SESUG 2020 Paper 137

Upgrading Clinical Trial Reports from ODS LISTING to ODS TAGSETS.RTF

Christopher J. Smith, Cytel; Joshua M. Horstman, Nested Loop Consulting

ABSTRACT

Several pharmaceutical and biotechnology companies are still using ODS LISTING as the primary method for producing outputs. While there is nothing inherently wrong with this approach, using either ODS RTF or ODS TAGSETS.RTF can provide more aesthetically pleasing outputs in the same font as the main clinical study report. In addition, it can provide a more harmonized approach between tables, listings and figures by having a file format compatible with graphics procedures. Lastly, it lays the foundation to have a more robust validation process by having the option to read the actual RTF file back into a SAS® data set. This paper provides examples of typical clinical tables and listings and how to create them in the TAGSETS.RTF destination. Several commonly used options found within the REPORT procedure such as HEADLINE, HEADSKIP, SPLIT=, SKIP, and many others are only available in the LISTING destination. We will provide alternative approaches including STYLE override options, inline formatting, and COMPUTE BLOCK statements when upgrading to using the TAGSETS.RTF destination. We will also touch on STYLE template basics in the TEMPLATE procedure to avoid lengthy PROC REPORT procedures. After covering these topics, the programmer will be equipped to produce any table or listing typically needed for a clinical trial report. SAS 9.4 M6 was used in the examples presented. The intended audience for this paper is beginner to intermediate SAS users with basic knowledge of Base SAS and PROC REPORT.

INTRODUCTION

In the beginning, SAS® produced simple text output using a monospace font. Today, this is known as the ODS LISTING destination as it is just one of many destinations available through the Output Delivery System (ODS). One of the most frequently used destinations is the ODS RTF destination, which was released with SAS version 8.1 and marks its 20-year anniversary this year.

Despite the popularity of ODS RTF, it has several shortcomings, particularly related to pagination. In SAS version 9.2, released in 2008, SAS addressed some of the deficiencies of ODS RTF through addition of the TAGSETS.RTF destination. TAGSETS.RTF better manages titles and footnotes of multi-page reports by attempting to vertically measure where page breaks should occur.

Despite these RTF destinations having been around for decades, many pharmaceutical and biotechnology companies have yet to upgrade their processes to producing RTF output. The RTF destinations offer many benefits such as including proportional fonts and other rich text elements, having a single destination for all outputs (i.e. tables, listings, and figures), and laying the foundation for a more robust validation process.

This paper addresses new techniques statistical programmers will need to know when producing RTF outputs. We will use the TAGSETS.RTF destination throughout this paper due to its improved vertical measurement capabilities and expanded reporting options compared to the standard RTF destination. Techniques discussed include ODS ESCAPECHAR

functions for inline formatting along with COMPUTE BLOCK statements, CALL DEFINE statements, and STYLE override options within the REPORT procedure.

REPORT PROCEDURE BASICS

Before we get started, we would like to cover some basics on how the pharmaceutical and biotechnology industries utilize the REPORT procedure. While PROC REPORT can perform summarization and analysis of variables, all calculations and derivations are performed prior to invoking PROC REPORT in most clinical trial report programs. This allows ultimate flexibility over the formatting and structure of the report. The calculations can be performed using the SUMMARY or FREQ procedures, or a company might have standard macros to perform summary statistics or counts and percentages. Figure 1 shows an example flow of a clinical trial program to create a simple demography summary. The main takeaway is that all results are calculated before reaching PROC REPORT.

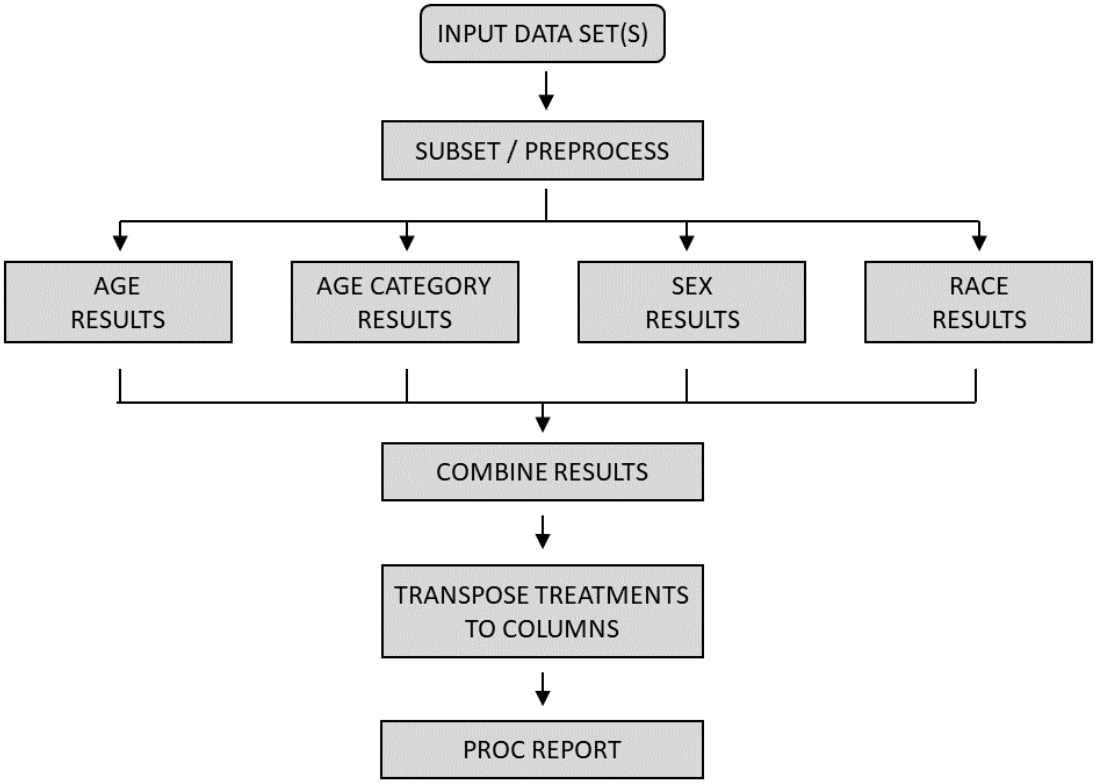


Figure 1. Example Flow of Clinical Trial Report Program

Since we are only using PROC REPORT as a glorified PRINT procedure, there are a few housekeeping rules to remember regarding options used within PROC REPORT.

MISSING OPTION

On the PROC REPORT statement, there is a MISSING option that we want active for every output:

```
proc report data=cars missing;
  <other REPORT statements>
run;
```

The reason we want MISSING on every PROC REPORT statement is so that no records are excluded from the report. Otherwise, variables defined as ORDER variables that have

missing values will have the entire record suppressed. Table 1 shows a data set, CARS, with a variable, MODEL, containing a missing value. MODEL will be used as an ORDER variable.

Make	Model	Type	MSRP
Acura	MDX	SUV	$36,945
Acura		Sedan	$23,820
Acura	TSX 4dr	Sedan	$26,990
Acura	TL 4dr	Sedan	$33,195
Acura	3.5 RL 4dr	Sedan	$43,755
Acura	NSX coupe 2dr manual S	Sports	$89,765

Table 1. Data Set with a Variable Containing a Missing Value

In clinical trial reporting, we want to provide all available information, even incomplete records that have missing values. However, using the REPORT procedure below without the MISSING option suppresses the second record in Table 1:

```
proc report data=cars;
  columns make model type;
  define make /order order=internal;
  define model/order order=internal;
run;
```

Note that the ORDER= option is different from the ORDER option. Defining a variable as an ORDER variable causes the report to have one row for every observation in the data set, and those rows will be sorted based on the values of the ORDER values. The ORDER= option is used to specify how the values of an ORDER variable are used to order the report. We can use the ORDER= option to order the rows based on the FORMATTED value of an ORDER variable or the non-formatted (or INTERNAL) value, amongst a few others. In this paper, we primarily use ORDER=INTERNAL.

Output 1 shows the report with no MISSING option.

```
Make          Model                          Type
Acura          3.5 RL 4dr                    Sedan
               MDX                           SUV
               NSX coupe 2dr manual S        Sports
               TL 4dr                        Sedan
               TSX 4dr                       Sedan
```

Output 1. PROC REPORT without MISSING Option

Notice the record having a missing value for MODEL is suppressed from the entire report. Including the MISSING option on the PROC REPORT statement will avoid this issue. Moving forward, we will use the unaltered SASHELP.CARS data set which does not contain any missing values for MODEL.

ANALYSIS OPTION

By default, PROC REPORT treats numeric variables as ANALYSIS variables and calculates the SUM statistic. Therefore, the following REPORT procedures will result in the same output:

```
proc report data=sashelp.cars missing;
  columns msrp;
  define msrp/analysis;
run;
```

```
proc report data=sashelp.cars missing;
  columns msrp;
run;
```

Output 2 shows the default behavior of numeric analysis variables.

MSRP
14027638

Output 2. DEFINE Statement Defaulting to ANALYSIS

This is behavior we do not want to see in our reports. We will analyze the data ourselves before passing it to PROC REPORT. The use of the ORDER and DISPLAY options on the DEFINE statement can prevent this analysis behavior of PROC REPORT.

GROUP OPTION

There is often some confusion between the GROUP and ORDER options on the DEFINE statement. Some programmers use these options interchangeably, but the options have very different behaviors. Defining a GROUP variable will attempt to combine rows with common values and calculate a statistic for another variable defined as an ANALYSIS variable. For example, the following report will calculate the sum of the values of MSRP within each level of MAKE:

```
proc report data=sashelp.cars missing;
  columns make msrp;
  define make/group order=internal;
run;
```

Output 3 shows the summation results for each value of the GROUP variable.

Make	MSRP
Acura	$300,570
Audi	$822,850
BMW	$865,705

... continued ...

Output 3. Variable Defined as GROUP

Recall the ORDER= option simply controls how rows are ordered throughout the report. The ORDER= option can be used in conjunction with either the GROUP or ORDER option. Since we calculate results prior to the PROC REPORT step for clinical trial reporting, there is usually no practical use for the GROUP option.

REMOVING THE RISK OF INCORRECT REPORTING

For reporting clinical trial data, especially for data listings, it is our intention to always report all records. Therefore, within the pharmaceutical and biotechnology industries we try to adhere to the following practices:

1. Do not DEFINE variables as GROUP variables.
2. Explicitly DEFINE every variable to be either an ORDER variable or a DISPLAY variable. The DISPLAY option makes sure there is a row in the report for each observation in the input data set.

The following code will produce one row in the report for every observation in the data set:

```
proc report data=sashelp.cars missing;
  columns make msrp;
  define make/order order=internal;
  define msrp/display;
run;
```

Output 4 shows that using ORDER and DISPLAY produces a report having the same number of rows as observations in the input data set.

```
                Make          MSRP
                Acura       $36,945
                            $23,820
                            $26,990
                            $33,195
                            $43,755
... continued ...
```

Output 4. Using ORDER and DISPLAY on Each DEFINE

OPTIONS ONLY FOR THE ODS LISTING DESTINATION

There are several options available in PROC REPORT that apply only to the ODS LISTING destination (i.e. text-based output). These include:

- HEADLINE (PROC REPORT statement)
- HEADSKIP (PROC REPORT statement)
- SPLIT= (PROC REPORT statement)
- SPACING= (PROC REPORT and DEFINE statement)
- FLOW (DEFINE statement)
- WIDTH= (DEFINE statement)
- SKIP (BREAK statement)
- @n placement (LINE statement)
- Repeat characters, like "__" or "--" for spanning headers

When using the RTF or TAGSETS.RTF destination, we avoid using these options. The one exception is the SPLIT= option. In the RTF destinations, SPLIT= option will still be available to force line breaks within column headers. However, split characters embedded within data values are recognized only in ODS LISTING. Otherwise, the options listed above have no utility and can confuse a less experienced programmer attempting to reuse or debug a program. Furthermore, it is not a good programming practice to include code that does nothing, also known as dead code.

ODS TAGSETS.RTF

Now that we've reviewed the basic use of PROC REPORT as it pertains to clinical trial reporting, we turn our attention to the reporting techniques necessary to generate clinical trial reports in the TAGSETS.RTF destination. We begin with a data set that already contains all necessary derivations and calculations and is ready for display using PROC REPORT. Display 1 shows the data set we will be reporting on throughout the paper, which contains summarized demographics data. The underlying data came from the CDISC Pilot Project, an example clinical trial provided by CDISC to demonstrate the use of the CDISC data standards. It is available at https://www.cdisc.org/pilot-project-submission-package.

	varord	varlbl	statord	statlbl	Placebo|(N=86)	Xanomeline|High Dose|(N=84)	p-value\[1]
1	1	Age (y)	1	n	86	84	0.5934
2	1	Age (y)	2	Mean	75.2	74.4	
3	1	Age (y)	3	SD	8.59	7.89	
4	1	Age (y)	4	Median	76.0	76.0	
5	1	Age (y)	5	Min	52	56	
6	1	Age (y)	6	Max	89	88	
7	1	Age (y)	101	<65	14 (16%)	11 (13%)	0.1439
8	1	Age (y)	102	65-80	42 (49%)	55 (65%)	
9	1	Age (y)	103	>80	30 (35%)	18 (21%)	
10	2	Sex	N	n	86	84	0.1409
11	2	Sex	1	Male	33 (38%)	44 (52%)	
12	2	Sex	2	Female	53 (62%)	40 (48%)	
13	3	Race (Origin)	N	n	86	84	0.6477
14	3	Race (Origin)	1	Caucasian	75 (87%)	71 (85%)	
15	3	Race (Origin)	2	African Descent	8 (9%)	9 (11%)	
16	3	Race (Origin)	3	Hispanic	3 (3%)	3 (4%)	
17	3	Race (Origin)	9	Other	0 (0%)	1 (1%)	

Display 1. Summarized Demographics Data Set

In order to make use of the TAGSETS.RTF destination, we close the LISTING output and send the report to an RTF file using the following code:

```
ods listing close;
ods tagsets.rtf file="d:\drug\study\outputs\t-dm.rtf";  ❶

title j=l "Protocol: CDISCPILOT01"
      j=c "Population: Intent-to-Treat"
      j=r "PAGE_X_OF_Y";  ❷

title3 "Summary of Demographic and Baseline Characteristics";

footnote1 "[1] P-values are results of ANOVA treatment group comparisons
for continuous variables and Pearson's chisquare test for categorical
variables.";

footnote2 "NOTE: Percentages are based on the number of non-missing
values.";

footnote3 j=l "Source: d:\t-dm-sgf.rtf"
          j=r "Date: &sysdate9.";

proc report data=comb split='|' missing nocenter;

  columns varord varlbl statord statlbl col: pvalue;

  *Order variables;  ❸
  define varord /order order=internal noprint;
  define varlbl /order order=internal noprint;
  define statord /order order=internal noprint;

  *Reported columns;  ❸
  define statlbl /display '';
  define col: /display;
  define pvalue /display;
```

```
  *Labels over statistics;  ❹
  compute before varlbl;
    line varlbl $50.;
  endcomp;

run;
ods tagsets.rtf close;
ods listing;
```

❶ First, we close the ODS LISTING destination and open the TAGSETS.RTF destination. While it is possible to write to both destinations simultaneously, we don't want to maintain code for two different destinations since the RTF file will be our final deliverable. Since we did not explicitly specify a STYLE= option on the ODS TAGSETS.RTF statement, the default style (STYLES.RTF) will be used.

❷ We use the string "PAGE_X_OF_Y" as a placeholder. Later in the paper we discuss post-processing the RTF file to replace this text and derive the actual page numbers ourselves. This approach generates page numbers as static text and facilitates incorporation of this table into a larger document such as a clinical trial report. In situations where that is not a concern, we could instead insert RTF field codes for page numbers by replacing this placeholder text with `~{pageof}` inside the double quotes. The "~" is an ODS escape character, and we will elaborate on what this is later in the paper.

❸ Recall from our REPORT basics described earlier, it is advisable to DEFINE all our variables as either ORDER or DISPLAY variables.

❹ Some programmers like to insert an extra row in the data for placing the variable label text over the statistics in the report. This is not necessary. We can insert lines of text into our report before each variable segment using the COMPUTE block.

The first two pages of the RTF output are shown in Output 5. There are some features of the default RTF style that may not be desirable, such as italicized titles and footnotes, the grey background of the column headers, and the default column widths. Later in this paper, we illustrate how to modify style elements and attributes of the output via the STYLE= option.

Although TAGSETS.RTF attempts to perform pagination based on how much vertical space the table elements occupy, it does not always accurately determine how much can fit on a page. In Output 5, we can see there are two footnotes on page 2 that should not have been placed on their own page. The light grey line indicates the start of page 2. To correct this, we can explicitly specify the column widths for every variable to avoid unexpected word wrapping (e.g. see row for Male). We address this later in the paper.

Protocol: CDISCPILOT01 *Population: Intent-to-Treat* *PAGE_X_OF_Y*

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	
Sex					
n	86	84	84	254	0.1409
Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	

(Continued)

[1] P-values are results of ANOVA treatment group comparisons for continuous variables and Pearson's chisquare test for categorical variables.

NOTE: Percentages are based on the number of non-missing values.
Source: d:\t-dm-sgf.rtf *Date: 09NOV2019*

Output 5. TAGSETS.RTF Output Using Default RTF Style

STYLE= AND ~S= OVERRIDE OPTIONS

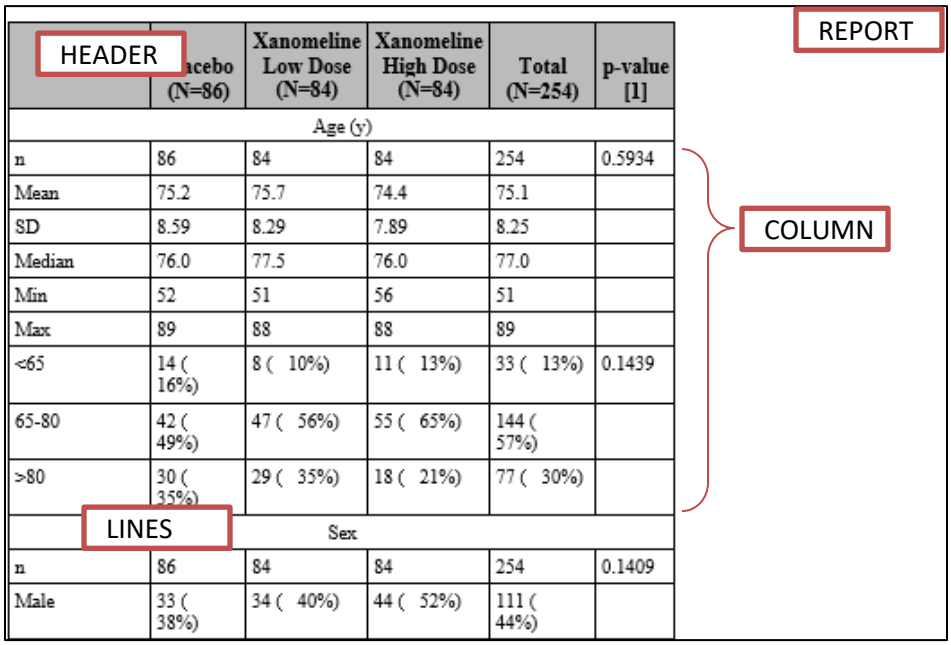
Many of the techniques for manipulating the RTF report involve the STYLE option. PROC REPORT styles are applied to a table based on named regions. The primary table regions we utilize are:

- REPORT
- HEADER
- COLUMN
- LINES

The syntax for the style option is `style(table-region)={style-attribute}`. This option can be used on the PROC REPORT, DEFINE, and/or COMPUTE statements as shown below:

```
proc report … style(table-region)={style-attribute};
  columns …;
  define … / style(table-region)={style-attribute};
  compute … / style(table-region)={style-attribute};
    …
  endcomp;
run;
```

All four regions are available in the PROC REPORT statement. The DEFINE statement accepts only the COLUMN and HEADER regions. The STYLE option on the COMPUTE block accepts just the LINES region. Style overrides given on the DEFINE and COMPUTE statements take precedence over those on the PROC REPORT statement. This paper includes many examples illustrating the use of STYLE= overrides. Display 2 shows the regions of a PROC REPORT output.



	cebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	
Sex					
n	86	84	84	254	0.1409
Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	

Display 2. Annotations of Table Regions for Style= Override

The REPORT region refers to the overall appearance of the table, including properties such as cell spacing, cell padding, and the borders between cells. The LINES region refers to cells inserted via the COMPUTE block and LINE statements. The COLUMN region consists of the body of the table while the HEADER region includes just the column headers.

There is also an in-line style override which allows us to alter the appearance of text in a title, footnote, or table cell. To do so, we use an ODS escape character and place the style override in-line with the data or statement containing text. The ODS escape character can be defined using the ODS ESCAPECHAR= statement. It is used to provide inline formatting such as bolding, superscript, subscript, font size, font color, insertion of Unicode characters, and much more. Moving forward we will use an ODS escape character of "~". To override text styling in the TITLE statement, we could do the following:

```
ods escapechar = "~";
title3 "~S={fontsize=12pt fontstyle=roman}Summary of Demographic and
Baseline Characteristics";
```

This changes the font size to 12pt and removes the *italic* style by specifying roman. One problem with this approach is that ODS does not account for the font size change in its vertical measurements. A better approach would be to use the HEIGHT= option with TITLE and FOOTNOTE statements:

```
title3 height=12pt "~S={fontstyle=roman}Summary of Demographic and Baseline
Characteristics";
```

Since we have the fontsize defined via the HEIGHT= option, we can remove the fontsize attribute from the in-line style override.

REPORT MODIFICATIONS

We now proceed through a series of modifications to our original report. The changes will progressively change the overall appearance of the report to achieve our desired output.

REMOVING CONTINUED TEXT AND EXTRA BLANK LINE BEFORE FOOTNOTES

TAGSETS.RTF automatically inserts a row with the text "(CONTINUED)" for multi-page reports. In addition, there is a blank line separating the body of the table and the titles and footnotes. Refer back to Output 5. To remove these, we can utilize the VSPACE and CONTINUE_TAG suboptions of the OPTIONS option (what a mouthful that was):

```
ods tagsets.rtf file="d:\drug\study\outputs\t-dm.rtf"
                options(continue_tag="off" vspace="no");
```

The result is shown in Output 6.

			Sex		
n	86	84	84	254	0.1409
Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	

[1] P-values are results of ANOVA treatment group comparisons for continuous variables and Pearson's chisquare test for categorical variables.

NOTE: Percentages are based on the number of non-missing values.

Source: d:\t-dm-sgf.rtf *Date: 09NOV2019*

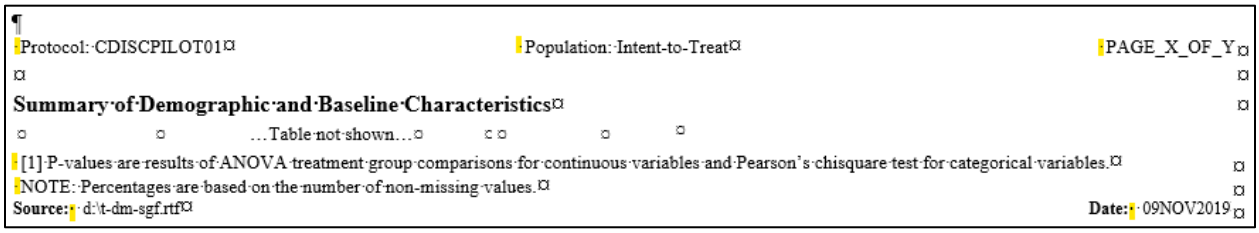
Output 6. Removal of "(CONTINUED)" and Blank Line

MODIFYING TITLES

By default, all titles and footnotes are **bold,** *italic*, and 13pt ***Times New Roman Font***. We can modify our titles and footnotes by using the HEIGHT= option and an in-line style override as shown:

```
title height=10pt j=l "~S={fontweight=light fontstyle=roman}Protocol:
CDISCPILOT01"
   j=c "~S={fontweight=light fontstyle=roman}Population: Intent-to-Treat"
   j=r "~S={fontweight=light fontstyle=roman}PAGE_X_OF_Y";

title3 height=12pt "~S={fontstyle=roman}Summary of Demographic and Baseline
Characteristics";

footnote1 height=10pt "~S={fontweight=light fontstyle=roman}[1] P-values
are results of ANOVA treatment group comparisons for continuous variables
and Pearson's chisquare test for categorical variables.";

footnote2 height=10pt "~S={fontweight=light fontstyle=roman}NOTE:
Percentages are based on the number of non-missing values.";

footnote3 height=9pt j=l "~S={fontweight=bold fontstyle=roman}Source:
~S={fontweight=light fontstyle=roman}d:\t-dm-sgf.rtf"
          j=r "~S={fontsize=9pt fontweight=bold fontstyle=roman}Date:
~S={fontweight=light fontstyle=roman}&sysdate9.";
```

Here, we changed the titles and footnotes to various sizes and weights. To unbold text, specify fontweight=light. However, there is an unfortunate side effect in that SAS inserts additional spaces as shown in Output 7.



Output 7. Unintentional Space Insertion

This is a known SAS bug. As a workaround, this problem can be avoided by handling title and footnotes styles directly in the style template by modifying the SystemTitle and SystemFooter style elements. This also eliminates the need to repeat the same style override in every title and footnote by consolidating that specification in one place. Style template modifications are beyond the scope of this paper but Haworth et al. (2009) give more details.

REMOVING GREY BACKGROUND IN COLUMN HEADERS

By default, the background color of the column headers is grey. We can alter all column headers by using a STYLE= override on the PROC REPORT statement. Instead of changing the background color to white, the following code will change the background to "No Color" in the HEADER region:

```
proc report data=comb split='|' missing
            style(header)={backgroundcolor=_undef_};
```

The result is shown below in Output 8.

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	

Output 8. Remove Column Header Color

CONTROLLING BORDERS BETWEEN CELLS

By default, black borders are present between cells. Some may prefer these border lines, but most pharmaceutical reports omit them. We will modify our code to only include lines above and below the table body so there is a visual break between titles, footnotes, and the table body. We will also place a line between the column headers and the table body. Lastly, we will decrease the cell padding and spacing between cells to 0 in order to fit more rows on each page. Here, we use another style override, but on the REPORT region:

```
proc report data=comb split='|' missing
            style(report)={rules=groups frame=hsides
                           cellspacing=0 cellpadding=0}
            style(header)={backgroundcolor=_undef_};
```

Using the RULES=GROUPS attribute we can obtain behavior analogous to the HEADLINE option used with the ODS LISTING destination. It places a blank line between the headers and the rest of the table. FRAME=HSIDES places horizontal lines above and below the table body. Note that the default value of FRAME=BOX places borders around all sides of the table. There are many combinations of RULES= and FRAME= options to get a desired appearance. See Haworth et al. (2009) for more details on these style attributes. Output 9 shows the cumulative effect of all our style overrides up to this point.

Protocol: CDISCPILOT01 Population: Intent-to-Treat PAGE_X_OF_Y

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
			Age (y)		
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	
			Sex		
n	86	84	84	254	0.1409
Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	
Female	53 (62%)	50 (60%)	40 (48%)	143 (56%)	
			Race (Origin)		
n	86	84	84	254	0.6477
Caucasian	75 (87%)	72 (86%)	71 (85%)	218 (86%)	
African Descent	8 (9%)	6 (7%)	9 (11%)	23 (9%)	
Hispanic	3 (3%)	6 (7%)	3 (4%)	12 (5%)	
Other	0 (0%)	0 (0%)	1 (1%)	1 (<1%)	
			MMSE		
n	86	84	84	254	0.5947
Mean	18.0	17.9	18.5	18.1	
SD	4.27	4.22	4.16	4.21	

[1] P-values are results of ANOVA treatment group comparisons for continuous variables and Pearson's chisquare test for categorical variables.

NOTE: Percentages are based on the number of non-missing values.

Source: d:\t-dm-sgf.rtf **Date:** 07DEC2019

Output 9. Cumulative Effect of Changes

As an alternative, all the STYLE= overrides used so far could be handled by creating a custom STYLE template using the TEMPLATE procedure. The font changes can be made by modifying the Fonts, SystemTitle, and SystemFooter style elements. Attribute changes in the REPORT region (i.e. cell spacing, padding and borders) can be accomplished by modifying the Table style element. The white header background can be achieved by modifying the Header style element. We refer the reader again to Haworth et al. (2009) for a thorough look at modifying Style Elements via PROC TEMPLATE.

CHANGING COLUMN WIDTH

Now, let's widen the widths of the columns to get rid of unwanted text wrapping. Unfortunately, the usual WIDTH= option we used with the ODS LISTING destination does not apply here. However, we can use the CellWidth attribute via a STYLE override:

```
*Reported columns;
define statlbl /display '' style(column)={cellwidth=2in};
define col: /display style(column)={cellwidth=1.25in};
define pvalue /display style(column)={cellwidth=1in};
```

Output 10 shows the new cell widths. Notice how the wider cell width provides more space so the text does not wrap for the age groups as it does in Output 9.

Protocol: CDISCPILOT01 Population: Intent-to-Treat PAGE_X_OF_Y

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
		Age (y)			
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	

Output 10. Changing Cell Widths

An alternative method to specifying individual widths for each column is to use the OutputWidth attribute on the PROC REPORT statement:

```
proc report data=comb split='|' missing
            style(report)={rules=groups frame=hsides
                           cellspacing=0 cellpadding=0
                           outputwidth=100%}
            style(header)={backgroundcolor=_undef_};
```

Output 11 shows how the columns now stretch to the right margin.

Protocol: CDISCPILOT01 Population: Intent-to-Treat PAGE_X_OF_Y

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
		Age (y)			
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	

Output 11. Force Column Widths to Span Width of Page

In most cases, we can get away with this alternative approach. However, in other situations we may wish to gain more control over the output and explicitly choose our own column widths on each DEFINE statement.

LINE JUSTIFICATION

Next, we left justify the variable labels currently being placed in the center of each column by using the LINES region with a STYLE override:

```
compute before varlbl /style(lines)={just=l};
  line varlbl $50.;
endcomp;
```

Output 12 shows the variable labels being left justified.

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	

Output 12. Variable Labels Left Justified

INDENTING STATISTICS

One approach to controlling indentation is to concatenate padded spaces to the beginning of the statistical text within the data, (e.g. `statlbl='  '||statlbl;`). If we use this approach, we need to make sure to specify the ASIS=ON attribute in the column region:

```
define statlbl /display '' style(column)={cellwidth=2in asis=on};
```

Alternatively, we can more directly affect the indentation without needing to concatenate padded spaces within the data. To do so, we modify the PADDINGLEFT attribute on the same define statement:

```
define statlbl /display '' style(column)={cellwidth=2in paddingleft=0.1in};
```

Output 13 shows indentation of statistical text. Regardless of which approach we use, the appearance will be similar.

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	

Output 13. Indentation of Statistical Text

HEADER AND COLUMN JUSTIFICATION

By default, the headers (i.e. the HEADER region) are center justified and the data (i.e. COLUMN region) are left justified. The following code would left justify all headers:

```
proc report data=comb split='|' missing style(header)={just=l};
```

Since the override is performed on the PROC REPORT statement, the justification is applied to all headers. If we wanted to apply different justification to different columns (e.g. in a listing report), we can utilize different overrides on the DEFINE statements:

```
define col:    /display style(header)={just=c};
define pvalue  /display style(header)={just=r};
```

This code would apply center justification to the treatment column headers and right justify the p-value column headers. Similarly, we can manipulate the body of the table (i.e. the COLUMN region) by using similar style overrides and replacing HEADER with COLUMN.

```
define col:    /display style(column)={just=c};
define pvalue  /display style(column)={just=r};
```

If we wanted the header and column regions to have the same justification, we can specify multiple regions at once:

```
define col:    /display style(header column)={just=c};
```

For the report we are currently working on, we will not apply any of these justifications. This section was added for completeness. In a clinical trial context, these justification options are used frequently with data listings.

DECIMAL ALIGNMENT

For decimal alignment of the results columns we can choose between two different approaches:

1) Left justify the COLUMN region with leading spaces concatenated in the data and specify ASIS=ON, or
2) Apply decimal tab alignment using the JUST=D option with the PADDINGRIGHT attribute.

In the ODS LISTING world, we would use the first approach but the ASIS attribute would not be needed (since the listing destination is simply text). The downside of (1) is that when using a proportional font like Times New Roman, the alignment will not be perfect. The second approach is predicated on decimals being present in the data and some trial and error is necessary with the PADDINGRIGHT attribute:

```
define col:  /display
              style(column)={cellwidth=1.25in just=d paddingright=0.3in};
define pvalue/display
              style(column)={cellwidth=1in just=d paddingright= 0.25in};
```

Output 14 shows the report with decimal alignment being used. PADDINGRIGHT is used to shift the results towards the center and under the header text.

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	

Output 14. Decimal Alignment Using Decimal Tab

LINE INSERTIONS AND PAGE BREAKS

Unlike the ODS LISTING destination, where the SKIP option of the BREAK statement can be used, we must use the COMPUTE block to insert lines in our RTF output. First, we need to derive a new variable to assist with placing the lines where we want them. At the same time, we create a page breaking variable so the pages do not break between rows of statistics as seen in Output 5 and Output 9:

```
data final;
  set comb;
  by varord statord;

  if varord in (1 2 3) then pg=1;   ❶
  else if varord in (4 5 ) then pg=2;
  else if varord in (6 7 8 ) then pg=3;
  else pg=4;

  if statord ge 100 then varord2=varord+100;    ❷
  else varord2=varord;
run;
```

❶ A variable PG is created that will be used as a pagination variable. With some trial and error, we've assigned values of PG so that page breaks will not occur in the middle of variable summaries. There are more dynamic programming approaches to handle this situation (e.g. counting the number of lines each variable uses), but this simpler approach meets the needs of this paper.

❷ Referring to the values of STATORD shown in Display 1, we see when there are continuous summaries and categorical group counts for the same variable (i.e. Age) that the groups are assigned 101, 102, etc. So, we know when STATORD increases to a value of 100 and up, we would like a line inserted. Assigning VARORD2 the value of VARORD + 100, will allow the age groups to have the same value but different from the continuous summary. See Display 3.

	pg	varlbl	varord	varord2	statord	statlbl	Placebo|(N=86)
1	1	Age (y)	1	1	1	n	86
2	1	Age (y)	1	1	2	Mean	75.2
3	1	Age (y)	1	1	3	SD	8.59
4	1	Age (y)	1	1	4	Median	76.0
5	1	Age (y)	1	1	5	Min	52
6	1	Age (y)	1	1	6	Max	89
7	1	Age (y)	1	101	101	<65	14 (16%)
8	1	Age (y)	1	101	102	65-80	42 (49%)
9	1	Age (y)	1	101	103	>80	30 (35%)
10	1	Sex	2	2	N	n	86
11	1	Sex	2	2	1	Male	33 (38%)
12	1	Sex	2	2	2	Female	53 (62%)
13	1	Race (Origin)	3	3	N	n	86
14	1	Race (Origin)	3	3	1	Caucasian	75 (87%)
15	1	Race (Origin)	3	3	2	African Descent	8 (9%)
16	1	Race (Origin)	3	3	3	Hispanic	3 (3%)
17	1	Race (Origin)	3	3	9	Other	0 (0%)

Display 3. Page Breaking and Line Insertion Variables

Now that we have these variables created, let's update our PROC REPORT code:

```
proc report data=comb split='|' missing nocenter
  style(report)={rules=groups frame=hsides cellspacing=0 cellpadding=0 }
  style(header)={backgroundcolor = _undef_};

  columns pg varord varlbl varord2 statord statlbl col: pvalue;

  *Order variables;
  define pg /order order=internal noprint;
  define varord /order order=internal noprint;
  define varlbl /order order=internal noprint;
  define varord2 /order order=internal noprint;
  define statord /order order =internal noprint;

  *Reported columns;
  define statlbl /display '' style(column)={cellwidth=2in
                                            paddingleft=0.1in};
  define col: /display style(column)={cellwidth=1.25in just=d
                                      paddingright=0.3in};
  define pvalue /display style(column)={cellwidth=1in just=d
                                        paddingright=0.25in};
  *Derived page breaking;
  break after pg/page;  ❶

  *Labels over statistics;
  compute before varlbl/style(lines)={just=l};
    line varlbl $50.;
  endcomp;

  *Line breaks after variable groups;
  compute after varord2; ❷
    line '';
  endcomp;

run;
```

❶ Fortunately for us, the PAGE option of the BREAK statement still works in RTF (unlike the SKIP option).

❷ Using a COMPUTE block, we insert a blank line each time the value of VARORD2 changes.

See Output 15 for the effect of these two changes to our report.

Protocol: CDISCPILOT01 Population: Intent-to-Treat PAGE_X_OF_Y

Summary of Demographic and Baseline Characteristics

	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value [1]
Age (y)					
n	86	84	84	254	0.5934
Mean	75.2	75.7	74.4	75.1	
SD	8.59	8.29	7.89	8.25	
Median	76.0	77.5	76.0	77.0	
Min	52	51	56	51	
Max	89	88	88	89	
<65	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
65-80	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
>80	30 (35%)	29 (35%)	18 (21%)	77 (30%)	
Sex					
n	86	84	84	254	0.1409
Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	
Female	53 (62%)	50 (60%)	40 (48%)	143 (56%)	
Race (Origin)					
n	86	84	84	254	0.6477
Caucasian	75 (87%)	72 (86%)	71 (85%)	218 (86%)	
African Descent	8 (9%)	6 (7%)	9 (11%)	23 (9%)	
Hispanic	3 (3%)	6 (7%)	3 (4%)	12 (5%)	
Other	0 (0%)	0 (0%)	1 (1%)	1 (<1%)	

[1] P-values are results of ANOVA treatment group comparisons for continuous variables and Pearson's chisquare test for categorical variables.
NOTE: Percentages are based on the number of non-missing values.
Source: d:\t-dm-sgf.rtf **Date:** 07DEC2019

Output 15. Report with Manual Pagination and Line Insertions

PAGE X OF Y

The final step in producing our clinical report is to replace the "PAGE_X_OF_Y" placeholders with actual page numbers such as "Page 1 or 4", "Page 2 of 4", etc. While RTF provides field codes that can be used to automate page numbering, we want our page numbers to consist of static text. This will ensure the page numbers remain the same if the table is used as part of a larger document such as a clinical study report. If we use RTF field codes, the page numbering is based on the number of pages in the document and changes as pages are added or removed.

Fortunately, RTF documents are nothing more than text files that utilize a specific markup syntax. Thus, we can easily post-process our RTF file using SAS to read the RTF file, make any desired changes, and write the modified RTF back out. In general, this process can be performed in a single pass through the file. However, since we do not know in advance the number of pages in the document, we need two passes, each of which is a separate DATA step.

In the first DATA step, we read the contents of the RTF file into a temporary data set. Each line in the RTF file becomes a record in the data set, and the entire line is stored in a single character variable. A second variable holds the current page number. Each time the "PAGE_X_OF_Y" placeholder is encountered, the page count variable is incremented. Once the entire file has been read, the value of the page count variable represents the total

number of pages in our document, which is the "Y" in "Page X of Y." That total page count is stored in a macro variable for use in the second DATA step.

```
* Read in RTF file and count number of times the placeholder string is ;
* found (which should correspond with total number of pages).          ;
data _readrtf_;
  infile "d:\drug\study\outputs\t-dm.rtf" missover length=l end=eof
    lrecl=2000;
  input line $varying2000. L;
  retain pagecount;
  if index(line,"PAGE_X_OF_Y") then pagecount + 1;

  * Store total number of pages in a macro variable.;
  if eof then call symput("_numpages",strip(put(pagecount,best.)));
run;
```

The second DATA step reads the temporary data set created in the first DATA step. It writes each record back out to the same RTF file (overwriting the original). When the "Page X of Y" placeholder is encountered, it is replaced with actual page numbers. The "X" comes from the page count variable, and the "Y" comes from the macro variable that holds the total page count.

```
*Rewrite the RTF file, replacing the placeholder with Page X of Y. ;
data _null_;
  file "d:\drug\study\outputs\t-dm.rtf" flowover;
  set _readrtf_;
  if index(line,"PAGE_X_OF_Y") then do;
    line = tranwrd(line, "PAGE_X_OF_Y",
                  "Page "||strip(put(pagecount,best.))||" of &_numpages");
  end;
  put line;
run;
```

Output 16 shows the header of the RTF document after this post-processing has been performed. The "Page X of Y" text has been replaced with the correct page numbers.

Protocol: CDISCPILOT01	Population: Intent-to-Treat	Page 1 of 4

Output 16. Page X of Y Implementation

OTHER RTF TECHNIQUES

There are several other techniques that are useful when producing RTF files. The following were not used in the demographics table but may be encountered when working with other types of clinical outputs.

LINE FEEDS – SPLIT ALTERNATIVE

As mentioned earlier, the SPLIT= option is honored in column headers but not within the body of the table when using the RTF destinations. Instead, we must insert RTF line feeds to replicate this behavior. This is done using a DATA step to embed an ODS escape character followed by an "n" within the data values:

```
ods escapechar='~';
data linefeeds;
  set ae;
  tox_out=catx('/~n', aetoxgr, aeout );
run;
```

Display 4 shows the line feed in the data as well as in the abbreviated adverse event listing output.

TOX_OUT
1/~nRecovered/Resolved

Toxicity·Grade/↵ Outcome¤
1/↵ Recovered/Resolved¤

Display 4. Line Feed

You can view the line feeds in the RTF document by turning on the option to show paragraph marks and other hidden formatting symbols in Microsoft Word®.

NON-BREAKABLE SPACE

Even if the ASIS attribute were active for the COLUMN region, there are some scenarios in which spaces are not honored. For example, when presenting adverse event listings, we often stack the MedDRA system organ class (SOC) and preferred term (PT) by concatenating the two variables and indenting the "bottom" variable:

```
socpt=catx("~n  ",aebodsys,aedecod);
```

However, in this case, the padded spaces after the line feed are not honored. The workaround for this is to use a non-breakable space (NBSP). Since non-breakable spaces are invisible to the human eye (i.e. they look just like normal spaces in the editor), it is a good programming practice to create a macro variable to store the non-breakable space. This communicates the use of NBSP to someone who may be reusing your program, as well as avoiding compilation errors. A small DATA step with a hexadecimal reference can accomplish this:

```
data _null_;
  nbsp='A0'x;
  call symputx('nbsp',nbsp);
run;
```

Now, we can update our concatenated variables:

```
socpt=catx("~n . ",aebodsys,aedecod);
```

Display 5 shows the NBSP in the abbreviated adverse event listing output.

socpt
CARDIAC DISORDERS~n ATRIAL FIBRILLATION

System·Organ·Class↵ °·Preferred·Term¤
CARDIAC·DISORDERS↵ °·ATRIAL·FIBRILLATION¤

Display 5. Non-breakable Space

You can view the NBSP in the RTF document by showing paragraph marks and other hidden formatting symbols in Microsoft Word. A non-breakable space is represented by a small superscript circle.

CALL DEFINE – INDENT CERTAIN ROWS

For adverse event reports, we usually summarize by SOC and PT. Furthermore, we usually indent PT and display it beneath system organ class as shown in Display 5. The difference in this scenario is that the data structure will have PT as its own observation and not concatenated to SOC on the same observation. See Display 6 for an abbreviated look at the data set.

	varord	varlbl	statlbl	note	Placebo|(N=86)
1	1	Number of subjects with an adverse event			65 (75.6%)
2	2	1st Occurrence of SOC Flag	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS		21 (24.4%)
3	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE PRURITUS	PT	6 (7.0%)
4	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE ERYTHEMA	PT	3 (3.5%)
5	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE DERMATITIS	PT	5 (5.8%)
6	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE IRRITATION	PT	3 (3.5%)
7	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE VESICLES	PT	1 (1.2%)
8	2	1st Occurrence of Preferred Term Flag	FATIGUE	PT	1 (1.2%)
9	2	1st Occurrence of Preferred Term Flag	OEDEMA PERIPHERAL	PT	2 (2.3%)
10	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE SWELLING	PT	0
11	2	1st Occurrence of Preferred Term Flag	APPLICATION SITE URTICARIA	PT	0
12	2	1st Occurrence of Preferred Term Flag	CHILLS	PT	1 (1.2%)

Display 6. Summarized Adverse Event Data

Note the variable called NOTE which identifies records that are preferred terms as opposed to the system organ class record. We can take advantage of this structure to eliminate the need for non-breakable spaces by using the CALL DEFINE statement within PROC REPORT:

```
proc report data=report split='|' missing ;

  columns varord ...<other variables>... note statlbl col:;

  *... Other PROC REPORT statements ...;

  * Indent PT records;
  compute statlbl;
    if note='PT' then do;
      call define(_col_,'style','style={paddingleft=.1in}');
    end;
  endcomp;
run;
```

The CALL DEFINE statement allows us to override style attributes to add padding to the left of the table cell. Even if the text wraps multiple lines, the indention is maintained. The use of the IF-THEN-DO block allows us to perform this action only for records identified as preferred terms as shown in Output 17.

Protocol: CDISCPILOT01 Population: Safety Page 1 of 13

Incidence of Treatment Emergent Adverse Events by Treatment Group

System Organ Class Preferred Term	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total Active (N=254)
Number of subjects with an adverse event	65 (75.6%)	77 (91.7%)	76 (90.5%)	218 (85.8%)
GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	21 (24.4%)	47 (56.0%)	40 (47.6%)	108 (42.5%)
APPLICATION SITE PRURITUS	6 (7.0%)	22 (26.2%)	22 (26.2%)	50 (19.7%)
APPLICATION SITE ERYTHEMA	3 (3.5%)	12 (14.3%)	15 (17.9%)	30 (11.8%)
APPLICATION SITE DERMATITIS	5 (5.8%)	9 (10.7%)	7 (8.3%)	21 (8.3%)
APPLICATION SITE IRRITATION	3 (3.5%)	9 (10.7%)	9 (10.7%)	21 (8.3%)
APPLICATION SITE VESICLES	1 (1.2%)	4 (4.8%)	6 (7.1%)	11 (4.3%)
FATIGUE	1 (1.2%)	5 (6.0%)	5 (6.0%)	11 (4.3%)
OEDEMA PERIPHERAL	2 (2.3%)	1 (1.2%)	2 (2.4%)	5 (2.0%)
APPLICATION SITE SWELLING	0	1 (1.2%)	2 (2.4%)	3 (1.2%)
APPLICATION SITE URTICARIA	0	2 (2.4%)	1 (1.2%)	3 (1.2%)
CHILLS	1 (1.2%)	1 (1.2%)	1 (1.2%)	3 (1.2%)

Output 17. Indentation of Preferred Terms

SPANNING HEADERS

Unlike the ODS LISTING destination, we cannot use repeat characters like "_" or "-". Typically, these are used in column headers to span an underline over multiple columns. Using TAGSETS.RTF, we need to use an in-line style override. Also, since there is no SPACING= option available in TAGSETS.RTF, we place an empty dummy variable between spanning headers:

```
proc report data=comb split='|' missing;
  columns
  sitegr1 siteid
  ("Placebo|(N=86)~S={borderbottomwidth=1}" itt0  eff0  com0) ❶
  _empty  ❷
  ("Xanomeline|Low Dose|(N=84)~S={borderbottomwidth=1}" itt54 eff54 com54)
  _empty
  ("Xanomeline|High Dose|(N=84)~S={borderbottomwidth=1}" itt81 eff81 com81)
  _empty
  ("Total|(N=254)~S={borderbottomwidth=1}" itt99 eff99 com99);

  *Reported columns;
  define sitegr1/display'Pooled|Id' style(column)={just=c} ;
  define siteid /display'Site|Id' style(column)={just=c} ;
  define itt:   /display style(column)={just=r paddingright=0.3in} 'ITT';
  define eff:   /display style(column)={just=r paddingright=0.3in} 'Eff';
  define com:   /display style(column)={just=r paddingright=0.3in} 'Com';
  define _empty /display '';
run;
```

❶ Similar to the approach used to change font sizes of titles and footnotes, we use in-line styles to override the *BorderBottomWidth* attribute to display a spanning header underline across multiple columns.

❷ A variable containing null or missing records can be used to provide space between columns so the spanning header underline does not run together.

Output 18 shows the spanning header underlines with spacing between them.

Protocol: CDISCPILOT01 Population: All Subjects Page 1 of 1

Summary of Number of Subjects by Site

Pooled Id	Site Id	Placebo (N=86)			Xanomeline Low Dose (N=84)			Xanomeline High Dose (N=84)			Total (N=254)		
		ITT	Eff	Com	ITT	Eff	Com	ITT	Eff	Com	ITT	Eff	Com
701	701	14	14	11	13	13	5	14	14	7	41	41	23
703	703	6	5	4	6	5	1	6	5	2	18	15	7
704	704	9	9	5	8	7	3	8	8	0	25	24	8
705	705	5	3	2	5	5	3	6	4	1	16	12	6
708	708	9	9	7	8	8	2	8	5	2	25	22	11
709	709	7	7	5	7	6	2	7	7	3	21	20	10
710	710	11	8	6	10	10	2	10	8	5	31	26	13
713	713	3	3	3	3	3	2	3	2	2	9	8	7
716	716	8	8	7	8	8	3	8	8	3	24	24	13
718	718	4	4	3	5	5	1	4	4	1	13	13	5
900	702	0	0	0	1	1	0	0	0	0	1	1	0
900	706	1	1	1	1	1	0	1	1	0	3	3	1
900	707	1	1	1	1	1	0	0	0	0	2	2	1
900	711	1	1	1	1	1	0	2	1	0	4	3	1
900	714	2	2	2	2	2	1	2	2	1	6	6	4
900	715	3	2	2	3	3	1	2	2	0	8	7	3
900	717	2	2	0	2	2	2	3	3	3	7	7	5
TOTAL		86	79	60	84	81	28	84	74	30	254	234	118

NOTE: ITT: Number of subjects in the ITT population, Eff: Number of subjects in the Efficacy population; Com: Number of subjects completing Week 24.

Source: d:\xanomeline\CDISCPILOT01\programs\t-sites.rtf **Date:** 14DEC2019

Output 18. Spanning Headers Underlines

SUPERSCRIPTS, SUBSCRIPTS, AND SPECIAL CHARACTERS

We can use an escape character again to insert superscript, subscript, and special Unicode characters. The following are 3 independent snippets of code:

- `by1lbl='Body Mass Index (kg/m~{super 2})';`
- `param=tranwrd(param,' max','~{sub max}');`
- `label AUCINF='AUC~{sub ~{unicode 221e}}';`

Display 7 shows these characters being rendered in the RTF file.

Body Mass Index (kg/m^2)	C_{max} (ng/mL)	
n	n	
Mean (SD)	Mean (SD)	**AUC_∞**
Median	Median	
Q1,Q3	Min, Max	
Min, Max	Geometric mean	
	Geometric CV%	

Display 7. Superscripts, Subscripts, and Special Characters in RTF

Notice how we can nest Unicode characters within a subscript for AUC∞.

DEALING WITH ORDER VARIABLES NOT REPEATING

Another unfortunate side effect when dealing with TAGSETS.RTF is that order variables do not repeat after a page break. Output 19 shows page 2 of an output where the subject number does not get displayed.

Listing of Vital Signs

Subject number	Study day	Visit	Systolic Blood Pressure (mmHg)
	29	AMBUL ECG REMOVAL	138
	42	WEEK 6	131
	54	WEEK 8	137
	83	WEEK 12	128
	111	WEEK 16	122
	139	WEEK 20	129
	172	WEEK 24	130
	180	WEEK 26	131
01-701-1033	-8	SCREENING 1	128
	-2	SCREENING 2	115
	1	BASELINE	136
	14	AMBUL ECG PLACEMENT	151
	15	WEEK 2	138
	28	WEEK 4	130
	182	RETRIEVAL	130

Output 19. Order Variable Repetition

SAS documentation, blogs, and forums will point us to the SPANROWS option of PROC REPORT, but this is not a robust solution for the following reasons:

- It is dependent on where LINE statements are placed.

- Page breaks do not always occur in the same places they would occur when not using SPANROWS.
- Using SPANROWS makes it more difficult to perform validation by reading the RTF output back into a SAS data set.
- At times, SPANROWS has a negative interaction with the FRAME=HSIDES attribute as shown in Output 20.

300-103	-13	29MAY2019 08:30	SCREENING	123/84	65	36.5	14	90.3
	-1	10JUN2019 08:31	DAY -1	124/81	73	36.5	14	88.8
	1	11JUN2019 09:03	DAY 1/ PRE-DOSE	115/77	69	36.9	14	

* Temperature: <36 or >38 degrees Celsius;
Pulse rate: >100 or <60, increase from baseline >20 or decrease from baseline >20 bpm;
Systolic blood pressure: >140 or <90, increase from baseline of >40 or decrease from baseline >30;
Diastolic blood pressure: >90 or < 50, increase from baseline of >30 or decrease from baseline >20;
Respiratory rate: <12 or >20 bpm;
Weight: >7% decrease or >7% increase from baseline.

Output 20. Horizontal Rule from FRAME=HSIDES Broken

For these reasons, we must resort to deriving our own paging variable:

```
data final;
  retain pg 0 ln 0;
  set vs;
  by usubjid;

  ln + 1;                          *Increment for each data row/observation;
  if first.usubjid then ln+1;      *Increment for blank lines between subjects;

  if ln gt 28 then do;
    Pg+1;                          *Increment page count;
    ln=0;                          *Reset lines;
    ln + 1;                        *Increment for each data row/observation;
    if first.usubjid then ln+1;    *Increment for blank lines between subjects;
  end;
run;
```

This method may take some trial and error. The number of observations that can fit on a page varies depends on the number of titles and footnotes. For this example, suppose 28 lines can fit on a page. Incrementing the line counter for each observation and for the blank lines we plan to insert between subjects, we know when it is appropriate to increment the page counter. After we increment the page counter, we must reset the line counter and recount the existing observation. Then we can utilize this page counter like we did in the PROC REPORT code that generated Output 15, which will force the subject ID to be displayed on each new page even if the subject data continues to the second page.

We recognize that this nullifies one of the main advantages of TAGSETS.RTF having vertical measurement. Despite this setback, TAGSETS.RTF does give you more control and flexibility that the original RTF destination does not (e.g. control over page panels, watermarks, and lines between titles/footnotes and body of report). For more information, see the SAS documentation for the ODS TAGSETS.RTF statement.

CONCLUSION

Change is always hard; fear of the unknown can prevent us from moving forward, developing better practices, and innovating. Although many pharmaceutical and biotechnology companies have been slow to move beyond ODS LISTING, we are not giving up from spreading the good news about the RTF destinations. Using either ODS RTF or ODS TAGSETS.RTF can harmonize fonts between outputs and the main clinical study report, as well as provide a common destination between tables, listings and figures by having a file

format compatible with graphics procedures. Most importantly, it lays the foundation for a more robust validation process by having the option to read the actual RTF file back into a SAS data set as a basis for comparison with output created by a quality control programmer. We hope this paper will get programmers over the main learning curve for creating RTF outputs and allow our industry to move beyond text-based output.

REFERENCES

Haworth, Lauren E., Zender, Cynthia L., and Burlew, Michele M. 2009. *Output Delivery System: The Basics and Beyond*. Cary, NC: SAS Institute Inc.

Clinical Data Interchange Standards Consortium. (2007, February). Pilot Project Submission Package. *CDISC.* Retrieved from https://www.cdisc.org/pilot-project-submission-package.

ACKNOWLEDGEMENTS

We'd like to thank Jane Eslinger for letting us pick her brain on all the little details about TAGSETS.RTF (sometimes via a SAS Support Ticket), and for getting us in touch with SAS developers to identify best practices when creating RTF reports.

We'd like to thank Scott Kosten and Richann Watson for their thorough review of this paper and for providing valuable feedback.

CONTACT INFORMATION

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

Christopher J. Smith
Cytel
chris.smith@cytel.com
https://www.linkedin.com/in/smithchr/

Joshua M. Horstman
Nested Loop Consulting
josh@nestedloopconsulting.com
https://www.linkedin.com/in/joshuahorstman/