. By default, it will use the internet explorer to open the report. You can also pass “chrome” to it to open the file in Google Chrome.

Below are some examples showing you how to call the macro to find different files and sort the results.

1. To Find all the files and folders inside the “C:\generate_html” folder and sort the results by type and descending modified date. The result will be saved into the “result.html” file on the C: drive. and Google Chrome will be used to open the report. You can call the macro in this way:

```
%Findfiles(dirnm=%str(C:\generate_html),sortvars=%str(type descending
date_modified),outhtml=%str(C:\result.html),browser_type=chrome);
```

2. To find all the html files inside the “C:\generate_html” folder and sort the results by location and type. The result will be saved into the “C:\result.html” file and the file will be opened by internet explorer by default. You can call the macro in this way:

```
%Findfiles(dirnm=%str(C:\generate_html),filetype=%str(.html),sortvars=%
str(loca tion type),outhtml=%str(C:\result.html));
```

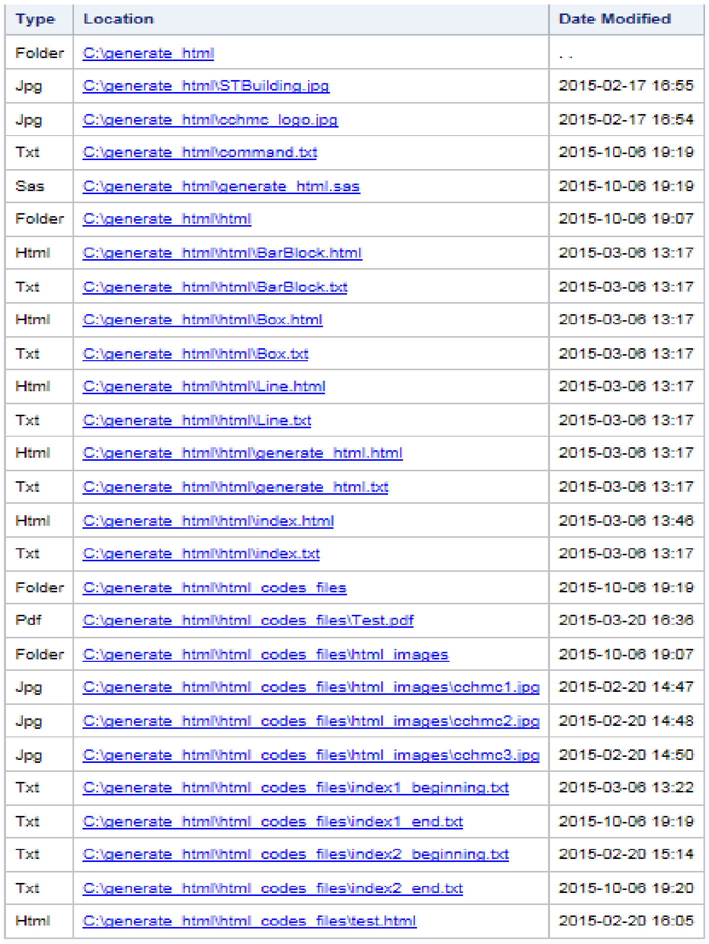
3. To find the “test.pdf” file inside the “C:\generate_html” folder, you can call the macro in this way:

```
%Findfiles(dirnm=%str(C:\generate_html),filetype=%str(test.pdf),
outhtml=%str(C:\result.html));
```

4. To find the folders inside the “C:\generate_html” folder, you can call the macro in this way:

```
%Findfiles(dirnm=%str(C:\generate_html),filetype=%str(folder),
outhtml=%str(C:\result.html));
```

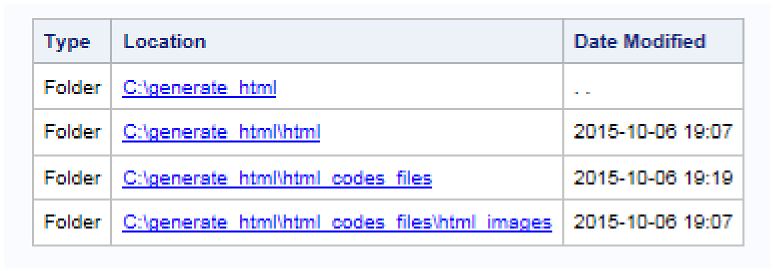
Below are the screenshots for a sample html report. Figure6 lists all the files and folders under directory "C:\generate_html". The report in figure 6 is sorted by "Location" column.



Type	Location	Date Modified
Folder	C:\generate_html	. .
Jpg	C:\generate_html\STBuilding.jpg	2015-02-17 16:55
Jpg	C:\generate_html\cchmc_logo.jpg	2015-02-17 16:54
Txt	C:\generate_html\command.txt	2015-10-06 19:19
Sas	C:\generate_html\generate_html.sas	2015-10-06 19:19
Folder	C:\generate_html\html	2015-10-06 19:07
Html	C:\generate_html\html\BarBlock.html	2015-03-06 13:17
Txt	C:\generate_html\html\BarBlock.txt	2015-03-06 13:17
Html	C:\generate_html\html\Box.html	2015-03-06 13:17
Txt	C:\generate_html\html\Box.txt	2015-03-06 13:17
Html	C:\generate_html\html\Line.html	2015-03-06 13:17
Txt	C:\generate_html\html\Line.txt	2015-03-06 13:17
Html	C:\generate_html\html\generate_html.html	2015-03-06 13:17
Txt	C:\generate_html\html\generate_html.txt	2015-03-06 13:17
Html	C:\generate_html\html\index.html	2015-03-06 13:46
Txt	C:\generate_html\html\index.txt	2015-03-06 13:17
Folder	C:\generate_html\html_codes_files	2015-10-06 19:19
Pdf	C:\generate_html\html_codes_files\Test.pdf	2015-03-20 16:36
Folder	C:\generate_html\html_codes_files\html_images	2015-10-06 19:07
Jpg	C:\generate_html\html_codes_files\html_images\cchmc1.jpg	2015-02-20 14:47
Jpg	C:\generate_html\html_codes_files\html_images\cchmc2.jpg	2015-02-20 14:48
Jpg	C:\generate_html\html_codes_files\html_images\cchmc3.jpg	2015-02-20 14:50
Txt	C:\generate_html\html_codes_files\index1_beginning.txt	2015-03-06 13:22
Txt	C:\generate_html\html_codes_files\index1_end.txt	2015-10-06 19:19
Txt	C:\generate_html\html_codes_files\index2_beginning.txt	2015-02-20 15:14
Txt	C:\generate_html\html_codes_files\index2_end.txt	2015-10-06 19:20
Html	C:\generate_html\html_codes_files\test.html	2015-02-20 16:05

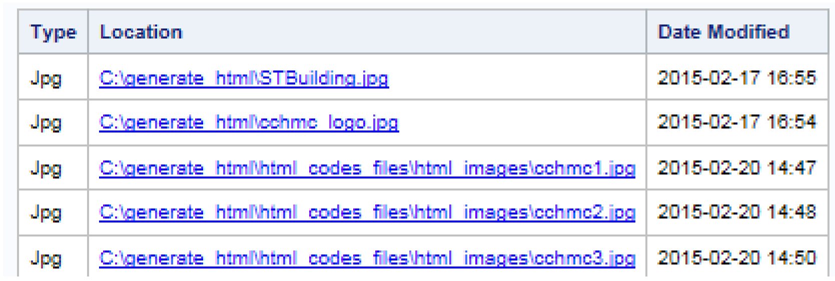
Figure 6. the Screenshot for a Sample HTML File Report

If you only want to select all the folders or select all the "jpg" files inside the folder "C:\generate_html" or find a file whose name is "test.pdf", the macro can help you do those things as well. See the screenshots in Figure7, 8 and 9.



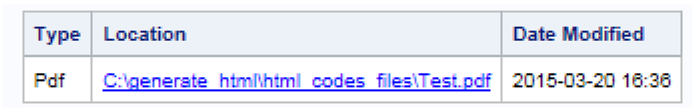
Type	Location	Date Modified
Folder	C:\generate_html	. .
Folder	C:\generate_html\html	2015-10-06 19:07
Folder	C:\generate_html\html_codes_files	2015-10-06 19:19
Folder	C:\generate_html\html_codes_files\html_images	2015-10-06 19:07

Figure 7. Find the Folders inside the "C:\generate_html" Folder



Type	Location	Date Modified
Jpg	C:\generate_html\STBuilding.jpg	2015-02-17 16:55
Jpg	C:\generate_html\cchmc_logo.jpg	2015-02-17 16:54
Jpg	C:\generate_html\html_codes_files\html_images\cchmc1.jpg	2015-02-20 14:47
Jpg	C:\generate_html\html_codes_files\html_images\cchmc2.jpg	2015-02-20 14:48
Jpg	C:\generate_html\html_codes_files\html_images\cchmc3.jpg	2015-02-20 14:50

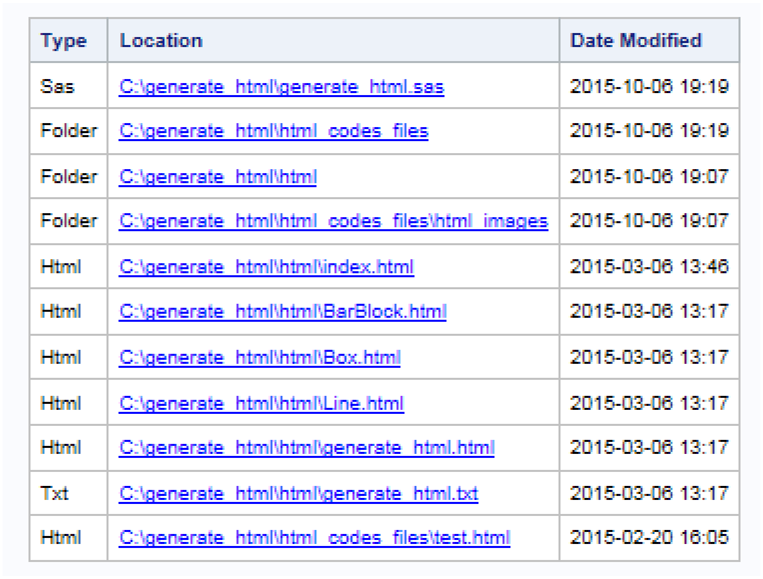
Figure 8. Find the "JPG" Files inside the "C:\generate_html" Folder



Type	Location	Date Modified
Pdf	C:\generate_html\html_codes_files\Test.pdf	2015-03-20 16:36

Figure 9. Find the "Test.pdf" File inside the "C:\generate_html" Folder

Also you can sort the listings using different ways. Figure 10 is the screenshot that sorting the results by the descending modified date.



Type	Location	Date Modified
Sas	C:\generate_html\generate_html.sas	2015-10-06 19:19
Folder	C:\generate_html\html_codes_files	2015-10-06 19:19
Folder	C:\generate_html\html	2015-10-06 19:07
Folder	C:\generate_html\html_codes_files\html_images	2015-10-06 19:07
Html	C:\generate_html\html\index.html	2015-03-06 13:46
Html	C:\generate_html\html\BarBlock.html	2015-03-06 13:17
Html	C:\generate_html\html\Box.html	2015-03-06 13:17
Html	C:\generate_html\html\Line.html	2015-03-06 13:17
Html	C:\generate_html\html\generate_html.html	2015-03-06 13:17
Txt	C:\generate_html\html\generate_html.txt	2015-03-06 13:17
Html	C:\generate_html\html_codes_files\test.html	2015-02-20 16:05

Figure 10. Find all the "HTML" Files inside the "C:\generate_html" Folder and Sort the Results by Descending Modified Date

THE SEARCHREPLACE MACRO

Below are the macro codes:

```
%macro SearchReplace
(foldernm=,searchstring=,replacestring=%str(),htmldir=%str(c:\result.html))
;
option mprint mlogic symbolgen NOXWAIT NOXSYNC;
filename ren pipe "dir ""&foldernm.\*.sas"" /b /s";
%put &foldernm.;
*dirinfo is a SAS data set that saves all the file information for the
searching folder;
data dirinfo;
infile ren pad;
```

```
input wholename $250.;
format filename $250.;
filename=cats(scan(scan(wholename,-1,'\'),1,'.'));
run;
data _null_;
set dirinfo end=end;
num=cats(_n_);
call symput("m"||num,cats(wholename));
call symput("n"||num,cats(filename));
if end then call symput("file_ct",num);
run;
%do i=1 %to &file_ct.;
*sasfile_&i. is the SAS data set that saves the SAS program codes;
data sasfile_&i.;
infile "&&m&i"
delimiter = '@@' missover dsd lrecl=32767 firstobs=1 TERMSTR=CRLF;
informat all $char5000. ;
input all $ ;
run;
data sasfile_&i.;
length wholename $200. filename $50.;
wholename="&&m&i";
filename="&&n&i";
set sasfile_&i.;
line_no=_n_;
run;
*contain_string_sasfile_&i. contains the SAS codes that have the search
string;
data contain_string_sasfile_&i.;
set sasfile_&i.;
if index(all,%str(&searchstring.)) >0;
run;
%end;
*search_result is the SAS data set that contains all the search_results;
data search_result;
set contain_string_sasfile_:;
run;
proc sort data=search_result;by wholename line_no;run;
*use this proc sql to check if the search_result data set is empty or not;
proc sql noprint;
select count(*) into :search_row_ct from search_result;
quit;
*htmlresult the data set that contains the data to be printed to the html
report;
data htmlresult;
length location $300;
%if &replacestring. ^= and &search_row_ct. ^= 0 %then %do;
length newloc $300;
%end;
set search_result;
location=cats("<a href='",wholename,"'>",wholename,"</a><br>");
%if &replacestring. ^= and &search_row_ct. ^= 0 %then %do;
newloc=cats("<a
href='",tranwrd(wholename,cats(filename,".sas"),""),cats("n_",filename,".sa
s"),"'>",tranwrd(wholename,cats(filename,".sas"),""),cats("n_",filename,".s
as"),"</a><br>");
label newloc="New File Location";
```

```
%end;
sascodes=tranwrd(all,&searchstring.,catx(" ","^S={background=yellow
font_weight=bold}",&searchstring.,"^S={}"));
run;
data htmlresult;
set htmlresult;
rename line_no=sas_codes_line_no;
drop all wholename filename;
run;
*print the result to the html report;
ods html file="&htmldir.";
ods escapechar='^';
%if &search_row_ct. ^= 0 %then %do;
proc print data=htmlresult noobs label;run;
%end;
/*if didn't find any files contain the search string, the following message
will print to the html report;*/
%else %do;
ods html text="<H1 align='center'>Don't find &searchstring. in the SAS
files.</H1>";
%end;
ods html close;
*use the internet explorer to open the html report;
x "start iexplore &htmldir.";
*do the replacement function;
%if &replacestring. ^= and &search_row_ct. ^= 0 %then %do;
*the all_sasfiles is the SAS data set that has all the SAS codes in all the
SAS programs;
data all_sasfiles;
set sasfile_:;
run;
* the new_result data set contains the new SAS codes that contain the
search string and
have been replaced by the replacement string;
data new_result;
set search_result;
newcodes=tranwrd(all,&searchstring.,&replacestring.);
run;
/*the following codes merge the replaced SAS codes to the all SAS codes
file, the data set
new_sas_files contains all the SAS codes with the newly replaced SAS codes
for those files that have
the search string;*/
proc sql;
create table sasfiles_contain_string as
select * from all_sasfiles
where wholename in (select distinct wholename from search_result)
order by wholename,line_no;
create table new_sas_files as
select s.*,newcodes
from sasfiles_contain_string as s left join new_result as n
on s.wholename=n.wholename and s.line_no=n.line_no;
quit;
data new_sas_files;
set new_sas_files;
if newcodes="" then newcodes=all;
run;
```

```
*the files data set contains the original file location and the new file
location information for those files
that have the search string;
proc sql;
create table files as
select distinct
wholename,cats(tranwrd(wholename,cats(filename,".sas"),""),cats("n_",filena
me,".sas")) as newloc
from new_sas_files;
quit;
data _null_;
set files end=end;
num=cats(_n_);
call symput("filenm"||num,cats(wholename));
call symput("loc"||num,cats(newloc));
if end then call symput("ct",num);
run;
*create new SAS files for those files that have the search string and
replace the search string
with the replacement string;
%do i=1 %to &ct.;
data f&i.;
set new_sas_files;
where wholename="&&filenm&i";
keep newcodes;
run;
data _null_;
set f&i.;
file "&&loc&i." notitles noprint;
put newcodes;
run;
proc sql;drop table f&i.;quit;
%end;
%end;
proc datasets lib=work kill;run;quit;
%MEND;
```

- The “foldernm” is used to indicate the name of the search folder.
- The “searchstring” is used to save the search string.
- The “replacestring“is used to save the replacement string. If you don’t want to use the replacement function, you don’t need to pass the value to this macro variable.
- The “htmldir“is used to indicate where you want to save the html report. If you don’t pass a value to this macro variable, by default, the html report will be saved as “c:\result.html”.

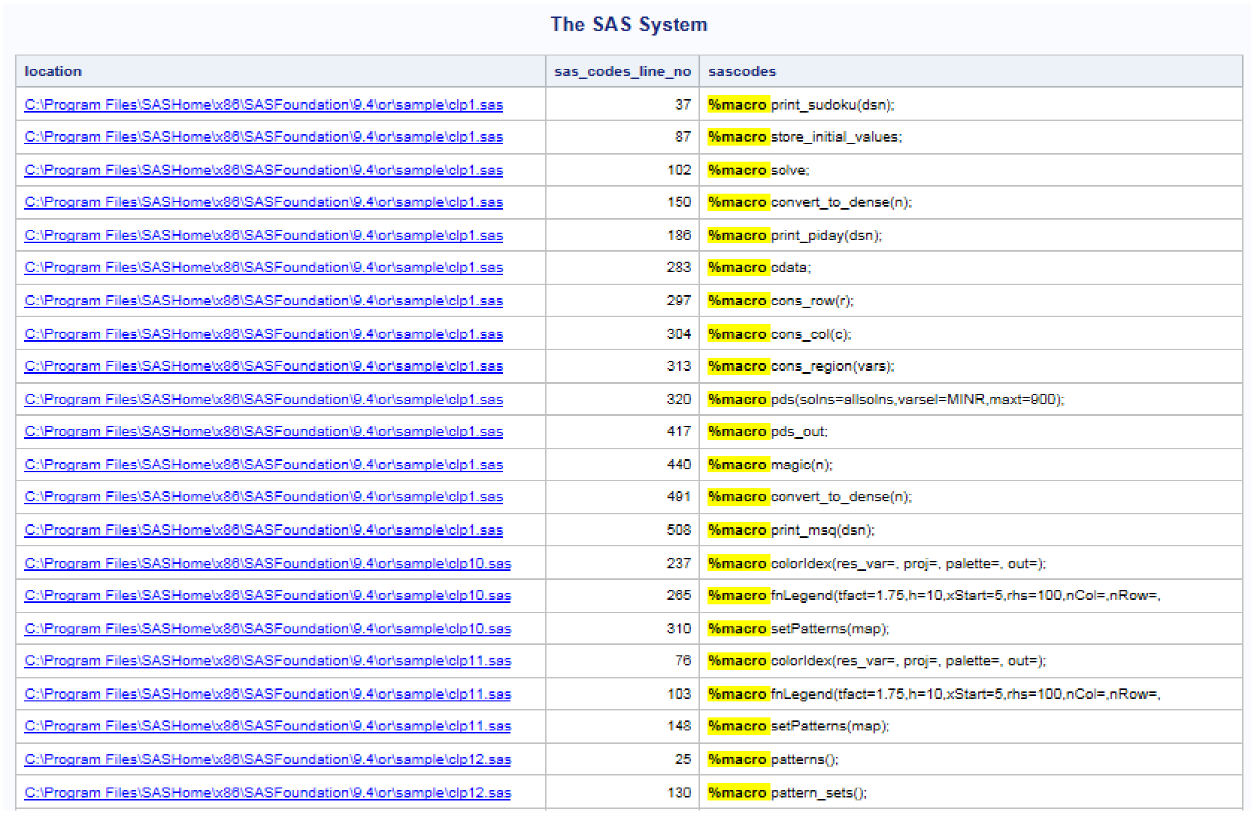
The following macro call will search the “%macro” in all the SAS programs located in the “C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample” folder and its subfolders. If the macro finds any SAS files that contain the search string, the macro will replace the search string with the replacement string “*this is a macro;%macro”. Then the macro will create new SAS files. The new SAS files are named as “n_” followed by the original file names, for e.g, if the original file name is “test. sas”, the new file name will be “n_test.sas”. The new files will be saved in the same folder as the original files. The html report will be saved as the “c:\result1.html”.

```
%SearchReplace(foldernm=%str(C:\Program Files\SASHome\x86\SASFoundation\
9.4\or\sample),searchstring=%str('%macro'),replacestring=%str('*this is a
macro;%macro'), htmldir=%str(c:\result1.html));
```

If you just want to use the search function, you don’t need to pass a value to the “replacestring” macro variable, the following SAS codes show you an example, by calling the macro in this way, it will search the string “%macro” in the “C:\test” folder and its subfolders, the html report will be saved as “c:\result.html” by default:

```
%SearchReplace(foldernm=%str(C:\test),searchstring=%str('%macro'));
```

Figure 11 shows you a sample html report generated by the macro, the search string in this example is “%macro” and the search folders are “C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample” and its subfolders.

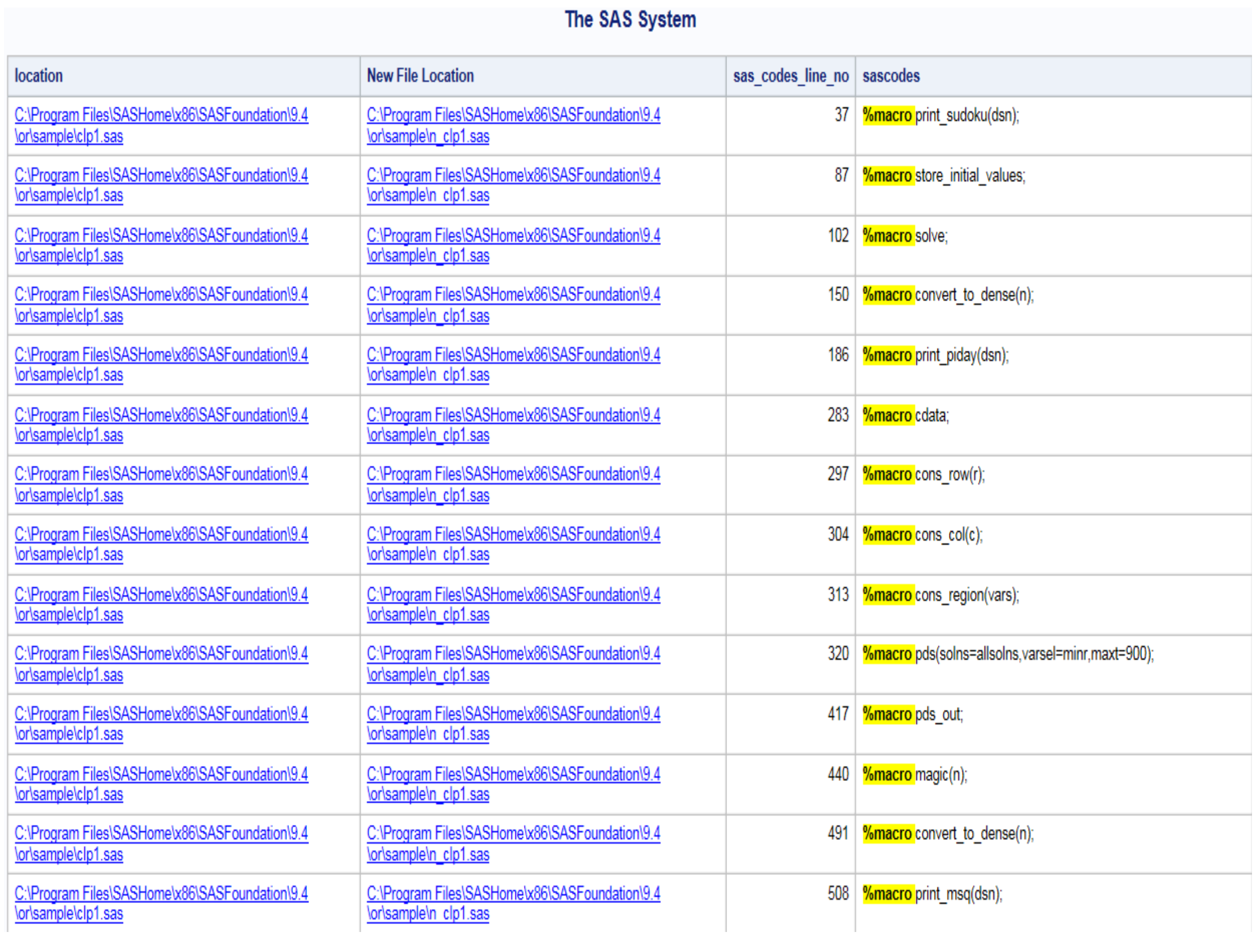


The SAS System

location	sas_codes_line_no	sascodes
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	37	%macro print_sudoku(dsn);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	87	%macro store_initial_values;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	102	%macro solve;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	150	%macro convert_to_dense(n);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	186	%macro print_piday(dsn);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	283	%macro cdata;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	297	%macro cons_row(r);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	304	%macro cons_col(c);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	313	%macro cons_region(vars);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	320	%macro pds(solns=allsolns,varsel=MINR,maxt=900);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	417	%macro pds_out;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	440	%macro magic(n);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	491	%macro convert_to_dense(n);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	508	%macro print_msq(dsn);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp10.sas	237	%macro colorIdex(res_var=, proj=, palette=, out=);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp10.sas	265	%macro fnLegend(tfact=1.75,h=10,xStart=5,rhs=100,nCol=,nRow=,
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp10.sas	310	%macro setPatterns(map);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp11.sas	76	%macro colorIdex(res_var=, proj=, palette=, out=);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp11.sas	103	%macro fnLegend(tfact=1.75,h=10,xStart=5,rhs=100,nCol=,nRow=,
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp11.sas	148	%macro setPatterns(map);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp12.sas	25	%macro patterns();
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp12.sas	130	%macro pattern_sets();

Figure 11. The Screenshot of a Sample HTML Report for the Search Function

If we pass the replacement string to the macro, the macro will create new files for those files that contain the search string, replace the search string with the replacement string in those new files. The new files’ names will have the prefix “n_” followed by the original file names. The new files will be saved in the same folder as the original files. Figure 12 shows you a sample html report for the replacement function. The search string is “%macro”, the replacement string is “*This is a macro.;%macro” and the search folders are “C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample” and its subfolders.



The SAS System

location	New File Location	sas_codes_line_no	sascodes
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	37	%macro print_sudoku(dsn);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	87	%macro store_initial_values;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	102	%macro solve;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	150	%macro convert_to_dense(n);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	186	%macro print_piday(dsn);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	283	%macro cdata;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	297	%macro cons_row(r);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	304	%macro cons_col(c);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	313	%macro cons_region(vars);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	320	%macro pds(solns=allsolns,varsel=minr,maxt=900);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	417	%macro pds_out;
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	440	%macro magic(n);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	491	%macro convert_to_dense(n);
C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\clp1.sas	C:\Program Files\SASHome\x86\SASFoundation\9.4\or\sample\n_clp1.sas	508	%macro print_msq(dsn);

Figure 12. The Screenshot of a Sample HTML Report for the Replacement Function

THE CHECKCHARVARTYPE MACRO

Below are the macro codes:

```
%macro checkCharVarType(libname=,datasets=%str(),filedir=);
%macro checkcharvars(libnm=,datanm=,varnm=,tno=);
proc sql;
create table tchar_&tno. as
select distinct "&libnm." as libnm length=32,
"&datanm." as datanm length=32,
"&varnm." as varnm length=32,
&varnm. as value length=1000
from &libnm..&datanm.(keep=&varnm.);
quit;
%mend;
%let datasets=%sysfunc(upcase(&datasets.));
proc sql noprint;
*select all the char variables from the data sets and save them in data set
tmp1;
create table tmp1 as
select libname,memname,memtype,name
from dictionary.columns
where libname=upcase("&libname.") and memtype="DATA" and type="char"
```

```
%if &datasets.^= %then %do;
and upcase(memname) in (&datasets.)
%end;
;
quit;
data tmp2;
length libnm datanm varnm $32.;
set tmp1;
keep libnm datanm varnm;
libnm=libname;
datanm=memname;
varnm=name;
attrib _all_ label='';
run;
proc sort data=tmp2;by libnm datanm varnm;run;
data tmp2;
set tmp2;
length sascodes $500.;
sascodes=cats('%checkcharvars(libnm=',libnm,',datanm=',datanm,',varnm=',varnm
,',tno=',_n_,');');
run;
data _null_;
set tmp2;
call execute(sascodes);
run;
data tmp3;
set tchar_:;
run;
proc sort data=tmp3;by value;run;
data tmp4;
set tmp3;
length chartype $50.;
if value="" then chartype="missing";
else do;
value1=compress(value,'0123456789');
if value1 in ("") or (substr(value,1,1) in ("+","-") and value1 in ("+","-"))
then chartype="integer";
else if value1 in (".") or (substr(value,1,1) in ("+","-") and value1 in
("+.","-.")) then chartype="decimal";
else if value1="--" and length(value) in (8,10) then chartype="date_hyphen";
else if value1="//" and length(value) in (8,10) then chartype="date_slash";
else if lowcase(value1) in ('jan','feb','mar','apr','may','jun',
                                'jul','aug','sep','oct','nov','dec') and
length(value) in (7,9) then chartype="date_dateM";
else if value1 in ("--::","--:::") then chartype="datetime_hyphen";
else if value1 in ("//::","//:::") then chartype="datetime_slash";
else if lowcase(value1) in ('jan::','feb::','mar::','apr::','may::','jun::',
                             'jul::','aug::','sep::','oct::','nov::','dec::',
                             'jan:::','feb:::','mar:::','apr:::','may:::',
     'jun:::', 'jul:::','aug:::','sep:::','oct:::',        'nov:::','dec:::')
then chartype="datetime_dateM";
else chartype="undefined";
end;
keep libnm datanm varnm value value1 chartype;
run;
proc sort data=tmp4 out=tmp5 nodupkey;by libnm datanm varnm chartype;run;
proc transpose data=tmp5 out=tmp5(drop=_name_) prefix=type;by libnm datanm
varnm;var chartype;run;
data tmp5;
length libnm datanm varnm $32.;
set tmp5;
alltype=catx(";",of type:);
drop type:;
run;
data tmp1;
set tmp1;
rowno=_n_;
rename name=varnm;
run;
proc sort data=tmp1;by varnm;run;
proc sort data=tmp5;by varnm;run;
data tmp1;
merge tmp1 tmp5;
by varnm;
if chartype="" then chartype="missing";
label libname="library" memname="Data Name" varnm="Variable Name"
alltype="Character Info";
run;
proc sort data=tmp1;by rowno;run;
PROC EXPORT DATA= tmp1(drop=memtype rowno libnm datanm chartype)
        OUTFILE= "&filedir."
        DBMS=EXCEL REPLACE label;
      SHEET="charinfo";
RUN;
proc datasets noprint;delete tchar: tmp:;run;quit;
%mend;
```

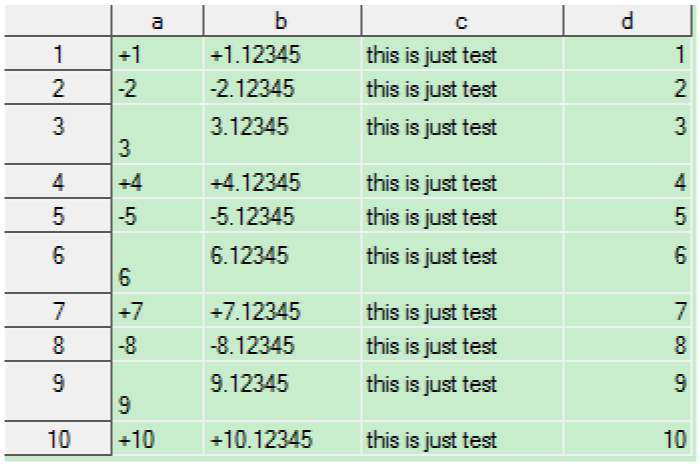
- The “libname” is used to indicate the library name for the input dataset.
- The “datasets” is used to indicate the input SAS dataset names. If you don’t pass a value to it, the macro will check all the data sets in the library.
- The “filedir” is used to indicate the file location you want

To show how the macro works, I created two SAS data sets “sample1” and “sample2” using the following SAS codes. Figure 13 and Figure 14 contain the screenshots of these two SAS data sets.

```
data sample1;
length a b c $100.;
do d=1 to 10;
if mod(d,3)=0 then a=d;
else if mod(d,3)=1 then a=cats("+",d);
else if mod(d,3)=2 then a=cats("-",d);
if mod(d,3)=0 then b=cats(d,".","12345");
else if mod(d,3)=1 then b=cats("+",d,".","12345");
else if mod(d,3)=2 then b=cats("-",d,".","12345");
c="this is just test";
output;
end;
run;
data sample2;
format e yymmdd10.;
```

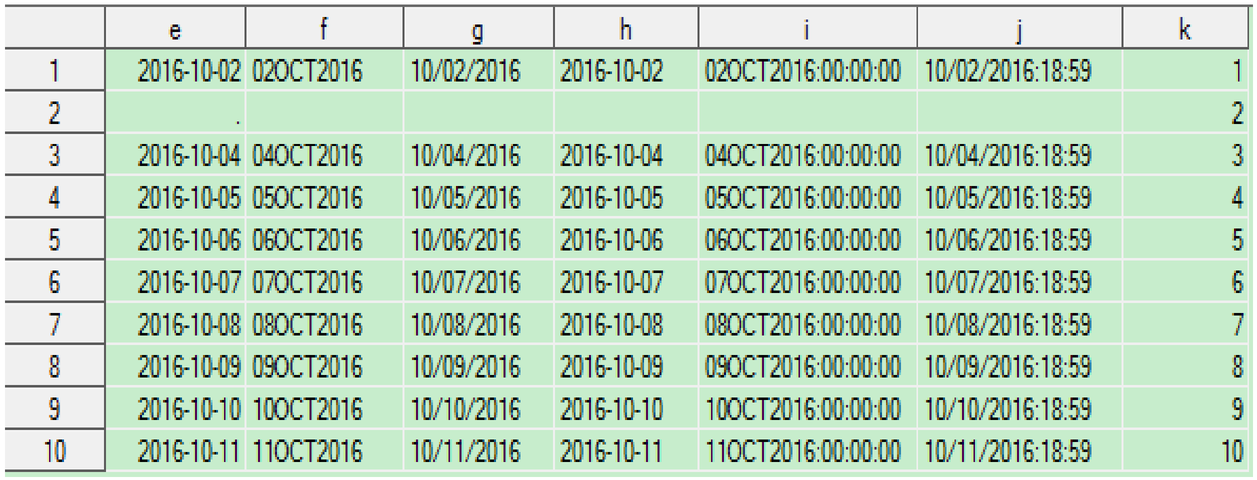
```
length f g h i j $100.;
do k=1 to 10;
e="01Oct2016"d+k;
f=put(e,date9.);
g=put(e,mmddyy10.);
h=put(e,yymmdd10.);
i=cats(put(e,date9.),":00:00:00");
j=cats(put(e,mmddyy10.),":18:59");
output;
end;
run;

data sample2;
set sample2;
if _N_=2 then call missing(of e--j);
run;
```



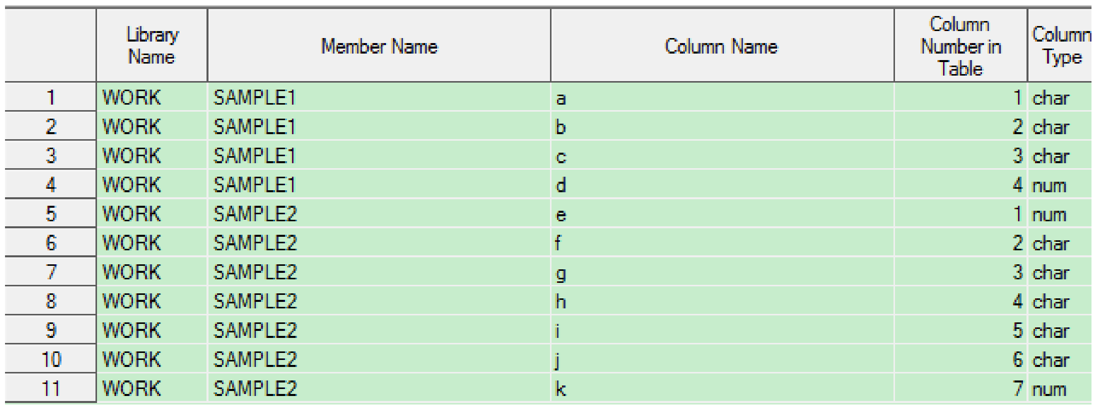
	a	b	c	d
1	+1	+1.12345	this is just test	1
2	-2	-2.12345	this is just test	2
3	3	3.12345	this is just test	3
4	+4	+4.12345	this is just test	4
5	-5	-5.12345	this is just test	5
6	6	6.12345	this is just test	6
7	+7	+7.12345	this is just test	7
8	-8	-8.12345	this is just test	8
9	9	9.12345	this is just test	9
10	+10	+10.12345	this is just test	10

Figure 13. the Screenshot for the Sample1 Data Set



	e	f	g	h	i	j	k
1	2016-10-02	02OCT2016	10/02/2016	2016-10-02	02OCT2016:00:00:00	10/02/2016:18:59	1
2	.						2
3	2016-10-04	04OCT2016	10/04/2016	2016-10-04	04OCT2016:00:00:00	10/04/2016:18:59	3
4	2016-10-05	05OCT2016	10/05/2016	2016-10-05	05OCT2016:00:00:00	10/05/2016:18:59	4
5	2016-10-06	06OCT2016	10/06/2016	2016-10-06	06OCT2016:00:00:00	10/06/2016:18:59	5
6	2016-10-07	07OCT2016	10/07/2016	2016-10-07	07OCT2016:00:00:00	10/07/2016:18:59	6
7	2016-10-08	08OCT2016	10/08/2016	2016-10-08	08OCT2016:00:00:00	10/08/2016:18:59	7
8	2016-10-09	09OCT2016	10/09/2016	2016-10-09	09OCT2016:00:00:00	10/09/2016:18:59	8
9	2016-10-10	10OCT2016	10/10/2016	2016-10-10	10OCT2016:00:00:00	10/10/2016:18:59	9
10	2016-10-11	11OCT2016	10/11/2016	2016-10-11	11OCT2016:00:00:00	10/11/2016:18:59	10

Figure 14. the Screenshot for the Sample2 Data Set



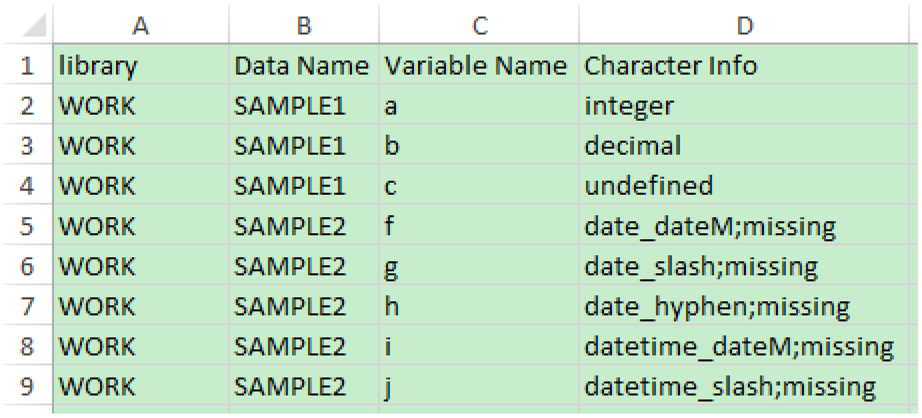
	Library Name	Member Name	Column Name	Column Number in Table	Column Type
1	WORK	SAMPLE1	a	1	char
2	WORK	SAMPLE1	b	2	char
3	WORK	SAMPLE1	c	3	char
4	WORK	SAMPLE1	d	4	num
5	WORK	SAMPLE2	e	1	num
6	WORK	SAMPLE2	f	2	char
7	WORK	SAMPLE2	g	3	char
8	WORK	SAMPLE2	h	4	char
9	WORK	SAMPLE2	i	5	char
10	WORK	SAMPLE2	j	6	char
11	WORK	SAMPLE2	k	7	num

Figure 15. the Data Type Informaton for the Sample1 and Sample2 Data Sets

For the “sample1” and “sample2” test data sets in this paper, we can call the macro using the following way and it will create an excel report “c:\charinfo.xlsx”.

```
%checkCharVarType(libname=work,datasets=%str('sample1','sample2'),
filedir=%str(C:\charinfo.xlsx));
```

Figure 16 is the screenshot for the “c:\charinfo.xlsx” report.



	A	B	C	D
1	library	Data Name	Variable Name	Character Info
2	WORK	SAMPLE1	a	integer
3	WORK	SAMPLE1	b	decimal
4	WORK	SAMPLE1	c	undefined
5	WORK	SAMPLE2	f	date_dateM;missing
6	WORK	SAMPLE2	g	date_slash;missing
7	WORK	SAMPLE2	h	date_hyphen;missing
8	WORK	SAMPLE2	i	datetime_dateM;missing
9	WORK	SAMPLE2	j	datetime_slash;missing

Figure 16. the Screenshot for the “c:\charinfo.xlsx” Report

Currently, the macro will check if a character variable contains those type of values:

(1) The macro can find out if a variable contains integers or not, even the integers contain “+” and “-“in front of the values.

(2) The macro can find out if a variable contains decimal values or not, even the decimal values contain “+” and “-“in front of the values.

(3) The macro can find out if a variable contains any missing values.

(4) The macro can find values that only contain digits and two “/”, this helps to identify those date values in mmddyy formats.

(5) The macro can find values that only contain digits and two “--”, this helps to identify those date values in yymmdd formats.

(6) The macro can find values that only contain digits and character values in this list: ('jan','feb','mar','apr','may','jun', 'jul','aug','sep','oct','nov','dec'), this helps to identify those date values in dateM formats.

(7) The macro can find values that only contain digits,":" and two "--", this helps to identify those datetime values where date is in yymmdd formats.

(8) The macro can find values that only contain digits, ":" and character values in this list: ('jan','feb','mar','apr','may','jun', 'jul','aug','sep','oct','nov','dec'), this helps to identify those datetime values where date is in dateM formats.

The macro will check all those type of values in the selected character variables, if those character variables contain those type of values, the macro will provide this information to the user. Figure 17 shows you the description for each value:

Value	**Description**
missing	contain missing values
integer	contain integer values
decimal	contain decimal values
date_hyphen	may contain date values with -
date_slash	may contain date values with /
date_dateM	may contain date values in dateM. format
datetime_hyphen	may contain datetime values with -
datetime_slash	may contain datetime values with /
datetime_dateM	may contain datetime values with dateM. format
missing value	may contain missing values
not in those defined type of character values	may contain characters values

Figure 17. the List of the Character Info Value Description

So if a variable contains missing values and integer values, the description for this variable in the final excel report will be "integer;missing".

Though the macro currently cannot be 100% accurate to check if a value is a date value or a datetime value, but it works in most of the cases.

Besides, the macro will be helpful to those users who want to change character variables to numeric variables because the macro can identify the integer and decimal values correctly. Also the macro is helpful to the users who want to delete those character variables that don't contain any values in batches.

Also the macro is working efficiently, for most of the clinical trial study, it runs fast and will give you the results in about 2 or 3 minutes at most. I just use the base SAS to run this macro. If you have a SAS server, it will be even quicker.

CONCLUSION

The six macros presented in this paper can be used as helpful tools for some common data tasks. All of the macros have their own published online paper. You can check the reference section to find each individual paper.

REFERENCES

1. Sa,T. 2016. "A Macro That Can Fix Data Length Inconsistency and Detect Data Type Inconsistency" SAS Global Forum 2016, Las Vegas, NV. Available at http://support.sas.com/resources/papers/proceedings16/10000-2016.pdf

2. Sa,T, Liu,Y.H. 2015. "A Simple Macro to Select Various Variables Lists" PharmaSUG 2015, Orlando, FL. Available at http://www.pharmasug.org/proceedings/2015/QT/PharmaSUG-2015-QT15.pdf

3. Sa,T. 2015. "Keep the Formats When Exporting to Excel" MWSUG 2015, Omaha, NE. Available at http://www.mwsug.org/proceedings/2015/PO/MWSUG-2015-PO-01.pdf

4. Sa,T. 2016. “Let SAS® Help You Easily Find and Access Your Folders and Files” SAS Global Forum 2016, Las Vegas, NV. Available at http://support.sas.com/resources/papers/proceedings16/11720-2016.pdf

5. Sa,T. 2016. “A Macro that can Search and Replace String in your SAS Programs” SAS Global Forum 2017, Orlando, FL. Available at http://support.sas.com/resources/papers/proceedings17/1136-2017.pdf

6. Sa,T. 2017. “Macro that can Provide More Information for your Character Variables” MWSUG 2017, Saint Louis, MO. Available at https://www.mwsug.org/proceedings/2017/RF/MWSUG-2017-RF05.pdf

ACKNOWLEDGMENTS

The author wishes to thank the Division of Biostatistics and Epidemiology at Cincinnati Children's Hospital Medical Center for its support.

CONTACT INFORMATION

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

Name: Ting Sa
Enterprise: Division of Biostatistics and Epidemiology, Cincinnati Children’s Hospital Medical Center
Address: 3333 Burnet Ave
City, State ZIP: Cincinnati, OH 45229
Work Phone: (513) 636-3674
E-mail: Ting.Sa@cchmc.org