PharmaSUG 2023 - Paper QT-085

Tips to Read In and Output Excel Spreadsheets in SAS®

Jaime Yan, Merck & Co., Inc., Rahway, NJ, USA
Chao Su, Merck & Co., Inc., Rahway, NJ, USA
Changhong Shi, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

Excel is a powerful and versatile tool widely used in clinical trials to save and transfer data. Especially in organizations where research software like SAS® and R is not commonly used, researchers can only manipulate data by relying on IT. Analysis and reporting programmers play an essential role in receiving different types of excel spreadsheets and creating customized excel reports according to requests from different departments. Usable code to read in different types of spreadsheets and generate friendly interface excel outputs is beneficial and meaningful. In this paper, different methods are presented to output excel files to meet different complex requirements. This will significantly relieve the Analysis and reporting (A&R) programmers' work and improve efficiency. It also helps to smooth the collaboration among different functional groups to successfully achieve the common goal.

INTRODUCTION

The major pain point for A&R programmers to deal with the excel worksheet is to read files of various excel worksheet formats and output customized reports with a friendly interface. In this paper, the difference between various excel versions is introduced. The methods and tips for reading in different versions of excel files and exporting requested format excel spreadsheets are described. The sample codes for importing and exporting files are appended as a reference.

MAIN DIFFERENCE OF THE FORMAT

With the development of Microsoft excel, different excel worksheet formats, like xls, xlsx, xlsm, xlsb, and csv, have been used. These differences always need to be clarified for the user before the proper way is selected to import/export the excel file.

Different versions of excel support various formats. Usually, the later version is backward compatible. The earlier version of excel could not be upward compatible with some features only supported by the later version. For example, the xls format is used for excel 2003 and the previous version. The xlsx, xlsm, and xlsb are supported by excel 2007 and later versions, and the xlsx is the general format; the xlsm would support the VBA macro inside the file. The xlsb is the binary text format of excel, which has a smaller file size and would support VBA macro but needs better compatibility. In other words, the early version of excel must adequately support the file in this format.

To sum up, the format supported by the later excel version could contain more rows and columns with a smaller file size. Also, the later features are not supported by the earlier version of excel.

TWO MAIN PROCEDURE FOR READING IN FROM EXCEL

SAS provides several different procedures to read in and output excel files. Among these procedures, considering the ease of use, scalability, and whether easy to use as a module, two input and two output procedures with sample code will be discussed respectively.

LIBNAME Engine procedure:

SAS provides three LIBNAME Engines for excel: Excel LIBNAME Engine, PCFILES LIBNAME Engine, and XLSX LIBNAME Engine.

The PCFILES LIBNAME relies on Microsoft Access Database Engine(ACE) and SAS PC Files Server and works on Windows and UNIX environments; The Excel LIBNAME Engine requires the ACE and would only be used in the Windows environment. Therefore, the user should set up the PC Files Server and make sure the ACE is ready before using them. These LIBNAME Engines support various excel formats like read xlsx, .xlsb, .xlsm, .xls and create the format of .xlsx, .xlsb, .xls.

The XLSX engine only supports the .xlsx (the most commonly used format of excel files) but would read the file directly without relying on ACE and PC File Server. For xlsx format, the XLSX engine has more flexibility and is easy to use without paying attention to setting up the PC Files Server. The user does not need to worry about the painful unaligned problem. Furthermore, it can work in both Windows and UNIX environments.

LIBNAME Engine sample code:

```
libname myexcel xlsx "&excel_file_path.&slash.xxxxxx_.&extension";

data content;
    set myexcel."&sheet1"n;

    if length(dataset) <= 8 and index(dataset, "AD") > 0;
    keep dataset %col(2);

proc sort;
    by dataset;
run;
```

Proc import procedure:

This procedure would work under both Windows/UNIX/Linux environments. Option DBMS is used to specify the data type to import/export, specifically for reading/creating excel files. XLS/XLSX/EXCEL/ EXCEL4/EXCEL5/EXCELCS are all identifiers used for the DBMS option.

The DBMS=XLS and DBMS=XLSX would deal with the excel file directly without PC Files Server or ACE. The DBMS=EXCEL uses Excel LIBNAME engine while DBMS=EXCELCS uses PC Files LIBNAME engine, and PC Files Server is needed.

Proc import:

For excel file reading in, the XLS and EXCEL4, and EXCEL5 support xls format; EXCEL/EXCELCS support .xls .xlsb.xlsm.xlsx; XLSX works for xlsx format.
Proc import sample code:

```
proc import datafile = "&excel_file_path/xxxxxx_.&extension"
    out = content (keep=adam_dataset adam_dataset
    _description key_variables)
    dbms = &extension replace;
    range = "Content$A5:G100";
    getnames = yes;
run;
```

Tips for importing:

Work Tips/flow for selecting the proper method to read the excel file:

- Check the environment: Windows or UNIX.
- Check the extension/format of the target excel file.
- If .xlsx, prefer to use XLSX LIBNAME Engine.
- If other formats, use proc import and select the proper identifier for the DBMS option.
- For the xls format, check whether UTF-8 data is within the excel file. If UTF-8 is within the file, recommended to use XLSX Engine or proc import with DBMS=EXCEL or save the xls format as xlsx first to avoid transcoding problems.

TWO MAIN PROCEDURE FOR OUTPUTING EXCEL

Proc export procedure:

Proc export is a simple and easy-to-use option. It could export the prepared dataset to excel directly. However, this procedure does not support setting the title, footnote, and other content formats. Specific for creating the excel file, The EXCEL/EXCELCS to DBMS option support for .xls .xlsb. xlsx; XLSX for xlsx format. The XLS and EXCEL4, and EXCEL5 support xls format.

```
proc export data=final
    outfile="&outpath./base_character_victor.xlsx"
    dbms= xlsx REPLACE label;
    sheet="abc";
run;
```

ODS EXCEL combine result output procedure:

The ODS EXCEL combined result output procedure is another option; it is more powerful and flexible. This procedure would export an XLSX file with a table, listing, and even SAS graphics in the output. To

customize their output, the user could flexibly combine ODS EXCEL with other output result procedures like proc print, proc report, proc SQL, etc. At the same time, more format setting code is needed.

ODS EXCEL combine with proc sql sample code:

```
ods excel file = "%sysfunc(pathname(fptotb,f))/abc.xlsx"
    options (embed_footnotes_once = 'on'
    embedded_footnotes='on'
    embed_titles_once = 'on'
    embedded_titles='on'
    gridlines='off'
    sheet_interval='proc'
    sheet_name="abc"
    autofilter ="all");
;
ods escapechar='^';
title 'Title 1';
title2 'Title 2';

proc sql flow;
    select * from final;
run;

ods excel close;
```

ODS EXCEL combine with proc report sample code:

```
ods excel file = "%sysfunc(pathname(fptotb,f))/&outname..xlsx"
    options (embed_footnotes_once = 'on'
    embedded_footnotes='on'
    embed_titles_once = 'on'
    embedded_titles='on'
    gridlines='off'
    sheet_interval='proc'
    sheet_name="&outname."
    AUTOFILTER ="ALL");
;
title 'Title 1';

proc report data=final;
    column cat ('Drug A' c1 c2) ('Placebo' c3 c4) ('Total' c5 c6 );
    define cat / ' ' style(column)={asis=on width=100%} width=100;
    define c1 / 'n(%)' style(column)={asis=on width=100%} width=100;
    define c2 / 'Mean' style(column)={asis=on width=100%} width=100;
    define c3 / 'n(%)' style(column)={asis=on width=100%} width=100;
    define c4 / 'Mean' style(column)={asis=on width=100%} width=100;
```

```
    define c5 / 'n(%)' style(column)={asis=on width=100%} width=100;
    define c6 / 'Mean' style(column)={asis=on width=100%} width=100;
run;

ods excel close;
```

Tips for Exporting:

The Tips for selecting which format to use for exporting depend on the following:

- The reviewer's excel version. Generally, it is exported with xlsx format. If the user needs to use the old excel version, then use proc export with DBMS=XLS/EXCEL4/EXCEL5.
- What Features are needed? If the user needs to include the plot, use ODS EXCEL with an xlsx format extension.
- Whether the file size is a concern? If so, use proc export with DBMS=EXCEL/EXCELCS and xlsb format since they have smaller file sizes than xls.

CONCLUSION

As an A&R programmer, we deal with all of these different excel worksheet formats. When reading from the worksheet, we need to pay attention to the input format; when output, we also need to select the proper format based on the use case. The tips discussed in this paper for selecting the proper method for importing/exporting the excel file would be helpful when programmers need to deal with the excel file.

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

REFERENCES

Benjamin Jr, William E, 2010, "The Little Engine That Could: Using EXCEL LIBNAME Engine Options to Enhance Data Transfers between SAS® and Microsoft® Excel Files", *Proceedings of PharmaSUG 2010,* Paper AD10, Orlando, FL.

Sun, Helen and Wong, Cindy, 2005, "A Macro for Importing Multiple Excel Worksheets into SAS® Data Sets", *Proceedings of SAS Users Group International 30*, paper 040-30, Philadelphia, PA.

Wu, Wenyu and Zhang, Liping, 2011, "A Recursive SAS Macro to Automate Importing Multiple Excel Worksheets into SAS Data Sets", *Proceedings of PharmaSUG 2011*, Paper CC10, Nashville, TN.

CONTACT INFORMATION

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

Jaime, Yan	Chao Su,	Changhong Shi
	+1 (732) 5946459	+1 (732) 5941383
mingyu.yan1@merck.com	chao.su@merck.com	changhong_shi@merck.com