PharmaSUG China 2019 - Paper CC-035

LST and RTF Output Check Tool

Mary Rose Sibayan, PPD, Manila, Philippines

ABSTRACT

In clinical research, several outputs are usually produced for each deliverable. Examples of which are tables, listings and figures. When reviewing the outputs, aside from checking the accuracy of results, we also check the overall layout and appearance of each output. Few of the simplest but most encountered issue is having page break issues or headskip issue. It can be difficult and time consuming to manually check all outputs for the mentioned issues, especially if an output has more than 100 pages.

This paper aims to present %check_pagination_issues, a macro which checks the pagination and headskip of an output. The macro checks the actual pagination of the final output against the planned pagination and checks if there are none or if there are extra headskips in the output. The macro writes the warnings in the SAS log of the specific output with issues. The macro also creates a data set containing a list of outputs that are run in the same directory having at least one of the mentioned issues. The macro can also check outputs in LST and RTF format.

INTRODUCTION

During development and maintenance of outputs, one of the difficulties is checking page break issues for each output. It will be tedious if more than a hundred outputs are needed for submission and if several of the outputs have many pages. If the process of checking page break issues can be automated, the time saved can instead be used to focus more on reviewing the results and actual content of the output.

In addition to page break issues, each output also needs to be checked if it is compliant with the requirements. One of the most overlooked requirement for tables and listings is having the need to output exactly 1 space for headskip. This headskip is needed so that there will be a space in between the column headers and body of the output. When there are several outputs to be reviewed, it is highly likely to miss checking this during the output review.

Some pharmaceutical companies require the use of a unique extension of an output, an example of which is an LST file. For commonly used file extensions such as RTF, DOC or DOCX files, checking for page break issues takes lesser time. If the total number of sections is not equal to the total number of pages, this already indicates a page break issue. In contrast, LST files do not have a section number so checking for page break issues would have to be done page per page. If checking was done manually, it will take longer time.

The %check_pagination_issues aims to solve the issues mentioned above. This macro is called right after PROC REPORT so that issues can immediately be checked and solved during the development process of the outputs.

PAGE BREAK ISSUE AND HEADSKIP ISSUE

Sometimes, the issue of having no headskip at all or having more than 1 headskip is overlooked specially if there are several outputs that need to be checked. Display 1 is an example of an LST output with more than 1 headskip, as indicated by the boxes in pink and yellow.

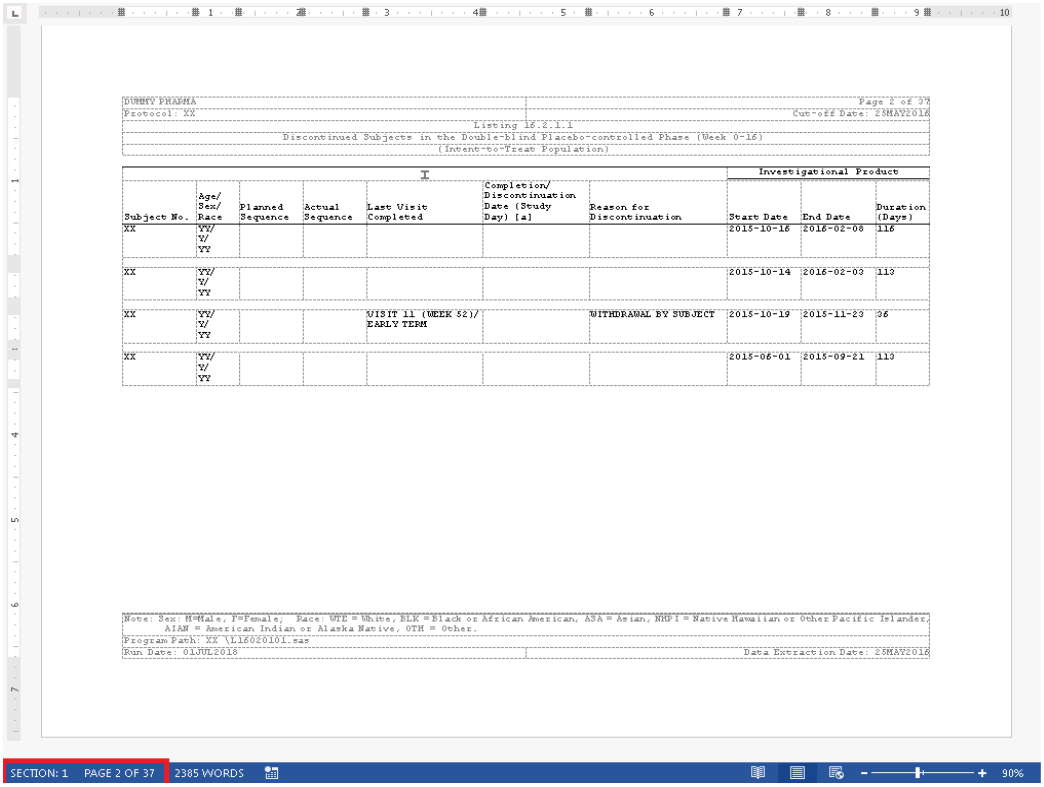
Page 1 of 1

Best Overall Response by ECOG PS (0, 1, 2)
All Response Evaluable Subjects

	Number of Subjects (%)			
	Total N =	ECOG (PS 0) N =	ECOG (PS 1) N =	ECOG (PS 2) N =
BEST OVERALL RESPONSE (RECIST 1.1, CONFIRMATION OF RESPONSE REQUIRED):				
COMPLETE RESPONSE (CR)				

Display 1: Output with >1 Headskip Issue

During creation of outputs, the programmers normally create a page variable which will dictate the planned pagination of the output, depending on how many rows would fit into each page. Page break issue often occurs when the planned pagination does not match the actual pagination needed. The actual pagination is commonly affected by the wrapping of the columns or the number of rows needed for titles, footnotes and column headers. Most of the time, planned pagination does not match the actual pagination. Display 2 is an example of an RTF output with page break issue, as indicated by the unequal section number and current page number.



Display 2: Output with Page Break Issue

PREREQUISITES OF LST AND RTF OUTPUT CHECK TOOL

Before the call of the %check_pagination_issues, there are 2 additional calls that the user will need to add before the macro can be used. First is to add an ODS OUTPUT REPORT call right before PROC REPORT. One of the uses of ODS OUTPUT REPORT is to store the output of PROC REPORT into a data set, which will then be used to determine the planned number of pages. If the report has subgroup

bylines, the number of output data set will depend on the number of subgroups in the output. Below is the first code that needs to be called:

```
ODS output Report(PERSIST=PROC match_all=rep)=<DATA SET NAME>;
```
1

The second call needed right after the ODS OUTPUT REPORT is a call to store the next macro that will be defined into a data set. This is needed to store the PROC REPORT statements to be defined in the succeeding statements inside a data set. The PROC REPORT statements will be used to find out the following information:

- The variable/s defined with the id option
- The variable used with page option
- The variable defined with the break statement
- The number of interleaved pages
- To check whether headskip option was used

The second call needed is stated below:

```
options nomprint;
filename mprint temp;
options mprint mfile;
```
2

The 2 statements above are important for %check_pagination_issues to execute properly. After the mentioned codes above has been called, the PROC REPORT statements need to be defined in a macro call and then executed.

It is also a requirement for the input data set to have a pagination variable available. Likewise, the PAGE option should also be used in the PROC REPORT. This pagination variable is going to be used by the macro to find the planned number of pages. If the PAGE option is not used along with the BREAK statement, an alert will be produced in the SAS Log and the pagination will not be checked. Below is a sample of the proc report call needed:

```
%macro report();
PROC REPORT DATA=FINALS NOWD HEADLINE MISSING SPLIT='^' SPACING = 1
STYLE(REPORT)={OUTPUTWIDTH=100%};
      COLUMN PAGEBK SUBJID ASR TRTSEQP TRTSEQA VIS DSSTDTC DSREA
("INVESTIGATIONAL PRODUCT ~R/RTF'\BRDRB\BRDRS" (TRTSDT TRTEDT
TRTDUR_));
      DEFINE PAGEBK           /ORDER NOPRINT;
      ...
      COMPUTE BEFORE SUBJID/STYLE=[CELLHEIGHT=1.2%];
            LINE " ";
      ENDCOMP;
      BREAK AFTER PAGEBK/PAGE;
RUN;
%mend;
%report;
```
3

%CHECK_PAGINATION_ISSUE

The %check_pagination_issues macro was mainly developed because of the need to check the page break and headskip of the outputs, regardless of whether the output has LST or RTF extension. Right after the execution for PROC REPORT in statement 3 above, the %check_pagination_issues can immediately be called. The data set name to be defined in statement 4 below is the same data set name defined from statement 1. The library name is the path where the output data set containing results from the macro will be stored. Below is the call that will execute the macro:

```
%check_pagination_issues(inreport=<DATA SET NAME>, reslib=<LIBRARY NAME>);
```
4

The macro will first save the PROC REPORT code into a data set before closing the output. In the macro, the code to close the output has already been included so statement 4 above can be set as the last line of the program. The macro will have 3 parts: the part to get the planned number of pages needed for the output, the part to get the actual number of pages used in the output itself and the part to check the number of headskips used in the output. The file extension of the output will also be checked. The checking of the actual number of pages in the output will depend on the extension of the file so it will be used in the latter part of the macro.

GETTING THE PLANNED NUMBER OF PAGES

For the first part of the macro, the data set produced from statement 1 above will be used to find the planned number of pages which is needed for the output. Considerations were included in the macro for cases wherein there are interleaving pages or when the output uses subgroup headers, which is common for clinical trial outputs. When the output has subgroup headers, there would be more than 1 data set produced from statement 1, depending on the number of subgroups. For cases like this, all data sets produced will be appended first. A variable named *dset* will then be derived within the macro to distinguish the records coming from different data sets. If the output will not have subgroup headers, then this *dset* would be assigned as 1. Below is the code that will handle the derivation when the output has subgroup headers:

```
%MACRO append_ds;
      %if &dsnum. gt 1 %then %do;
            %let dsnum_x=%eval(&dsnum. -1);
              %do i=1 %to &dsnum_x.;
                   DATA &INREPORT&I.;
                     SET &INREPORT&I. END=LAST;
                     DSET=&I;
                   RUN;

                   DATA &INREPORT;
                     SET &INREPORT &INREPORT&I.;
                   RUN;

                   PROC DATA SETS NOLIST;
                     DELETE &INREPORT&I.;
                   QUIT; RUN;
            %end;
      %end;
      %else %do;
            DATA &INREPORT;
              SET &INREPORT;
              DSET=1;
            RUN;
      %end;
%MEND;
%append_ds;
```

If the PAGE option has been used with the BREAK statement as checked from the PROC REPORT codes defined, the macro will proceed to create a variable to count the planned number of pages using the appended data set. A simple by statement within the DATA step will be used to derive the number of pages needed for the output. Below is the code to get the planned number of pages:

```
DATA TMP2;
  SET &INREPORT;
  RETAIN BLOCKS;
```

```
  BY DSET &PAGEVAR.;
  IF FIRST.&PAGEVAR THEN BLOCKS+1;
RUN;
```

Afterwards, the maximum value of the variable blocks will be derived. The macro will then find the number of interleaved pages there will be in the output depending on the total number of variables which has been defined with the ID option and PAGE option in the DEFINE statement. The maximum value of the variable block will then be multiplied with the number of interleaved pages. This will now be set as the number of planned pages.

GETTING THE ACTUAL NUMBER OF PAGES

For the second part of the macro, the final output will be checked to find the number of actual pages used. If the extension of the output is LST, a simple INFILE statement will be used to import the actual output into a data set. From the "Page X of Y" line in the data set, the numeric Y part will be taken and will be set as the actual number of pages used. Below is the code used to derive the actual number of pages for LST:

```
data Import_output;
  length line $3000;
  infile "&path.\&TLF." missover dsd length=lg;
  input @;
  input line ;
run;
**count the number of records with the page string**;
data tmp1;
  set Import_output;
  flags=ifn(indexw(upcase(line), 'PAGE') gt 0 and indexw(upcase(line),
'OF') gt 0, 1, 0);
  if flags=1;
  Totalpages=input(scan(substr(upcase(line), findw(upcase(line),
'OF')+2), 1, '\'), best.);
run;
```

If the extension used is RTF, DDE commands will be used to open the output, go to the end of the document and save the output as a temporary text file. Afterwards, this text file will then be imported into a data set. The way the records are stored is different between text file and LST file. In text files, all the titles in the output will be included. To make sure that the Y part from "Page X of Y" will be taken, PRXPARSE and PRXMATCH function will first be used to check if there are any alphabet after the Y part. If there is, the position of the first alphabet will be derived. Afterwards, SUBSTR function will be used to get the value starting from the numeric Y part until the position before the first alphabet part in the string. Below is the part that will count actual number of pages for RTF files.

```
options noxwait noxsync ;
%let rc=%sysfunc(system(start winword));
data _null_;
x=sleep(2);
run;
filename word DDE 'winword|system' notab;

**go to the end of the file and save as txt**;
data _null_;
      file word;
      put '[FileOpen .Name = "' "&path.\&TLF." '"]';
      put "[EndofDocument]";
      put '[FileSaveAs .Name = "' "&path.\temp_out.txt" '",
.format=2]';
      put '[FileClose]';
```

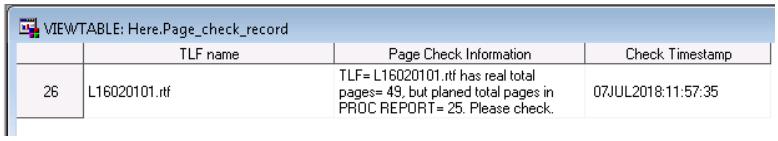
```
            put '[FileExit]';
      run;

      **input the txt and save into data set**;
      data tmp;
        length lines $32000;
        infile "&path.\temp_out.txt" dsd dlm='09'x truncover LRECL=32000;
        input lines $;
      run;

      %sysexec del "&path.\*.txt";

      **get the actual number of pages by using the page line and using
      prxparse to  get y from page x of y**;
      data tmp1(keep=lines pages);
        set tmp;
        where index(upcase(lines), 'PAGE') gt 0;
        y=compress(scan(substr(lines, find(upcase(lines), 'PAGE')), 4, ' '),
, 's');

        if _n_ = 1 then pattern_num = prxparse("/[A-Za-z]/");
        retain pattern_num;
        position = prxmatch(pattern_num, y);
        if position > 1 then do;
            pages=input(substr(y, 1, position-1), best.);
        end;
        else do;
            pages=input(y, best.);
        end;
      run;
```

The number of planned pages will be compared with the actual number of pages used. If it is not equal, a log message alert will be printed, just like in Display 3 below:

```
ALERT_P: TLF= L16020101.rtf has real total pages= 49, but planed total pages in PROC REPORT= 25. Please check.
```

Display 3: SAS Log for Outputs with Page Break Issue

In addition, a data set named as page_check_record will be created in the library name defined in statement 4. This data set will store all the important alerts from the macro for the specific output. The specific date and time when the output was run will also be included in the page_check_record. Display 4 below is a screenshot of the page check record data set that will be created by the macro and the row that will be added when the planned total pages is not equal to actual number of pages.



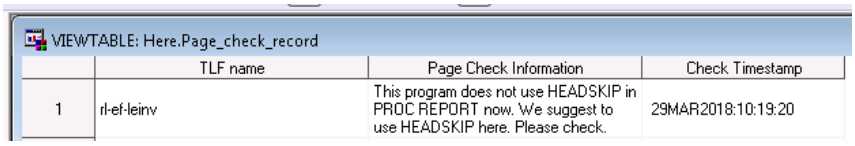
	TLF name	Page Check Information	Check Timestamp
26	L16020101.rtf	TLF= L16020101.rtf has real total pages= 49, but planed total pages in PROC REPORT= 25. Please check.	07JUL2018:11:57:35

Display 4: Page Check Record Output of %check_pagination_issues for Page Break Issues

CHECKING THE NUMBER OF HEADSKIPS USED

The third part of the macro checks if there are no headskip used or if there are extra headskips in the output. If the HEADSKIP option has not been used based on the PROC REPORT statements defined, a

record will be output in the page check record data set. Display 5 below is a screenshot of the page check record data set and the row that will be created when there is no HEADSKIP option.


VIEWTABLE: Here.Page_check_record

	TLF name	Page Check Information	Check Timestamp
1	rl-ef-leinv	This program does not use HEADSKIP in PROC REPORT now. We suggest to use HEADSKIP here. Please check.	29MAR2018:10:19:20

Display 5: Page Check Record Output of %check_pagination_issues for No Headskip issue

The macro will also check if there are greater than 1 headskips at the beginning of each page. If there is already a HEADSKIP option, the first record of each data set produced from statement 1 above should not have a row whose columns all have missing values. All additional records created by the PROC REPORT for the COMPUTE BEFORE/AFTER statements will be removed first before checking for the first record. Display 6 below is a screenshot of a PROC REPORT data set when there are extra headskips.

Display 6: Extra Headskip in PROC REPORT Output Data set

The macro was made to still check the headskip issue even if there is no PAGE option in the PROC REPORT statement. If there is no PAGE option with the BREAK statement, the by variables for the DATA step below would only be the variable *dset*. In addition, considerations were made to make sure that not only the variables in the DEFINE statement of PROC REPORT would be checked for missing columns, but also the by variables defined in the PROC REPORT if there any. A part was also made to check if the original data set has no observations. If the data set has no observations, then it should not be flagged for extra headskip issue anymore. Below is the code that will count the number of headskips in the output:

```
**check first if the whole output has no observations.;
PROC SORT DATA=&INREPORT OUT=TMP3A; BY DSET &PAGEVAR.; WHERE
MISSING(_BREAK_); RUN;
DATA _NULL_;
      SET TMP3A END=EOF;
      BY DSET &PAGEVAR.;
      IF FIRST.&PAGEVAR. AND EOF THEN CALL SYMPUTX('NOREPORT', 1);
      ELSE CALL SYMPUTX('NOREPORT', 0);
RUN;
**count number of by variables;
%if &byvars ne %str() %then %do;
      DATA _NULL_;
            CALL SYMPUTX('BYVARS', TRANWRD("&BYVARS", " ", ", "));
      RUN;
%end;
%let bycnt=.;
DATA _NULL_;
      CALL SYMPUTX('BYCNT', COUNTW("&BYVARS."));
RUN;

PROC SORT DATA=&INREPORT OUT=TMP3; BY DSET &PAGEVAR.; WHERE
~MISSING(_BREAK_); RUN;

DATA TMP3;
  SET TMP3 END=EOF;
  BY DSET &PAGEVAR.;
  TMP_MISS=CMISS(&COLUMNLIST.);
```

```
  %IF  &BYCNT > 0 %THEN %DO;
      IF (TMP_MISS=&COLUMNS.) AND FIRST.&PAGEVAR AND &HEADSKIP_FLAG.>0
AND CMISS(&BYVARS.)=  &BYCNT. AND &NOREPORT. NE 1 THEN FLAG2=1;
  %END;
  %ELSE %DO;
      IF (TMP_MISS=&COLUMNS.) AND FIRST.&PAGEVAR AND &HEADSKIP_FLAG.>0
AND &NOREPORT. NE 1 THEN FLAG2=1;
  %END;
  IF FLAG2=1 THEN DO;
    put "ALERT_P: TLF= &TLF. has one blank row under header line for
some pages. Please check.";
  END;
  IF FLAG2=1;
RUN;
```

If there is an extra headskip, a row will be added to the page check record data set. Display 7 below is a screenshot of the page check record data set when the output has an extra headskip.

VIEWTABLE: Work.Page_check_record

	TLF name	Page Check Information	Check Timestamp
348	rt-im-imslaeeecog.lst	TLF=rt-im-imslaeeecog.lst has one blank row under header line for some pages. Please check.	08JUL2018:10:14:17

Display 7: Page Check Record Output of %check_pagination_issues for Extra Headskip Issue

CONCLUSION

For every submission in the clinical industry, there is a need to ensure that all outputs have high quality in terms of both results and aesthetics. Since page break issue is only a part of the aesthetics, it would be better to automate the checking for this issue. The time saved from checking for aesthetics issue can be used to focus more on checking for the results and contents of each output. %check_pagination_issues macro provides help in checking for page break and headskip issues by providing a list of outputs with issues for every run. The macro can provide the checks for both LST and RTF outputs.

REFERENCES

Gupta, Ajay. 2013. “Combining First Page of Multiple RTF Outputs in SAS ® using Bookmark and VBA Macro”. *Proceedings of PharmaSUG 2013 Conference, paper CC24*.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. In case the reader wants a copy of the code that was used, please contact the authors at:

Mary Rose Alpha Sibayan
PPD
Taguig City, Philippines
Email: MaryRoseAlpha.Sibayan@ppdi.com

ACKNOWLEDGEMENTS

Special thanks to Jirui Jiang for all his inputs for this paper.