Paper PO-01-2015

Keep the Formats When Exporting to Excel

Ting Sa, Cincinnati Children's Hospital Medical Center, Cincinnati, OH

ABSTRACT

When using SAS to export data to excel, the formats got lost. In this paper, we introduce a macro 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.

INTRODUCTION

When using the PROC EXPORT or the EXCELXP to export the SAS data to an excel files, the formats are lost in the output excel file. For e.g, the following SAS codes will generate a SAS data set in Figure 1:

```
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 1. A Sample Input Data Set "test" Saved in the "work" Libaray

Figure 2 lists the formats used in the "test" SAS data set:

	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 2. Formats in the "test" SAS Data Set

Then using the PROC EXPORT codes below to export the data set to an excel file, Figure 3 shows the output excel file "test.xlsx" file:

```
PROC EXPORT DATA= WORK.TEST
            OUTFILE= "C:\test.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="test";
RUN;
```

	A	B	C	D	E	F
1	id	gender	race	yn	birthdate	birthtime
2	1	1	1	N	1/2/2015	1/0/1900
3	2	0	2	Y	1/3/2015	1/0/1900
4	3	1	0	N	1/4/2015	1/0/1900
5	4	0	1	Y	1/5/2015	1/0/1900
6	5	1	2	N	1/6/2015	1/0/1900
7	6	0	0	Y	1/7/2015	1/0/1900

Figure 3. the ""test.xlsx" File

From the output, we can see that we lost all the formats and the "birthtime" values are even not displayed correctly.

One way to solve this is to create a view from the SAS data set, and the formatted variables in the view are then exported correctly with PROC EXPORT [1]. For example, I can use the following codes to create a view "test_view" based on the "test" data set and export the test_view to excel, by doing this, all the formats will be kept in the excel file, Figure 4 shows the excel result:

```
proc sql;
       create view test_view as
       select id, put(gender,GENDERFMT.) as gender,
       put(race,RACEFMT.) as race,put(yn,$YNFMT.) as yn,
       put(birthdate,YYMMDD10.) as birthdate,
       put(birthtime,TIME8.) as birthtime from test;
quit;

PROC EXPORT DATA= test_view
            OUTFILE= "C:\test.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="test";
RUN;
```

	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 4. the ""test.xlsx" File with the Formats

Although the above way solves the problem, if you have a lot of variables and formats in your data set, using the above way to write the codes manually will not be fun, therefore, in this paper, I create a macro that uses the above way to export SAS data sets to excels and keep all the formats in the excel file. To use the macro, 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.

THE EXPORTEXCELWITHFORMAT MACRO SAS CODES

Presented below are the SAS codes for the exportExcelWithFormat macro. If you need to include source code:

```
%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.

For the sample SAS data set “test”, you can call the macro like this:

```
%exportExcelWithFormat(libname=work,dataname=test,outputname=%str(C:\test.xlsx),
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”.

HOW THE MACRO EXPORTEXCELWITHFORMAT WORKS

Let me explain the macro step by step:

1. The following SAS codes will select all the variables in the input data set from the dictionary.columns table and save the variable names and the formats to the tmp_vars SAS data set. If you happen to have a data set whose name is tmp_vars also, you can either change the macro or rename your own data set.

```
Proc sql noprint;
create table tmp_vars as
select name,format from dictionary.columns
where libname=upcase("&libname.") and memname=upcase("&dataname.");
quit;
```

   Figure 5 is a screenshot for the tmp_vars data set while using the input data set "test" in this paper.

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

Figure 5. the "tmp_vars" Data Set Using the Input SAS Data Set "test"

2. The following SAS codes will create a new column "formatcode" in the tmp_vars data set. This column is used to create the "put" statement for each variable if they have a format so that later we will use those "put" statements to create the view. Figure 6 is the screenshot for the new tmp_vars data set that contains the "formatcode" statement.

```
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;
```

	Column Name	Column Format	formatcode
1	id		id,
2	gender	GENDERFMT.	put(gender,GENDERFMT.) as gender ,
3	race	RACEFMT.	put(race,RACEFMT.) as race ,
4	yn	$YNFMT.	put(yn,$YNFMT.) as yn ,
5	birthdate	YYMMDD10.	put(birthdate,YYMMDD10.) as birthdate ,
6	birthtime	TIME8.	put(birthtime,TIME8.) as birthtime

Figure 6. the "tmp_vars" Data Set with the New Column "formatcode"

3. The following SAS codes will create a macro variable formatcodes and combine all the values in the formatcode columns in the tmp_vars data set together and save them to the macro variable formatcodes.

```
%let formatcodes=;
data _null_;
set tmp_vars;
```

```
call symput('formatcodes', trim(resolve('&formatcodes.')||' '||trim
(formatcode)));
run;
```

If you use "%put &formatcodes.;" after the following codes, Figure 7 shows you the value for the macro variable formatcodes based on the input data set "test" in this paper.

```
MLOGIC(EXPORTEXCELWITHFORMAT):  %PUT &formatcodes.
SYMBOLGEN:  Macro variable FORMATCODES resolves to  id, put(gender,GENDERFMT.) as gender ,
            put(race,RACEFMT.) as race , put(yn,$YNFMT.) as yn , put(birthdate,YYMMDD10.) as
            birthdate , put(birthtime,TIME8.) as birthtime
id, put(gender,GENDERFMT.) as gender , put(race,RACEFMT.) as race , put(yn,$YNFMT.) as yn ,
put(birthdate,YYMMDD10.) as birthdate , put(birthtime,TIME8.) as birthtime
MLOGIC(EXPORTEXCELWITHFORMAT):  Ending execution.
```

Figure 7. the Value for the Macro Variable "formatcodes"

4. The following SAS codes will create the view tmp_view that will contain all the formatted data. Figure 8 is the screenshot of this tmp_view based on the input SAS data set "test".

```
proc sql;
create view tmp_view as
select &formatcodes.
from &libname..&dataname.;
quit;
```

	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 8. the "tmp_view"

5. Once the view is created, the macro just uses the proc export to export the view to the excel files and reset the macro variable formatcodes and drop the table tmp_vars and the view tmp_view.

CONCLUSION

The macro presented in this paper provides an easy way to export the SAS data sets to excel files without losing the formats. Currently the macro will keep all the formats in the data sets. There are situations that you just want to keep some of the formats, you can easily update the macro in this paper to satisfy this need as well. Also besides using the SAS view method, some people will use the ExcelXP tagset and the style statement to keep the formats. You can check reference 2 for more details about this method.

REFERENCES

1. KNOWLEDGE BASE / SAMPLES & SAS NOTES. “Usage Note 18406: Exporting formatted variables to Excel” http://support.sas.com/kb/18/406.html
2. Derby,N and McGahan,C. 2013. “Maintaining Formats when Exporting Data from SAS® into Microsoft® Excel®.” SAS Global Forum 2013, San Francisco, CA. http://support.sas.com/resources/papers/proceedings13/316-2013.pdf

ACKNOWLEDGMENTS

The author wishes to thank the Division of Biostatistics and Epidemiology at Cincinnati Children's Hospital Medical Center for its support, and particularly Robert Tamer for his helpful feedback.

CONTACT INFORMATION

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

Name: Ting Sa
Enterprise: Cincinnati Children’s Hospital Medical Center
Address: 3333 Burnet Ave
City, State ZIP: Cincinnati, OH45229
Work Phone: 513-636-3674
E-mail: ting.sa@cchmc.org