PharmaSUG 2026 - Paper DV-294

Composite TLFs – A Combined Approach to Data Visualization

Jesse Pratt, PPD, part of Thermo Fisher Scientific
Rayce Wiggins, PPD, part of Thermo Fisher Scientific

ABSTRACT

Clinical trial reviewers are routinely required to interpret results across separate tables, listings, and figures (TLFs). While each output serves a specific purpose, reviewing them independently introduces inefficiencies, increases cognitive burden, and raises the risk of missed signals. This paper introduces Composite TLFs, a unified reporting approach that programmatically integrates tables, listings, and figures into single, cohesive outputs.

Composite TLFs consolidate logically related summary, subject-level, and graphical information into one output, allowing reviewers to assess results holistically without navigating between files. Examples are presented using simulated data and implemented in both SAS® and R, demonstrating that this approach is software-agnostic and readily adoptable across clinical programming environments.

INTRODUCTION

Tables, listings, and figures form the foundation of clinical trial reporting. Tables summarize, listings provide traceability, and figures highlight trends. Traditionally, these outputs are generated and reviewed independently, requiring reviewers to cross-reference multiple files to fully interpret results.

Composite TLFs address these limitations by integrating complementary TLF components into a single output designed around a focused clinical question.

COMPOSITE TLF CONCEPT

An Composite TLF is a single output that combines two or more traditional TLF components, such as a table and figure or a listing and figure, into a unified display. Each component addresses the same analytical objective and is visually synchronized through layout, color, and scale.

TEHCNICAL IMPLEMENTATION

SAS IMPLEMENTATION

In SAS, Composite TLFs are implemented using ODS Graphics and Graph Template Language (GTL). Table and figure data sets are generated from a common source data set, then combined. Tabular components are rendered using text plot elements within GTL, allowing precise alignment with graphical components. Consistent attribute maps ensure identical treatment-level color usage across tables and figures.

R IMPLEMENTATION

In R, Composite TLFs are constructed using ggplot2 for both tabular and graphical elements, with patchwork used to assemble multiple plots into a single cohesive layout. Using the patchwork package allows us to keep the table and figure data sets separate while still allowing a single combined image to be generated. Shared color mappings and scales ensure visual synchronization across components.

EXAMPLE 1: COMBINED TABLE-FIGURE OUTPUT - SAS

Figure 1 presents a Composite TLF that combines a categorical summary table with a corresponding bar chart displaying the number of patients experiencing infrequent bowel movements by treatment group. By presenting these components together, reviewers can immediately reconcile numeric values with graphical trends.

Number of Patients Experiencing Infrequent Bowel Movements	IMP-101 5 mg QD (N=109) n (%)	IMP-101 10 mg QD (N=82) n (%)	Placebo (N=129) n (%)	Total (N=320) n (%)
0 Events	11 (10.1)	12 (14.6)	8 (6.2)	31 (9.7)
1 Event	10 (9.2)	8 (9.8)	11 (8.5)	29 (9.1)
2 Events	13 (11.9)	10 (12.2)	7 (5.4)	30 (9.4)
3 Events	7 (6.4)	9 (11.0)	9 (7.0)	25 (7.8)
4 Events	10 (9.2)	7 (8.5)	10 (7.8)	27 (8.4)
5 Events	10 (9.2)	10 (12.2)	6 (4.7)	26 (8.1)
>5 Events	6 (5.5)	15 (18.3)	5 (3.9)	26 (8.1)

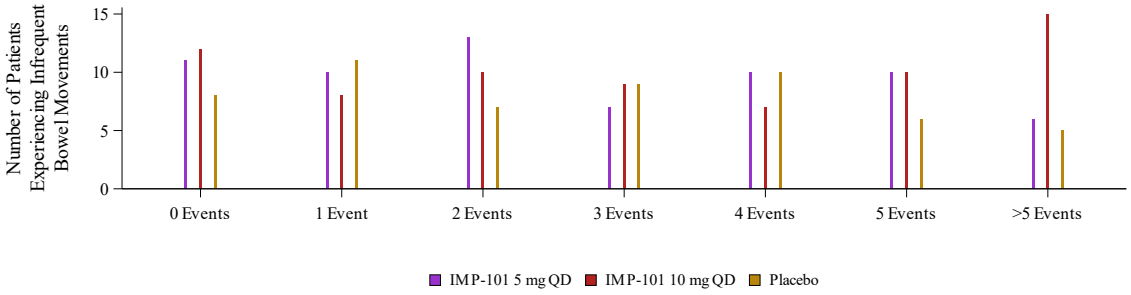


Figure 1. Composite Table–Figure Output Combining Summary Table and Bar Chart.

Let's explore the technical implementation of this in SAS. First, we start with a single common data set. We'll assume that, beginning with one or more ADaM data sets, we perform manipulations that lead us to the data set given in Figure 2.

COL1	N_1	N_2	N_3	N_99	PCNT_1	PCNT_2	PCNT_3	PCNT_99	YVAR	XVAR	TRT01PN
Number of Patien...									0	0	1
0 Events	11	12	8	31	10.09174311926...	14.63414634146...	6.2015503875969	9.6875	1	0	2
1 Event	10	8	11	29	9.1743119266055	9.756097560975...	8.527131782945...	9.0625	2	0	3
2 Events	13	10	7	30	11.92660550458...	12.19512195121...	5.426356589147...	9.375	3	0	99
3 Events	7	9	9	25	6.422018348623...	10.97560975609...	6.976744186046...	7.8125	4	0	
4 Events	10	7	10	27	9.1743119266055	8.536585365853...	7.751937984496...	8.4375	5	0	
5 Events	10	10	6	26	9.1743119266055	12.19512195121...	4.651162790697...	8.125	6	0	
>5 Events	6	15	5	26	5.5045871559633	18.29268292682...	3.875968992248...	8.125	7	0	

BIGN	COL2	COL3	COL4	COL5
109				
82	11 (10.1)	12 (14.6)	8 (6.2)	31 (9.7)
129	10 (9.2)	8 (9.8)	11 (8.5)	29 (9.1)
320	13 (11.9)	10 (12.2)	7 (5.4)	30 (9.4)
	7 (6.4)	9 (11.0)	9 (7.0)	25 (7.8)
	10 (9.2)	7 (8.5)	10 (7.8)	27 (8.4)
	10 (9.2)	10 (12.2)	6 (4.7)	26 (8.1)
	6 (5.5)	15 (18.3)	5 (3.9)	26 (8.1)

Figure 2. Starting Data Set for Table-Figure Output

A few notes:

- This data set contains both numeric and character variables for our quantities of interest.
- The numeric variables will be reserved for the figure portion of the output.
- The character variables will be reserved for the table portion of the output.
- The variables XVAR and YVAR are utilized in the table portion as positioning variables.

After some straightforward manipulations, we arrive at the following data sets for the table (Figure 3) and figure (Figure 4) portions of our output.

COL1	YVAR	XVAR	COL2	COL3	COL4	COL5
	0	0				
0 Events	1	0	11 (10.1)	12 (14.6)	8 (6.2)	31 (9.7)
1 Event	2	0	10 (9.2)	8 (9.8)	11 (8.5)	29 (9.1)
2 Events	3	0	13 (11.9)	10 (12.2)	7 (5.4)	30 (9.4)
3 Events	4	0	7 (6.4)	9 (11.0)	9 (7.0)	25 (7.8)
4 Events	5	0	10 (9.2)	7 (8.5)	10 (7.8)	27 (8.4)
5 Events	6	0	10 (9.2)	10 (12.2)	6 (4.7)	26 (8.1)
>5 Events	7	0	6 (5.5)	15 (18.3)	5 (3.9)	26 (8.1)

Figure 3. Table Data Set

XVAR_F	TRT	YVAR_F
1	N_1	11
1	N_2	12
1	N_3	8
2	N_1	10
2	N_2	8
2	N_3	11
3	N_1	13
3	N_2	10
3	N_3	7
4	N_1	7
4	N_2	9
4	N_3	9
5	N_1	10
5	N_2	7
5	N_3	10
6	N_1	10
6	N_2	10
6	N_3	6

Figure 4. Figure Data Set

Note that these data sets have no variable names in common. This is because we need to merge them together into a single input data set (GTL supports only one data set at a time). When we do, we have the following:

COL1	YVAR	XVAR	COL2	COL3	COL4	COL5	XVAR_F	TRT	YVAR_F
	0	0					1	N_1	11
0 Events	1	0	11 (10.1)	12 (14.6)	8 (6.2)	31 (9.7)	1	N_2	12
1 Event	2	0	10 (9.2)	8 (9.8)	11 (8.5)	29 (9.1)	1	N_3	8
2 Events	3	0	13 (11.9)	10 (12.2)	7 (5.4)	30 (9.4)	2	N_1	10
3 Events	4	0	7 (6.4)	9 (11.0)	9 (7.0)	25 (7.8)	2	N_2	8
4 Events	5	0	10 (9.2)	7 (8.5)	10 (7.8)	27 (8.4)	2	N_3	11
5 Events	6	0	10 (9.2)	10 (12.2)	6 (4.7)	26 (8.1)	3	N_1	13
>5 Events	7	0	6 (5.5)	15 (18.3)	5 (3.9)	26 (8.1)	3	N_2	10
							3	N_3	7
							4	N_1	7
							4	N_2	9
							4	N_3	9
							5	N_1	10
							5	N_2	7
							5	N_3	10
							6	N_1	10
							6	N_2	10
							6	N_3	6

Figure 5. Combined Table-Figure Data Set for GTL

Now let's explore the GTL code that generates our combined Table-Figure output from the above data set. The structure of this code is as follows:

- A LAYOUT LATTICE block with two cells – the top cell generating the table portion of the output and the bottom cell generating the figure portion of the output.
- The table portion of the output consists of another LAYOUT LATTICE block consisting of 5 side-by-side cells – one cell for each column of the table – and a series of annotations for the column headers.
- The figure portion of the output consists of a LAYOUT OVERLAY block containing a needle plot to create the bars, and our legend.

Each of the five table columns will contain its own LAYOUT OVERLAY block, so it will be to our benefit to write a brief macro to cover this like so:

```
%macro loopit;
%do i=1 %to 5;
       layout overlay / xaxisopts=(display=none)
                        yaxisopts=(display=none reverse=true)
                    walldisplay=(fill) pad=(top=40px);
          scatterplot x=XVAR y=YVAR / markerattrs=(color=white);
            innermargin / align=left;
              axistable value=COL&i y=YVAR / showmissing=true display=(values) valueattrs=(size=9);
          endinnermargin;
       endlayout;
%end;
%mend;
```

Notes:

- We use a y-axis table to plot the elements contained in the table.
- Since plotting statements are required within a LAYOUT OVERLAY block, we generate a simple scatter plot of XVAR and YVAR and make the markers white so that they aren't visible.

We now start our GTL code with building our legend items and attribute maps:

```
proc template;
   define statgraph tf1402040102;
      begingraph;
       legenditem name="lgnd1" type=fill / label="IMP-101 5 mg QD" fillattrs=(color=darkorchid);
       legenditem name="lgnd2" type=fill / label="IMP-101 10 mg QD" fillattrs=(color=firebrick);
       legenditem name="lgnd3" type=fill / label="Placebo" fillattrs=(color=darkgoldenrod);
       DiscreteAttrVar attrvar=ATTR_TRT var=TRT attrmap="myattrmap";
       DiscreteAttrMap name="myattrmap";
         value "N_1" / lineattrs=(color=darkorchid);
         value "N_2" / lineattrs=(color=firebrick);
         value "N_3" / lineattrs=(color=darkgoldenrod);
       enddiscreteattrmap;
```

Next, the outer LAYOUT LATTICE:

```
layout lattice / rows=2 rowweights=(0.45 0.55) rowgutter=20px;
```

We then build our table by calling the previous macro, along with a series of DRAWTEXT statements:

```
layout lattice / rows=2 rowweights=(0.45 0.55) rowgutter=20px;
   layout lattice / rows=1 columns=5 columnweights=(0.3 .175 .175 .175 .175) columngutter=0px;
     %loopit;
     drawtext textattrs=(weight=bold) "Number of Patients Experiencing Infrequent Bowel Movements" /
           x=1 y=90 drawspace=graphpercent anchor=left width=25;
     drawtext textattrs=(color=darkorchid) "IMP-101 5 mg QD" / x=33 y=96 drawspace=graphpercent
           anchor=center width=100;
     drawtext textattrs=(color=darkorchid) "(N=&BIGN1)" / x=33 y=92.5 drawspace=graphpercent
           anchor=center width=100;
     drawtext textattrs=(color=darkorchid) "n (%)" / x=33 y=89 drawspace=graphpercent anchor=center
           width=100;
     drawtext textattrs=(color=firebrick) "IMP-101 10 mg QD" / x=50 y=96 drawspace=graphpercent
           anchor=center width=100;
     drawtext textattrs=(color=firebrick) "(N=&BIGN2)" / x=50 y=92.5 drawspace=graphpercent
           anchor=center width=100;
     drawtext textattrs=(color=firebrick) "n (%)" / x=50 y=89 drawspace=graphpercent anchor=center
           width=100;
     drawtext textattrs=(color=darkgoldenrod) "Placebo" / x=67 y=96 drawspace=graphpercent
           anchor=center width=100;
     drawtext textattrs=(color=darkgoldenrod) "(N=&BIGN3)" / x=67 y=92.5 drawspace=graphpercent
           anchor=center width=100;
     drawtext textattrs=(color=darkgoldenrod) "n (%)" / x=67 y=89 drawspace=graphpercent
           anchor=center width=100;
     drawtext "Total" / x=84 y=96 drawspace=graphpercent anchor=center width=100;
     drawtext "(N=&BIGN4)" / x=84 y=92.5 drawspace=graphpercent anchor=center width=100;
     drawtext "n (%)" / x=84 y=89 drawspace=graphpercent anchor=center width=100;
     drawline x1=27 x2=99 y1=99 y2=99 / drawspace=graphpercent lineattrs=(color=black pattern=1);
     drawline x1=27 x2=99 y1=86 y2=86 / drawspace=graphpercent lineattrs=(color=black pattern=1);
  endlayout;
```

Notes:

- Our Big N values were previously sent into macro variables.
- The number of annotations seems daunting, however there are ways of determining the positions efficiently. This, however, is not within the scope of this paper.
- We are able to color-code elements of the table – a feature not readily available in standard tables.
- An alternative method to the annotations will be presented in a later example using R.
- Since our table code appears first in our GTL block, the table portion of the output will appear on top.

To create the bar chart at the bottom of our combined output, we have:

```
layout overlay / xaxisopts=(label=" " linearopts=(viewmin=1 viewmax=7 tickvaluesequence=(start=1
                           end=7 increment=1)))
            yaxisopts=(label="Number of Patients!Experiencing Infrequent!Bowel Movements"
                       labelfitpolicy=splitalways labelsplitchar="!"
                       linearopts=(integer=true thresholdmax=1))
            walldisplay=(fill);
   needleplot x=XVAR_F y=YVAR_F / group=ATTR_TRT groupdisplay=cluster clusterwidth=0.3
         lineattrs=(pattern=1 thickness=10);
   discretelegend "lgnd1" "lgnd2" "lgnd3" / down=1 border=false;
endlayout;
```

Notes:

- We utilize our previously defined attribute map and legend items in this block.
- NEEDLEPLOT was used to construct the bar chart, however it could have just as easily been done with BARCHART or BARCHARTPARM statements; this will be left as an exercise for the reader to investigate.

Finally, we close out our GTL block:

```
        endlayout;
      endgraph;
    end;
run;
```

Run PROC SGRENDER to generate the output given in Figure 1.

EXAMPLE 2: COMBINED LISTING-FIGURE OUTPUT - SAS

Figure 6 presents a Composite TLF combining a subject-level listing with a corresponding graphical summary. This graphical summary is presented on each page and is the same aggregate plot across pages. If preferred, the figure could also correspond to each page individually. The listing provides traceability, while the figure highlights overall trends. Note the color-coded treatment header. Conditional formatting within the listing is used to highlight subjects meeting predefined criteria, as given by the green text below.

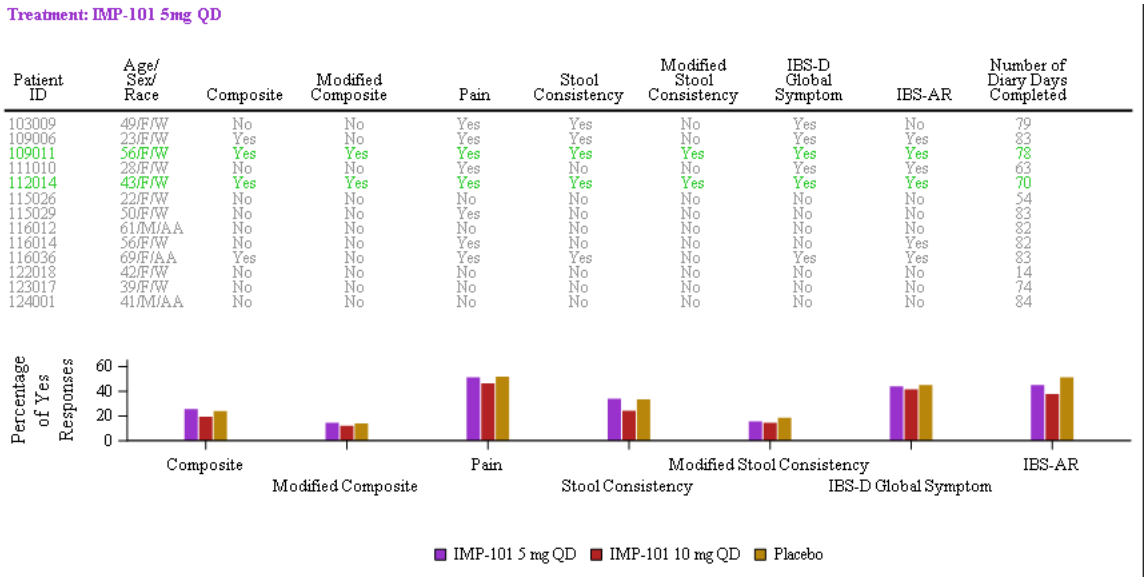


Figure 6. Composite Listing–Figure Output Combining Subject-Level Data and Graphical Summary.

As we did with the Table-Figure, let's investigate the process used to generate the Listing-Figure, beginning with our data. As prior, we will be starting from a common data set, generating a "listing data set" and a "figure data set" from our common source, then merging the resulting two together. Assume that, beginning from ADaM, we derive the following:

TRT01PN	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9
1	103009	49/F/W	No	No	Yes	Yes	No	Yes	No
1	109006	23/F/W	Yes	No	Yes	Yes	No	Yes	Yes
1	109011	56/F/W	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1	111010	28/F/W	No	No	Yes	No	No	Yes	Yes
1	112014	43/F/W	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1	115026	22/F/W	No	No	No	No	No	No	No
1	115029	50/F/W	No	No	Yes	No	No	No	No
1	116012	61/M/AA	No	No	No	No	No	No	No
1	116014	56/F/W	No	No	Yes	No	No	No	Yes
1	116036	69/F/AA	Yes	No	Yes	Yes	No	Yes	Yes
1	122018	42/F/W	No	No	No	No	No	No	No
1	123017	39/F/W	No	No	No	No	No	No	No
1	124001	41/M/AA	No	No	No	No	No	No	No
1	125002	29/F/AA	No	No	No	No	No	No	No
1	125046	61/F/W	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1	125047	63/F/W	No	No	No	No	No	No	Yes
1	125057	65/F/AA	No	No	No	No	No	No	No
1	130013	49/F/W	Yes	No	Yes	Yes	No	Yes	Yes
1	131022	63/M/W	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1	132003	37/M/W	No	No	No	No	No	No	No

Figure 7. Starting Data Set for Listing-Figure Output

Some notes about this data set:

- Our listing will be ordered by treatment and subject and will include an Age/Sex/Race variable.
- The rest of the variables have answers of Yes/No only. Percentages will be calculated for each variable to create the figure data set a little later.

After some manipulations, we arrive at our listing data set:

TRT01PN	COL1	COL2	COL10	COL3	COL4	COL5	COL6	COL7	COL8	COL9	XVAR
1	103009	49/F/W	79	No	No	Yes	Yes	No	Yes	No	0
1	109006	23/F/W	83	Yes	No	Yes	Yes	No	Yes	Yes	0
1	109011	56/F/W	78	Yes	Yes	Yes	Yes	Yes	Yes	Yes	0
1	111010	28/F/W	63	No	No	Yes	No	No	Yes	Yes	0
1	112014	43/F/W	70	Yes	Yes	Yes	Yes	Yes	Yes	Yes	0
1	115026	22/F/W	54	No	No	No	No	No	No	No	0
1	115029	50/F/W	83	No	No	Yes	No	No	No	No	0
1	116012	61/M/AA	82	No	No	No	No	No	No	No	0
1	116014	56/F/W	82	No	No	Yes	No	No	No	Yes	0
1	116036	69/F/AA	83	Yes	No	Yes	Yes	No	Yes	Yes	0
1	122018	42/F/W	14	No	No	No	No	No	No	No	0
1	123017	39/F/W	74	No	No	No	No	No	No	No	0
1	124001	41/M/AA	84	No	No	No	No	No	No	No	0

YVAR	PAGENUM	COLORFL
0	1	
1	1	
2	1	1
3	1	
4	1	1
5	1	
6	1	
7	1	
8	1	
9	1	
10	1	
11	1	
12	1	
0	2	
1	2	1
2	2	
3	2	
4	2	
5	2	1

Figure 8. Listing Data Set

Let's look at a few notes on this data set:

- Like the table data set, the variables XVAR and YVAR are for positioning the columns on and across pages.
- XVAR retains a value of zero across all observations.
- We wish to have 13 subjects on each page, thus YVAR ranges from 0 to 12. Note the corresponding PAGENUM variable that indexes along with YVAR.
- The COLORFL variable flags observations where each of COL3 to COL9 have a value of "Yes". This will be used to give these observations the green text seen in the output.

Next we perform the necessary manipulations to arrive at our figure data set:

XVAR_F	YVAR_F	TRT
1	25.68807339449...	1
2	14.67889908256...	1
3	51.37614678899...	1
4	33.94495412844...	1
5	15.59633027522...	1
6	44.03669724770...	1
7	44.954128440367	1
1	19.51219512195...	2
2	12.19512195121...	2
3	46.34146341463...	2
4	24.390243902439	2
5	14.63414634146...	2
6	41.46341463414...	2
7	37.80487804878...	2
1	24.031007751938	3
2	13.953488372093	3
3	51.937984496124	3

Figure 9. Figure Data Set

Notes:

- XVAR_F indicates the respective column COL3 to COL9. A format will display the values as seen in the output.
- YVAR_F is the percentage of "Yes" responses.
- This data set will need to be replicated once for each page number that we have, with all of these then being merged onto our listing data set in order to display the figure on each page.

Once we have our combined listing-figure data set, we can begin looking at the code to plot. First, we need to define some macro variables to be used by annotating our header:

```
%let H1_1=;                  %let H2_1=Patient;           %let H3_1=ID;              %let xpos1=4;
%let H1_2=%str(Age/);        %let H2_2=%str(Sex/);        %let H3_2=Race;            %let xpos2=13;
%let H1_3=;                  %let H2_3=;                  %let H3_3=Composite;       %let xpos3=22;
%let H1_4=;                  %let H2_4=Modified;          %let H3_4=Composite;       %let xpos4=31;
%let H1_5=;                  %let H2_5=;                  %let H3_5=Pain;            %let xpos5=42;
%let H1_6=;                  %let H2_6=Stool;             %let H3_6=Consistency;     %let xpos6=51;
%let H1_7=Modified;          %let H2_7=Stool;             %let H3_7=Consistency;     %let xpos7=61;
%let H1_8=%str(IBS-D);       %let H2_8=Global;            %let H3_8=Symptom;         %let xpos8=71;
%let H1_9=;                  %let H2_9=;                  %let H3_9=%str(IBS-AR);    %let xpos9=81;
%let H1_10=%str(Number of);  %let H2_10=%str(Diary Days); %let H3_10=Completed;     %let xpos10=90;
```

Some notes:

- Our headers allow for up to 3 lines of text. The H1_XXX, H2_XXX, H3_XXX variables correspond to the first, second, and third lines respectively.
- The XPOS variables indicated horizontal positioning values.
- While this looks like a manual process, methodology outside the scope of this paper allows us to obtain these values efficiently.

As we will be looping through each page in order to build the entire listing, we'll need to generate some additional macro variables, using PROC SQL:

```
proc sql;
   select distinct TRT01PN,PAGENUM,COLOR
   into :TRT1-,:PAGE1-,:C1-
   from here.lf16020602
   where not missing(TRT01PN)
   order by TRT01PN,PAGENUM;
quit;
```

We will follow a similar methodology to what we did for our Table-Figure output. The structure of our GTL code is as follows:

- A LAYOUT LATTICE block with two cells – the top cell generating the listing portion of the output and the bottom cell generating the figure portion of the output.
- The listing portion of the output consists of another LAYOUT LATTICE block consisting of 10 side-by-side cells – one cell for each column of the table – and a series of annotations for the column headers.
- The figure portion of the output consists of a LAYOUT OVERLAY block containing a needle plot to create the bars, and our legend.
- We will create one image per page and loop through them to construct the full listing.

Let's first define a short macro to generate each of the ten columns of our listing:

```
%macro loopit;
%do i=1 %to 10;
      layout overlay / xaxisopts=(display=none)
                       yaxisopts=(display=none reverse=true linearopts=(viewmax=13))
                  walldisplay=(fill) pad=(top=80px);
        scatterplot x=XVAR y=YVAR / markerattrs=(color=white);
        innermargin / align=left;
            axistable value=COL&i y=YVAR / showmissing=true display=(values) valueattrs=(size=9)
                colorgroup=ATTR_TXT;
        endinnermargin;
        drawtext "&&H1_&i" / x=&&XPOS&i y=-4 xspace=graphpercent yspace=datavalue anchor=center width=100;
        drawtext "&&H2_&i" / x=&&XPOS&i y=-3 xspace=graphpercent yspace=datavalue anchor=center width=100;
        drawtext "&&H3_&i" / x=&&XPOS&i y=-2 xspace=graphpercent yspace=datavalue anchor=center width=100;
     endlayout;
%end;
%mend;
```

The DRAWTEXT statements will generate the headers. Next, we will write a macro that loops through the template and rendering for each page. Let's look at the first part of this macro:

```
%macro plotit;
%do j=1 %to &NPAGE;
proc template;
   define statgraph lf&j;
      begingraph;
        legenditem name="lgnd1" type=fill / label="IMP-101 5 mg QD" fillattrs=(color=darkorchid);
        legenditem name="lgnd2" type=fill / label="IMP-101 10 mg QD" fillattrs=(color=firebrick);
        legenditem name="lgnd3" type=fill / label="Placebo" fillattrs=(color=darkgoldenrod);
        DiscreteAttrVar attrvar=ATTR_TRT var=TRT attrmap="myattrmap";
        DiscreteAttrMap name="myattrmap";
           value "1" / lineattrs=(color=darkorchid);
           value "2" / lineattrs=(color=firebrick);
           value "3" / lineattrs=(color=darkgoldenrod);
        enddiscreteattrmap;
        DiscreteAttrVar attrvar=ATTR_TXT var=COLORFL attrmap="myattrmap2";
        DiscreteAttrMap name="myattrmap2";
           value "1" / textattrs=(color=limegreen);
        enddiscreteattrmap;
```

Notes:

- NPAGE is a macro variable previously generated dynamically that contains the total number of pages that we will have.
- Our indexing begins with “j” and not “i”, as our previously defined macro for looping through columns uses the latter as an index.
- We have two separate attribute maps – one for our treatment groups and the other for our color flag that highlights the rows of the listing.

Next, our outer LAYOUT LATTICE block:

```
layout lattice / rows=2 rowweights=(0.6 0.4) rowgutter=20px;
```

Due to the use of our first macro, the listing block of our GTL code looks rather simple:

```
layout lattice / rows=1 columns=10 columnweights=preferred columngutter=0px;
   %loopit;
endlayout;
```

Our figure block looks very similar to our Table-Figure output:

```
layout overlay / xaxisopts=(label=" " linearopts=(viewmin=1 viewmax=7 tickvaluesequence=(start=1
                          end=7 increment=1) tickvaluefitpolicy=stagger))
                 yaxisopts=(label="Percentage!of Yes!Responses"
                      labelfitpolicy=splitalways labelsplitchar="!"
                      linearopts=(integer=true thresholdmax=1))
                 walldisplay=(fill);
    needleplot x=XVAR_F y=YVAR_F / group=ATTR_TRT groupdisplay=cluster clusterwidth=0.3
       lineattrs=(pattern=1 thickness=10);
    discretelegend "lgnd1" "lgnd2" "lgnd3" / down=1 border=false;
 endlayout;
```

We wrap up our template with a couple of additional annotations and then close up our blocks:

```
           drawtext textattrs=(weight=bold color=&&C&j) "Treatment: &&TRT&j" / x=1 y=98 drawspace=graphpercent
              anchor=left width=100;
           drawline x1=1 x2=99 y1=81.5 y2=81.5 / lineattrs=(pattern=1 color=black) drawspace=graphpercent;
        endlayout;
     endgraph;
  end;
run;
```

Finally, we generate an output by running PROC SGRENDER, looping through the pages, and calling the final macro:

```
ods graphics / noborder height=4.5in width=9in;
proc sgrender data=here.lf16020602 template=lf&j;
   where PAGENUM=&j;
   format XVAR_F xf.;
run;
%end;
%mend;

%plotit;
```

EXAMPLE 3: COMBINED TABLE-FIGURE OUTPUT - R

As mentioned earlier, this approach is software-agnostic. We will illustrate this by producing the output in Figure 1 using R. R has a small advantage in that it can process multiple data sets simultaneously via the PATCHWORK package, as we shall see. We can begin with two separate data sets, one for our table portion and one for our figure portion. Let's look at the table data set first.

COL	XVAR	YVAR	COLOR
n (%)	2	0	1
N=(109)	2	-0.5	1
IMP-101 5 mg QD	2	-1	1
n (%)	3	0	2
N=(82)	3	-0.5	2
IMP-101 10 mg QD	3	-1	2
n (%)	4	0	3
N=(129)	4	-0.5	3
Placebo	4	-1	3
n (%)	5	0	.
N=(320)	5	-0.5	.
Total	5	-1	.
Events	1	-0.5	4
Number of	1	-1	4
	1	0	.
0 Events	1	1	.
1 Event	1	2	.
2 Events	1	3	.
3 Events	1	4	.
4 Events	1	5	.
5 Events	1	6	.
>5 Events	1	7	.
	2	0	.
11 (10.1)	2	1	.
10 (9.2)	2	2	.
13 (11.9)	2	3	.
7 (6.4)	2	4	.
10 (9.2)	2	5	.

Figure 10. Table Data Set

Here are some notes:

- This is merely a snip of the first few rows of our table data set. Each value in each column of our table will be represented by a value of the COL variable.
- XVAR and YVAR, again, are positioning variables.
- The COLOR variable will be used to color-code our column headers.
- In the SAS version, we utilized a large amount of annotations to build our table. However, in this example, we will be using a different approach by having all elements of the table contained in the data set and use a text plot to produce the image. It will be left as an exercise for the reader to use the same methodology in SAS.

Let's now look at our figure data set:

XVAR_F	TRT	YVAR_F
1	N_1	11
1	N_2	12
1	N_3	8
2	N_1	10
2	N_2	8
2	N_3	11
3	N_1	13
3	N_2	10
3	N_3	7
4	N_1	7
4	N_2	9
4	N_3	9
5	N_1	10
5	N_2	7
5	N_3	10
6	N_1	10
6	N_2	10
6	N_3	6
7	N_1	6
7	N_2	15

Figure 11. Figure Data Set

Notes:

- XVAR_F is our x-axis variable for the bar chart.
- YVAR_F represents our counts.
- TRT represents our treatments.

With our two data sets in mind, let's now shift to the R code that generates each. We begin by reading in the necessary packages and data sets, and defining our color scheme for our text elements:

```
library(ggplot2)
library(grid)
library(patchwork)
library(dplyr)
library(tidyr)
library(haven)

# Sample data
data <- read_sas(data_file = "XXX/table.sas7bdat")
data2 <- read_sas(data_file = "XXX/figure1.sas7bdat")

# Define color mapping
color_mapping <- c("1" = "darkorchid", "2" = "firebrick", "3" = "darkgoldenrod", "4" = "black")
```

The next piece of code generates the table portion of our output:

```
# Create the plot
table <- ggplot(data, aes(x = XVAR, y = YVAR, label = COL, color = factor(COLOR))) +
  geom_text(size = 2.5, show.legend = FALSE) +
  scale_color_manual(values = color_mapping) +
  scale_x_continuous(limits = c(1, 5), breaks = 1:5) +
  scale_y_reverse(limits = c(7, -2), breaks = -2:7) +
  geom_hline(yintercept = 0.6, linetype = "solid", color = "black") +
  geom_hline(yintercept = -1.5, linetype = "solid", color = "black") +
  theme(panel.border=element_blank(),
        axis.text=element_blank(),
        axis.title=element_blank(),
        axis.ticks=element_blank(),
        panel.background = element_rect(fill = "white"))
```

Note that GEOM_TEXT is doing the majority of the heavy lifting for this portion.

Now we generate the figure portion of our output:

```
# Define colors
colors2 <- c("N_1" = "darkorchid", "N_2" = "firebrick", "N_3" = "darkgoldenrod")

# Create the plot
figure <- ggplot(data2, aes(x = XVAR_F, y = YVAR_F, fill = TRT)) +
  #geom_segment(aes(xend = XVAR_F, yend = 0), size = 10) +
  geom_bar(stat="identity", position="dodge")+
  scale_fill_manual(name="Treatment", values = colors2, labels = c("IMP-101 5 mg QD", "IMP-101 10 mg QD", "Placebo")) +
  scale_x_continuous(breaks = 1:7, limits = c(0, 8), label = c("0 Events","1 Event","2 Events","3 Events","4 Events","5 Events",">5 Events")) +
  scale_y_continuous(breaks = seq(0, 20, by = 5)) +
  labs(x = " ", y = "Number of Patients\nExperiencing Infrequent\nBowel Movements") +
  theme(panel.border=element_blank(),
        panel.grid.major= element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"),
        axis.text=element_text(color="black", size=12),
        axis.title=element_text(size=12),
        legend.text=element_text(size=10),
        legend.position="bottom",
        axis.text.x = element_text(angle = 0, hjust = 0.5,
                                   vjust = 0),
        panel.background = element_rect(fill = "white"))
```

Finally, we put them both together using PATCHWORK and save the image file:

```
TableFigure <- table / figure

ggsave("TableFigure.png", TableFigure, width = 9, height = 4.5)
```

Our combined table-figure output will then appear like so:

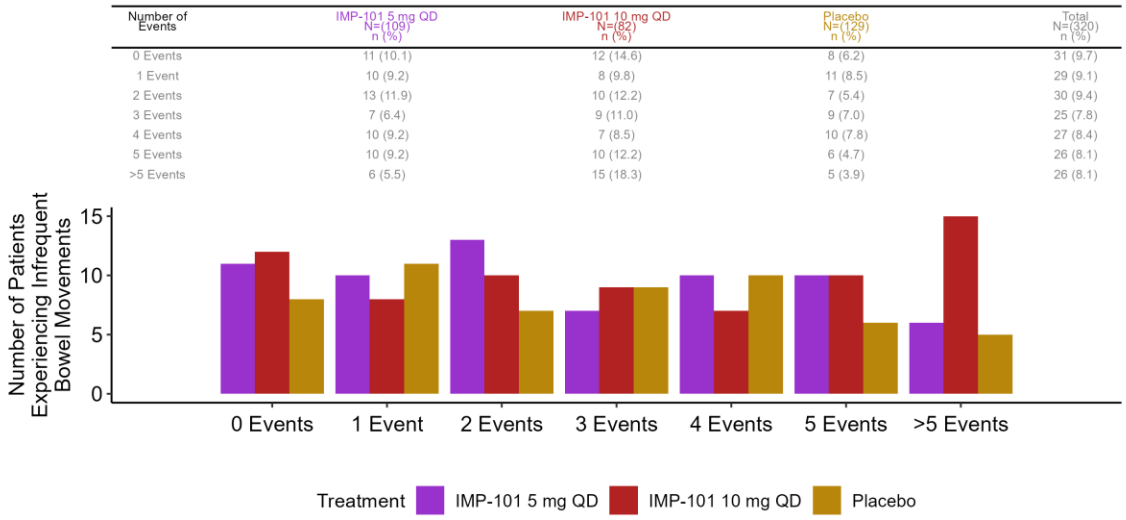


Figure 12. Combined Table-Figure Output – R

ADDITIONAL EXAMPLES

While the above examples were kept moderately simple in order to highlight the methodology, these outputs are highly customizable. Here are a few examples (code not included) that further illustrate the power of this technique:

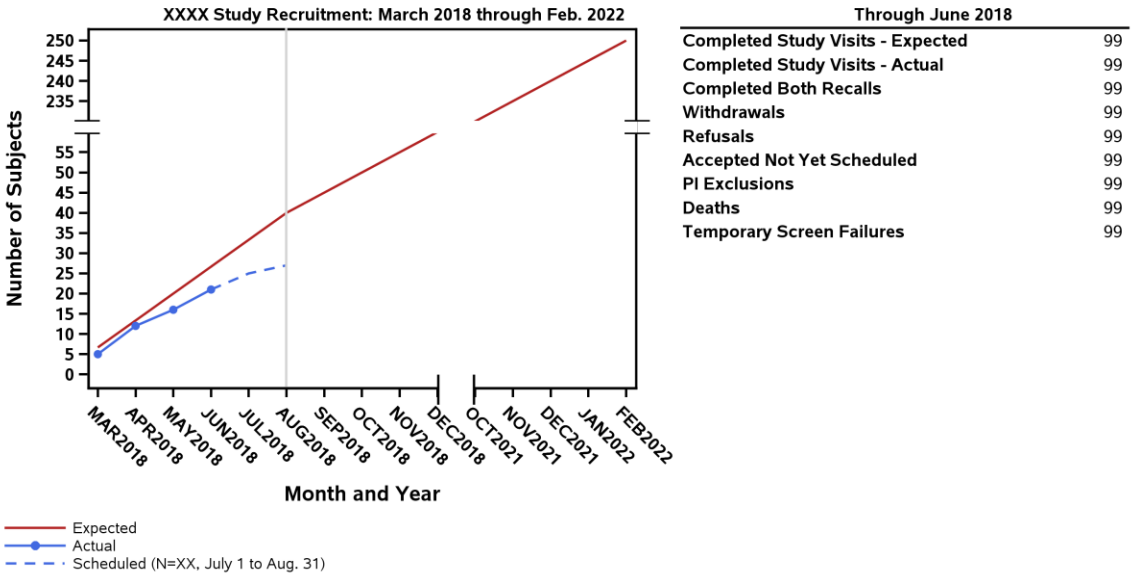


Figure 13. Combined Table-Figure Showing Recruitment Data

	Placebo (N=534)	Treatment (N=538)	Hazard Ratio (95% CI)	P-Value
Event-Free Survival (days)			1.02 (0.853, 1.210)	0.861
Number of Subjects with Events n (%)	263 (49)	244 (45)		
Mild	134 (25)	116 (22)		
Moderate	81 (15)	82 (15)		
Severe	48 (9)	46 (9)		
Number of Subjects Censored n (%)	271 (51)	294 (55)		
Mild	126 (24)	129 (24)		
Moderate	78 (15)	108 (20)		
Severe	67 (13)	57 (11)		
25th Percentile (95% CI)	669.0 (668.00, 669.00)	669.0 (668.00, 669.00)		
Median (95% CI)	672.0 (671.00, 673.00)	673.0 (672.00, 673.00)		
75th Percentile (95% CI)	680.0 (677.00, 682.00)	680.0 (677.00, 686.00)		
Kaplan-Meier Estimate (95% CI)				
1 Month	0.99 (0.973, 0.994) [n = 518]	0.99 (0.978, 0.996) [n = 525]		
2 Months	0.98 (0.968, 0.991) [n = 515]	0.99 (0.975, 0.995) [n = 523]		
6 Months	0.95 (0.923, 0.962) [n = 474]	0.96 (0.939, 0.974) [n = 486]		
12 Months	0.92 (0.897, 0.944) [n = 443]	0.93 (0.904, 0.949) [n = 451]		
18 Months	0.90 (0.873, 0.925) [n = 418]	0.91 (0.880, 0.931) [n = 424]		
24 Months	0.06 (0.019, 0.136) [n = 1]	0.00 (0.000, 0.000) [n = 0]		

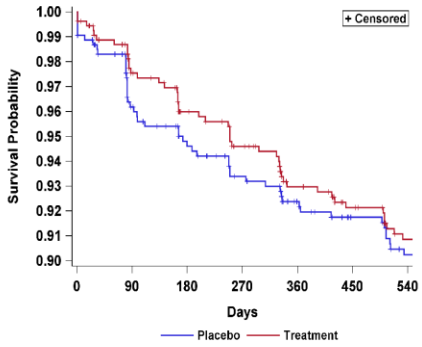


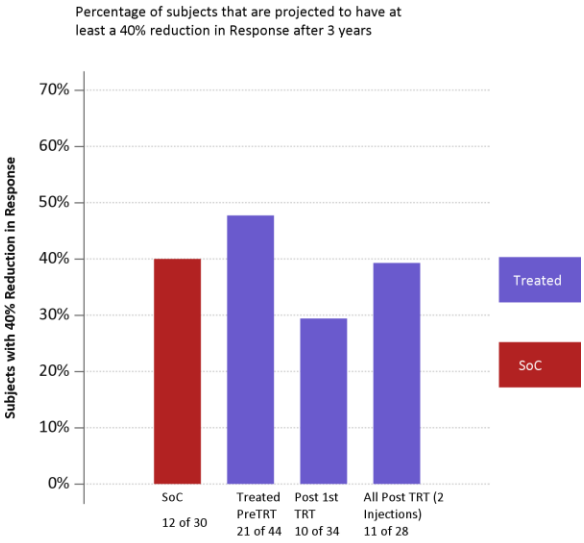
Figure 14. Table-Figure of Survival Analysis with Kaplan-Meier Plot

Response: Projected Percent Change

SUITABLE LOGO

SoC		Treated	Treated PreTRT	Post 1st TRT	All Post TRT (2 Injections)
ID-1-1*	-28%	ID-1-1*	-28%	89%	-47%
ID-1-2*	-56%	ID-1-2*	-56%	-36%	28%
ID-1-3	-97%	ID-1-4*	-9%	-107%	-95%
ID-1-4*	-9%	ID-1-5	-120%	40%	72%
ID-2-2*	-99%	ID-10-1	-243%	37%	-13%
ID-2-3	-159%	ID-10-3	-121%		
ID-3-2*	-55%	ID-11-1	-101%	-116%	-93%
ID-3-4*	-92%	ID-13-1	-56%		
ID-4-1*	4%	ID-14-6	-163%		
ID-4-4*	-6%	ID-15-2	24%		
ID-5-3*	-22%	ID-15-3	41%		
ID-6-3	-28%	ID-2-1	241%	-64%	-63%
ID-7-2*	-29%	ID-2-2*	-99%	-206%	-51%
ID-7-3	-63%	ID-3-1	217%	283%	-78%
ID-8-1	11%	ID-3-2*	-55%	25%	-14%
ID-9-1	-58%	ID-3-3	-44%	4%	-48%
ID-9-4	81%	ID-3-4*	-92%		
ID-9-7	-156%	ID-4-1*	4%		
ID-9-8*	-2%	ID-4-2	-76%	528%	
ID-9-9*	-30%	ID-4-3	39%	57%	7%
ID-9-10*	-46%	ID-4-4*	-6%		
ID-9-14*	-36%	ID-4-5	291%	13%	52%
	40% 12 of 30		48% 21 of 44	29% 10 of 34	39% 11 of 28

% of Subjects having at least 40% reduction in Response



Red: Subjects projected to have at least 40% reduction in Response
Green: Subjects not projected to have at least 40% reduction in Response

Figure 15. Listing-Figure Displaying Multiple Outcomes in Listing and Figure Format

CONCLUSION

Composite TLFs represent a pragmatic evolution of traditional clinical reporting. By integrating tables, listings, and figures into single cohesive outputs, this approach reduces reviewer burden, improves interpretability, and streamlines production workflows.

Demonstrated in both SAS and R, Composite TLFs are software-agnostic and scalable across clinical domains.

CONTACT INFORMATION

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

Jesse Pratt
jesse.pratt@ppd.com

Rayce Wiggins
rayce.wiggins@ppd.com