Paper CT11

Elevating Clinical Data Presentation: Unlocking the Power of rtables in R

Pooja Jasuja, Johnson & Johnson, Mumbai, India

ABSTRACT

In clinical research, effective data presentation not only supports decision making but also ensures clarity, reproducibility, and regulatory readiness. The rtables package in R simplifies this process by delivering publication-ready tables through an intuitive layout system and minimal coding effort. Its flexible design supports complex structures, combining categorical and continuous summaries while ensuring consistency and transparency.
This paper explores advanced rtables capabilities such as conditional formatting to highlight critical results and improve readability. Practical examples demonstrate how rtables accelerates report generation and enhances stakeholder communication. By adopting rtables, teams can elevate data presentation standards and convey insights with clarity and precision in the evolving landscape of clinical research.

INTRODUCTION

Clinical trial data is the foundation for evaluating safety, efficacy, and regulatory decisions. Statistical analysis of this data provides meaningful insights, but presenting these findings clearly is just as important. Typically, these results appear in Clinical Study Reports (CSRs) through structured tables that summarize key endpoints.

Several tools are commonly used to create these tables—SAS, R packages like **tidytlg**, and even Python. Each offers unique strengths but also limitations in flexibility, readability, or coding effort. Among these, the **rtables** package in R stands out for its ability to produce publication-ready tables with minimal code and maximum control over layout and formatting.

In this paper, we focus on practical coding tips and tricks for rtables, demonstrating how to build simple summary tables, create nested layouts for categorical and continuous data, design shift tables for safety analysis, and apply conditional formatting to highlight critical results. These techniques showcase the versatility of rtables and its potential to streamline reporting while maintaining clarity and compliance.
Before diving into these practical examples, the next section provides an overview of rtables—what it is, how it works, and why it is becoming a preferred choice for clinical reporting.

Overview of rtables

The rtables package in R is purpose-built for creating publication-ready tables for clinical reporting. It uses a layout-driven approach, where rows and columns are defined first and analysis functions are applied seamlessly, making even complex tables easy to design and audit. With support for categorical and continuous summaries, **nested structures (to organize multiple grouping levels such as treatment → site → gender), shift tables (to compare baseline and post-baseline values for safety analysis), and conditional formatting (to highlight critical results)**, rtables combines flexibility with simplicity. These capabilities are essential for presenting clinical data in a way that is both comprehensive and easy to interpret.

Building on this foundation, the next section highlights the key features and practical use cases that demonstrate how rtables moves from simple summaries to advanced layouts for real-world clinical reporting.

Key Features & Use Cases

rtables makes it easy to move from basic to advanced clinical tables without adding complexity. In the following sections, we focus on three core capabilities that demonstrate its power:

- **Transforming Simple Summary Tables into Nested Layouts** for richer insights across categorical and continuous data.
- **Shift Tables** to capture baseline versus post-baseline changes, essential for safety reporting.
- **Conditional Formatting** to emphasize critical results and improve readability for publications.

These features illustrate how rtables combines flexibility, clarity, and efficiency to meet the demands of modern clinical reporting.

In the next section, we explain these aspects by showcasing practical examples and code snippets that illustrate how to implement each feature effectively.

1. Transforming Simple Summary Tables into Nested Layouts

1.1. Context

Clinical tables often start with basic summaries, such as treatment-arm counts or mean values. However, stakeholders frequently need deeper insights—like breaking results down by site, sex, or other subgroups—without switching tools or rewriting large portions of code. This requirement calls for a flexible approach that scales from simple summaries to multi-level structures.

1.2. Goal

Create an arm-wise table and expand it from a simple summary into a nested layout that combines categorical and continuous summaries.

1.3. R code for Simple Summary

```
# Getting started with rtables
library(rtables)

# Simulated data

set.seed(123)
df <- data.frame(
ARM = sample(c("Treatment" , "Placebo" ), 100, replace = TRUE),
GENDER = sample(c("Male", "Female"), 100, replace = TRUE),
AGE = round(rnorm(100, mean = 50, sd = 10), 1)
)

# Assume df has variables: ARM, GENDER, AGE
lyt <- basic_table() %>%
  split_cols_by("ARM") %>%
  split_rows_by("GENDER") %>%
  analyze("AGE", afun = mean)

tbl <- build_table(lyt, df)
tbl
```

1.4. Output

```
         Treatment   Placebo
---------------------------
Male
  mean     49.40      49.50
Female
  mean     51.65      44.22
```

1.5. R code for Nesting + Categorical & Continuous Summaries

```
# Getting started with rtables
library(rtables)

# Simulated clinical trial data
set.seed(123)
data2 <- data.frame(
  ARM = sample(c("Treatment" , "Placebo" ), 100, replace = TRUE),
  SITE = rep(c("Site 1", "Site 2"), times = 100),
  GENDER = sample(c("Male", "Female"), 100, replace = TRUE),
  AGE = round(rnorm(100, mean = 50, sd = 10), 1)
)

# Custom function for multiple statistics
multi_stats <- function(x) {
  in_rows(
    "Mean (SD)" = sprintf("%.2f (%.3f)", mean(x), sd(x)),
    "Median [IQR]" = sprintf("%.2f [%.1f-%.1f]", median(x),
                             quantile(x, 0.25), quantile(x,0.75))
    )
```

```
}
# Assume df has variables: ARM,SITE,GENDER, AGE
lyt2 <- basic_table() %>%
  split_cols_by("ARM") %>%
  add_colcounts()%>%
  split_rows_by("SITE") %>%
  summarize_row_groups(format="xx") %>%
  split_rows_by("GENDER") %>%
  summarize_row_groups() %>%
  analyze("AGE", afun = multi_stats)

tbl2 <- build_table(lyt2, data2)
print(tbl2)
```

1.6. Output

	Treatment (N=114)	Placebo (N=86)
Site 1	60	40
Male	34 (29.8%)	16 (18.6%)
Mean (SD)	49.44 (7.427)	53.27 (5.820)
Median [IQR]	46.50 [43.9-53.3]	55.40 [49.7-57.6]
Female	26 (22.8%)	24 (27.9%)
Mean (SD)	51.59 (12.517)	42.77 (8.417)
Median [IQR]	50.40 [40.5-61.1]	41.80 [36.2-51.4]
Site 2	54	46
Male	18 (15.8%)	24 (27.9%)
Mean (SD)	49.33 (9.048)	46.98 (8.505)
Median [IQR]	47.40 [46.3-50.8]	47.00 [40.7-51.7]
Female	36 (31.6%)	22 (25.6%)
Mean (SD)	51.69 (11.844)	45.79 (5.681)
Median [IQR]	51.65 [43.5-57.7]	45.80 [40.6-49.1]

1.7. Key Highlights of This Enhancement

a. **Nesting with Group Summaries**
Adding split_rows_by("SITE") and split_rows_by("GENDER") introduces hierarchical structure. Using summarize_row_groups() at each level automatically displays **row counts and percentages**, giving clear context for subgroup sizes.

b. **Column Counts for Transparency**
The add_colcounts() function adds **N per treatment arm** in column headers, ensuring denominators are visible for all summaries.

c. **Descriptive Statistics with Controlled Formatting**
The custom multi_stats function uses sprintf() to format **Mean (SD)** and **Median [IQR]** with precise decimal control, producing clean, publication-ready outputs.

2. Shift Tables

2.1. Context

In laboratory safety assessments, toxicity grades (e.g., Grade 0, 1, 2, 3) indicate the severity of abnormalities. Regulatory reviewers and clinicians need to see not just the current grade but how it changes after treatment—whether patients progress from normal (Grade 0) to mild (Grade 1) or severe (Grade 3). This progression is critical for evaluating drug safety. Shift tables provide a clear way to display these transitions from baseline to post-baseline, making patterns of worsening or improvement easy to interpret.

2.2. Goal

Create baseline vs post-baseline shift tables to display toxicity grade transitions (e.g., Grade 0→1, 1→2).

2.3. R code (illustrative)

```
# Getting started with rtables
library(rtables)
```

```r
set.seed(2025)

# --- Simulated data ---
lab <- data.frame(
  PARAM = rep(c("ALT", "AST", "Hemoglobin"), each = 30),
  BASE = sample(c("Low", "Normal", "High"), 90, TRUE, prob = c(0.2, 0.6, 0.2)),
  POST = sample(c("Low", "Normal", "High"), 90, TRUE, prob = c(0.1, 0.7, 0.2)),
  AVISIT = rep(c("Day 2","Day 4", "EOS"), length.out = 90)  # Added visit variable
)

# --- Analysis function: counts for POST categories by BASE ---
shift_afun <- function(x, .spl, .var, ...) {
  # x = POST values for a given BASE category
  counts <- table(factor(x, levels = c("Low", "Normal", "High")))
  # add total row
  total <- sum(counts)
  # Convert to named list for rtables
  c(as.list(counts), Total = total)
}

# Convert to factors with correct order' and define header
lab$BASE_header <- "Baseline"
lab$BASE <- factor(lab$BASE, levels = c("Low", "Normal", "High"))
lab$POST <- factor(lab$POST, levels = c("Low", "Normal", "High"))


# --- Layout for shift table ---
# Assume lab df has variables: PARAM, BASE, POST, AVISIT
lyt <- basic_table() %>%
  split_cols_by("BASE_header") %>%
  split_cols_by("BASE") %>%
  add_overall_col("Total") %>%  # ☑ Adds Total column
  split_rows_by("PARAM" ,
                label_pos = "topleft",
                child_labels = "visible",
                split_label = "Laboratory Test "
                ) %>%
  split_rows_by("AVISIT", split_label = "Visit") %>%  # Added AVISIT
  analyze("POST", afun = shift_afun)

# --- Build table ---
tbl <- build_table(lyt, lab)
tbl
```

2.4. Output

Laboratory Test	Baseline Low	Baseline Normal	Baseline High	Total
ALT				
Day 2				
Low	0	1	0	1
Normal	1	2	2	5
High	2	1	1	4
Total	3	4	3	10
Day 4				
Low	0	2	0	2
Normal	2	1	1	4
High	0	4	0	4
Total	2	7	1	10
EOS				
Low	1	0	0	1
Normal	0	5	2	7
High	0	1	1	2
Total	1	6	3	10

2.5. Key Highlights of This Enhancement

a. **Baseline-as-columns with a clean total:**
The layout defines columns by Baseline (Low, Normal, High) and appends an overall Total using add_overall_col("Total"), so reviewers can see per-baseline distributions and the aggregate in one glance.

b. **Stable shift counts per slice:**
shift_afun() computes POST category counts with table(factor(x, levels = ...)), enforcing clinical order and preserving zero-count levels—yielding a consistent baseline → post shift block for every PARAM × AVISIT.

c. **Readable blocks by test and visit:**
Row splits on PARAM and AVISIT group results into intuitive blocks—"for this lab test, at this visit, here's how values moved after baseline"—making the narrative clear without extra footnotes or reshaping.

3. Conditional Formatting

3.1. Context

Demographic tables often include variables like age, sex, and race across treatment arms. While these summaries are essential, large tables can make it difficult for reviewers to quickly spot critical patterns—such as an imbalance in age distribution or a skewed sex ratio between arms. Conditional formatting helps address this by visually emphasizing such differences based on predefined rules. For example, highlighting median age values above a threshold or bolding counts where one arm significantly outweighs another improves readability and draws attention to potential safety or efficacy concerns.

3.2. Goal

Emphasize statistical or clinical significance using styling such as bold, italics, trend arrows, special characters or colors.

3.3. R code (concept)

```
# Getting started with rtables
library(rtables)
set.seed(123)

# Simulated clinical data

data2 <- data.frame(
  ARM = sample(c("Treatment", "Placebo"), 100, replace = TRUE),
  SITE = rep(c("Site 1", "Site 2"), times = 100),
  SEX = sample(c("Male", "Female"), 100, replace = TRUE),
  AGE = round(rnorm(100, mean = 50, sd = 10), 1)
```

```
)

# Custom function for multiple statistics
multi_stats <- function(x) {
  mean_val <- mean(x)
  median_val <- median(x)
  q1 <- quantile(x, 0.25)
  q3 <- quantile(x, 0.75)

  mean_sd <- if (mean_val > 50) {
    paste0("<i>", "<span style='color:red;'>", sprintf("%.1f", mean_val),"</i>"
,"</span>", " \u2191", " (",
           "<span style='color:green;'>", sprintf("%.1f", sd(x)),"</span>", ")")
  } else {
    sprintf("%.1f (%.1f)", mean_val, sd(x))
  }

  med_iqr <- if (median_val > 50) {

    paste0("<b>", sprintf("%.1f", median_val), "**", "</b>",
           " [", sprintf("%.1f", q1), "-", sprintf("%.1f", q3), "]")

  } else {
    sprintf("%.1f [%.1f-%.1f]", median_val,
            q1, q3)
  }

  in_rows(
    "Mean (SD)" = mean_sd,
    "Median [IQR]" = med_iqr,
    .formats = "xx"
  )
}

lyt2 <- basic_table() %>%
  split_cols_by("ARM") %>%
  add_colcounts()%>%
  split_rows_by("SITE") %>%
  summarize_row_groups(format="xx") %>%
  split_rows_by("SEX") %>%
  summarize_row_groups() %>%
  analyze("AGE", afun = multi_stats)

tbl2 <- build_table(lyt2, data2)
print(tbl2)

# Export to HTML
html_output <- as.character(as_html(tbl2))

# Unescape HTML tags in cell content
html_output_clean <- gsub("&lt;", "<", html_output)
html_output_clean <- gsub("&gt;", ">", html_output_clean)

# Write to file
html_path <- file.path(Sys.getenv("USERPROFILE"), "Downloads", "formatted_table.html")

writeLines(html_output_clean,html_path )
```

3.4. Output

	Treatment (N=114)	Placebo (N=86)
Site 1	60	40
Male	34 (29.8%)	16 (18.6%)
Mean (SD)	49.4 (7.4)	*53.3* ↑ (5.8)
Median [IQR]	46.5 [43.9–53.3]	**55.4**** [49.7–57.6]
Female	26 (22.8%)	24 (27.9%)
Mean (SD)	*51.6* ↑ (12.5)	42.8 (8.4)
Median [IQR]	**50.4**** [40.5–61.1]	41.8 [36.2–51.4]
Site 2	54	46
Male	18 (15.8%)	24 (27.9%)
Mean (SD)	49.3 (9.0)	47.0 (8.5)
Median [IQR]	47.4 [46.3–50.8]	47.0 [40.7–51.7]
Female	36 (31.6%)	22 (25.6%)
Mean (SD)	*51.7* ↑ (11.8)	45.8 (5.7)
Median [IQR]	**51.7**** [43.5–57.7]	45.8 [40.6–49.1]

3.5. Key Highlights of This Enhancement

a. Clinically-aware, in-cell emphasis (no extra columns)
   - The table encodes **clinical relevance directly inside cells**—red italicized **Mean** with an ↑ arrow when thresholds are exceeded, and **bold Median** to flag potential concern.
   - This **reduces cognitive load** for reviewers: they can spot elevated signals instantly **without scanning additional annotations** or columns.

b. Multi-stat rows along with conditional formatting from a single reusable function
   - multi_stats() produces **multiple labelled rows** ("Mean (SD)" and "Median [IQR]") and applies **conditional HTML styling** in one pass.
   - This approach ensures **consistent formatting and logic reuse**, making it easy to scale or adapt thresholds and styles across different tables.

c. Display separated from computation for flexible outputs
   - Numeric calculations remain clean, while styling is layered on top for HTML outputs.
   - This separation allows easy toggling between **styled HTML**, **plain text**, or **image exports** without rewriting analytical code.

CONCLUSION

The rtables package makes it easier to create clinical tables while keeping them clear, reproducible, and ready for regulatory use. Using a layout-first approach, combining categorical and continuous summaries, adding shift tables for safety analysis, and applying conditional formatting where needed helps teams produce publication-ready outputs with minimal coding effort. These practices fit well into modern R workflows, making it easier to iterate quickly, maintain clean audits, and communicate results effectively.

Looking ahead, rtables can go beyond traditional tables. It can work with interactive reporting tools like Shiny or Quarto for drill-down views, combine tables with graphics for better safety and efficacy insights, and use packages such as flextable or gt for conditional formatting in Word, PowerPoint, and PDF outputs. These directions will make reporting faster, more flexible, and easier to adapt to different formats.

REFERENCES

1. **rtables Development Team. (2025). *rtables: Reporting Tables*. R package version 0.6.15.**
   Available at:
   https://cran.r-project.org/package=rtables

2. **Insights Engineering Team. (2025). *rtables: Reporting Tables with R* — Official Documentation.**
   A user-friendly guide showing how to build clinical reporting tables using rtables, including examples for layout creation, formatting, and multi-level tabulation.
   Available at:
   https://github.com/insightsengineering/rtables/blob/main/inst/extdata/Clinical_Trial_Reporting_Tables_in_R.pdf

ACKNOWLEDGMENTS

Special thanks to Statistical Programming India team for their continual feedback and learning forums that inspired the examples presented here. Generative AI tools, including GitHub Copilot, were used to assist with drafting portions of the text; all analyses, explanations, and code snippets were subsequently validated to ensure accuracy, reproducibility, and alignment with clinical reporting standards.

CONTACT INFORMATION

Author Name: Pooja Jasuja
Company: Johnson & Johnson
Email: pjasuja@its.jnj.com