Changing ODS Outputting Methods to Streamline the TLF Output Process

Rose O'Nians Michalowicz, Plus-Project, Knutsford, UK
Richard Fryer, Plus-Project, Leeds, UK

ABSTRACT

The process of converting TLF outputs into their final format, normally RTF to PDF, and compiling them for delivery is time consuming and often riddled with issues. This task is required during a period on the critical delivery path where time is limited. With intricate bookmarking and multiple format requirements to adhere to, the efficient set-up of in-program replaying of outputs straight to the final destination is often overlooked. The resultant default PDF output with automated bookmarking saves significant time, and memory usage, in the TLF process and comes with a list of additional benefits, including ease of compiling. The use of ODS Document has been covered in a multitude of papers in the past, however this paper will document the real-life practical implementation of the procedure, including the necessary process updates required to navigate issues which are not well documented in previous papers.

INTRODUCTION

TLF outputting is thought to be a simple task. Once it is set up correctly, things run smoothly, and the programmers can interpret any issues that arise. At the end of the study the programming lead just needs to convert and compile the outputs, but this part is never easy. Often blank pages can appear at the end of outputs, the bookmarks can appear in the wrong places, special characters can appear incorrectly, or macro variables can fail to resolve. Several previous papers detail how to output straight to PDF and use PROC DOCUMENT to update this process, completing some of these steps within SAS® without having to make any manual adjustments outside of SAS. However, none of these papers have discussed bookmark processing with respect to the handling of outputs that use by groups/variables or dealing with figure outputs. We will explore these points below, describing how they can be handled.

TLFs are standardly sent to RTF within TLF programs, transformed into PDF using a tool external to SAS, and then compiled into the final document using other macros, CMD commands, or Adobe® Acrobat® Pro. Following this, bookmarks often need adding, removing or correcting. Many companies do this manually or use their own tools, commonly outside of SAS, but in general these tools are problematic, and issues can arise last minute, especially if outputs require non-typical packaging. Using an automated process, these issues can be avoided, and uniformity and reproducibility become easier to obtain.

The process outlined in this paper creates outputs straight to PDF, or any required destination(s). Titles, footnotes and bookmarks are then added from the titles file (titles.xlsx converted into a SAS dataset) directly to the individual outputs, rather than added at compiling time, and processed to ensure desired formats.

The main tools requiring set up for this process are:

- titles.xlsx
- %ODSSTART
- %ODSEND

%ODSSTART and %ODSEND are macros that have been developed by Plus-Project and are not SAS default macros.

When each output program is run the following occurs:

1. An ODS (Output Delivery System) document location is opened
2. The output is sent to ODS Document
3. Special characters within the bookmarks are masked
4. Excess bookmarks from by variable processing and figure rendering are removed
5. The output is replayed from ODS Document to the specified location(s)

PROC DOCUMENT

PROC DOCUMENT utilises ODS Document to store an output in a buffer and reprint it later. This is useful when outputting TLFs to several destinations and can be done in a single step. There are other advantages to using ODS Document but many are beyond the scope of this paper. [1] covers further PROC DOCUMENT applications.

There are three main steps to outputting TLFs with ODS:

- Open required file destinations. In this example, there are two.

```
ods rtf "<study path>\output\t01.rtf"
ods pdf "<study path>\output\t01.pdf";
```

- Create the output. Most commonly done using PROC REPORT, PROC MEANS or similar.

```
proc report data=output.t_01;
  by trtn trt;
  columns pagen col1 col2 col3 col4;

  define pagen / order noprint;
  define col1 / 'Subject';
  define col2 / 'Age';
  define col3 / 'Sex';
  define col4 / 'Race';

  break before pagen;
run;
```

- Close opened file destinations.

```
ods pdf close;
ods rtf close;
```

This example creates PDF and RTF outputs containing demography data. It is also common practice to output to RTF and post-process the outputs externally to SAS to convert them to PDF format for compiling. Figure 1 shows this process in a diagrammatical format, highlighting a linear outputting method with no means for consistent adjustments to bookmarks between output formats and high risk of style issues occurring during the conversion process externally to SAS. The blue steps are performed within SAS and the red steps are performed post-production outside of SAS.

Using ODS DOCUMENT vastly improves the process and results in moving the processes which are external to SAS in Figure 1 into programmatical steps, reducing the likelihood of error in post-processing. Figure 2 highlights the step of opening ODS Document in yellow, completed in the %ODSSTART macro. The output is then produced using a standard outputting procedure such as PROC REPORT or PROC SGRENDER. The green steps are completed in the %ODSEND macro. These steps do not involve data manipulation and can be run independently if required, without the re-QC of datasets. This part of the process prints data in requested formats to destinations of choice, featuring processed bookmarks with special characters masked, using the DOCUMENT procedure. Styles and other formatting options are applied here. One of the biggest benefits of this three-step process is that the once the data manipulation has been done in the output program, the %ODSSTART and %ODSEND macros can be used across multiple programs without adjustments, enabling standardisation and consistent results across all outputs. The risks of formatting issues occurring during post-processing, per Figure 1, are removed and the time-consuming conversion process often undertaken is now redundant.

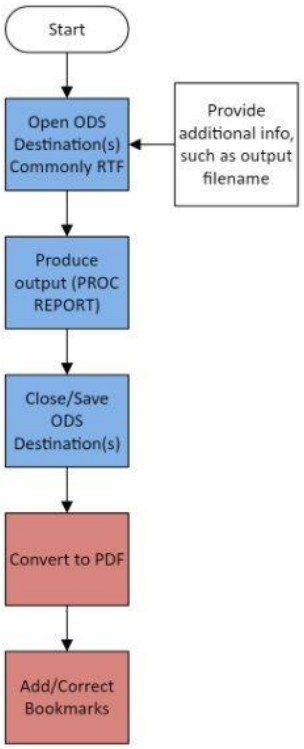

Figure 1 An overview of the standard process.

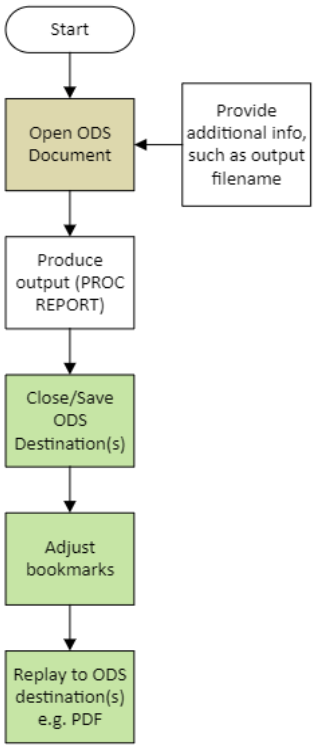

Figure 2 Improved process highlighting %ODSSTART/%ODSEND procedures.

ENVIRONMENT SET-UP

TITLES DATASET

As with most companies, a titles dataset is created from a spreadsheet which is used to apply titles, footnotes, and bookmarks to an output. The column names and standard texts (e.g. “title1”) in the spreadsheet are critical and no changes can be made to the structure without updates to associated tools. The key columns required for bookmarking in the titles file are:

- Program name
- Report ID – to connect the output to the set of titles, footnotes and bookmarks. This is especially important if more than one output is created from any program.
- Title Type – noting bookmarks, titles and footnotes. title2 is used here as title1 is expected to be the study name.
- Text – text string to be output

A	B	C	F	
Program Name	Report Id	Report Type	Title Type	Text
Chemistry_Table	1	Table	bookmark	PHUSE Paper Table &g_reportno: %Changes in Chemistry &Haematology by Treatment
Chemistry_Table	1	Table	footnote1	j=l "PHUSE Paper test output footnote.";
Chemistry_Table	1	Table	title2	j=l "PHUSE Paper Table &g_reportno: Percent Changes in Chemistry and Haematology by Treatment ";
Age_Group_Figure	1	Figure	bookmark	PHUSE Paper Figure &g_reportno: %of Subjects Screened &Treated per Age Group
Age_Group_Figure	1	Figure	footnote1	j=l "PHUSE Paper test output footnote.";
Age_Group_Figure	1	Figure	title2	j=l "PHUSE Paper Figure &g_reportno: Percent of Subjects Screened and Treated per Age Group";

Figure 3 Screenshot of necessary columns in the titles.xlsx required for bookmarking

Special characters must be masked, otherwise they will not display correctly. This is not handled in the titles file as is often done, but within PROC DOCUMENT itself. This allows the text in the titles spreadsheet to be written as expected, mitigating any risk of special character issues appearing at the final run before delivery.

MACROS

There are two macros used in place of the ODS statements on either side of the PROC REPORT, named %ODSSTART and %ODSEND.

%ODSSTART

%ODSSTART is simple, containing statements to do the following:

- `ods listing close;` - close the listing destination and associated files, ensuring that ODS does not send an output to that destination and freeing system resources.
- `ods results off;` - turn off output tracking that ODS generates in the results window.
- `ods document name=work.&odsoutputdataset (write) dir=(path=\<file>);` - open a new ODS document that the output can be routed to, producing a temporary, replayable document.
- `%global &odsdocopen;` - create a global macro variable to feed into %ODSEND to confirm that %ODSSTART has successfully opened a new document.

These statements are required before the output procedure.

%ODSEND

%ODSEND is used to write the output to each of the required formats and set any requirements. It is in this macro that company-specific requirements can be set, such as default output options. There are checks programmed into this macro to ensure that requirements are met, otherwise messages are written to the log. The bookmarks are also handled here to ensure the compiling of outputs works as desired.

%ODSEND only runs if %ODSSTART has been run prior, indicated by the &odsdocopen global macro variable. If this has not been set, the macro is exited.

The style of the output, including headers, page orientation, margins and formatting of text fonts, is taken care of through the input of the pre-defined style in the style= macro parameter. In this example a template, created in a styles sheet using PROC TEMPLATE, is used. The macro input defines the type of output required and multiple options can be added, for example `%odsend(odstype=rtf pdf)` gives RTF and PDF outputs. Other possible destinations include html5, powerpoint, excel and word. To get a list of all the output items that have been loaded into the ODS document, the ODS document that was opened in ODSSTART is closed, `ods document close`. All properties from the closed document are then output with the statement `ods output properties=&odsoutputname`. This ODS output is then closed, `ods output close`.

The special characters are masked (see page 5) and following this, the program loops through each output type defined in the macro call and sets the extension type macro variable. An ODS statement is used to open each required destination.

In the below code &odstype_x refers to each of the values of ODSTYPE input to the macro call (e.g. rtf, pdf, html5, ppt, xlsx, docx), and this code is looped over when multiple options are specified.

```
ods &odstype_x. style=&style file= "&destination..&reportname..&extension." nogtitle nogfootnote %if
&odstype_x=pdf %then pdftoc=&pdftoc;;
```

&odstype is defined in the macro call. &destination and &reportname are given by the initialisation macro. &pdftoc is defined in the macro call and determines the level of bookmarking in the compiled PDF. 1 is the default, giving one bookmark per TLF with no lower-level bookmarks for by groups.

nogtitle and nogfootnote are used to prevent all titles and footnotes from appearing in the graphics file [2]. Instead, titles and footnotes become part of the printer file, meaning they can be handled and formatted consistently with other table and listing outputs and style options can be applied.

Next, the output is updated within the DOCUMENT procedure. Additional bookmarks and secondary bookmark levels are removed to leave a single top-level bookmark. This bookmark is then assigned with the processed label previously derived, as in the next section, and each object is replayed into the final destination.

Finally, it is important that the session is reset. %ODSEND deletes all temporary datasets, closes open ODS destinations and resets options that were saved at the start of the macro.

FEATURES

BOOKMARK CONTROL

Bookmarking in PDFs can be tedious, due to the tendencies for SAS to output unwanted bookmarks. There are ways to handle this within the SAS code, for example using contents = "" in the PROC REPORT on a break statement and description = "" in PROC SGPLOT before post-processing. However, with PROC DOCUMENT it can be done automatically, using bookmarks assigned in the titles file. Bookmarks are produced differently when using different output procedures. SAS automatically creates a bookmark for each new group when there are by groups in the PROC REPORT, as in Figure 4. There are papers that explore the handling of bookmarks for PROC REPORT and other methods, see [3]. However, these methods do not work with PROC SGRENDER, so a different process must be followed. To fix this the extra bookmarks need to be removed, and bookmark levels controlled. This is done by first reading in the list of non-directory bookmarks, where TYPE does not equal "Dir" (directory). From Figure 4 three rows would be selected for the table and from Figure 5 two for the figure.

	PATH	TYPE
1	\file#1	Dir
2	\file#1\Report#1	Dir
3	\file#1\Report#1\ByGroup1#1	Dir
4	\file#1\Report#1\ByGroup1#1\Report#1	Table
5	\file#1\Report#1\ByGroup2#1	Dir
6	\file#1\Report#1\ByGroup2#1\Report#1	Table
7	\file#1\Report#1\ByGroup3#1	Dir
8	\file#1\Report#1\ByGroup3#1\Report#1	Table

Figure 4 Contents of ODS output file showing BYGROUP<N> in the pathname which refers to the different by groups set in the PROC REPORT.

	PATH	TYPE
1	\file#1	Dir
2	\file#1\SGRender#1	Dir
3	\file#1\SGRender#1\SGRender#1	Graph
4	\file#1\SGRender#2	Dir
5	\file#1\SGRender#2\SGRender#1	Graph

Figure 5 Contents of ODS output file showing SGRENDER<N> in the pathname which refers to multiple figures being rendered within an individual output.

```
proc document name=work.&odsoutput. (read);
  list ^(where=(_type_ ne 'Dir')) / levels=all;
run;
quit;
```

The resulting datasets, after the addition of two new required variables (OBJ and OBJPATH) derived from the path variable, are shown in Figure 6 and Figure 7. OBJ and OBJPATH are then assigned to macro variables which are used as input into the PROC DOCUMENT code to update the ODS output document.

	PATH	TYPE	OBJ	OBJPATH
1	\file#1\Report#1\ByGroup1#1\Report#1	Table	Report#1	\file#1\Report#1\ByGroup1#1\
2	\file#1\Report#1\ByGroup2#1\Report#1	Table	Report#1	\file#1\Report#1\ByGroup2#1\
3	\file#1\Report#1\ByGroup3#1\Report#1	Table	Report#1	\file#1\Report#1\ByGroup3#1\

Figure 6 List of non-directory bookmarks for a table with by-groups.

	PATH	TYPE	OBJ	OBJPATH
1	\file#1\SGRender#1\SGRender#1	Graph	SGRender#1	\file#1\SGRender#1\
2	\file#1\SGRender#2\SGRender#1	Graph	SGRender#1	\file#1\SGRender#2\

Figure 7 List of non-directory bookmarks for a figure output with multiple figures being rendered.

When multiple figures are rendered within an output procedure, multiple bookmarks are created. To handle this, it is necessary to remove additional bookmarks prior to assigning the bookmark labels defined in the titles file. This is done using rename and move statements at the start of PROC DOCUMENT, before the correct bookmarks are generated. Previous papers [3, 4] have demonstrated methods of bookmark level generation control such as using PROC DOCUMENT to create a new Dir in the document file, applying required bookmarks to the new Dir and copying the outputs and objects into it, removing labels and bookmarks from each object individually. Whilst this method is successful with PROC REPORT outputs and most SG procedures, it leaves two levels of bookmarks for SGRENDER, as in Figure 8. Simply making the second bookmark levels null also does not yield desired results, as in Figure 9. Instead, moving all objects to the top level rather than a new Dir, and replaying those removes all level two bookmarks. With all objects now being at level one, the first bookmark can be overwritten with the desired label defined in the titles file. This results in a single bookmark for the file and is successful with any PROC output method, as shown in Figure 10.

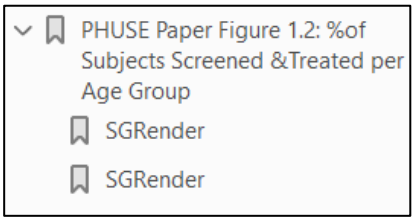

Figure 8 Automatic bookmarking for multiple plots when using SGRender and attempting to mask the bookmarks.

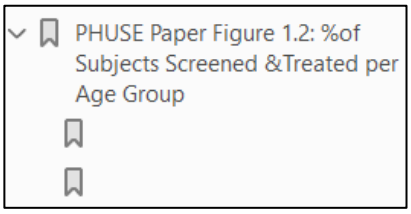

Figure 9 Trying to remove the automatic bookmarking, setting description=" ".

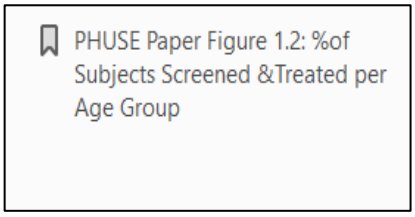

Figure 10 Desired bookmarking using PROC DOCUMENT.

Bookmarks are added for each replaying of an object in PDF outputs. Due to this we must control specifically which objects generate bookmarks when replayed. To do this we force the first object to be output once replayed, using the run statement in PROC DOCUMENT and include the defined bookmark here. Then the generation of bookmarks is turned off and the remaining objects are replayed without the risk of additional bookmarks being created. The option nobookmarkgen is applied to the objects following the first replayed object to control the outputted bookmarks. This issue does not occur with other formats, so the following code is used to prevent the additional bookmarks in PDF files only.

```
%if %index(%upcase(&odstype),%str(PDF)) %then %do;
  ods pdf(local) nobookmarkgen;
%end;
```

It is essential to include contents=" not only on the main PROC REPORT but also on break statements to avoid extra unwanted bookmarks appearing in outputs.

```
break before nbook / page contents='';
```

This produces the correct bookmarking as in Figure 11, without this an additional bookmark is generated as in Figure 12. The NBOOK variable in this example is used for paging, but this would be necessary for any break variable.

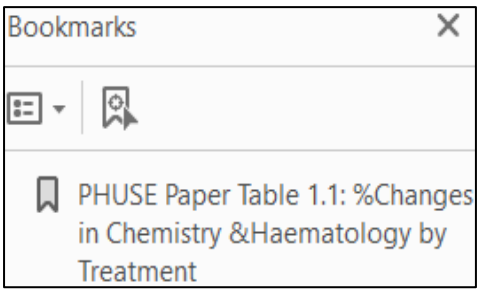

Figure 11 When contents = "" is applied, there is no additional bookmark.

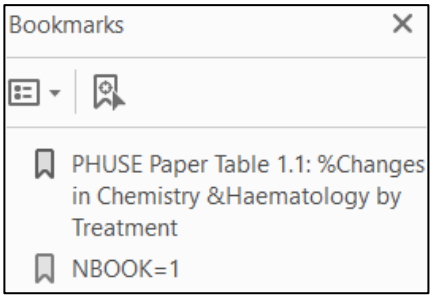

Figure 12 When contents=" is missing, NBOOK=1 appears as an additional bookmark.

MASKING SPECIAL CHARACTERS

Any time that there are special characters in bookmark text, there is risk that they will affect the running of the program if they are not handled correctly. Especially tricky are characters that are used to identify macros and macro variables in SAS code, namely % and &. To handle these, %ODSEND checks if the bookmark text in the titles file contains % or & followed immediately by any other valid characters, i.e. alphanumeric characters or an underscore, not a space. If so, a check runs to see if there exists a macro or macro variable with that name. This can be useful when, for example, the macro variable containing the report number, &g_reportname, is required to be resolved in the bookmark text, but you

don't want SAS to attempt to resolve an & used as "and". Figure 3 shows a screenshot of the titles file with some characters that need to be handled in different ways.

To check for a macro the following code is run using %sysmacexist [5], a standard SAS macro function:

```
if string=:'%' then do;
  call execute(cats('%let sysmacexist=%sysmacexist(',substr(string,2),');'));
  exist=input(symget('sysmacexist'),1.);
end;
```

To check for a macro variable a similar line is run, using symexist [6], a standard SAS function:

```
else if string=:'&' then do;
  exist=symexist(substr(string,2));
end;
```

If exist=1, no further manipulations are made to the string. For records where exist=0, the following code is executed for each partial string:

```
bookmarknew=tranwrd(bookmark,strip(string),cats('%nrstr(',string,')'));
```

The new bookmarks for the example outputs are shown in Figure 13 and Figure 14.

	BOOKMARK	BOOKMARKNEW	STRING	EXIST
1	PHUSE Paper Figure &g_reportno: %of Subjects Screened &Treated per Age Group	PHUSE Paper Figure &g_reportno: %nrstr(%of) Subjects Screened %nrstr(&Treated) per Age Group	&Treated	0

Figure 13 Resulting text in the manipulated titles dataset for the figure output.

	BOOKMARK	BOOKMARKNEW	STRING	EXIST
1	PHUSE Paper Table &g_reportno: %Changes in Chemistry &Haematology by Treatment	PHUSE Paper Table &g_reportno: %nrstr(%Changes) in Chemistry %nrstr(&Haematology) by Treatment	&Haematology	0

Figure 14 Resulting text in the manipulated titles dataset for the table output.

In these examples &g_reportno would be resolved but the other % and & in the text would be displayed correctly. This is done by the macro looping through the string until it comes across the position containing the special character and then surrounding the symbol and following text with %nrstr(). This ensures that the text is displayed properly, masking the % or & as plain text when output. Otherwise, the existing macro or macro variable are not masked and allowed to resolve.

In the two examples shown in Figure 13 and Figure 14 the bookmarks are updated to read as displayed in Figure 15.

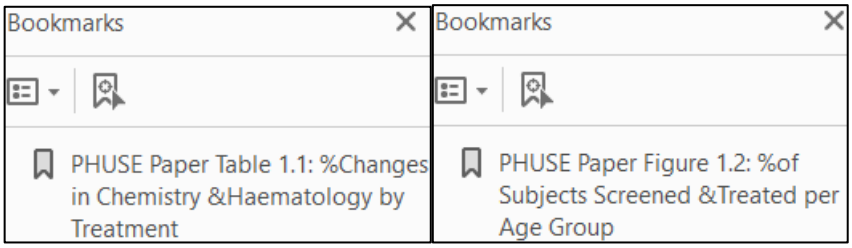

Figure 15 Bookmarks for the example table and figure.

COMPILING

The next step in the process is to compile all the individual PDFs into the final document(s). This can be done in various ways but is outside of the scope of this paper. There are multiple compiling related benefits of the process detailed in this paper:

- Natively creating bookmarks in the individual files mean that they don't have to be recreated at the compiling stage, meaning compiling is significantly faster and simpler.
- As the outputs are already in their desired format for compiling, no conversion is required and the efficiency of the compiling process is improved.
- Fewer files means that less storage space is used, reducing server space requirements.
- As the original output files are compiled, the time taken for doing final QC checks can be reduced as checking for blank pages, page size, page numbering, etc. are all QC'd at an earlier stage in the process. The compiled outputs will be the exact same as the individual files.

ADDITIONAL CONSIDERATIONS FOR FIGURES

A lot of previous papers, which suggested replaying outputs directly to PDF format using ODS DOCUMENT, offered code or advice that work with listings and tables. However, figures have additional considerations which the code and advice in previous papers failed to fully account for. Below are some suggestions regarding these extra considerations.

As GTL (Graph Template Language) can be used for creating templates for figure outputs, reading in different data types to produce outputs in the same style, a template code library can be set up to support ongoing re-use of common figure code. This allows consistent outputs to be produced over individual or multiple studies, streamlining the programming process.

Another issue that can arise when replaying figure outputs is that PROC SGPLOT and PROC GPLOT ignore predefined templates when outputs are replayed to PDF format. This causes fonts and styles not to appear as expected and results in formatting issues. These plotting methods also fail to save titles and footnotes when replaying. When using GTL to create a template and outputting a figure using PROC SGRENDER, PROC DOCUMENT can be used to store the style created in the template. This means that the output can be replayed with access to the styles and allows closer control of styles and formatting features. Titles, footnotes and predefined styles are all retained when replaying figure outputs to PDF using this method.

Changes may be required for the styles sheet, but the %ODSSTART and %ODSEND macros can be reused with no modifications, resulting in issue-free compiling and bookmarking every time. Rinse and repeat.

CONCLUSION

There are many papers detailing parts of the information given here, but this paper attempts to tie the learning from each together into a solution that works for all outputs, including figures. A few standard macros can prevent all sorts of issues when it comes to delivery. Once set up, they are simple to maintain and rerun each time for consistent, accurate deliverables. There is significant set up for this system, requiring the creation of %ODSSTART and %ODSEND, as well as ensuring that the titles file is in the correct format. However, these can be used across every study, reducing complications and increasing efficiency across the board.

CONTACT INFORMATION

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

Rose O'Nians Michalowicz
Email: rose.oniansmichalowicz@plus-project.co.uk
Richard Fryer
Email: richard.fryer@plus-project.co.uk

BIBLIOGRAPHY

[1] M. Tuchman, PROC DOCUMENT by Example Using SAS®, Cary, NC: SAS Institute Inc., 2012.

[2] SAS, "TITLE, FOOTNOTE, and NOTE Statements," [Online]. Available: https://support.sas.com/documentation/cdl/en/graphref/63022/HTML/default/viewer.htm#titlechap.htm. [Accessed 26 07 2024].

[3] T. Dumas-Robertson, "One-Level PDF Bookmark Created by ODS DOCUMENT and PROC DOCUMENT," in *PharmaSUG*, Seattle, WA, 2018.

[4] D. Spruck and M. Kawohl, "Sort your SAS Graphs and Create a Bookmarked PDF Document Using ODS PDF," in *Konferenz der SAS Anwender in Forschung und Entwicklung*, Potsdam, Germany, 2005.

[5] SAS, "%SYSMACEXIST Function," [Online]. Available: https://support.sas.com/documentation/cdl/en/mcrolref/62978/HTML/default/viewer.htm#n0xwysoo8i2j3kn13ls4thkn3xbp.htm. [Accessed 25 July 2024].

[6] SAS, "SYMEXIST Function," SAS, [Online]. Available: https://support.sas.com/documentation//cdl/en/mcrolref/62978/HTML/default/viewer.htm#n1j6dly37s4370n14hdiznpbxso4.htm. [Accessed 25 July 2024].

[7] R. D. Muller, "Proc DOCUMENT, The Powerful Utility for ODS Output," in *SAS WUSS*, SAn Francisco, California, 2016.

[8] B. Lawhorn, "Let's Give 'Em Something to TOC about: Transforming the Table of Contents of Your PDF File," in *SAS Global Forum*, Las Vegas, NV, 2011.

ACKNOWLEDGEMENTS

The authors would like to thank the programmers at Plus-Project involved in the development of the tools contributing to this paper. The authors would also like to acknowledge the review team, namely Tony Cooper, Liz Meeson, Rachel Dunk and David Meek, who's expert advice and constructive comments helped to build this paper. A special thank you goes to Will Greenway who had a strong hand in the development of the tools mentioned and provided continuous support and advice throughout the creation of this paper.

APPENDIX A: CODE SNIPS

```
%odsstart;

proc report data=output.&g_reportname nowd missing split="#" spanrows contents='';
  by trtn;
  columns nbook npage trtn usubjid sex arm;

  define nbook   / order noprint;
  define npage   / order noprint;
  define trtn    / order noprint;

  define usubjid / order style(header)=[asis=yes just=left] style(column)=[asis=yes just=left   width=50%] "Subject";
  define sex     /       style(header)=[just=center]        style(column)=[         just=center width=24%] "Sex";
  define arm     /       style(header)=[just=center]        style(column)=[         just=center width=24%] "Arm";

  break before nbook / page contents='';

  break after npage / page;

  compute after npage;
    line @1 ' ';
  endcomp;
run;

%odsend(odstype = pdf rtf);
```

Figure 16 Typical code used to output a table with by processing, including calls to %ODSSTART and %ODSEND. This outputs to both PDF and RTF destinations.

```
%odsstart;
  proc template;
    define statgraph scatterplot;
      begingraph / designwidth=1024px;
        dynamic group xvar yvar n;
        layout overlay /

          yaxisopts=(type=log logopts=(base=10 viewmin=0.1 viewmax=100))

          xaxisopts=(type=log logopts=(base=10 viewmin=0.1 viewmax=100));

          scatterplot y=yvar x=xvar / group=group name="scatter";

        endlayout;
      endgraph;
    end;
  run;

  proc sgrender data=age_data template=scatterplot description=' ';
    format trtn ntreat.;
    dynamic group="TRTN" xvar="years65" yvar="years80" n="flag";
    label years65="65 Age Group"
          years80="80 Age Group";
  run;
%odsend(odstype = pdf rtf);
```

Figure 17 Typical GTL template for a scatterplot with SGRender being used to output the figure, including calls to %ODSSTART and %ODSEND. This outputs to both PDF and RTF destinations.

APPENDIX B: GLOSSARY

Graphics output – An image file containing the graph body, which cannot be edited, output from PROC SGRENDER (or similar).

Printer file – The actual output, including the graphics output and titles, footnotes, page numbering, etc. This can be in PDF, RTF, etc.