PharmaSUG China 2026 - Paper DV-159

The reporter package: A Powerful Reporting Package to Replicate SAS® Output

Bill Huang

ABSTRACT

With the increasing application of the R programming language in clinical trial research, a primary objective is to successfully replicate SAS output results. The "reporter" package represents a comprehensive and user-friendly reporting solution, offering capabilities comparable to SAS's PROC REPORT/ODS. This package supports multiple output formats including TXT, PDF, RTF, DOCX, and HTML—while efficiently handling large-scale data. Leading pharmaceutical companies, such as Amgen, have adopted "reporter" as a key component of their reporting workflows. This paper provides a systematic introduction to the fundamental usage of "reporter", covering applications from basic to advanced levels. Furthermore, it demonstrates how "reporter" can generate outputs nearly identical to SAS through concise code, illustrated by standard clinical trial tables and listings. Finally, this paper highlights useful features such as break labels and group cohesion, and outlines the future development roadmap, including RTF code insertion.

INTRODUCTION

The **reporter** R package was designed to give you similar capabilities as PROC REPORT and ODS, which allows you to reproduce SAS reports with a high degree of fidelity. The package lets you:

- Generate tables, figures, and listings
- Write in RTF, PDF, TXT, DOCX and HTML file formats
- Has many options for placement and justification of titles and footnotes
- Supports page break, page wrap, and page by functionality
- Allows mixing of figures, text, and tables on the same page

The purpose of this paper is to introduce you to the most basic features of the package. Full documentation is available at https://reporter.r-sassy.org/index.html.

SAMPLE DATA

The following sample data will be used for the example below:

```
# Create sample data
dat <- read.table(header = TRUE, text = '
VAR      LABEL            A             B
Age          N            9            10
Age    MEANSTD  "13.3 (1.00)" "13.3 (1.89)"
Age     MEDIAN           13            13
Age       Q1Q3    13.0-14.0     12.0-15.0
Age     MINMAX        12-15         11-16
Height       N            9            10
Height MEANSTD  "62.1 (3.88)" "62.5 (6.26)"
Height  MEDIAN        62.50         64.55
Height    Q1Q3    59.8-63.5     57.5-66.5
Height  MINMAX    56.5-69.0     51.3-72.0
Weight       N            9            10
Weight MEANSTD "95.9 (12.36)" "103.7 (29.49)"
Weight  MEDIAN        98.00        105.75
Weight    Q1Q3   84.0-102.5    85.0-128.0
Weight  MINMAX   83.0-112.5    50.5-150.0
```

```
Sex          N            9            10
Sex          F    "5 (55.6%)"  "4 (40.0%)"
Sex          M    "4 (44.4%)"  "6 (60.0%)"')
```

SIMPLE REPORRT

Creating a report with the **reporter** package involves three steps:

- Create report content (a table, text, or plot)
- Create a report object and attach the content
- Write out the report in the desired output format

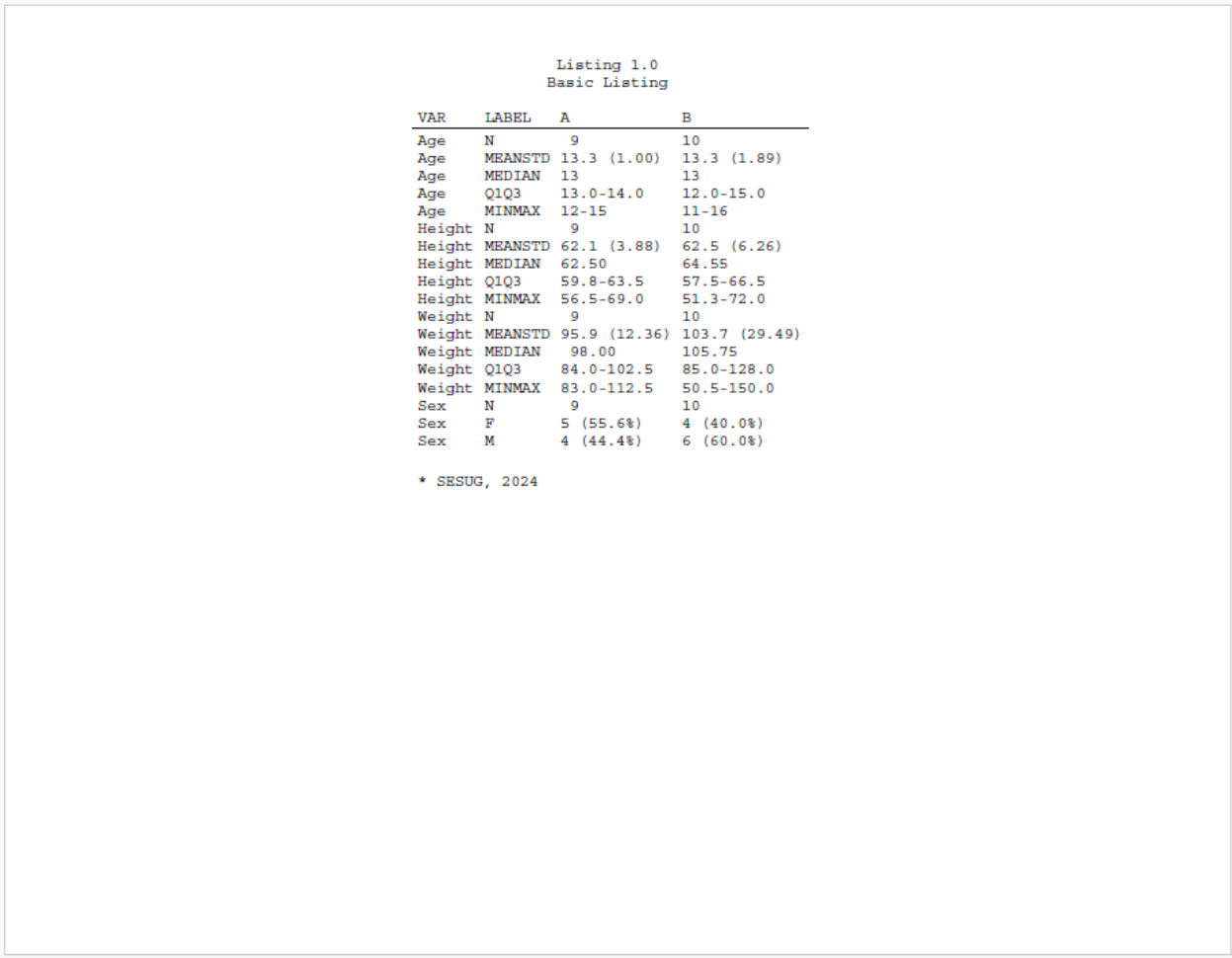
Here is an example that shows how to create a minimal report using the sample data from above:

```
library(reporter)

# Create table
tbl <- create_table(dat) |>
  titles("Listing 1.0", "Basic Listing") |>
  footnotes("* SESUG, 2024")

# Create the report
rpt <- create_report("./report/example1",
                     font = "Courier",
                     output_type = "PDF") |>
  add_content(tbl)

# Write the report
write_report(rpt)
```

In the above example, notice the following:

- The input data is passed to the `create_table()` function. Any calculations should be done prior to reporting. The reporting functions will not perform any calculations.
- The table object `tbl` is then added to the report object `rpt` via the `add_content()` function. You can add one or more pieces of content to a report.
- You can change to a different file format by changing the value passed to the `output_type` parameter of `create_report()`.
- The `write_report()` function renders the report in the desired output type and writes it to the file system. The file format can also be specified on the `output_type` parameter of `write_report()`.

The above code will produce the following PDF report. Observe that the columns and column widths are all determined automatically based on the data. Also, the orientation, font size, and margins all have sensible defaults.



Listing 1.0
Basic Listing

VAR	LABEL	A	B
Age	N	9	10
Age	MEANSTD	13.3 (1.00)	13.3 (1.89)
Age	MEDIAN	13	13
Age	Q1Q3	13.0-14.0	12.0-15.0
Age	MINMAX	12-15	11-16
Height	N	9	10
Height	MEANSTD	62.1 (3.88)	62.5 (6.26)
Height	MEDIAN	62.50	64.55
Height	Q1Q3	59.8-63.5	57.5-66.5
Height	MINMAX	56.5-69.0	51.3-72.0
Weight	N	9	10
Weight	MEANSTD	95.9 (12.36)	103.7 (29.49)
Weight	MEDIAN	98.00	105.75
Weight	Q1Q3	84.0-102.5	85.0-128.0
Weight	MINMAX	83.0-112.5	50.5-150.0
Sex	N	9	10
Sex	F	5 (55.6%)	4 (40.0%)
Sex	M	4 (44.4%)	6 (60.0%)

* SESUG, 2024

Figure 1: Simple Report Example

REPLICATE SAS OUTPUTS FOR CLINICAL TRIALS

Now, let's replicate SAS outputs with the **reporter** package for clinical trials. There are some common layouts in clinical trials. This paper illustrates how to replicate a demographics table with a hierarchy column and a shift table with spanning headers.

DEMOGRAPHICS TABLE WITH A HIERARCHY COLUMN

The following demographics table is generated from SAS.

This layout contains:

- A hierarchy column containing two-layer information. For example, group "Sex" contains smaller groups "Male" and "Female".
- Blank lines between each group.

Table 14-2.1. Baseline Demographics
(Safety Analysis Set)

	Treatment A (N = 145)	Treatment B (N = 153)	Total (N = 298)
Sex - n (%)			
Male	75 (51.7)	91 (59.5)	166 (55.7)
Female	70 (48.3)	62 (40.5)	132 (44.3)
Ethnicity - n (%)			
Hispanic/Latino	0 (0.0)	2 (1.3)	2 (0.7)
Not Hispanic/Latino	143 (98.6)	145 (94.8)	288 (96.6)
Unknown/Missing	2 (1.4)	6 (3.9)	8 (2.7)
Race - n (%)			
Asian	18 (12.4)	10 (6.5)	28 (9.4)
Black (or African American)	0 (0.0)	1 (0.7)	1 (0.3)
White	123 (84.8)	130 (85.0)	253 (84.9)
Other	1 (0.7)	6 (3.9)	7 (2.3)
Age (years)			
n	145	153	298
Mean	64.7	65.8	65.3
SD	9.7	8.3	9.0
Median	65.0	66.0	66.0
Q1, Q3	59.0, 73.0	60.0, 72.0	59.0, 72.0
Min, Max	40, 83	43, 85	40, 85
Age group - n (%)			
18 - 64 years	71 (49.0)	62 (40.5)	133 (44.6)
65 - 74 years	49 (33.8)	67 (43.8)	116 (38.9)
75 - 84 years	25 (17.2)	23 (15.0)	48 (16.1)
≥ 85 years	0 (0.0)	1 (0.7)	1 (0.3)

Page 1 of 1

Program: /dummy/tables/t-dm-sum-fas.sas
Output: t14-02-001-dm-sum-fas.rtf (Date Generated: 04NOV24:23:21:00) Source: adsl

Figure 2: Demographics table from SAS

Now, let's replicate this table with the **reporter** package. The input data will be like:

	item_group	item	TRT01AN_1	TRT01AN_2	TRT01AN_3
1	Sex - n (%)	Male	75 (51.7)	91 (59.5)	166 (55.7)
2	Sex - n (%)	Female	70 (48.3)	62 (40.5)	132 (44.3)
3	Ethnicity - n (%)	Hispanic/Latino	0 (0.0)	2 (1.3)	2 (0.7)
4	Ethnicity - n (%)	Not Hispanic/Latino	143 (98.6)	145 (94.8)	288 (96.6)
5	Ethnicity - n (%)	Unknown/Missing	2 (1.4)	6 (3.9)	8 (2.7)
6	Race - n (%)	Asian	18 (12.4)	10 (6.5)	28 (9.4)
7	Race - n (%)	Black (or African American)	0 (0.0)	1 (0.7)	1 (0.3)
8	Race - n (%)	White	123 (84.8)	130 (85.0)	253 (84.9)
9	Race - n (%)	Other	1 (0.7)	6 (3.9)	7 (2.3)
10	Age (years)	n	145	153	298
11	Age (years)	Mean	64.7	65.8	65.3
12	Age (years)	SD	9.7	8.3	9.0
13	Age (years)	Median	65.0	66.0	66.0
14	Age (years)	Q1, Q3	59.0, 73.0	60.0, 72.0	59.0, 72.0
15	Age (years)	Min, Max	40, 83	43, 85	40, 85
16	Age group - n (%)	18 - 64 years	71 (49.0)	62 (40.5)	133 (44.6)
17	Age group - n (%)	65 - 74 years	49 (33.8)	67 (43.8)	116 (38.9)
18	Age group - n (%)	75 - 84 years	25 (17.2)	23 (15.0)	48 (16.1)
19	Age group - n (%)	≥ 85 years	0 (0.0)	1 (0.7)	1 (0.3)

Figure 3: Input Data for Demographics Table

Here is the code that shows how to create this demographics table:

```
# Create table content
tbl <- create_table(output_data,
                    borders = "outside") |>
  # ----- Combine two columns for hierarchy display  ----- #
  stub(c("item_group", "item")) |>
  # ----- Set row label for stub column ----- #
  define(item_group, label_row = T, blank_before = T) |>
  # ----- Basic settings for columns ----- #
  define(item, indent = .25, width = item_width) |>
  define(TRT01AN_1, label = trt_label_1, width =  TRT01AN_3,
         align = "center", n = big_n_1) |>
  define(TRT01AN_2, label = trt_label_2, width =  TRT01AN_3,
         align = "center", n = big_n_2) |>
  define(TRT01AN_3, label = trt_label_3, width =  TRT01AN_3,
         align = "center", n = big_n_3)|>
  # ----- Titles and Footnotes ----- #
  titles("Table 14-2.1.  Baseline Demographics",
         "(Safety Analysis Set)", bold = T, font_size = 11)
  footnotes("Page [pg] of [tpg]", align = "right", blank_row = "none") |>
  footnotes(paste0("Program:",program_path), italics = TRUE) |>
  footnotes(output_path, blank_row = "none", italics = TRUE)

# Create report and add content
rpt <- create_report(outfile,
                     orientation = "portrait",
                     output_type = "RTF",
                     font_name = "Arial",
                     font_size = 9)  |>
  set_margins(top = 1, bottom = 0.75, right = 1, left = 1.5) |>
  add_content(tbl)
# Write the report
write_report(rpt)
```

Note the following in the above code:

- The `stub()` function merges *"item_group"* and *"item"* into one column for hierarchy display.

- The `define()` function controls basic attributes of each column such like width, label, and alignment.
- The parameter `label_row` in `define()` function lets *"item_group"* be the label of rows for hierarchy display.
- The parameter `blank_before` in `define()` function creates blank lines before each *"item_group"*.
- The parameter `indent` in `define()` function indents *"item"* for 0.25 inches.
- The `titles()` and `footnotes()` functions are attached to `create_table()` so that titles and footnotes are in the scope of this table.

Below is the table generated by the **reporter** package. We can see that it looks the same as SAS:

Table 14-2.1. Baseline Demographics
(Safety Analysis Set)

	Treatment A (N=145)	Treatment B (N=153)	Total (N=298)
Sex - n (%)			
Male	75 (51.7)	91 (59.5)	166 (55.7)
Female	70 (48.3)	62 (40.5)	132 (44.3)
Ethnicity - n (%)			
Hispanic/Latino	0 (0.0)	2 (1.3)	2 (0.7)
Not Hispanic/Latino	143 (98.6)	145 (94.8)	288 (96.6)
Unknown/Missing	2 (1.4)	6 (3.9)	8 (2.7)
Race - n (%)			
Asian	18 (12.4)	10 (6.5)	28 (9.4)
Black (or African American)	0 (0.0)	1 (0.7)	1 (0.3)
White	123 (84.8)	130 (85.0)	253 (84.9)
Other	1 (0.7)	6 (3.9)	7 (2.3)
Age (years)			
n	145	153	298
Mean	64.7	65.8	65.3
SD	9.7	8.3	9.0
Median	65.0	66.0	66.0
Q1, Q3	59.0, 73.0	60.0, 72.0	59.0, 72.0
Min, Max	40, 83	43, 85	40, 85
Age group - n (%)			
18 - 64 years	71 (49.0)	62 (40.5)	133 (44.6)
65 - 74 years	49 (33.8)	67 (43.8)	116 (38.9)
75 - 84 years	25 (17.2)	23 (15.0)	48 (16.1)
≥ 85 years	0 (0.0)	1 (0.7)	1 (0.3)

Page 1 of 1

Program:/dummy/tables/t-dm-sum-fas.R
Output: t14-02-001-dm-sum-fas.rtf (Date Generated: 20MAR2026:12:43:50) Source: adsl

Figure 4: Demographic table by the reporter package

SHIFT TABLE WITH SPANNING HEADERS

The following shift table is generated by SAS:

Table 14-7.2.1. Shifts in Grade From Baseline - Two Way - Aspartate Aminotransferase (U/L)
(Safety Analysis Set)

	Maximum Grade for Increased Value							Maximum Grade for Decreased Value						
Baseline Grade	NA n (%)	0 n (%)	1 n (%)	2 n (%)	3 n (%)	4 n (%)	Total n (%)	NA n (%)	0 n (%)	1 n (%)	2 n (%)	3 n (%)	4 n (%)	Total n (%)
Treatment A (N = 205)														
NA	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	205 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	205 (100.0)
0	0 (0.0)	143 (69.8)	40 (19.5)	1 (0.5)	3 (1.5)	1 (0.5)	188 (91.7)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
1	0 (0.0)	0 (0.0)	13 (6.3)	2 (1.0)	0 (0.0)	0 (0.0)	15 (7.3)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
3	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
4	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
Total	0 (0.0)	144 (70.2)	53 (25.9)	3 (1.5)	4 (2.0)	1 (0.5)	205 (100.0)	205 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	205 (100.0)
Treatment B (N = 190)														
NA	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	190 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	190 (100.0)
0	0 (0.0)	135 (71.1)	33 (17.4)	4 (2.1)	0 (0.0)	1 (0.5)	173 (91.1)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
1	0 (0.0)	0 (0.0)	14 (7.4)	1 (0.5)	1 (0.5)	0 (0.0)	16 (8.4)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
3	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
4	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
Total	0 (0.0)	136 (71.6)	47 (24.7)	5 (2.6)	1 (0.5)	1 (0.5)	190 (100.0)	190 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	190 (100.0)

Page 1 of 1

Safety Analysis Set includes all subjects who are enrolled and received at least one dose of study drug.
Parameters included: Aspartate Aminotransferase (U/L).
Grading categories determined using CTCAE version 4.0.

Program:
/dummy/tables/t-lb-shift-two-way-ast.sas
Output: t14-07-002-001-lb-shift-two-way.rtf (Date Generated: 03DEC23:20:49:28) Source: adlb, adsl

Figure 5: Sift table from SAS

This layout contains:

- Two spanning headers with a gap between adjacent columns.
- Row labels for each treatment group

Now, let's replicate this table with the **reporter** package. The input data will be like:

	trt_stub	item	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14
1	Treatment A (N=205)	NA	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	205 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	205 (100.0)
2	Treatment A (N=205)	0	0 (0.0)	143 (69.8)	40 (19.5)	1 (0.5)	3 (1.5)	1 (0.5)	188 (91.7)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
3	Treatment A (N=205)	1	0 (0.0)	0 (0.0)	13 (6.3)	2 (1.0)	0 (0.0)	0 (0.0)	15 (7.3)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
4	Treatment A (N=205)	2	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
5	Treatment A (N=205)	3	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
6	Treatment A (N=205)	4	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
7	Treatment A (N=205)	Total	0 (0.0)	144 (70.2)	53 (25.9)	3 (1.5)	4 (2.0)	1 (0.5)	205 (100.0)	205 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	205 (100.0)
8	Treatment B (N=190)	NA	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	190 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	190 (100.0)
9	Treatment B (N=190)	0	0 (0.0)	135 (71.1)	33 (17.4)	4 (2.1)	0 (0.0)	1 (0.5)	173 (91.1)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
10	Treatment B (N=190)	1	0 (0.0)	0 (0.0)	14 (7.4)	1 (0.5)	1 (0.5)	0 (0.0)	16 (8.4)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
11	Treatment B (N=190)	2	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
12	Treatment B (N=190)	3	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
13	Treatment B (N=190)	4	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
14	Treatment B (N=190)	Total	0 (0.0)	136 (71.6)	47 (24.7)	5 (2.6)	1 (0.5)	1 (0.5)	190 (100.0)	190 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	190 (100.0)

Figure 6: Input Data for Sift table

Here is the code that shows how to create this shift table:

```
# Create table content
tbl <- create_table(output_data, borders = "outside") |>
  # ----- Column setting -----#
  column_defaults(from = "col1", to = "col14", align = "center") |>
  # ----- Spanning header setting -----#
  spanning_header(from = "col1", to = "col7",
                  label = "Maximum Grade for Increased Value") |>
  spanning_header(from = "col8", to = "col14",
                  label = "Maximum Grade for Decreased Value") |>
  # ----- Combine columns for hierarchy display -----#
  stub(c("trt_stub", "item"), label = "Baseline\nGrade") |>
  define(trt_stub, label_row = T, blank_before = T) |>
  # ----- Label setting -----#
  define(col1, label = "NA\nn(%)") |>
  define(col2, label = "0\nn(%)") |>
  define(col3, label = "1\nn(%)") |>
  define(col4, label = "2\nn(%)") |>
  define(col5, label = "3\nn(%)") |>
  define(col6, label = "4\nn(%)") |>
  define(col7, label = "Total\nn(%)") |>
  define(col8, label = "NA\nn(%)") |>
  define(col9, label = "0\nn(%)") |>
  define(col10, label = "1\nn(%)") |>
  define(col11, label = "2\nn(%)") |>
  define(col12, label = "3\nn(%)") |>
  define(col13, label = "4\nn(%)") |>
  define(col14, label = "Total\nn(%)") |>
  # ----- Title setting -----#
  titles(title_string1, title_string2, bold = T, font_size = 11) |>
  # ----- Footnote setting -----#
  footnotes("Page [pg] of [tpg]", align = "right", blank_row = "none",
            valign = "top") |>
  footnotes(ft1, align = "left", blank_row = "none", valign = "top") |>
  footnotes(ft2, align = "left", blank_row = "none", valign = "top") |>
  footnotes(ft3, align = "left", blank_row = "none", valign = "top") |>
  footnotes("Program:") |>
```

```
  footnotes(program_path, blank_row = "none", italics = TRUE) |>
  footnotes(last_footnote, blank_row = "none", italics = TRUE)
# Create report and add content
rpt <- create_report(file.path(paste0(output_path,output_name)),
                     orientation = "landscape", font_name = "Arial",
                     font_size = 9, output_type = "RTF")  |>
  set_margins(top = 1.5, bottom = 1.0, right = 1.0, left = 0.75) |>
  add_content(tbl)
# Write the report
write_report(rpt)
```

Note the following in the above code:

- The `column_defaults()` function sets common attributes for multiple columns.
- The `spanning_header()` function creates spanning headers.
- The `stub()` function merges *"trt_stub"* and *"item"* into one column for hierarchy display.
- The `define()` function creates labels of row and blank lines and also controls basic attributes of each column such like width, label, and alignment.

The output generated by the **reporter** package is shown below:

Table 14-7.2.1. Shifts in Grade From Baseline - Two Way - Aspartate Aminotransferase (U/L)
(Safety Analysis Set)

	Maximum Grade for Increased Value							Maximum Grade for Decreased Value						
Baseline Grade	NA n(%)	0 n(%)	1 n(%)	2 n(%)	3 n(%)	4 n(%)	Total n(%)	NA n(%)	0 n(%)	1 n(%)	2 n(%)	3 n(%)	4 n(%)	Total n(%)
Treatment A (N=205)														
NA	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	205 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	205 (100.0)
0	0 (0.0)	143 (69.8)	40 (19.5)	1 (0.5)	3 (1.5)	1 (0.5)	188 (91.7)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
1	0 (0.0)	0 (0.0)	13 (6.3)	2 (1.0)	0 (0.0)	0 (0.0)	15 (7.3)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
3	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
4	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
Total	0 (0.0)	144 (70.2)	53 (25.9)	3 (1.5)	4 (2.0)	1 (0.5)	205 (100.0)	205 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	205 (100.0)
Treatment B (N=190)														
NA	0 (0.0)	1 (0.5)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (0.5)	190 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	190 (100.0)
0	0 (0.0)	135 (71.1)	33 (17.4)	4 (2.1)	0 (0.0)	1 (0.5)	173 (91.1)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
1	0 (0.0)	0 (0.0)	14 (7.4)	1 (0.5)	1 (0.5)	0 (0.0)	16 (8.4)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
3	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
4	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
Total	0 (0.0)	136 (71.6)	47 (24.7)	5 (2.6)	1 (0.5)	1 (0.5)	190 (100.0)	190 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	190 (100.0)

Page 1 of 1

Safety Analysis Set includes all subjects who are enrolled and received at least one dose of study drug.
Parameters included: Aspartate Aminotransferase (U/L).
Grading categories determined using CTCAE version 4.0.

Program:
/dummy/tables/t-lb-shift-two-way-ast.R
Output: t14-07-002-002-lb-shift-two-way-ast (Date Generated: 20MAR2026:12:56:12) Source: adsl, adlb

Figure 7: Sift table from the reporter package

The blank gap between two spanning headers is automatically generated. We can see that it looks the same as SAS. With these two examples, we know that the **reporter** package can replicate the general SAS outputs for clinical trials.

MORE FEATURES

In addition to above functionality, the **reporter** package has more features for users to design the outputs. Please see the illustrations below.

PAGE WRAP FOR MULTIPLE COLUMNS

When a listing contains too many columns, the **reporter** package automatically wraps columns across pages. In this case, user can choose to fix the column so that it is shown across pages, which is similar to ID statement of PROC REPORT in SAS.

The primary code for this output is as follows:

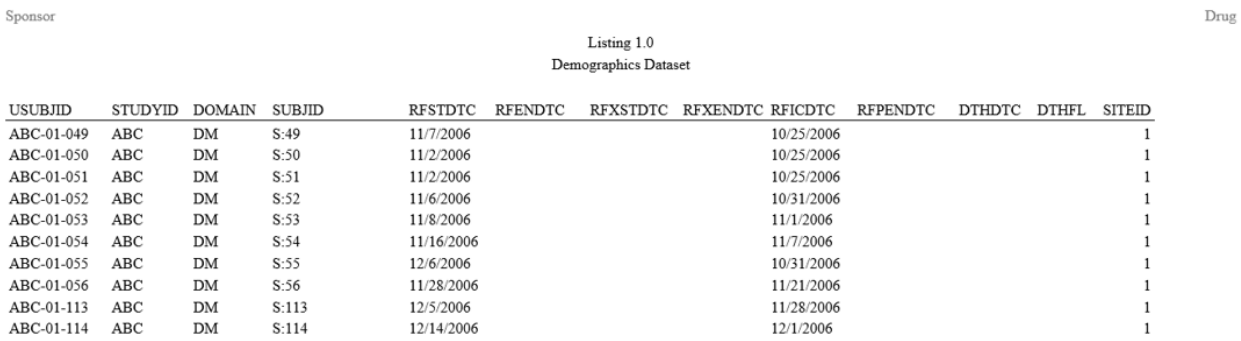
```
tbl <- create_table(data_demo) |>
  # ----- Set fixed column -----#
  define(USUBJID, id_var = TRUE) |>
  titles("Listing 1.0",
         "Demographics Dataset") |>
  footnotes("My footnotes")
```

The highlights of this program are as follows:

- The `id_var` parameter in `define()` function fixes *"USUBJID"* so that it is shown across wrapping pages.

In this case, the input data set has 23 columns, which would exceed the total width of the page. The **reporter** package would distribute these columns into different pages according to the widths. When `id_var` parameter is set as `TRUE`. The fixed column would be shown on every page.

The page 1 output is shown below:



Sponsor Drug

Listing 1.0
Demographics Dataset

USUBJID	STUDYID	DOMAIN	SUBJID	RFSTDTC	RFENDTC	RFXSTDTC	RFXENDTC	RFICDTC	RFPENDTC	DTHDTC	DTHFL	SITEID
ABC-01-049	ABC	DM	S:49	11/7/2006				10/25/2006				1
ABC-01-050	ABC	DM	S:50	11/2/2006				10/25/2006				1
ABC-01-051	ABC	DM	S:51	11/2/2006				10/25/2006				1
ABC-01-052	ABC	DM	S:52	11/6/2006				10/31/2006				1
ABC-01-053	ABC	DM	S:53	11/8/2006				11/1/2006				1
ABC-01-054	ABC	DM	S:54	11/16/2006				11/7/2006				1
ABC-01-055	ABC	DM	S:55	12/6/2006				10/31/2006				1
ABC-01-056	ABC	DM	S:56	11/28/2006				11/21/2006				1
ABC-01-113	ABC	DM	S:113	12/5/2006				11/28/2006				1
ABC-01-114	ABC	DM	S:114	12/14/2006				12/1/2006				1

Figure 8: Page 1 of Page Wrapping Output

As illustrated in above output, because the reporter package detects the total widths of all columns exceed the page width, the first 13 columns are shown on page 1, including the fixed column *"USUBJID"*.

The page 2 output is shown below:

Sponsor Drug

Listing 1.0
Demographics Dataset

USUBJID	BRTHDTC	AGE	AGEU	SEX	RACE	ETHNIC	ARMCD	ARM	ACTARMCD	ACTARM
ABC-01-049	11/12/1966	39	YEARS	M	WHITE	NOT HISPANIC OR LATINO	4	ARM D		
ABC-01-050	12/19/1958	47	YEARS	M	WHITE	NOT HISPANIC OR LATINO	2	ARM B		
ABC-01-051	5/2/1972	34	YEARS	M	WHITE	NOT HISPANIC OR LATINO	1	ARM A		
ABC-01-052	6/27/1961	45	YEARS	F	WHITE	UNKNOWN	3	ARM C		
ABC-01-053	4/7/1980	26	YEARS	F	WHITE	NOT HISPANIC OR LATINO	2	ARM B		
ABC-01-054	9/13/1962	44	YEARS	M	WHITE	NOT HISPANIC OR LATINO	4	ARM D		
ABC-01-055	6/11/1959	47	YEARS	F	BLACK OR AFRICAN AMERICAN	UNKNOWN	3	ARM C		
ABC-01-056	5/2/1975	31	YEARS	M	WHITE	NOT HISPANIC OR LATINO	1	ARM A		
ABC-01-113	2/8/1932	74	YEARS	M	WHITE	UNKNOWN	4	ARM D		
ABC-01-114	7/9/1934	72	YEARS	F	WHITE	UNKNOWN	2	ARM B		

Figure 9: Page 2 of Page Wrapping Output

The remaining 10 columns are shown on page 2. The fixed column *"USUBJID"* is still on this page.

VARIOUS BORDER STYLES

The **reporter** package provides different kinds of borders. To apply ALL borders to the entire table, please refer to the following code:

```
# Create table content
tbl <- create_table(final, first_row_blank = TRUE, width = 8.5,
                    header_bold = TRUE, borders = "all", ) |>
  # ---- Combine columns and Set Cell Style ---- #
  stub(vars = v(group, cat), "Completion Status",
       style = cell_style(bold = TRUE, indicator = "lblind")) |>
  define(lblind, visible = FALSE)
  # ---- Title and Footnote Border ---- #
  titles(tl, bold = TRUE, font_size = 11, borders = "outside",
         blank_row = "none") |>
  footnotes(ft, borders = "outside", blank_row = "none")
```

The highlights of this code are as follows:

- The `borders` parameter in `create_table()` function is set to `"all"` to create all borders for the table.
- The `borders` parameters in `titles()` and `titles()` functions are set to `"outside"` to create outside borders.
- The `cell_style()` function makes bolding style for cell values.

The resulting output is shown below:

Table 5.2.3
Subject Disposition by Category and Treatment Group
Safety Population

Completion Status	Placebo (N=20)	Drug 50mg (N=21)	Drug 100mg (N=21)	Competitor (N=23)
Subjects who Completed Study	19 (95.0%)	17 (81.0%)	16 (76.2%)	20 (87.0%)
Subject Completed All Visits And Protocol Requirements	19 (95.0%)	16 (76.2%)	16 (76.2%)	20 (87.0%)
Subject Completed All Visits But With Major Protocol Violations	0 (0.0%)	1 (4.8%)	0 (0.0%)	0 (0.0%)
Subjects terminated due to non-compliance	0 (0.0%)	1 (4.8%)	0 (0.0%)	0 (0.0%)
Non-Compliance With Study Drug	0 (0.0%)	1 (4.8%)	0 (0.0%)	0 (0.0%)
Subjects who terminated early	1 (5.0%)	3 (14.3%)	5 (23.8%)	3 (13.0%)
Lack Of Efficacy	1 (5.0%)	0 (0.0%)	3 (14.3%)	2 (8.7%)
Lost To Follow Up	0 (0.0%)	3 (14.3%)	2 (9.5%)	1 (4.3%)
Program: DS_Table.R NOTE: Denominator based on number of non-missing responses.				

Figure 10: Table with All Borders

In addition to ALL borders, the **reporter** package also provides the GROUP border feature to enhance the visual clarity between groups. please refer to the following code:

```
# Step 1. Create table content
tbl <- create_table(final, first_row_blank = TRUE, width = 8.5,
                    header_bold = TRUE, borders = "outside", ) |>
  # ---- Combine columns and Set Cell Style ---- #
  stub(vars = v(group, cat), "Completion Status",
       style = cell_style(bold = TRUE, indicator = "lblind")) |>
  define(lblind, visible = FALSE) |>
  # ---- Group Border ---- #
  define(group, format = group_fmt, blank_after = TRUE,
         group_border = TRUE) |>
```

The highlights of this code are as follows:

- The `group_border` parameter in `define()` function is set to `TRUE` to create bottom line for each group.

The resulting output is shown below:

Table 5.2.3
Subject Disposition by Category and Treatment Group
Safety Population

Completion Status	Placebo (N=20)	Drug 50mg (N=21)	Drug 100mg (N=21)	Competitor (N=23)
Subjects who Completed Study	19 (95.0%)	17 (81.0%)	16 (76.2%)	20 (87.0%)
Subject Completed All Visits And Protocol Requirements	19 (95.0%)	16 (76.2%)	16 (76.2%)	20 (87.0%)
Subject Completed All Visits But With Major Protocol Violations	0 (0.0%)	1 (4.8%)	0 (0.0%)	0 (0.0%)
Subjects terminated due to non-compliance	0 (0.0%)	1 (4.8%)	0 (0.0%)	0 (0.0%)
Non-Compliance With Study Drug	0 (0.0%)	1 (4.8%)	0 (0.0%)	0 (0.0%)
Subjects who terminated early	1 (5.0%)	3 (14.3%)	5 (23.8%)	3 (13.0%)
Lack Of Efficacy	1 (5.0%)	0 (0.0%)	3 (14.3%)	2 (8.7%)
Lost To Follow Up	0 (0.0%)	3 (14.3%)	2 (9.5%)	1 (4.3%)

Program: DS_Table.R
NOTE: Denominator based on number of non-missing responses.

Figure 11: Table with GROUP Borders

MULTPLIE TABLES ON SAME PAGE

Sometimes, we need to put multiple tables on the same page such as patient profiles. Suppose we've created two table objects `tb1` and `tb2` by the `create_table()` function. In the step of `create_report()`, we just need to use two `add_content()` functions to aggregate these two tables into one report such as below code:

```
# Add different tables into one report
rpt <- create_report(output_path, font = "Arial", output_type = "RTF") |>
  add_content(tb1, page_break = FALSE) |>
  add_content(tb2)
```

The highlights of this code are as follows:

- The `page_break` parameter in the `add_content()` function is set to `FALSE` so that the next table `tb2` would be added into the same page.

The resulting output is shown below:

Listing 1.1 Subjects Narratives of Adverse Events
Subject: ABC-01-049

USUBJID	Treatment Group	Centre	Subject	Sex	Age (yrs)	Race	Birth Date
ABC-01-049	Competitor	01	049	Male	39	White	1966-11-12

		Study Day					
Event Start Date	Event Stop Date	Start	End	System Organ Class	Severity[a]	Serious	Related
2006-12-28	2007-03-13	52	127	Investigations	Moderate	No	No
2006-12-28	2007-03-13	52	127	Investigations	Moderate	No	No
2007	2007			Nervous system disorders	Mild	No	No
2007-01-11	2007-03-13	66	127	Investigations	Moderate	No	No
2007	2007			Musculoskeletal and connective tissue disorders	Mild	No	No

[a]Severity: 01=Mild, 02=Moderate, 03=Severe, 04=Life Threatening, 05=Fatal
[b]Action Taken: 01=None, 02=Investigational product dose altered, 03=Medication taken, 04=Hospitalized, 05=Removed from study, 06=Investigational product discontinued, 07=Transfusion performed,88=Other

Figure 12: Multiple tables on the same page

As shown in above output, the first table is the demographic information of the subject, and the second table is the disposition information of the subject.

On the other hand, with `page_break = TRUE` in the `add_content()` function, the next object will be shown on the new page.

BREAK LABELS

For some types of reports, it is desirable to have a label that indicates when a group has been broken across pages. This feature is supported with the `break_label` parameter on the `define()` function.

Here is the primary code to control the break labels:

```
tbl <- create_table(adae,
                    borders = "outside", n_format = custom_n_format) |>
  # ----- Column setting -----#
  column_defaults(from = trt1, to = trt3, align = "center",
                  width = trt_width) |>
  define(blank_grp, blank_before = T, visible = FALSE) |>
  stub(vars = c("total", "AESOC", "AEDECOD", "AESEV"),
       label = "System Organ Class\n  Preferred Term\n    Grade",
       width = item_width) |>
  define(AESOC, break_label = "(Continued)") |>
  define(AEDECOD, indent = 0.16, break_label = "(Cont.)") |>
  define(AESEV, indent = 0.32)
```

The highlights of this code are as follows:

- The `break_label` parameter in the `define()` function is set to "`(Continued)`" in `AESOC` and "`(Cont.)`" in `AEDECOD` so that the values would be repeated with the corresponding suffix when they are broken across pages.

Page 1 of resulting output is shown below:

Table 14-6.1.3.1. Treatment-emergent Adverse Events by System Organ Class, Preferred Term and Grade (Safety Analysis Set)

System Organ Class Preferred Term Grade	Treatment A (N = 11) n(%)	Treatment B (N = 7) n(%)	Treatment C (N = 15) n(%)
Number of subject with adverse events	x (x.x%)	x (x.x%)	x (x.x%)
Gastrointestinal disorders	x (x.x%)	x (x.x%)	x (x.x%)
CHEILITIS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
DIARRHOEA	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
STOMACH DISCOMFORT	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
General disorders and administration site conditions	x (x.x%)	x (x.x%)	x (x.x%)
APPLICATION SITE ERYTHEMA	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
INFLUENZA LIKE ILLNESS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
Hepatobiliary disorders	x (x.x%)	x (x.x%)	x (x.x%)
GALLBLADDER DISORDER	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
Infections and infestations	x (x.x%)	x (x.x%)	x (x.x%)
BRONCHITIS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)

Page 1 of 3

Figure 13: Page 1 of the table with break labels

Page 2 of resulting output is shown below:

Table 14-6.1.3.1. Treatment-emergent Adverse Events by System Organ Class, Preferred Term and Grade (Safety Analysis Set)

System Organ Class Preferred Term Grade	Treatment A (N = 11) n(%)	Treatment B (N = 7) n(%)	Treatment C (N = 15) n(%)
Infections and infestations (Continued)			
NAIL INFECTION	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
PARONYCHIA	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
RASH PUSTULAR	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
RESPIRATORY TRACT INFECTION	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
RHINITIS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
Injury, poisoning and procedural complications	x (x.x%)	x (x.x%)	x (x.x%)
FALL	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
HEAT CRAMPS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
MUSCLE INJURY	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)

Page 2 of 3

Figure 14: Page 2 of the table with break labels

Because the group of System Organ Class *"Infections and infestations"* is broken across pages, it is shown again on the top of page 2 with the assigned suffix *"(Continued)"*.

Page 3 of resulting output is shown below:

Table 14-6.1.3.1. Treatment-emergent Adverse Events by System Organ Class, Preferred Term and Grade (Safety Analysis Set)

System Organ Class Preferred Term Grade	Treatment A (N = 11) n(%)	Treatment B (N = 7) n(%)	Treatment C (N = 15) n(%)
Injury, poisoning and procedural complications (Continued)			
MUSCLE INJURY (Cont.)			
Severe	x (x.x%)	x (x.x%)	x (x.x%)

Page 3 of 3

Figure 15: Page 3 of the table with break labels

The `break_label` parameter supports break labels for multiple columns. On page 3, because both group of System Organ Class *"Injury, poisoning and procedural complications"* and the group of Preferred Term *"MUSCLE INJURY"* are broken, they are repeated on the top of page 3 with the assigned suffixes.

GROUP COHESION

Following the previous example, now we want to avoid splitting groups across pages. To keep groups together on the same page, we can use `group_cohesion` parameter in the `define()` function.

To illustrate, let's make a small modification to the code above to turn on group cohesion:

```
tbl <- create_table(adae, borders = "outside",
                    n_format = custom_n_format) |>
  # ----- Column setting -----#
  column_defaults(from = trt1, to = trt3, align = "center",
                  width = trt_width) |>
  define(blank_grp, blank_before = T, visible = FALSE) |>
  stub(vars = c("total", "AESOC", "AEDECOD", "AESEV"),
       label = "System Organ Class\n  Preferred Term\n    Grade",
       width = item_width) |>
  define(AESOC, break_label = "(Continued)", group_cohesion = TRUE) |>
  define(AEDECOD, indent = 0.16, break_label = "(Cont.)",
         group_cohesion = TRUE) |>
  define(AESEV, indent = 0.32)
```

The highlights of this code are as follows:

- The `group_cohesion` parameter in the `define()` function is set to `TRUE` in `AESOC` and `AEDECOD` so that the groups would be on the same page as possible as they can.

Page 1 of resulting output is shown below:

Table 14-6.1.3.1. Treatment-emergent Adverse Events by System Organ Class, Preferred Term and Grade (Safety Analysis Set)

System Organ Class Preferred Term Grade	Treatment A (N = 11) n(%)	Treatment B (N = 7) n(%)	Treatment C (N = 15) n(%)
Number of subject with adverse events	x (x.x%)	x (x.x%)	x (x.x%)
Gastrointestinal disorders	x (x.x%)	x (x.x%)	x (x.x%)
CHEILITIS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
DIARRHOEA	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
STOMACH DISCOMFORT	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
General disorders and administration site conditions	x (x.x%)	x (x.x%)	x (x.x%)
APPLICATION SITE ERYTHEMA	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
INFLUENZA LIKE ILLNESS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
Hepatobiliary disorders	x (x.x%)	x (x.x%)	x (x.x%)
GALLBLADDER DISORDER	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)

Page 1 of 3

Figure 16: Page 1 of the table with group cohesion

Page 2 of resulting output is shown below:

Table 14-6.1.3.1. Treatment-emergent Adverse Events by System Organ Class, Preferred Term and Grade (Safety Analysis Set)

System Organ Class Preferred Term Grade	Treatment A (N = 11) n(%)	Treatment B (N = 7) n(%)	Treatment C (N = 15) n(%)
Infections and infestations	x (x.x%)	x (x.x%)	x (x.x%)
BRONCHITIS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
NAIL INFECTION	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
PARONYCHIA	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
RASH PUSTULAR	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
RESPIRATORY TRACT INFECTION	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
RHINITIS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
Injury, poisoning and procedural complications	x (x.x%)	x (x.x%)	x (x.x%)
FALL	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)
HEAT CRAMPS	x (x.x%)	x (x.x%)	x (x.x%)
Mild	x (x.x%)	x (x.x%)	x (x.x%)
Moderate	x (x.x%)	x (x.x%)	x (x.x%)
Severe	x (x.x%)	x (x.x%)	x (x.x%)

Page 2 of 3

Figure 17: Page 2 of the table with group cohesion

Compared to the previous example (Figure 13 and 14), the last System Organ Class group on page 1 *"Infections and infestations"* is moved to page 2 so that it would not be split across pages. In this way, the report looks nicer and becomes more readable. For more information about group cohesion, please refer to Example 16 in the **reporter** website.

RTF CODE INSERTION

We can insert RTF code in both **reporter** code and the data set with the `report_options()` function and its parameter `allow_code`. This function works for the `create_report()` object. Please refer to the following code:

```
# Create table
tbl <- create_table(df, first_row_blank = TRUE, borders = c("all")) |>
  stub(c("var", "label"), width = .8) |>
  define(var, blank_after = TRUE, label_row = TRUE,
         format = c(ampg = "\\i Here is a italic label \\i0.",
                    cyl = "\\ul Under line label \\ul0")) |>
  define(A, label = "\\shad {{\\cf6 Group A}} {common::supsc('2')}\\shad0",
         align = "center", n = 19) |>
  define(B, label = "\\outl Group B \\outl0", align = "center", n = 13) |>
  footnotes("\\i\\fs14 * Italic Small Table Footnote \\fs18")
# Create report and add content
rpt <- create_report(fp, orientation = "portrait", output_type = "RTF",
                     font = "Times") |>
  report_options(allow_code = TRUE) |>
  page_header(left = "\\strike Strikethrough header \\strike0",
              right = "Study: Cars") |>
  titles("\\b\\i {{\\highlight7 Yellow italic highlight}} \\b0\\i0",
         "\\fs0\\fs28 {{\\cf2 This is a blue big title}} \\fs18") |>
  add_content(tbl) |>
  footnotes("\\i\\fs14 * Italic Small Report Footnote \\fs18")
res <- write_report(rpt)
```

We can insert RTF code in the data set:

	var	label	A	B
1	ampg	\li360 N	19	13
2	ampg	\li360 Mean	18.8	22.0 (4.9)
3	ampg	\li360 Median	16.4{\sub mm}	21.4{\super 2}
4	ampg	\li360 {\cf12 Q1 - Q3}	15.1 - 21.2	19.2 - 22.8
5	ampg	\li360 Range	{\cf2 10.4 - 33.9}	14.7 - 32.4
6	cyl	\li360 8\line Cylinder	\ul 10 (52.6%) \ul0	4 (30.8%)
7	cyl	\li360 6\line {\cf11 Cylinder and more perhaps more}	4 (21.1%)	3 (23.1%)
8	cyl	\li360 4\line Cylinder	\ul 5 (26.3%) \ul0	6 (46.2%)

Figure 18: Data set with RTF code insertion

Observe that there are several RTF formatting codes in **reporter** code and the data set:

- Italic with `\\i`
- Underline with `\\ul`
- Shadow with `\\shad`
- Outline with `\\outl`
- Strikethrough with `\\strike`
- Highlight with `\\highlight`
- Coloring with `\\cf`
- Subscript and Superscript with `\\sub` and `\\super`
- Changing font size with `\\fs`

Please note that curly braces should be doubled, i.e.`{{\\cf6 Group A}}`. The single curly braces will still be available as glue replacements.

Resulting output is shown below:

~~Strikethrough header~~ Study: Cars

Yellow italic highlight

This is a blue big title

	Group A [2] (N=19)	**Group B** (N=13)
Here is a italic label .		
N	19	13
Mean	18.8	22.0 (4.9)
Median	16.4[mm]	21.4[2]
Q1 - Q3	15.1 - 21.2	19.2 - 22.8
Range	10.4 - 33.9	14.7 - 32.4
Under line label		
8 Cylinder	10 (52.6%)	4 (30.8%)
6 Cylinder and more perhaps more	4 (21.1%)	3 (23.1%)
4 Cylinder	5 (26.3%)	6 (46.2%)

** Italic Small Table Footnote*

Figure 19: Output with RTF code insertion

Observe that the RTF codes work as expected:

- Header is formatted with strike through.
- Titles are formatted with yellow highlights, italic, bold, blue color, and bigger font size.
- Column labels are formatted with red, shadow, and outline.
- Row labels are formatted with italic and underline.
- Subscript and superscript are displayed.
- Data values are formatted with blue, purple, underline, and indentation.
- Footnotes are formatted with italic and smaller font size.

HTML CODE INSERTION

Following the previous example, we can also insert HTML code with `report_options(allow_code = TRUE)` when output type is HTML. Please refer to the following code:

```
# Create table
tbl <- create_table(dat, borders = "outside") |>
  titles('<p style="color: blue; font-size: 18px;">Blue 18 px <b>Title</b>:
         Page [pg] of [tpg]</p>',
         "<b>Bold table title</b>") |>
  footnotes("<i>Italic footnote: Page [pg] of [tpg]</i>",
            "Emphasize <sup>Super</sup>") |>
  define(var, label = "Name") |>
  define(A, label = "Treatment A mg<sup>2</sup>") |>
  define(B, label = "Treatment B mg<sup>2</sup>")
# Create Report
rpt <- create_report(fp, output_type = "html", font = fnt,
                     font_size = fsz, orientation = "landscape") |>
  report_options(allow_code = TRUE) |>
  set_margins(top = 1, bottom = 1) |>
  add_content(tbl) |>
  titles("<mark>Mark Title: Page [pg] of [tpg]</mark>") |>
  footnotes("<small>Small footnote</small>: Page [pg] of [tpg]",
            borders = "none") |>
  page_header('<code style="color:red;">Code Style Page [pg] of
              [tpg]</code>',
              right = '<del>Strikethrough</del>') |>
  page_footer("Footer: Page <sub>[pg]</sub> of [tpg]",
              right = "Footer: Page <sub>[pg]</sub> of [tpg]")
```

We can also insert HTML code in the data set:

	var	label	A	B
1	<u>Underline</u>	N	19_{max}	13²
2	ampg	Mean	18.8 (6.5)	22.0 (4.9)
3	ampg	Indent Value	16.4	21.4
4	ampg	Q1 - Q3	15.1 - 21.2	19.2 - 22.8
5	ampg	<code style="color:purple;">Purple Code Style</code>	10.4 - 33.9	14.7 - 32.4
6	cyl	8 Cylinder	10 (52.6%)	4 (30.8%)
7	cyl	Red Words and Bold and Change Lines	4 (21.1%)	3 (23.1%)
8	cyl	Background Color and Bold	5 (26.3%)	6 (46.2%)

Figure 19: Data set with HTML code insertion

Resulting output is shown below:

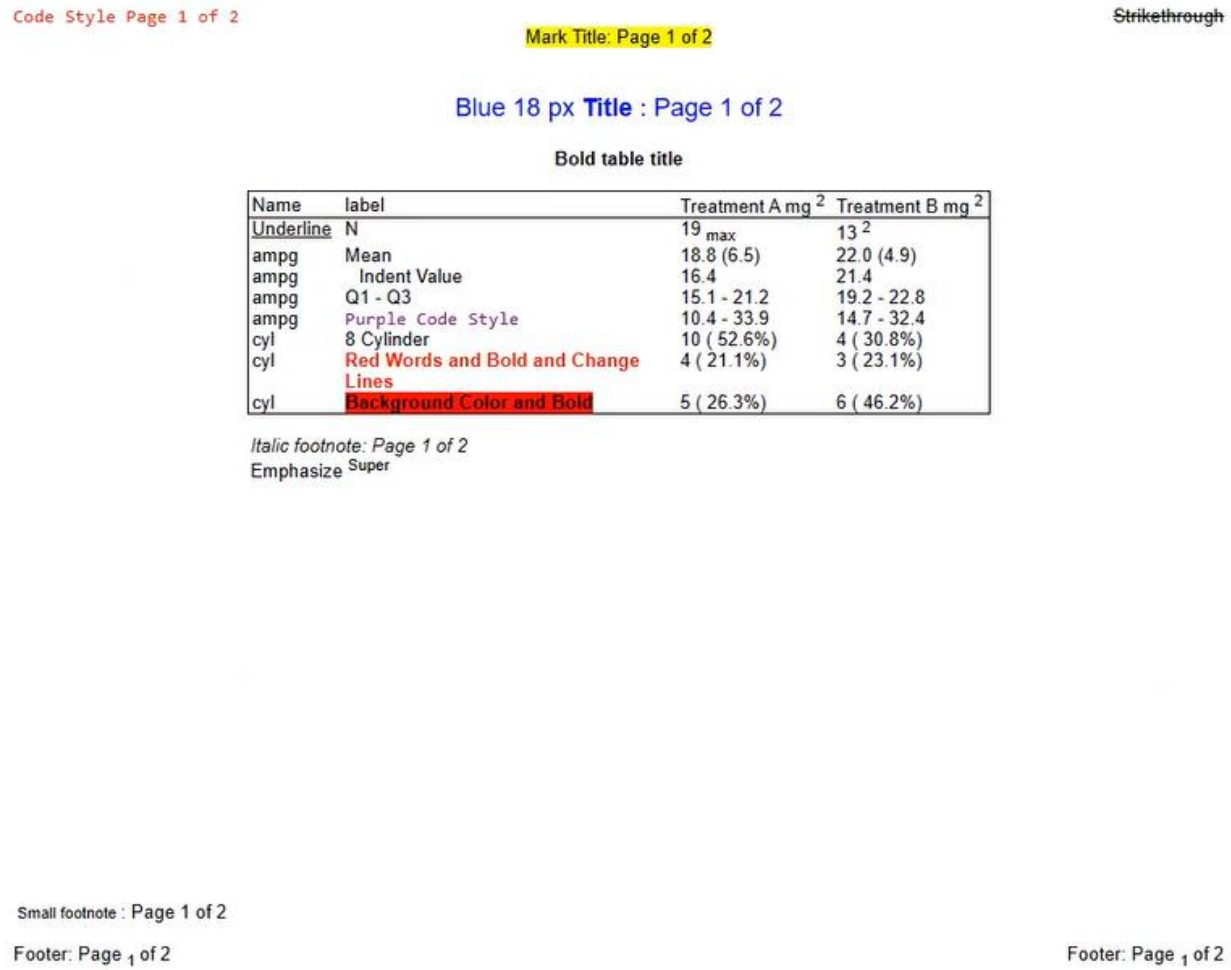


Figure 20: Output with HTML code insertion

Observe that the HTML codes work as expected:

- Headers are formatted with red code style and strike through.

- Titles are formatted with yellow highlight, blue 18px, and bold.
- Column headers are formatted with superscript.
- Data values are formatted with underline, indentation, color, bold, and background color.
- Footnotes are formatted with italic, superscript, and smaller font size.
- Footers are formatted with subscripts.

CONCLUSION

The **reporter** R package can create almost any report that you can create with SAS for clinical trials, such as demographics tables with hierarchy column and shift tables with spanning headers shown in this paper. The **reporter** package was designed specifically to have similar capabilities and a similar conceptual framework. It has a variety of options to control table layout including page wrapping, border styles, multiple tables on the same page, break labels, and group cohesions. It also allows you to export into five different file formats. In addition, the **reporter** package allows users to insert RTF code and HTML code to customize the formats. In summary, the package is not only easy to use and produces results that are nearly identical to SAS but also contains various options for users to design the outputs flexibly.

REFERENCES

Bosak D (2024). *Replicating SAS® Reports in R: The REPORTER Package*, SESUG Paper 113-2024, https://sesug.org/proceedings/sesug_2024_SAAG/PresentationSummaries/Papers/113_Final_PDF.pdf

Bosak D (2023). *Getting Started with Reporter*, Accessed August 13, 2024. https://reporter.r-sassy.org/articles/reporter.html

Bosak D (2023). *An Overview of the SASSY System*, WUSS Paper 185-2023, https://www.lexjansen.com/wuss/2023/WUSS-2023-Paper-185.pdf

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to David J. Bosak for inviting me to co-author the **reporter** package. His invaluable guidance has profoundly enriched my knowledge. Over the past year, our tireless dedication to continuous development has been driven by a shared mission: to empower R users within the clinical trial industry to generate outputs with greater efficiency, ease, and precision.

CONTACT INFORMATION

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

Bill Huang
bill.huang@toastmasters.org.tw

David J. Bosak
Archytas Clinical Solutions
dbosak01@gmail.com
www.r-sassy.org