PharmaSUG 2018 - Paper HT-01

Manipulating Statistical and Other Procedure Output to Get the Results That You Need

Vincent DelGobbo, SAS Institute Inc.

ABSTRACT

Many scientific and academic journals require that statistical tables be in a specific format (for example, the American Psychological Association [APA] style). This paper shows you how to change the output of any SAS® procedure to conform to any journal or corporate style. You'll learn how to save data from any SAS procedure, change the data format using the DATA step, and dynamically create a format based on your data. You'll also use the SAS Output Delivery System (ODS) inline formatting functions and style overrides, and produce several output formats, including HTML, RTF (for Microsoft Word), PDF, and Microsoft Excel files. Finally, you'll learn how to create and deliver the output on-demand using SAS server technology. This paper is appropriate for all SAS skill levels.

INTRODUCTION

This paper focuses on the data manipulation techniques used to transform procedure output, rather than on the statistical methods and procedures that produce the report. The report shown in Figure 1 was created using the results from the SAS SUMMARY and MIXED procedures, and is displayed using the REPORT procedure. The report shows statistics for the change from baseline cholesterol values for two dosages of the drug Xanomeline (54 mg and 81 mg) and for a placebo over time. While this type of output is common for those performing clinical drug trials in the United States, the concepts and techniques that you learn here can be applied to other data and output.

You could create this report manually using word processing software, but copying and pasting SAS procedure results is both laborious and error-prone. Instead, we programmatically create the report without the need for editing, resulting in a process that is accurate and repeatable with different data.

The report is categorized into three main sections:

- Section 1 - Summary statistics (N, MEAN, MIN, MAX, and so on) created by the SUMMARY procedure.
- Section 2 - LS mean standard estimates and confidence intervals, created by the MIXED procedure.
- Section 3 - Differences between the two active dosages of the drug and the placebo, created by the MIXED procedure.

The JOURNAL style, supplied by SAS and used here, provides results that closely match the requirements of the APA, but there are some differences. Refer to an earlier paper by this author to learn how to create a style that conforms to the APA requirements (DelGobbo 2015).

The code in this paper was tested using SAS 9.4M5 and Microsoft Word 2010 software. A copy of the data and code are available http://support.sas.com/papers/delgobbo-ods2018.

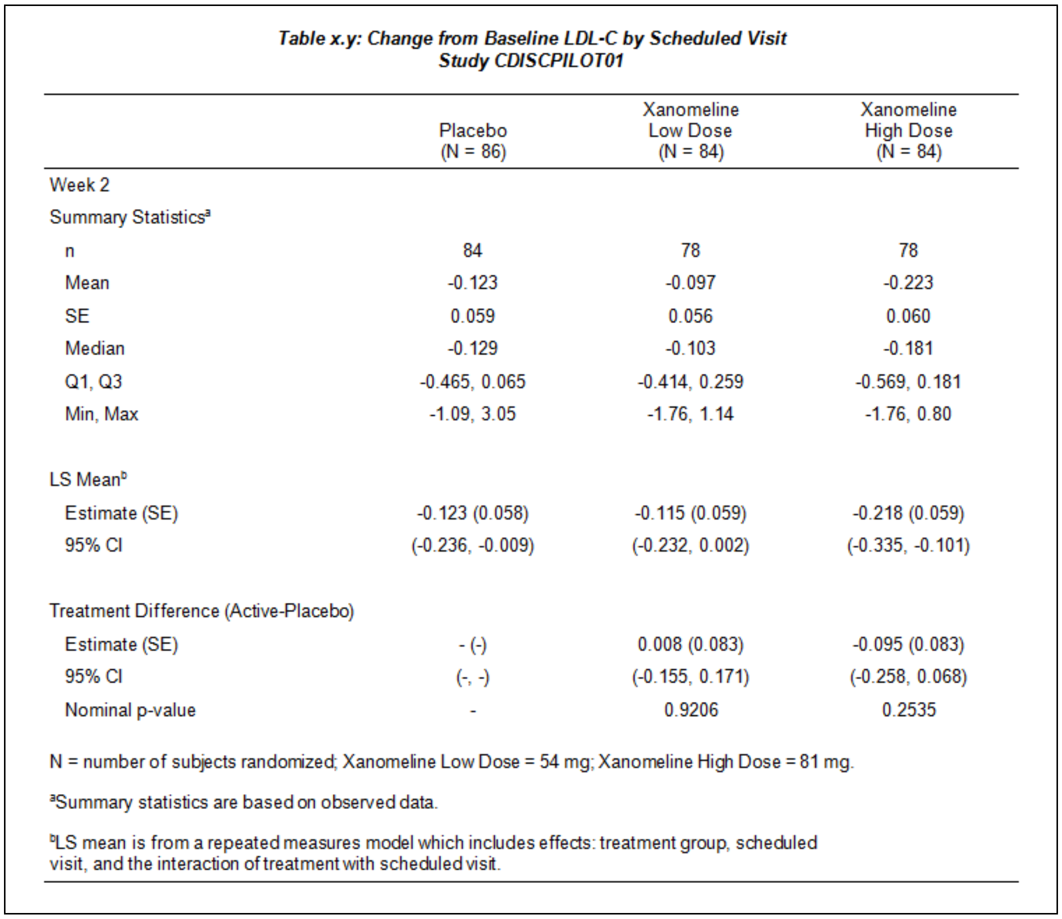


Table x.y: Change from Baseline LDL-C by Scheduled Visit
Study CDISCPILOT01

	Placebo (N = 86)	Xanomeline Low Dose (N = 84)	Xanomeline High Dose (N = 84)
Week 2			
Summary Statistics[a]			
n	84	78	78
Mean	-0.123	-0.097	-0.223
SE	0.059	0.056	0.060
Median	-0.129	-0.103	-0.181
Q1, Q3	-0.465, 0.065	-0.414, 0.259	-0.569, 0.181
Min, Max	-1.09, 3.05	-1.76, 1.14	-1.76, 0.80
LS Mean[b]			
Estimate (SE)	-0.123 (0.058)	-0.115 (0.059)	-0.218 (0.059)
95% CI	(-0.236, -0.009)	(-0.232, 0.002)	(-0.335, -0.101)
Treatment Difference (Active-Placebo)			
Estimate (SE)	- (-)	0.008 (0.083)	-0.095 (0.083)
95% CI	(-, -)	(-0.155, 0.171)	(-0.258, 0.068)
Nominal p-value	-	0.9206	0.2535

N = number of subjects randomized; Xanomeline Low Dose = 54 mg; Xanomeline High Dose = 81 mg.

[a]Summary statistics are based on observed data.

[b]LS mean is from a repeated measures model which includes effects: treatment group, scheduled visit, and the interaction of treatment with scheduled visit.

Figure 1. Final Report for Week 2 of the Study

SAMPLE DATA

The data used to produce the output of Figure 1 was obtained from the Clinical Data Interchange Standards Consortium (CDISC) pilot project (CDISC 2018). The column headings for the report and part of the title were dynamically generated from data in the subject-level demographic and characteristic data in the ADSL SAS table. Information about the columns of interest for this table are shown in Table 1. The values of TRT01PN represent the treatment that the subject received, either Placebo (0), Xanomeline Low Dose (54), or Xanomeline High Dose (81). There is one record for each subject in the study.

Column Name	Type	Label	Values
STYDYID	Character	Study Identifier	CDISCPILOT01
TRT01PN	Numeric	Planned Treatment for Period 01 (N)	0, 54, and 81
TRT01P	Character	Planned Treatment for Period 01	Placebo, Xanomeline Low Dose, and Xanomeline High Dose

Table 1. Column Properties and Data Values for the ADSL SAS Table

The ANALYSIS SAS table is a subset of the ADLBC laboratory results table provided by CDISC, and is used to compute the statistics in the body of the report. Table 2 shows the properties of the columns in this table. Only records for cholesterol results are included.

Column Name	Type	Label	Typical Values
TRT01PN	Numeric	Planned Treatment (N)	0, 54, and 81
TRT01P	Character	Planned Treatment	Placebo, Xanomeline Low Dose, and Xanomeline High Dose
PARAMCD	Character	Parameter Code	CHOL
USUBJID	Character	Unique Subject Identifier	01-701-1015, 01-701-1023, ...
AVISITN	Numeric	Analysis Visit (N)	2, 4, 6, 8, 12, 16, 20, 24, and 26
CHG	Numeric	Change from Baseline	-1.9395, -0.46548, 1.49988, ...

Table 2. Column Properties and Data Values for the ANALYSIS SAS Table

OUTPUT DELIVERY SYSTEM (ODS) BASICS

ODS is the part of Base SAS® software that enables you to generate different types of output from your procedure code. An ODS destination controls the type of output that is generated (HTML, RTF, PDF, and so on). An ODS style controls the appearance of the output.

The report shown in Figure 1 was created using the RTF ODS destination and the JOURNAL ODS style supplied by SAS. All formatting and layout are performed by SAS; there is no need to hand-edit the RTF file. You simply use an application such as Microsoft Word to open the file created by ODS.

Here are the general ODS statements to generate the Rich Text Format (RTF) file.

```
❶ ods _all_ close;

❷ ods rtf file='file-name.rtf' style=style-name;
   * Your SAS procedure code here;
❸ ods rtf close;
```

The first ODS statement (❶) closes all destinations that are open because we want to generate only RTF output.

The second ODS statement (❷) uses the RTF destination to generate the output and then store it in a file (SAS Institute Inc. 2017d). The STYLE option controls the appearance of the output, such as the font and color scheme. To see a list of ODS styles that are available for use at your site, submit the following SAS code.

```
   ods _all_ close;
   ods listing;
   proc template; list styles; run; quit;
```

To find the SAS code that generates sample output for the ODS styles available on your system, click the **Full Code** tab in SAS Sample 36900 (SAS Institute Inc. 2009).

The third ODS statement (❸) closes the RTF destination and releases the file so that it can be opened with Microsoft Word.

Note: If you place the files where users can access them over a network, you should set file permissions to prevent accidental alteration.

OVERVIEW OF SAVING DATA FROM A PROCEDURE

Here are the general steps to capture any procedure output data, and then manipulate it meet your reporting needs:

1. Use the ODS TRACE statement to list the output objects available from the procedures.
2. Run the procedures to create the statistics or other data.
3. Examine the log to see the available output objects.
4. Use the ODS OUTPUT statement to create SAS tables from the relevant ODS output objects.
5. Restructure and combine the SAS tables to create a single table with all the information needed to create the report.
6. Create a new ODS style that provides the appearance that you want (optional).
7. Create the report using PROC PRINT or PROC REPORT with ODS style overrides, inline formatting functions, or the newly created style.

This paper guides you through all steps except Step 6, creating a new ODS style. That topic is discussed in DelGobbo 2015.

SETTING UP THE PROGRAM ENVIRONMENT

The code below specifies system options, defines the missing value and ODS escape characters, and then creates global macro variables that are used later.

```
options nodate nonumber missing='-';

*  Close all ODS destinations, and then open when needed;

ods _all_ close;

*  ODS escape character used for inline formatting;

ods escapechar = '^';

*;
*  Create a global macro variable with the value of the study ID for
*  use in a title.
*;

data _null_;
set sample.adsl(obs=1);
call symputx('STUDYID', studyid);
run;

*  Format used for N;

%let STATFMT0D=12.;

*  Format used for MIN and MAX requires two decimal places;

%let STATFMT2D=%sysevalf(&STATFMT0D + 0.2);
```

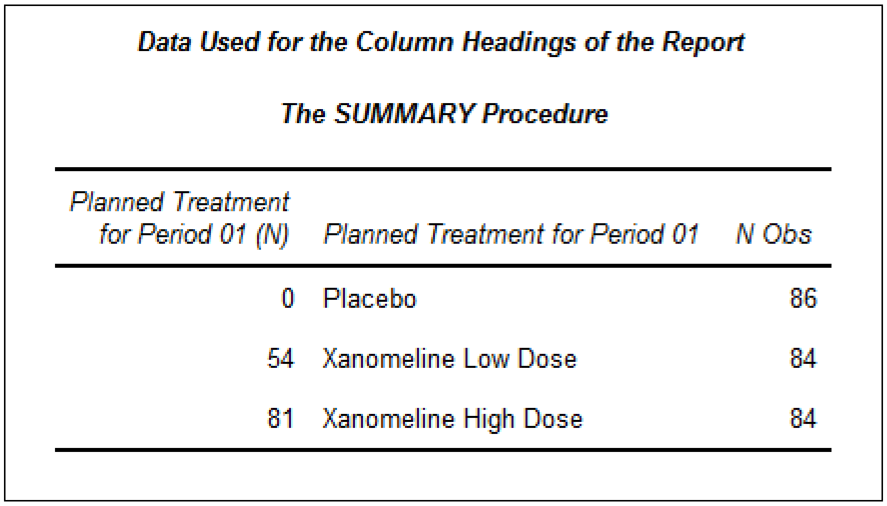
```
  *  Format used for other statistics requires three decimal places;

  %let STATFMT3D=%sysevalf(&STATFMT2D + 0.1);
```

SAS CODE TO CREATE THE STATISTICAL OUTPUT

This code creates output used for the column headings and body of the report:

```
  *  Turn ODS graphics off;

  ods graphics off;

  *  Open the HTML destination to create the procedure output;

  ods html path='directory-location' file='Statistics.htm' style=Journal;

  *  Create data for the column headings of the report;

  title 'Data Used for the Column Headings of the Report';

❶ proc summary print data=sample.adsl;
    class trt01pn trt01p;
  run; quit;

  *  Create the statistics for Section 1 of the report;

  title 'Statistics for Section 1 of the Report';

❷ proc summary print data=sample.analysis missing stackods
               n mean stderr median q1 q3 min max;
    class avisitn trt01pn;
    var chg;
  run; quit;

  *;
  *  Create the statistics for Section 2 and Section 3 of the report.
  *  Differences are computed in reference to the placebo
  *  (Active Drug - Placebo).
  *;

  title 'Statistics for Sections 2 and 3 of the Report';

❸ proc mixed data=sample.analysis;
    class usubjid avisitn trt01pn;
    model chg = avisitn trt01pn avisitn*trt01pn / ddfm=kenwardroger;
    repeated avisitn / subject=usubjid type=un;

    lsmeans avisitn*trt01pn / cl;                   ❹
    slice   avisitn*trt01pn / cl diff=control('2' '0') sliceby=avisitn;
  run; quit;

  *  Close the HTML destination;

  ods html close;
```

(❶) The first instance of PROC SUMMARY computes the number of subjects for each of the three treatment groups (see Figure 2). These values are used in the **N =** text in the column headings of Figure 1.



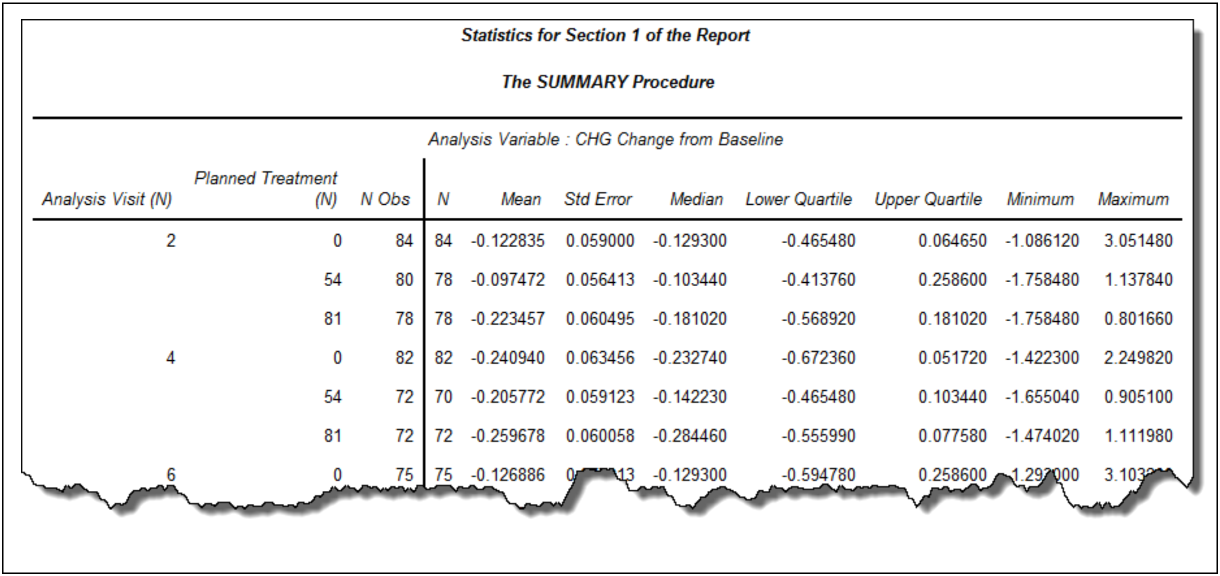
Data Used for the Column Headings of the Report

The SUMMARY Procedure

Planned Treatment for Period 01 (N)	*Planned Treatment for Period 01*	*N Obs*
0	Placebo	86
54	Xanomeline Low Dose	84
81	Xanomeline High Dose	84

Figure 2. Output from PROC SUMMARY Run with the ADSL Table

(❷) Running PROC SUMMARY on the ANALSIS table computes the change from baseline statistics displayed in the **Summary Statistics** section of Figure 1. The statistics are computed for each distinct combination of visit and treatment (see Figure 3).



Statistics for Section 1 of the Report

The SUMMARY Procedure

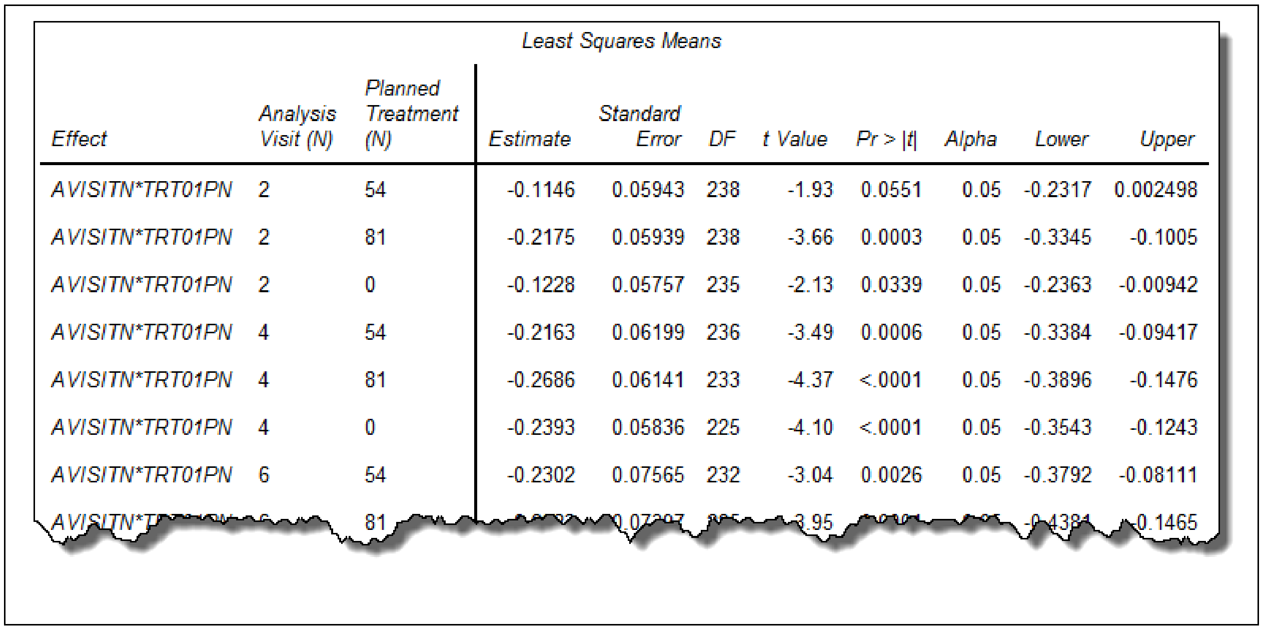
Analysis Variable : CHG Change from Baseline

Analysis Visit (N)	*Planned Treatment (N)*	*N Obs*	*N*	*Mean*	*Std Error*	*Median*	*Lower Quartile*	*Upper Quartile*	*Minimum*	*Maximum*
2	0	84	84	-0.122835	0.059000	-0.129300	-0.465480	0.064650	-1.086120	3.051480
	54	80	78	-0.097472	0.056413	-0.103440	-0.413760	0.258600	-1.758480	1.137840
	81	78	78	-0.223457	0.060495	-0.181020	-0.568920	0.181020	-1.758480	0.801660
4	0	82	82	-0.240940	0.063456	-0.232740	-0.672360	0.051720	-1.422300	2.249820
	54	72	70	-0.205772	0.059123	-0.142230	-0.465480	0.103440	-1.655040	0.905100
	81	72	72	-0.259678	0.060058	-0.284460	-0.555990	0.077580	-1.474020	1.111980
6	0	75	75	-0.126886	[illegible]	-0.129300	-0.594780	0.258600	[illegible]	3.103

Figure 3. Output from PROC SUMMARY Run with the ANALYSIS Table

(❸) Running the MIXED procedure on the ANALYSIS table computes the statistics used in the **LS Mean** and **Treatment Difference (Active - Placebo)** sections of Figure 1. The REPEATED statement specifies a repeated measures analysis, with AVISITN as the repeated measures effect and USUBJID as the subject identifier.

The LS means and associated statistics for the **LS Mean** section of the report are computed for each visit-treatment combination (see Figure 4).



Least Squares Means

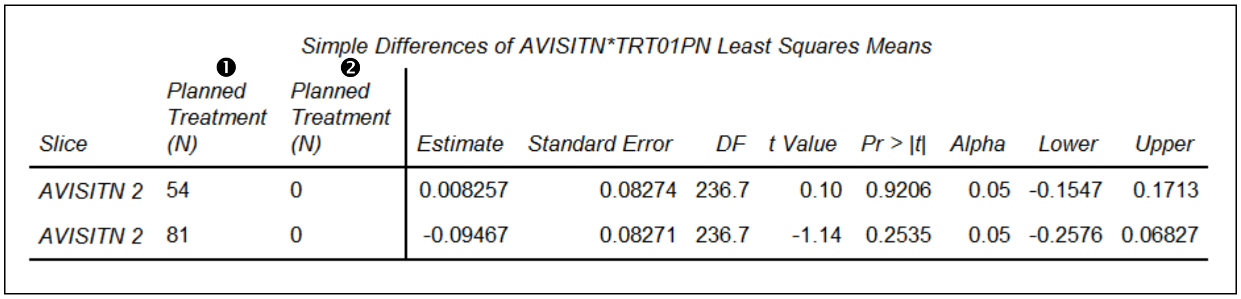
Effect	Analysis Visit (N)	Planned Treatment (N)	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
AVISITN*TRT01PN	2	54	-0.1146	0.05943	238	-1.93	0.0551	0.05	-0.2317	0.002498
AVISITN*TRT01PN	2	81	-0.2175	0.05939	238	-3.66	0.0003	0.05	-0.3345	-0.1005
AVISITN*TRT01PN	2	0	-0.1228	0.05757	235	-2.13	0.0339	0.05	-0.2363	-0.00942
AVISITN*TRT01PN	4	54	-0.2163	0.06199	236	-3.49	0.0006	0.05	-0.3384	-0.09417
AVISITN*TRT01PN	4	81	-0.2686	0.06141	233	-4.37	<.0001	0.05	-0.3896	-0.1476
AVISITN*TRT01PN	4	0	-0.2393	0.05836	225	-4.10	<.0001	0.05	-0.3543	-0.1243
AVISITN*TRT01PN	6	54	-0.2302	0.07565	232	-3.04	0.0026	0.05	-0.3792	-0.08111
AVISITN*T	[illegible]	81	[illegible]	[illegible]	[illegible]	-3.95	[illegible]	[illegible]	-0.4384	-0.1465

Figure 4. LS Mean Output from PROC MIXED

The SLICE statement produces LS mean differences and associated statistics (see Figure 5) used in the **Treatment Difference (Active - Placebo)** section of the report.

(❹) A value for the AVISITN and TRT01PN variables are required in the CONTROL specification, even though the SLICEBY option causes only the value of TRT01PN to be considered. Any value of AVISITN present in the data can be specified. We specify **'0'** for TRT01PN to indicate that Placebo is taken to be the control for each value of AVISITN.

Figure 5 shows estimates for the within-visit TRT01PN differences for visit 2. Each row represents the estimated difference of the mean value of CHG for the active dose (❶) with that of Placebo (❷). The first row shows statistics for the difference between the low dose and the placebo and the second row shows the statistics for the high dose versus the placebo. Output such as this is produced for the remaining eight values of AVISITN.



Simple Differences of AVISITN*TRT01PN Least Squares Means

Slice	❶ Planned Treatment (N)	❷ Planned Treatment (N)	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
AVISITN 2	54	0	0.008257	0.08274	236.7	0.10	0.9206	0.05	-0.1547	0.1713
AVISITN 2	81	0	-0.09467	0.08271	236.7	-1.14	0.2535	0.05	-0.2576	0.06827

Figure 5. Active - Placebo Differences of LS Means for Visit 2 from PROC MIXED

LISTING THE ODS OUTPUT OBJECTS CREATED BY SAS PROCEDURES

ODS output objects contain output data resulting from the execution of SAS procedures. You can save the output objects as SAS tables, and then use the tables to create customized reports. Use the ODS TRACE statement (SAS Institute Inc. 2017e) to list the output objects created by the SUMMARY and MIXED procedures.

```
ods trace on;

title 'Data Used for the Column Headings of the Report';

proc summary print data=sample.adsl ...; run; quit;

title 'Statistics for Section 1 of the Report';

proc summary print data=sample.analysis ...; run; quit;

title 'Statistics for Sections 2 and 3 of the Report';

proc mixed data=sample.analysis ...; run; quit;

ods trace off;
```

The information shown in Figure 6 is displayed twice in the log, once for each instance of the PROC SUMMARY code.

The SUMMARY objects contain the data for the output of Figure 2 and Figure 3.

```
Output Added:
-------------
Name:         Summary
Label:        Summary statistics
Template:     base.summary
Path:         Summary.Summary
-------------
```

Figure 6. ODS Output Object Available from the SUMMARY Procedure

Figure 7 lists some of the ODS output objects available from PROC MIXED. The LSMEANS object contains the data for the output of Figure 4, and the data for Figure 5 is stored in the SLICEDIFFS object.

```
Output Added:
-------------
Name:         Dimensions
Label:        Dimensions
Template:     Stat.Mixed.Dimensions
Path:         Mixed.Dimensions
-------------

Output Added:
-------------
Name:         LSMeans
Label:        Least Squares Means
Template:     Stat.Mixed.LSMeans
Path:         Mixed.LSMeans
-------------

Output Added:
-------------
Name:         SliceDiffs
Label:        AVISITN*TRT01PN Diffs
Template:     Stat.LMR.Diffs
Path:         Mixed.Slices.SliceDiffs
-------------
```

Figure 7. Partial List of ODS Output Object Available from the MIXED Procedure

CREATING SAS TABLES FROM ODS OUTPUT OBJECTS

The ODS OUTPUT statement creates SAS tables from ODS output objects (SAS Institute Inc. 2017c). Here is the general syntax:

```
ods output output-object-name1 = SAS-table-name1
           output-object-name2 = SAS-table-name2 ...;

* Your SAS procedure code here;
```

Because we need to manipulate data from only some of the output objects (see Figure 6 and Figure 7), we add the following ODS OUTPUT statements to our code:

```
ods output Summary=work.colhead;

title 'Data Used for the Column Headings of the Report';

proc summary print data=sample.adsl ...; run; quit;

ods output Summary=work.summary;

title 'Statistics for Section 1 of the Report';

proc summary print data=sample.analysis ...; run; quit;

ods output LSMeans=work.lsmeans SliceDiffs=work.diffs;

title 'Statistics for Sections 2 and 3 of the Report';

proc mixed data=sample.analysis ...; run; quit;
```

The SLICEDIFFS output object does not contain data for the AVISITN variable or a record for the placebo. This code creates the AVISITN variable (❶) and adds the placebo record (❷) for each visit.

```
  *;
  *  Add a record to the DIFFS table to represent the placebo. Statistics for
  *  these records are missing.
  *;

  proc sort data=work.diffs out=work.diffs_sorted; by slice; run; quit;

  data work.diffs2(drop=stmtno effect slice _trt01pn);
  set work.diffs_sorted;
  by slice;
  length avisitn 8.;
❶ avisitn = input(scan(slice, 2), best.);
  output;
❷ if (last.slice) then do;
    trt01pn = _trt01pn;
    call missing(of estimate stderr df tvalue probt alpha lower upper);
    output;
  end;
  run;
```

_TRT01PN is our control variable and always has a value of zero (Placebo). All statistics for the placebo record have missing values and are displayed as a dash (-) in our report.

USING THE FORMAT PROCEDURE WITH AN INPUT CONTROL DATA SET

We need to create column headings based on the formatted value of the TRT01PN variable. We could create a SAS format by hard-coding FORMAT procedure statements using data from Figure 2:

```
proc format;
  value colhead  0 = 'Placebo~(N=86)'
                54 = 'Xanomeline~Low Dose~(N=84)'
                81 = 'Xanomeline~High Dose~(N=84)';
run; quit;
```

But dynamically creating the format from an input control data set (SAS Institute Inc. 2018c) provides a more flexible, data-driven approach. The properties of the input control data set are shown in Table 3.

Column Name	Type	Description	Values
FMTNAME	Character	Format name	COLHEAD
TYPE	Character	Type of format	N
START	Numeric	Beginning value of the range	0, 54, 81
END	Numeric	Ending value of the range	0, 54, 81
LABEL	Character	Formatted value	Placebo~(N=86), Xanomeline~Low Dose~(N=84), Xanomeline~High Dose~(N=84)

Table 3. Properties of the Control Input Data Set for the COLHEAD Format

The COLHEAD table created earlier by the SUMMARY procedure (see Figure 2) has the data needed to create the format. Here is the code:

```
  data work.colhead_cntlin(drop=trt01pn trt01p nobs first_space);
  set work.colhead;
  length fmtname $7 type $1 start end 8 label $30 first_space 8;

  *  Replace the first blank space in the label with the split character;

❶ first_space = index(strip(trt01p), ' ');

❷ if (first_space ne 0)
    then substr(trt01p, first_space, 1) = '~';

  *  Specify the required information for the format;

❸ fmtname = 'COLHEAD';
  type    = 'N';
  start   = trt01pn;
  end     = trt01pn;
  label   = strip(trt01p) || '~(N = ' || strip(put(nobs, &STATFMT0D)) || ')';
  run;

  proc format cntlin=work.colhead_cntlin; run; quit;
```

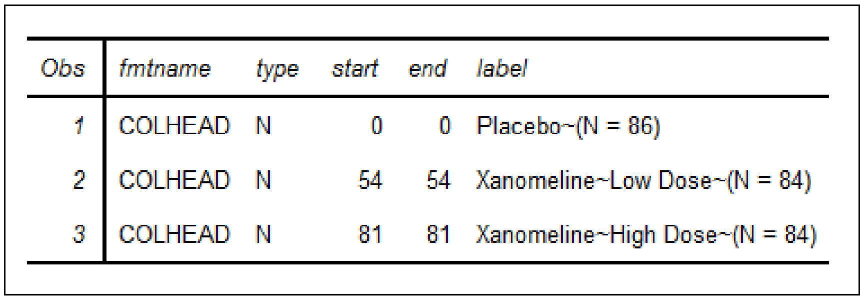
(❶) The INDEX function searches TRT01P for the first blank space, and then returns either the position of the blank space or zero if no blank space is found.

(❷) The SUBSTR function, when used to the left of the equals sign, modifies the value of TRT01P (SAS Institute Inc. 2018e). The first blank space is replaced with the split character. Table 4 shows the values of TRT01P.

Initial Value	Modified Value
Placebo	Placebo
Xanomeline Low Dose	Xanomeline~Low Dose
Xanomeline High Dose	Xanomeline~High Dose

Table 4. Initial and Modified Values of TRT01P

(❸) Specify values for the required variables of the COLHEAD_CNTLIN input control data set. We add the appropriate N= value to the end of TRT01P to obtain the value of the format label (see Figure 8).



```
Obs | fmtname  type  start  end  label
  1 | COLHEAD  N         0    0  Placebo~(N = 86)
  2 | COLHEAD  N        54   54  Xanomeline~Low Dose~(N = 84)
  3 | COLHEAD  N        81   81  Xanomeline~High Dose~(N = 84)
```

Figure 8. The COLHEAD_CNTLIN Input Control Data Set

RESTRUCTURE AND COMBINE THE PROCEDURE OUTPUT

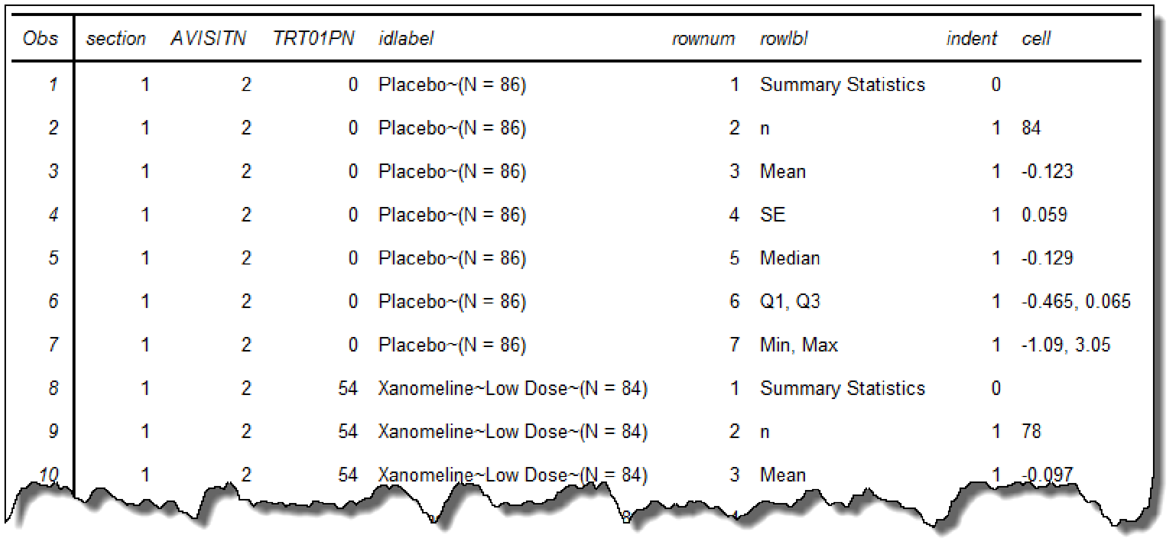
The three output data sets from the SUMMARY and MIXED procedures need to be altered and combined before they can be used to produce the report. These are the general steps:

1. Create a row for the section heading.
2. Create a row for each statistic from a column value.
3. Combine the three data sets into one.
4. Use the TRANSPOSE procedure to create the final output table.

In the following sections we apply this process to each of the three SAS tables, discussing the specifics of each case.

PREPARING THE DATA FOR THE SUMMARY STATISTICS SECTION

This section of the report corresponds to the **Summary Statistics** section in Figure 1. The SUMMARY procedure output stored in the SUMMARY SAS table is used to create this section. Figure 9 shows the data that we need to create.



Obs	section	AVISITN	TRT01PN	idlabel	rownum	rowlbl	indent	cell
1	1	2	0	Placebo~(N = 86)	1	Summary Statistics	0	
2	1	2	0	Placebo~(N = 86)	2	n	1	84
3	1	2	0	Placebo~(N = 86)	3	Mean	1	-0.123
4	1	2	0	Placebo~(N = 86)	4	SE	1	0.059
5	1	2	0	Placebo~(N = 86)	5	Median	1	-0.129
6	1	2	0	Placebo~(N = 86)	6	Q1, Q3	1	-0.465, 0.065
7	1	2	0	Placebo~(N = 86)	7	Min, Max	1	-1.09, 3.05
8	1	2	54	Xanomeline~Low Dose~(N = 84)	1	Summary Statistics	0	
9	1	2	54	Xanomeline~Low Dose~(N = 84)	2	n	1	78
10	1	2	54	Xanomeline~Low Dose~(N = 84)	3	Mean	1	-0.097

Figure 9. Partial View of the Data Used for the Summary Statistics Section

Table 5 lists the report items from Figure 1, the variables in Figure 9 that correspond to the report items, and the variables from the SUMMARY table used to derive these values.

Report Item	Variables in Figure 9	Variables in the SUMMARY Table
Column Headings	IDLABEL	TRT01PN
Week Number	AVISITN	AVISITN
N	CELL	N
Mean	CELL	MEAN
SE	CELL	STDERR
Median	CELL	MEDIAN
Q1, Q3	CELL	Q1, Q3
Min, Max	CELL	MIN, MAX

Table 5. Variables from the SUMMARY SAS Table Used in the Report

This code creates the REPORT_SUMMARY SAS table shown in Figure 9:

```
  data work.report_summary(keep=section rownum indent rowlbl cell
                                avisitn trt01pn idlabel);
  set work.summary;
  by avisitn trt01pn;

  length idlabel $30 rownum indent 8 rowlbl $40 cell $20;

❶ section = 1;

❷ if (first.trt01pn) then rownum = 0;
```

```
  *  Column heading;

❸ idlabel = strip(put(trt01pn, colhead.));

  * Row Heading;

  rownum +1;
  indent = 0;
  rowlbl = 'Summary Statistics';
  cell   = '';
  output;

❹ * Row Values;

  indent = 1;

  rownum +1;
  rowlbl = 'n';
  cell   = strip(put(n, &STATFMT0D));
  output;

  rownum +1;
  rowlbl = 'Mean';
  cell   = strip(put(mean, &STATFMT3D));
  output;

  rownum +1;
  rowlbl = 'SE';
  cell   = strip(put(stderr, &STATFMT3D));
  output;

  rownum +1;
  rowlbl = 'Median';
  cell   = strip(put(median, &STATFMT3D));
  output;

  rownum +1;
  rowlbl = 'Q1, Q3';
  cell   = strip(put(q1, &STATFMT3D)) || ', ' || strip(put(q3, &STATFMT3D));
  output;

  rownum +1;
  rowlbl = 'Min, Max';
  cell   = strip(put(min, &STATFMT2D)) || ', ' ||
           strip(put(max, &STATFMT2D));
  output;

  run;
```

(❶) The SECTION variable indicates that this data is used for the **Summary Statistics** section.

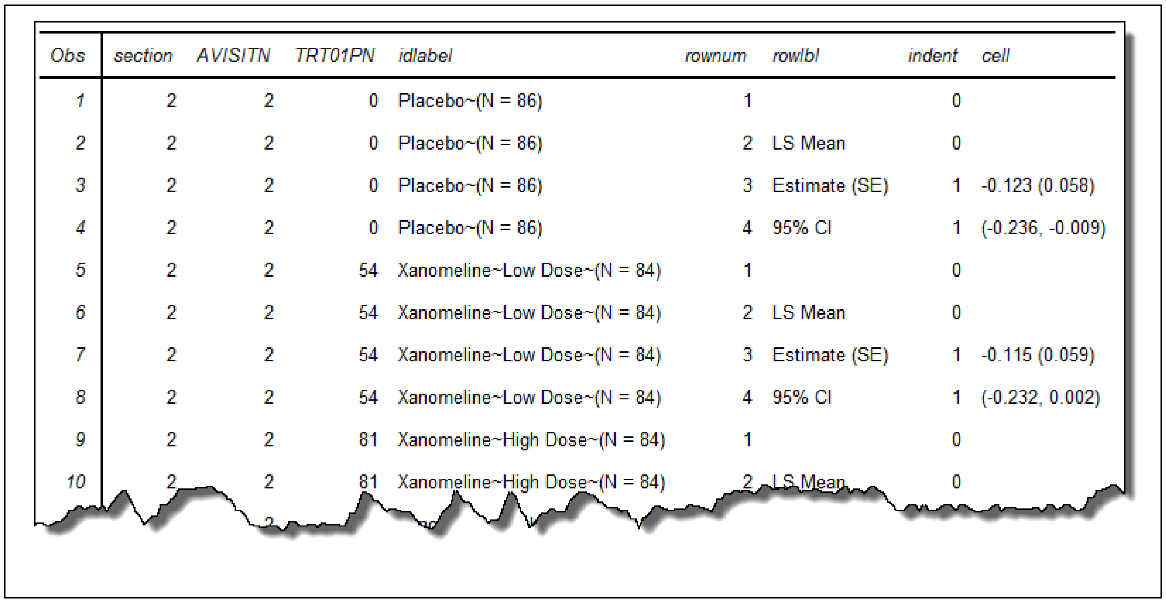
(❷) A row number is created for each report item based on the distinct visit-treatment combination.

(❸) The IDLABEL variable, used later to format the column headings, is created using the COLHEAD format.

(❹) The variables used in the data rows are created. Setting INDENT to 1 indicates that these row values are indented underneath the section heading. The ROWLBL and CELL variables are then created and output. The ROWLBL variable contains the row labels for the report, and all data values are stored in the CELL character variable. Using the STATFMT0D, STATFMT2D, and STATFMT3D macro variables ensures that the numeric values are displayed with the appropriate number of decimal places, resulting in the SAS table shown in Figure 9.

PREPARING THE DATA FOR THE LS MEAN SECTION

This section of the report corresponds to the information presented under the **LS Mean** heading of Figure 1. The MIXED procedure output stored in the LSMEANS SAS table is used to create this section. Figure 10 shows the data that we need to create.



Obs	section	AVISITN	TRT01PN	idlabel	rownum	rowlbl	indent	cell
1	2	2	0	Placebo~(N = 86)	1		0	
2	2	2	0	Placebo~(N = 86)	2	LS Mean	0	
3	2	2	0	Placebo~(N = 86)	3	Estimate (SE)	1	-0.123 (0.058)
4	2	2	0	Placebo~(N = 86)	4	95% CI	1	(-0.236, -0.009)
5	2	2	54	Xanomeline~Low Dose~(N = 84)	1		0	
6	2	2	54	Xanomeline~Low Dose~(N = 84)	2	LS Mean	0	
7	2	2	54	Xanomeline~Low Dose~(N = 84)	3	Estimate (SE)	1	-0.115 (0.059)
8	2	2	54	Xanomeline~Low Dose~(N = 84)	4	95% CI	1	(-0.232, 0.002)
9	2	2	81	Xanomeline~High Dose~(N = 84)	1		0	
10	2	2	81	Xanomeline~High Dose~(N = 84)	2	LS Mean	0	

Figure 10. Partial View of the Data Used for the LS Mean Section

Table 6 lists the report items from Figure 1, the variables in Figure 10 that correspond to the report items, and variables from the LSMEANS table used to derive these values.

Report Item	Variables in Figure 10	Variables in the LSMEANS SAS Table
Column Headings	IDLABEL	TRT01PN
Week Number	AVISITN	AVISITN
Estimate (SE)	CELL	ESTIMATE, STDERR
95% CI	CELL	LOWER, UPPER

Table 6. Variables from the LSMEANS SAS Table Used in the Report

This code creates the REPORT_LSMEANS SAS table shown in Figure 10:

```
proc sort data=work.lsmeans out=work.lsmeans_sorted;
  by avisitn trt01pn;
run; quit;
```

```
  data work.report_lsmeans(keep=section rownum indent rowlbl cell
                                avisitn trt01pn idlabel);
  set work.lsmeans_sorted;
  by avisitn trt01pn;

  length idlabel $30 rownum indent 8 rowlbl $40 cell $20;

❶ section = 2;

❷ if (first.trt01pn) then rownum = 0;

  *  Column heading;

❸ idlabel = strip(put(trt01pn, colhead.));

❹ * Blank row;

  rownum +1;
  indent = 0;
  rowlbl = '';
  cell   = '';
  output;

  * Row Heading;

  indent = 0;
  rownum +1;
  rowlbl = 'LS Mean';
  cell   = '';
  output;

❺ * Row Values;

  indent = 1;

  rownum +1;
  rowlbl = 'Estimate (SE)';
  cell   = strip(put(estimate, &STATFMT3D)) ||
           ' (' || strip(put(stderr, &STATFMT3D)) || ')';
  output;

  rownum +1;
  rowlbl = '95% CI';
  cell   = '('  || strip(put(lower, &STATFMT3D)) ||
           ', ' || strip(put(upper, &STATFMT3D)) || ')';
  output;

  run;
```

(❶) The SECTION variable indicates that this data is used for the **LS Mean** section.

(❷) A row number is created for each report item based on the distinct visit-treatment combination.

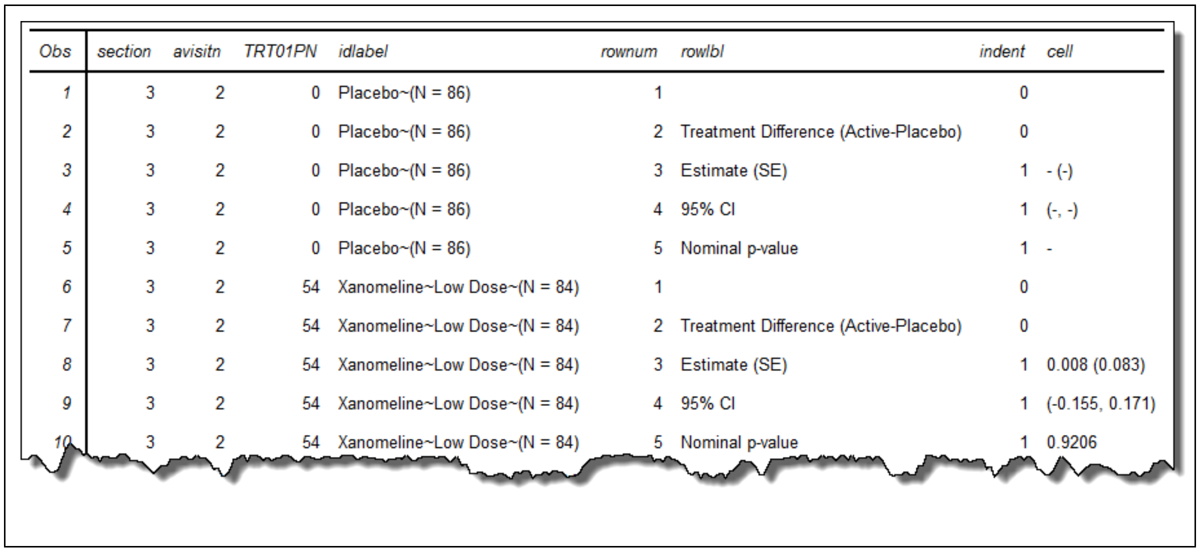
(❸) The IDLABEL variable, used later to format the column headings, is created using the COLHEAD format.

(❹) A blank row creates space between this section and the **Summary Statistics** section.

(❺) The variables used in the data rows are created. Setting INDENT to 1 indicates that these row values are indented underneath the section heading. The ROWLBL and CELL variables are then created and output. The ROWLBL variable contains the row labels for the report, and all data values are stored in the CELL character variable. Using the STATFMT3D macro variable ensures that the numeric values are displayed with the appropriate number of decimal places.

PREPARING THE DATA FOR THE DIFFERENCES IN LS MEANS SECTION

This section of the report corresponds to the information presented under the **Treatment Difference (Active-Placebo)** section of Figure 1. The MIXED procedure output stored in the DIFFS2 SAS table is used to create this section. Figure 11 shows the data that we need to create.



```
Obs  section  avisitn  TRT01PN  idlabel                        rownum  rowlbl                                indent  cell
  1        3        2        0  Placebo~(N = 86)                    1                                             0
  2        3        2        0  Placebo~(N = 86)                    2  Treatment Difference (Active-Placebo)      0
  3        3        2        0  Placebo~(N = 86)                    3  Estimate (SE)                              1  - (-)
  4        3        2        0  Placebo~(N = 86)                    4  95% CI                                     1  (-, -)
  5        3        2        0  Placebo~(N = 86)                    5  Nominal p-value                            1  -
  6        3        2       54  Xanomeline~Low Dose~(N = 84)        1                                             0
  7        3        2       54  Xanomeline~Low Dose~(N = 84)        2  Treatment Difference (Active-Placebo)      0
  8        3        2       54  Xanomeline~Low Dose~(N = 84)        3  Estimate (SE)                              1  0.008 (0.083)
  9        3        2       54  Xanomeline~Low Dose~(N = 84)        4  95% CI                                     1  (-0.155, 0.171)
 10        3        2       54  Xanomeline~Low Dose~(N = 84)        5  Nominal p-value                            1  0.9206
```

Figure 11. Partial View of the Data Used for the Differences in LS Means Section

Table 7 lists the report items from Figure 1, the variables in Figure 11 that correspond to the report items, and variables from the DIFFS2 table used to derive these values.

Report Item	Variables in Figure 11	Variables in the DIFFS2 SAS Table
Column Headings	IDLABEL	TRT01PN
Week Number	AVISITN	AVISITN
Estimate (SE)	CELL	ESTIMATE, STDERR
95% CI	CELL	LOWER, UPPER
Nominal p-value	CELL	PROBT

Table 7. Variables from the DIFFS2 SAS Table Used in the Report

This code creates the REPORT_DIFFS SAS table shown in Figure 11:

```
proc sort data=work.diffs2 out=work.diffs2_sorted;
  by avisitn trt01pn;
run; quit;

data work.report_diffs(keep=section rownum indent rowlbl cell
                            avisitn trt01pn idlabel);
set work.diffs2_sorted;
```

```
  by avisitn trt01pn;

  length idlabel $30 rownum indent 8 rowlbl $40 cell $20;

❶ section = 3;

❷ if (first.trt01pn) then rownum = 0;

  *  Column heading;

❸ idlabel = strip(put(trt01pn, colhead.));

❹ * Blank row;

  rownum +1;
  indent = 0;
  rowlbl = '';
  cell   = '';
  output;

  * Row Heading;

  indent = 0;
  rownum +1;
  rowlbl = 'Treatment Difference (Active-Placebo)';
  cell   = '';
  output;

❺ * Row Values;

  indent = 1;

  rownum +1;
  rowlbl = 'Estimate (SE)';
  cell   = strip(put(estimate, &STATFMT3D)) ||
           ' (' || strip(put(stderr, &STATFMT3D)) || ')';
  output;

  rownum +1;
  rowlbl = '95% CI';
  cell   = '('  || strip(put(lower, &STATFMT3D)) ||
           ', ' || strip(put(upper, &STATFMT3D)) || ')';
  output;

  rownum +1;
  rowlbl = 'Nominal p-value';
  cell   = strip(put(probt, pvalue6.4));
  output;

  run;
```

(❶) The SECTION variable indicates that this data is used for the **Treatment Difference (Active-Placebo)** section.

(❷) A row number is created for each report item based on the distinct visit-treatment combination.

(❸) The IDLABEL variable, used later to format the column headings, is created using the COLHEAD format.

(❹) A blank row creates space between this section and the **LS Mean** section.

(❺) The variables used in the data rows are created. Setting INDENT to 1 indicates that these row values are indented underneath the section heading. The ROWLBL and CELL variables are then created and output. The ROWLBL variable contains the row labels for the report, and all data values are stored in the CELL character variable. Using the STATFMT3D macro variable and the PVALUE format (SAS Institute Inc. 2017f) ensures that the numeric values are appropriately displayed.

COMBINING THE DATA FROM THE THREE SECTIONS

Create the final table by combining the REPORT_SUMMARY, REPORT_LSMEANS, and REPORT_DIFFS tables, sorting the table, and then transposing it to get the final structure needed for the report.

```
*  Combine the preliminary data into a single table;

data work.report1;
set work.report_summary
    work.report_lsmeans
    work.report_diffs;
run;

*  Sort in preparation for transposition;

proc sort data=work.report1 out=work.report1_sorted;
  by avisitn section rownum rowlbl indent trt01pn;
run; quit;

*  Transpose the data to get the layout needed for the report;
                                                   ❸
proc transpose data=work.report1_sorted prefix=trt_
               out=work.report_final(drop=_name_);
  by avisitn section rownum rowlbl indent;
❶ var cell;
❷ id trt01pn;
❹ idlabel idlabel;
run; quit;
```

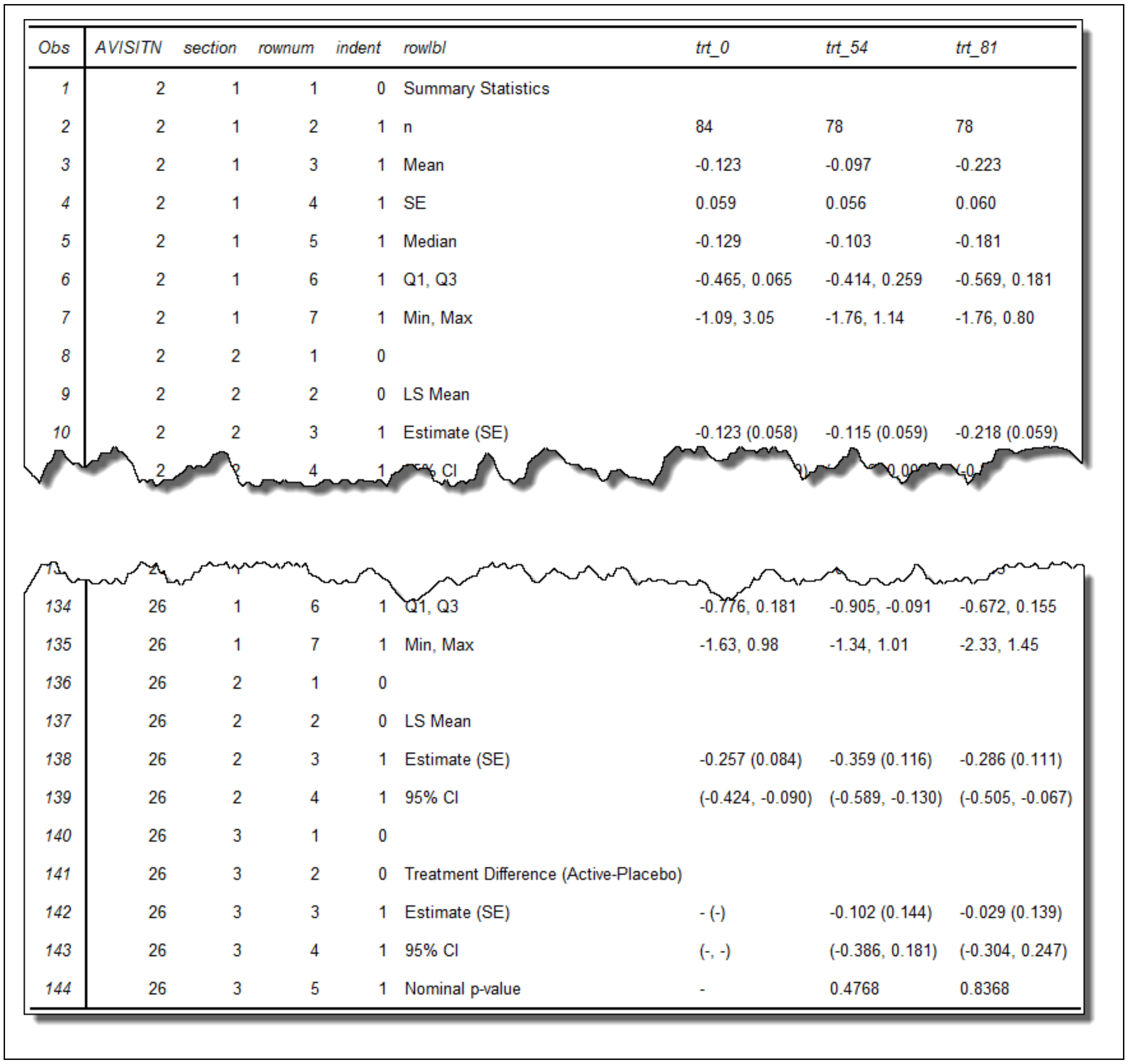
(❶) The BY statement specifies grouping of the data in the REPORT_FINAL SAS table. These variables appear in the output data set, but are not transposed. The VAR statement specifies the variable to be transposed.

(❷) Because TRT01PN has three distinct values (0, 54, and 81), the output table contains three new columns corresponding to the placebo, the low dose active drug, and the high dose active drug.

(❷) and (❸) The values of the TRT01PN variable are used to name the three new columns in the output table. Because 0, 54, and 81 are not valid SAS names, the PREFIX option is used to specify valid column names: TRT_0, TRT_54, and TRT_81.

(❹) The values of the IDLABEL variable are used to apply the labels "Placebo~(N=86)", Xanomeline~Low Dose~(N=84)", and "Xanomeline~High Dose~(N=84)" to the TRT_0, TRT_54, and TRT_81 columns, respectively.

A partial view of the REPORT_FINAL SAS table without labels is shown in Figure 12.



Obs	AVISITN	section	rownum	indent	rowlbl	trt_0	trt_54	trt_81
1	2	1	1	0	Summary Statistics			
2	2	1	2	1	n	84	78	78
3	2	1	3	1	Mean	-0.123	-0.097	-0.223
4	2	1	4	1	SE	0.059	0.056	0.060
5	2	1	5	1	Median	-0.129	-0.103	-0.181
6	2	1	6	1	Q1, Q3	-0.465, 0.065	-0.414, 0.259	-0.569, 0.181
7	2	1	7	1	Min, Max	-1.09, 3.05	-1.76, 1.14	-1.76, 0.80
8	2	2	1	0				
9	2	2	2	0	LS Mean			
10	2	2	3	1	Estimate (SE)	-0.123 (0.058)	-0.115 (0.059)	-0.218 (0.059)
[illegible]	2	2	4	1	95% CI	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
134	26	1	6	1	Q1, Q3	-0.776, 0.181	-0.905, -0.091	-0.672, 0.155
135	26	1	7	1	Min, Max	-1.63, 0.98	-1.34, 1.01	-2.33, 1.45
136	26	2	1	0				
137	26	2	2	0	LS Mean			
138	26	2	3	1	Estimate (SE)	-0.257 (0.084)	-0.359 (0.116)	-0.286 (0.111)
139	26	2	4	1	95% CI	(-0.424, -0.090)	(-0.589, -0.130)	(-0.505, -0.067)
140	26	3	1	0				
141	26	3	2	0	Treatment Difference (Active-Placebo)			
142	26	3	3	1	Estimate (SE)	- (-)	-0.102 (0.144)	-0.029 (0.139)
143	26	3	4	1	95% CI	(-, -)	(-0.386, 0.181)	(-0.304, 0.247)
144	26	3	5	1	Nominal p-value	-	0.4768	0.8368

Figure 12. The REPORT_FINAL SAS Table

CREATING THE REPORT

Use the REPORT procedure with the REPORT_FINAL SAS table to create the report:

```
*  Open the RTF destination to create the procedure output;
                                                                 ❶
ods rtf file='directory-location\CHOL_Report.rtf' style=Journal bodytitle;

title1 'Table x.y: Change from Baseline LDL-C by Scheduled Visit';
title2 "Study &STUDYID";

proc report data=work.report_final nowd missing split='~';
❷ columns avisitn section rownum indent rowlbl trt_: ;

  define avisitn / order order=internal noprint;
  define section / order order=internal noprint;
```

```
    define rownum  / order order=internal noprint;
    define indent  / noprint;
    define rowlbl  / display ' ';
  ❷ define trt_:   / display center;

    break after avisitn / page;

  ❸ compute before avisitn / style=[just=l];
      text = catx(' ', 'Week', avisitn);
      line text $;
    endcomp;

  ❹ compute rowlbl;
      *  Placeholder for superscript and indenting;
      if (rowlbl eq 'Summary Statistics')
        then;
      else if (rowlbl eq 'LS Mean')
        then;
      if (indent.sum ne 0)
        then;
    endcomp;

  ❺ compute after avisitn / style=[just=l];
      line ' ';
      line 'N = number of subjects randomized; Xanomeline Low Dose = 54 mg;
            Xanomeline High Dose = 81 mg.';
      line ' ';
      line 'Summary statistics are based on observed data.';
      line ' ';
      line 'LS mean is from a repeated measures model
            which includes effects: treatment group, scheduled';
      line 'visit, and the interaction of treatment with scheduled visit.';
    endcomp;

  run; quit;

  *  Close the RTF destination;

  ods rtf close;
```

(❶) The BODYTITLE option in the ODS statement specifies that title text is placed into the body of the RTF document instead of the header section (SAS Institute Inc. 2017d). The SPLIT option in the PROC REPORT statement ensures that the column labels display as shown in Figure 1.

(❷) The TRT_: name prefix in the COLUMNS and DEFINE statements is used to specify the TRT_0, TRT_54, and TRT_81 variables (SAS Institute Inc. 2017a). Using this type of syntax adds flexibility to our program when used with different data. The BREAK statement ensures that data for each visit begins on a new page.

(❸) The first COMPUTE block displays the visit number before the data. The CATX function removes leading and trailing blanks from its arguments, concatenates the arguments separated by a delimiter, and then returns the concatenated string (SAS Institute Inc. 2018b). This is the general syntax:

```
  catx(delimiter, argument1, argument2, argumentN)
```

We use a blank space as the delimiter to create values like "Week 2", "Week 16", and so on.

(❹) The second COMPUTE block is a placeholder for where we will add superscript and indented text to the row labels.

(❺) The third COMPUTE block prints information after the body of the report.

Figure 13 shows the result of running this code.

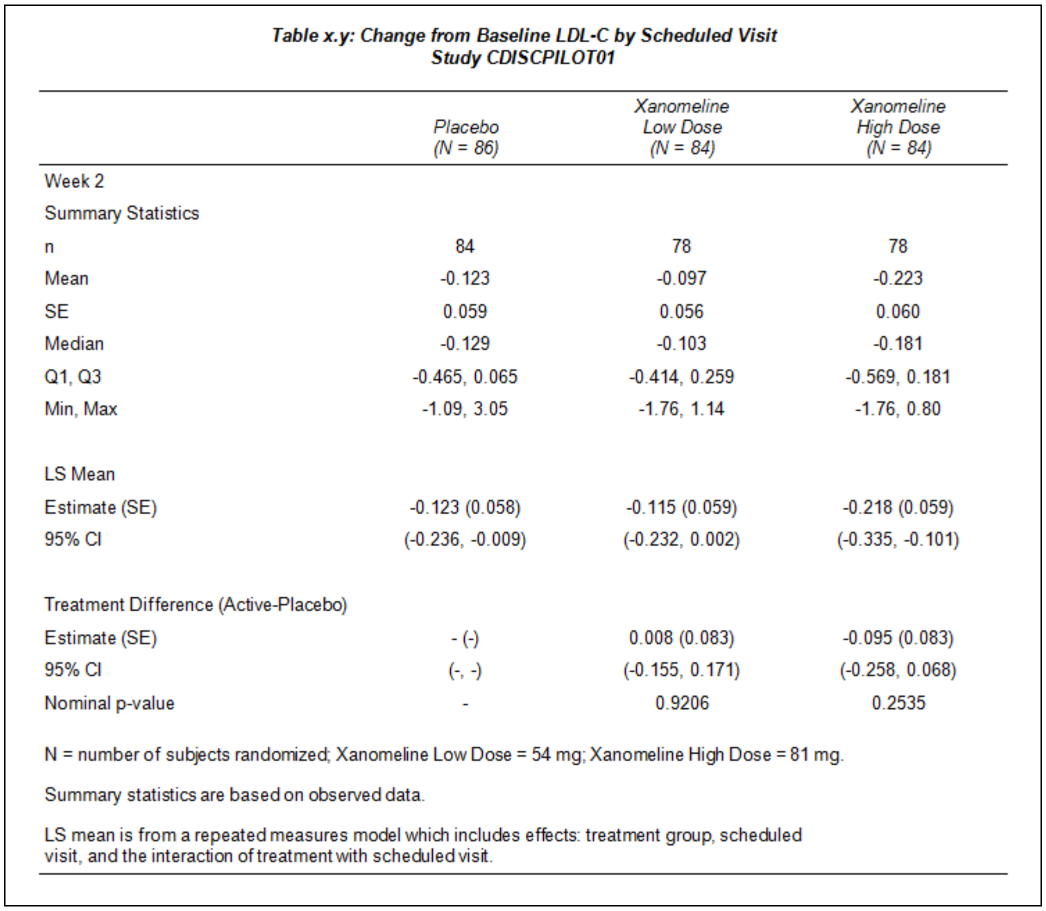


Table x.y: Change from Baseline LDL-C by Scheduled Visit
Study CDISCPILOT01

	Placebo (N = 86)	*Xanomeline Low Dose (N = 84)*	*Xanomeline High Dose (N = 84)*
Week 2			
Summary Statistics			
n	84	78	78
Mean	-0.123	-0.097	-0.223
SE	0.059	0.056	0.060
Median	-0.129	-0.103	-0.181
Q1, Q3	-0.465, 0.065	-0.414, 0.259	-0.569, 0.181
Min, Max	-1.09, 3.05	-1.76, 1.14	-1.76, 0.80
LS Mean			
Estimate (SE)	-0.123 (0.058)	-0.115 (0.059)	-0.218 (0.059)
95% CI	(-0.236, -0.009)	(-0.232, 0.002)	(-0.335, -0.101)
Treatment Difference (Active-Placebo)			
Estimate (SE)	- (-)	0.008 (0.083)	-0.095 (0.083)
95% CI	(-, -)	(-0.155, 0.171)	(-0.258, 0.068)
Nominal p-value	-	0.9206	0.2535

N = number of subjects randomized; Xanomeline Low Dose = 54 mg; Xanomeline High Dose = 81 mg.

Summary statistics are based on observed data.

LS mean is from a repeated measures model which includes effects: treatment group, scheduled visit, and the interaction of treatment with scheduled visit.

Figure 13. The Initial Report

ADDING SUPERSCRIPTS AND INDENTING TO THE REPORT

The report shown in Figure 13 is missing the superscript text, and the statistic row labels are not indented under their section headings. We can use ODS inline formatting functions in a COMPUTE block to add these features to the text without modifying the REPORT_FINAL SAS table (SAS Institute Inc. 2017b).

The ODS ESCAPECHAR statement defines a special character to indicate that inline formatting operations follow. We specified ^ for an escape character because it is not used in any of our text:

```
ods escapechar '^';
```

Inline formatting instructions are included in text strings, start with the escape character, and are enclosed within braces. The general syntax for using an inline function is:

```
escape-character{function-name function-arguments}
```

We use the SUPER function to add superscript text, and the NBSPACE function to indent text with non-breaking spaces:

```
regular-text^{super superscript-text}

^{super superscript-text}regular-text

^{nbspace number-of-spaces}text-to-indent
```

Our second and third COMPUTE blocks become the following:

```
compute rowlbl;
  *  Placeholder for superscript and indenting;
  if (rowlbl eq 'Summary Statistics')
    then rowlbl = cats(rowlbl, '^{super a}');
  else if (rowlbl eq 'LS Mean')
    then rowlbl = cats(rowlbl, '^{super b}');
  if (indent.sum ne 0)
    then rowlbl = cats('^{nbspace 3}', rowlbl);
endcomp;

compute after avisitn / style=[just=l];
  line ' ';
  line 'N = number of subjects randomized; Xanomeline Low Dose = 54 mg;
        Xanomeline High Dose = 81 mg.';
  line ' ';
  line '^{super a}Summary statistics are based on observed data.';
  line ' ';
  line '^{super b}LS mean is from a repeated measures model
        which includes effects: treatment group, scheduled';
  line 'visit, and the interaction of treatment with scheduled visit.';
endcomp;
```

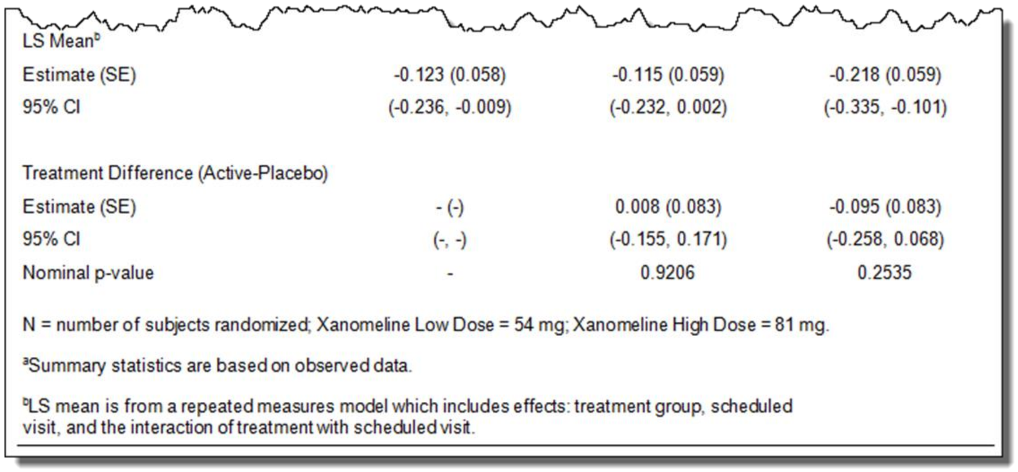
The CATS function removes leading and trailing blanks from its arguments, concatenates the arguments, and then returns the concatenated string (SAS Institute Inc. 2018a).

Table 8 shows some of the values of ROWLBL.

Summary Statistics^{super a}
LS Mean^{super b}
^{nbspace 3}Estimate (SE)
^{nbspace 3}95% CI
^{super a}Summary statistics are based on observed data.

Table 8. Sample Values of the ROWLBL Column

Partial PROC REPORT output is shown in Figure 14.



LS Mean[b]			
Estimate (SE)	-0.123 (0.058)	-0.115 (0.059)	-0.218 (0.059)
95% CI	(-0.236, -0.009)	(-0.232, 0.002)	(-0.335, -0.101)
Treatment Difference (Active-Placebo)			
Estimate (SE)	- (-)	0.008 (0.083)	-0.095 (0.083)
95% CI	(-, -)	(-0.155, 0.171)	(-0.258, 0.068)
Nominal p-value	-	0.9206	0.2535

N = number of subjects randomized; Xanomeline Low Dose = 54 mg; Xanomeline High Dose = 81 mg.

[a]Summary statistics are based on observed data.

[b]LS mean is from a repeated measures model which includes effects: treatment group, scheduled visit, and the interaction of treatment with scheduled visit.

Figure 14. Partial PROC REPORT Output Showing ODS Inline Formatting

However, notice that the row label text is not indented. The ODS RTF destination does not honor leading blanks by default. The Applying ODS Style Overrides to the Column Headings and Data Cells section discusses how to correct this issue.

UNDERSTANDING AND USING ODS STYLE OVERRIDES

The row label text is not indented, and the column headings in our final report are not italicized. We could use PROC TEMPLATE to change the Journal style to correct these issues, but it is easier instead to use ODS style overrides.

You can alter the appearance of specific parts of your PRINT, REPORT, and TABULATE procedure output by using style overrides. These specific parts of your SAS output are called locations. Figure 15 shows the locations of the REPORT procedure output (SAS Institute Inc. 2008).

The HEADER location controls the appearance of column headings and the COLUMN location applies to the data cells.

Style overrides can be specified in several ways. These are the two most common formats:

```
❶ style(location)=[style-attribute-name1=value1
                   style-attribute-name2=value2 ...]

❷ style(location)=style-element-name
```

The first format (❶) uses individual style attributes that are defined inline. For example, this PROC REPORT code alters three attributes of the COLUMN location (data cells) for only the MYVAR column:

```
define myvar / style(column)=[background=yellow font_size=10pt just=left];
```

Although this is the most commonly used format, it has some disadvantages. To use the same style override for different variables, you must apply it in multiple places, making your SAS code harder to read and maintain. In addition, if you want to use the style overrides in other SAS programs, you must copy the list of attribute name/value pairs to the new code. Because of these drawbacks, inline style overrides should be used sparingly.

The second format (❷) overcomes these problems by referencing a style element. Using this format involves creating a new style element, setting the style attributes within the element, and then using the style element name in your style override. This results in code that is easier to read, maintain, and reuse.

Earlier papers by this author provide additional information about using this format (DelGobbo 2008, 2009, 2010, 2011).

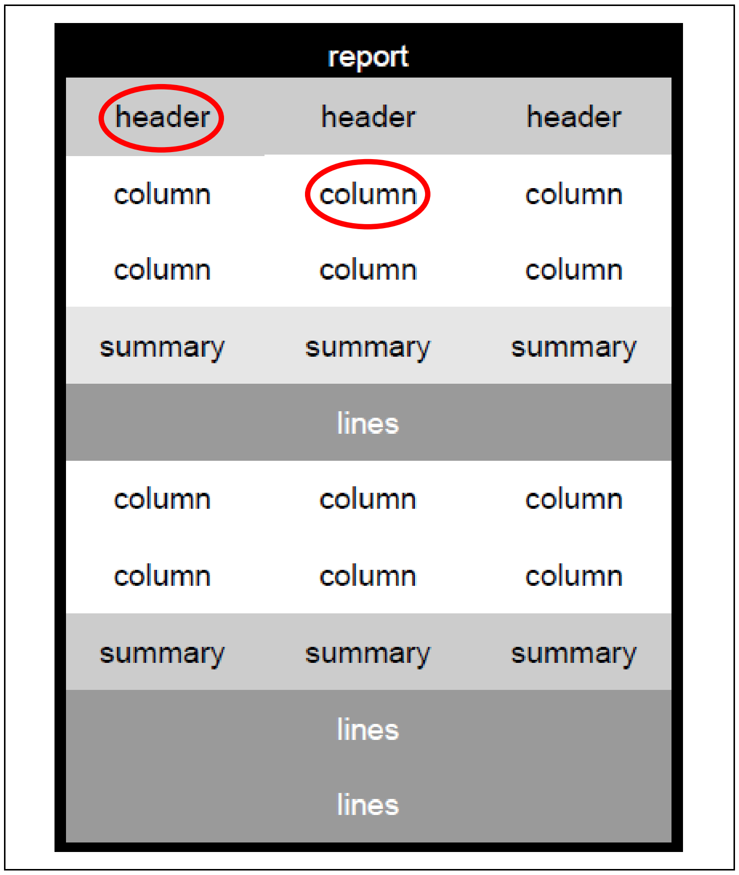


Figure 15. Style Locations for the REPORT Procedure

You can use a style override in the PROC REPORT statement to change the appearance of *all* columns:

```
proc report style(column)=[background=yellow font_size=10pt just=left] ...
```

This code specifies that all data cells in the report have a yellow background, and use left-justified 10-point text.

Refer to the ODS documentation for a full listing of style attributes (SAS Institute Inc. 2017g).

APPLYING ODS STYLE OVERRIDES TO THE COLUMN HEADINGS AND DATA CELLS

There are two differences between the output of Figure 13 and the final report (Figure 1): the column headings are displayed with italic text, and the statistic row labels are not indented under their section headings. ODS style overrides can correct both issues.

You could change the appearance of the column headings by applying a style override in the DEFINE statement for each column displayed in the report, but this is cumbersome when there are several columns. A more efficient way to change the appearance of all column headings is to specify the style override in the PROC statement.

The leading blanks in our statistic labels are honored when we specify `asis=on` in the DEFINE statement for the ROWLBL column:

```
proc report data=work.report_final nowd missing split='~'
    style(header)=[font_style=roman];
  columns avisitn section rownum indent rowlbl trt_:;

  define avisitn / order order=internal noprint;
  define section / order order=internal noprint;
  define rownum  / order order=internal noprint;
  define indent  / noprint;
  define rowlbl  / display ' ' style(column)=[asis=on];
  define trt_:   / display center;

  ...;

run; quit;
```

Our output now matches Figure 1.

THE FINAL SAS CODE

```
options nodate nonumber missing='-';

*  Close all ODS destinations, and then open when needed;

ods _all_ close;

*  ODS escape character used for inline formatting;

ods escapechar = '^';

*;
*  Create a global macro variable with the value of the study ID for
*  use in a title.
*;

data _null_;
set sample.adsl(obs=1);
call symputx('STUDYID', studyid);
run;

*  Format used for N;

%let STATFMT0D=12.;

*  Format used for MIN and MAX requires two decimal places;

%let STATFMT2D=%sysevalf(&STATFMT0D + 0.2);

*  Format used for other statistics requires three decimal places;

%let STATFMT3D=%sysevalf(&STATFMT2D + 0.1);

*  Turn ODS graphics off;

ods graphics off;
```

```
*  Open the HTML destination to create the procedure output;

ods html path='directory-location' file='Statistics.htm' style=Journal;

*  Create data for the column headings of the report;

ods trace on;

ods output Summary=work.colhead;

title 'Data Used for the Column Headings of the Report';

proc summary print data=sample.adsl;
  class trt01pn trt01p;
run; quit;

*  Create the statistics for Section 1 of the report;

ods output Summary=work.summary;

title 'Statistics for Section 1 of the Report';

proc summary print data=sample.analysis missing stackods
             n mean stderr median q1 q3 min max;
  class avisitn trt01pn;
  var chg;
run; quit;

*;
*  Create the statistics for Section 2 and Section 3 of the report.
*  Differences are computed in reference to the placebo
*  (Active Drug - Placebo).
*;

ods output LSMeans=work.lsmeans SliceDiffs=work.diffs;

title 'Statistics for Sections 2 and 3 of the Report';

proc mixed data=sample.analysis;
  class usubjid avisitn trt01pn;
  model chg = avisitn trt01pn avisitn*trt01pn / ddfm=kenwardroger;
  repeated avisitn / subject=usubjid type=un;

  lsmeans avisitn*trt01pn / cl;
  slice   avisitn*trt01pn / cl diff=control('2' '0') sliceby=avisitn;
run; quit;

ods trace off;

*  Close the HTML destination;

ods html close;

*;
*  Add a record to the DIFFS table to represent the placebo. Statistics for
*  these records are missing.
*;
```

```
proc sort data=work.diffs out=work.diffs_sorted; by slice; run; quit;

data work.diffs2(drop=stmtno effect slice _trt01pn);
set work.diffs_sorted;
by slice;
length avisitn 8.;
avisitn = input(scan(slice, 2), best.);
output;
if (last.slice) then do;
  trt01pn = _trt01pn;
  call missing(of estimate stderr df tvalue probt alpha lower upper);
  output;
end;
run;

*;
*  Create the input control data set and format used later to create the
*  column headings.
*;

data work.colhead_cntlin(drop=trt01pn trt01p nobs first_space);
set work.colhead;
length fmtname $7 type $1 start end 8 label $30 first_space 8;

*  Replace the first blank space in the label with the split character;

first_space = index(strip(trt01p), ' ');

if (first_space ne 0)
  then substr(trt01p, first_space, 1) = '~';

*  Specify the required information for the format;

fmtname = 'COLHEAD';
type    = 'N';
start   = trt01pn;
end     = trt01pn;
label   = strip(trt01p) || '~(N = ' || strip(put(nobs, &STATFMT0D)) || ')';
run;

proc format cntlin=work.colhead_cntlin; run; quit;

*;
*  Create a separate row from specific column values of the
*  Summary data - Section 1.
*;

data work.report_summary(keep=section rownum indent rowlbl cell
                              avisitn trt01pn idlabel);
set work.summary;
by avisitn trt01pn;

length idlabel $30 rownum indent 8 rowlbl $40 cell $20;

section = 1;

if (first.trt01pn) then rownum = 0;
```

```
*  Column heading;

idlabel = strip(put(trt01pn, colhead.));

* Row Heading;

rownum +1;
indent = 0;
rowlbl = 'Summary Statistics';
cell   = '';
output;

* Row Values;

indent = 1;

rownum +1;
rowlbl = 'n';
cell   = strip(put(n, &STATFMT0D));
output;

rownum +1;
rowlbl = 'Mean';
cell   = strip(put(mean, &STATFMT3D));
output;

rownum +1;
rowlbl = 'SE';
cell   = strip(put(stderr, &STATFMT3D));
output;
rownum +1;
rowlbl = 'Median';
cell   = strip(put(median, &STATFMT3D));
output;

rownum +1;
rowlbl = 'Q1, Q3';
cell   = strip(put(q1, &STATFMT3D)) || ', ' || strip(put(q3, &STATFMT3D));
output;

rownum +1;
rowlbl = 'Min, Max';
cell   = strip(put(min, &STATFMT2D)) || ', ' ||
         strip(put(max, &STATFMT2D));
output;

run;

*;
*  Create a separate row from specific column values of the
*  LS Means data - Section 2.
*;

proc sort data=work.lsmeans out=work.lsmeans_sorted;
  by avisitn trt01pn;
run; quit;
```

```
data work.report_lsmeans(keep=section rownum indent rowlbl cell
                              avisitn trt01pn idlabel);
set work.lsmeans_sorted;
by avisitn trt01pn;

length idlabel $30 rownum indent 8 rowlbl $40 cell $20;

section = 2;

if (first.trt01pn) then rownum = 0;

*  Column heading;

idlabel = strip(put(trt01pn, colhead.));

* Blank row;

rownum +1;
indent = 0;
rowlbl = '';
cell   = '';
output;

* Row Heading;

indent = 0;
rownum +1;
rowlbl = 'LS Mean';
cell   = '';
output;
* Row Values;

indent = 1;

rownum +1;
rowlbl = 'Estimate (SE)';
cell   = strip(put(estimate, &STATFMT3D)) ||
         ' (' || strip(put(stderr, &STATFMT3D)) || ')';
output;

rownum +1;
rowlbl = '95% CI';
cell   = '('  || strip(put(lower, &STATFMT3D)) ||
         ', ' || strip(put(upper, &STATFMT3D)) || ')';
output;

run;

*;
*  Create a separate row from specific column values of the
*  differences of LS Means data - Section 3.
*;

proc sort data=work.diffs2 out=work.diffs2_sorted;
  by avisitn trt01pn;
run; quit;
```

```
data work.report_diffs(keep=section rownum indent rowlbl cell
                            avisitn trt01pn idlabel);
set work.diffs2_sorted;
by avisitn trt01pn;

length idlabel $30 rownum indent 8 rowlbl $40 cell $20;

section = 3;

if (first.trt01pn) then rownum = 0;

*  Column heading;

idlabel = strip(put(trt01pn, colhead.));

* Blank row;

rownum +1;
indent = 0;
rowlbl = '';
cell   = '';
output;

* Row Heading;

indent = 0;
rownum +1;
rowlbl = 'Treatment Difference (Active-Placebo)';
cell   = '';
output;
* Row Values;

indent = 1;

rownum +1;
rowlbl = 'Estimate (SE)';
cell   = strip(put(estimate, &STATFMT3D)) ||
         ' (' || strip(put(stderr, &STATFMT3D)) || ')';
output;

rownum +1;
rowlbl = '95% CI';
cell   = '('  || strip(put(lower, &STATFMT3D)) ||
         ', ' || strip(put(upper, &STATFMT3D)) || ')';
output;

rownum +1;
rowlbl = 'Nominal p-value';
cell   = strip(put(probt, pvalue6.4));
output;

run;
```

```
*  Combine the preliminary data into a single table;

data work.report1;
set work.report_summary
    work.report_lsmeans
    work.report_diffs;
run;

*  Sort in preparation for transposition;

proc sort data=work.report1 out=work.report1_sorted;
  by avisitn section rownum rowlbl indent trt01pn;
run; quit;

*  Transpose the data to get the layout needed for the report;

proc transpose data=work.report1_sorted prefix=trt_
               out=work.report_final(drop=_name_);
  by avisitn section rownum rowlbl indent;
  var cell;
  id trt01pn;
  idlabel idlabel;
run; quit;

*  Open the RTF destination to create the procedure output;

ods rtf file='directory-location\CHOL_Report.rtf' style=Journal bodytitle;

title1 'Table x.y: Change from Baseline LDL-C by Scheduled Visit';
title2 "Study &STUDYID";

proc report data=work.report_final nowd missing split='~'
    style(header)=[font_style=roman];
  columns avisitn section rownum indent rowlbl trt_: ;
  define avisitn / order order=internal noprint;
  define section / order order=internal noprint;
  define rownum  / order order=internal noprint;
  define indent  / noprint;
  define rowlbl  / display ' ' style(column)=[asis=on];
  define trt_:   / display center;

  break after avisitn / page;

  compute before avisitn / style=[just=l];
    text = catx(' ', 'Week', avisitn);
    line text $;
  endcomp;

  compute rowlbl;
    *  Placeholder for superscript and indenting;
    if (rowlbl eq 'Summary Statistics')
      then rowlbl = cats(rowlbl, '^{super a}');
    else if (rowlbl eq 'LS Mean')
      then rowlbl = cats(rowlbl, '^{super b}');
    if (indent.sum ne 0)
      then rowlbl = cats('^{nbspace 3}', rowlbl);
  endcomp;
```

```
    compute after avisitn / style=[just=l];
      line ' ';
      line 'N = number of subjects randomized; Xanomeline Low Dose = 54 mg;
            Xanomeline High Dose = 81 mg.';
      line ' ';
      line '^{super a}Summary statistics are based on observed data.';
      line ' ';
      line '^{super b}LS mean is from a repeated measures model
            which includes effects: treatment group, scheduled';
      line 'visit, and the interaction of treatment with scheduled visit.';
    endcomp;

  run; quit;

  *  Close the RTF destination;

  ods rtf close;
```

CREATING THE REPORT IN HTML, PDF, AND EXCEL XLSX FORMAT

Add these ODS statements to the final SAS code to create PDF and HTML output:

```
*  Open the RTF destination to create the procedure output;

ods rtf file='directory-location\CHOL_Report.rtf' style=Journal bodytitle;

*  Open the PDF and HTML destinations to create the procedure output;

ods pdf  file='directory-location\CHOL_Report.pdf'        style=Journal;
ods html path='directory-location' file='CHOL_Report.htm' style=Journal;

title1 ...;
title2 ...;
proc report data=work.report_final ...; run; quit;

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

Add a BY statement to the PROC REPORT code to create Microsoft Excel output with worksheets named according to the value of AVISITN:

```
*  Open the Excel destination to create the procedure output;

ods excel file='directory-location\CHOL_Report.xlsx' style=Journal
  options(orientation='landscape'
          embedded_titles='yes'
          suppress_bylines='yes'
          sheet_name='Week #byval(avisitn)');

title1 ...;
title2 ...;

proc report data=work.report_final ...;
  by avisitn;
  ...;
run; quit;
```

```
ods excel close;
```

Table 9 briefly explains the Excel-specific options used in this code.

Option	Description
orientation	Printed output fits to a single page in landscape orientation.
embedded_titles	Title text appears in the workbook, instead of the print header.
suppress_bylilnes	BY line text is not included in the output.
sheet_name	Worksheet names begin with **Week,** followed by the week number.

Table 9. Explanation of ODS Excel Destination Options

Creating attractive and function Excel output using SAS is discussed in earlier papers (DelGobbo, 2018).

SAS SERVER TECHNOLOGY

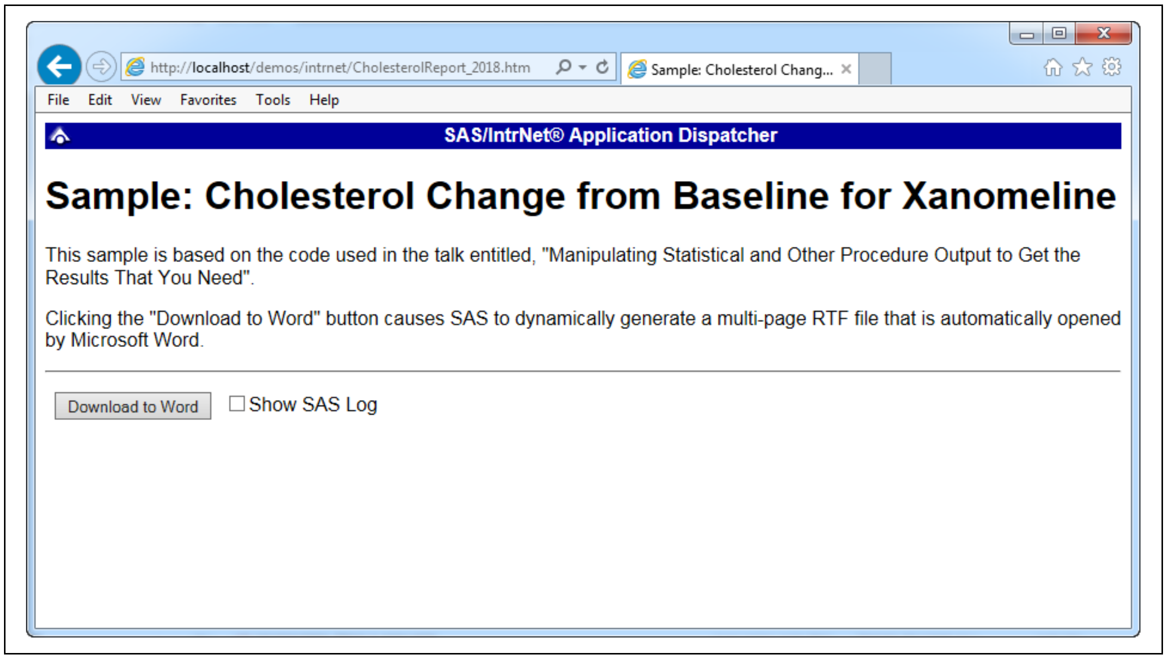
You can deliver dynamically generated SAS output in Microsoft Word using the Application Dispatcher or the SAS® Stored Process Server. The Application Dispatcher is part of SAS/IntrNet® software. The SAS Stored Process Server is available starting with SAS®9 as part of SAS® Integration Technologies, and is included with server offerings that use the SAS® Business Analytics infrastructure (for example, SAS® BI Server and SAS® Enterprise BI Server).

These products enable you to execute SAS programs from a Web browser or any other client that can open an HTTP connection to the Application Dispatcher or the SAS Stored Process Server. Both of these products can run on any platform where SAS is licensed. SAS software does not need to be installed on the client machine.

The SAS programs that you execute from the browser can contain any combination of DATA step, procedure, macro, or SCL code. Thus, all of the code that has been shown up to this point can be executed by both the Application Dispatcher and the SAS Stored Process Server.

Program execution is typically initiated by accessing a URL that points to the SAS server program. Parameters are passed to the program as name/value pairs in the URL. The SAS server takes these name/value pairs and constructs SAS macro variables that are available to the SAS program.

Figure 16 shows a Web page that can deliver SAS output directly to Microsoft Word, using a Web browser as the client.


http://localhost/demos/intrnet/CholesterolReport_2018.htm
Sample: Cholesterol Chang...
File Edit View Favorites Tools Help
SAS/IntrNet® Application Dispatcher

Sample: Cholesterol Change from Baseline for Xanomeline

This sample is based on the code used in the talk entitled, "Manipulating Statistical and Other Procedure Output to Get the Results That You Need".

Clicking the "Download to Word" button causes SAS to dynamically generate a multi-page RTF file that is automatically opened by Microsoft Word.

Download to Word
Show SAS Log

Figure 16. Web Page to Drive a SAS/IntrNet Application

Clicking **Download to Word** executes a slightly modified version of the SAS code that we have been working on. The modifications are as follows:

```
❶ %let RV=%sysfunc(appsrv_header(Content-type, application/msword));
❷ %let RV=%sysfunc(appsrv_header(Content-disposition, %str(attachment;
  filename="CHOL_Report.rtf")));  * Ignore line wrapping;

  ods _all_ close;

  ods rtf file=_webout style=Journal bodytitle;

  *  Remainder of the "final" SAS code;

  ods rtf close;
```

(❶) Set a MIME header that causes the SAS output to be opened by Microsoft Word, instead of being rendered by the Web browser. This statement is required.

(❷) Set a MIME header that causes the filename to be displayed in the File Download dialog box. As you can see in Figure 17, the filename appears as **CHOL_Report.rtf**. This header might cause problems with some versions of Microsoft Word, so be sure to test your applications before deploying them in a production environment. This statement is optional.

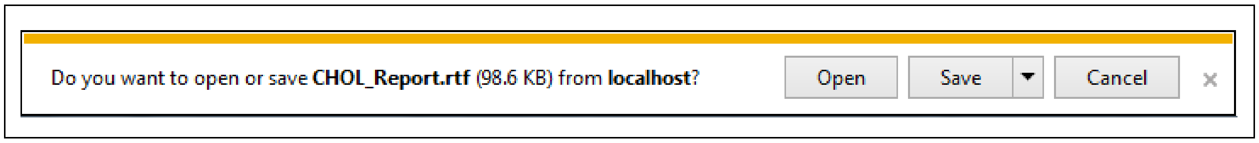

Do you want to open or save **CHOL_Report.rtf** (98.6 KB) from **localhost**?

Figure 17. File Download Dialog Box

The reserved _WEBOUT fileref is defined by the SAS server and directs output from the SAS server to the client. Modify your existing ODS statement to direct the output to this fileref instead of to an external disk file.

When you click the **Download to Word** button on the Web page, you are presented with the File Download dialog box (Figure 17). You can then click **Open** to immediately open your SAS output using Microsoft Word, or you can click **Save**.

For more detailed information and other examples, see the SAS/IntrNet Application Dispatcher and SAS Stored Process documentation (SAS Institute Inc. 2018d, 2018f). This author's earlier papers provide examples of dynamically delivering output to Excel (DelGobbo 2018)

CONCLUSION

Although SAS provides a rich set of procedures to perform statistical analyses, the format of the output usually cannot be used directly in journal articles or other documents. By using DATA step code and the SAS Output Delivery System (ODS), you can create customized output, reducing or eliminating the need for manual editing.

REFERENCES

Clinical Data Interchange Standards Consortium (CDISC). 2018. "Pilot Project Submission Package". Available at https://www.cdisc.org/pilot-project-submission-package.

DelGobbo, Vincent. 2008. "Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008/192-2008.pdf.

DelGobbo, Vincent. 2009. "More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/152-2009.pdf.

DelGobbo, Vincent. 2010. "Traffic Lighting Your Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/153-2010.pdf.

DelGobbo, Vincent. 2011. "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/170-2011.pdf.

DelGobbo, Vincent and Peter Flom. 2015. "Preparing Output from Statistical Procedures for Publication, Part 1: PROC REG to APA Format". *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings15/3307-2015.pdf.

DelGobbo, Vincent. 2018. "Related Papers". Available at http://support.sas.com/rnd/papers/intro-multisheet-excel-with-sas/index.html.

SAS Institute Inc. 2008. "SAS®9 Reporting Procedure Styles Tip Sheet". Available at http://support.sas.com/rnd/base/ods/scratch/reporting-styles-tips.pdf.

SAS Institute Inc. 2009. "Sample 36900: Instructions for viewing all of the style templates that are shipped with SAS®". Available at http://support.sas.com/techsup/notes/v8/36/900.html.

SAS Institute Inc. 2017a. "Name Prefix Lists". *SAS® 9.4 Language Reference: Concepts, Sixth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=lrcon&docsetTarget=p0wphcpsfgx6o7n1sjtqzizp1n39.htm&docsetVersion=9.4#n1dg3poi3joxh3n1tec2e293spdy.

SAS Institute Inc. 2017b. "ODS ESCAPECHAR Statement". *SAS® 9.4 Output Delivery System: User's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=odsug&docsetTarget=p11xia2ltavr8ln17srq8vn4rnqc.htm&docsetVersion=9.4.

SAS Institute Inc. 2017c. "ODS OUTPUT Statement". *SAS® 9.4 Output Delivery System: User's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=odsug&docsetTarget=p0oxrbinw6fjuwn1x23qam6dntyd.htm&docsetVersion=9.4.

SAS Institute Inc. 2017d. "ODS RTF Statement". *SAS® 9.4 Output Delivery System: User's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=odsug&docsetTarget=p1vvsv8ucnjzjnn1wq5wrlp74mdb.htm&docsetVersion=9.4.

SAS Institute Inc. 2017e. "ODS TRACE Statement". *SAS® 9.4 Output Delivery System: User's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=odsug&docsetTarget=p1fpt3uuow90o3n155hs7bp1mo7f.htm&docsetVersion=9.4.

SAS Institute Inc. 2017f. "PVALUE*w.d* Format". *SAS® 9.4 Formats and Informats: Reference*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=leforinforref&docsetTarget=p0pzklcc1zxivxn1js2y5nkg2wq9.htm&docsetVersion=9.4.

SAS Institute Inc. 2017g. "Style Attributes Detailed Information". *SAS® 9.4 Output Delivery System: User's Guide, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=odsug&docsetTarget=p0xi2cygmfk0wkn1ei625zq5r488.htm&docsetVersion=9.4.

SAS Institute Inc. 2018a. "CATS Function". *SAS® 9.4 Functions and CALL Routines: Reference, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=n1e21rr6al5m2nn19r1fat5qxwrt.htm&docsetVersion=9.4.

SAS Institute Inc. 2018b. "CATX Function". *SAS® 9.4 Functions and CALL Routines: Reference, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=n0p7wxtk0hvn83n1pveisbcp2ae9.htm&docsetVersion=9.4.

SAS Institute Inc. 2018c. "Input Control Data Set". *Base SAS® 9.4 Procedures Guide, Seventh Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=proc&docsetTarget=p0owa4ftikc2ekn1q0rmpulg86cx.htm&docsetVersion=9.4#p05g1cdvff8h8tn1frtzxrnkzezo.

SAS Institute Inc. 2018d. "Introduction to the Application Dispatcher". *SAS/IntrNet® 9.4: Application Dispatcher*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=dispatch&docsetTarget=p06h82ux8glu1pn16k9dxw8tjpyz.htm&docsetVersion=9.4.

SAS Institute Inc. 2018e. "SUBSTR (left of =) Function". *SAS® 9.4 Functions and CALL Routines: Reference, Fifth Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=p0uev77ebdwy90n1rsd7hwjd2qc3.htm&docsetVersion=9.4.

SAS Institute Inc. 2018f. "What Are SAS Stored Processes?" *SAS® 9.4 Stored Processes: Developer's Guide, Third Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=stpug&docsetTarget=n180km1hadyuton13gx4bzlqhpwr.htm&docsetVersion=9.4.

ACKNOWLEDGMENTS

The author would like to thank John King of Ouachita Clinical Data Services, and Chris Barrett and Randy Tobias of SAS Institute Inc. for their valuable contributions to this paper.

RECOMMENDED READING

SAS Institute Inc. 2017. "REPORT Procedure". *Base SAS® 9.4 Procedures Guide, Seventh Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=proc&docsetTarget=p0bqogcics9o4xn17yvt2qjbgdpi.htm&docsetVersion=9.4.

SAS Institute Inc. 2017. "SUMMARY Procedure". *Base SAS® 9.4 Procedures Guide, Seventh Edition*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=proc&docsetTarget=p0aq3hsvflztfzn1xa2wt6s35oy6.htm&docsetVersion=9.4.

SAS Institute Inc. 2017. "The MIXED Procedure". *SAS/STAT 14.3 User's Guide*. Cary, NC: SAS Institute Inc. Available at http://documentation.sas.com/?docsetId=statug&docsetVersion=14.3&docsetTarget=statug_mixed_toc.htm

CONTACT INFORMATION

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

Vincent DelGobbo
SAS Institute Inc.
100 SAS Campus Drive
Cary, NC 27513

sasvcd@sas.com
http://www.sas.com

LinkedIn: https://www.linkedin.com/in/vince-delgobbo-a717b88/

Papers and Resources: http://www.sas.com/reg/gen/corp/867226?page=Resources

If your registered in-house or local SAS users group would like to request this presentation as your annual SAS presentation (as a seminar, talk, or workshop) at an upcoming meeting, please submit an online User Group Request Form (goo.gl/yNCxeT) at least eight weeks in advance.