Building CDISC TFLs in R with {pharmaRTF} and {huxtable}: a practical Table 1

Brian Carter

2025-08-12

Table of contents

## Introduction

Tables, Figures, and Listings (TFLs) are the backbone of CDISC statistical reporting. In many organizations, SAS has been the primary tool, with teams leaning heavily on PROC REPORT and PROC TEMPLATE. The initial learning curve and the tendency to “lock into” one table design can make moving to R feel daunting. The goal of this post is to show a simple, reproducible path in R that you can *copy-paste and extend*: we’ll build a classic **Table 1** (baseline demographics) from an ARD (Analysis Results Data) style dataset using **huxtable** for table styling and **pharmaRTF** for regulator-friendly RTF output.

We’ll keep the pattern minimal and reusable: shape the ARD to a wide format, print a readable table for HTML/PDF, then export the same table to RTF with proper titles, footnotes, and pagination. The data used for this post are [available online for download.](https://github.com/buddha2490/pharmaRTFBlog.git)

What you’ll learn

* How ARD-shaped results roll up into TFLs
* A “starter” Table 1 you can replicate across studies
* Practical huxtable tricks: multi-row headers, cell merges, alignment, and borders
* How pharmaRTF adds titles, footnotes, and document properties for submission-ready RTF

## Data: the ARD idea

ARD (Analysis Results Data) standardizes computed statistics (counts, percentages, means, SDs, etc.) into a tidy structure, so TFLs can be automated from a consistent input. Many teams now generate ARDs directly in R; a popular option in the pharmaverse is the cards package, which provides constructors and utilities for ARD objects (and extensions in cardx for more analyses). We’ll start from a prepared ARD-like file, data.xlsx, and restructure it into a Table 1 input using following code.

df <- readxl::read\_excel("data.xlsx")  
  
table1 <- df %>%  
 filter(program == "program1.R") %>%  
 filter(variable\_context == "char") %>%  
 select(-stat\_label) %>%  
 pivot\_wider(  
 names\_from = stat\_name,  
 values\_from = stat  
 ) %>%  
 mutate(stat = as.character(glue::glue("{p} ({n})"))) %>%  
 group\_by(variable\_label, group\_level) %>%  
 mutate(variable\_label = ifelse(row\_number() != 1, "", variable\_label)) %>%  
 ungroup() %>%  
 mutate(stat\_label = "N (%)") %>%  
 bind\_rows(  
 df %>%  
 filter(program == "program1.R") %>%  
 filter(variable\_context == "num") %>%  
 select(-stat\_name) %>%  
 group\_by(variable\_label, group\_level) %>%  
 mutate(variable\_label = ifelse(row\_number() != 1, "", variable\_label)) %>%  
 ungroup() %>%  
 group\_by(variable\_level, group\_level) %>%  
 mutate(variable\_level = ifelse(row\_number() != 1, "", variable\_level)) %>%  
 ungroup()  
 ) %>%  
 select(group\_level, variable\_label, variable\_level, stat\_label, stat) %>%  
 mutate(group\_level = factor(group\_level, levels = c("All Patients", "Placebo", "Treatment")))

## Building your first huxtable

The huxtable package creates customizable, publication-quality tables in R with intuitive functions for styling, alignment, cell merging, and formatting. Paired with pharmaRTF, it enables seamless formatting of TFLs for regulatory submission, bridging statistical output and polished RTF tables, making R a viable SAS PROC REPORT alternative.

This minimal version is already sufficient for many tables: feed it a similarly shaped ARD and you’ll get consistent output. Next, let’s layer in the styling tricks you often need in clinical reporting.

ht <- table1 %>%  
 pivot\_wider(  
 names\_from = group\_level,  
 values\_from = stat,  
 id\_cols = c(variable\_level, variable\_label, stat\_label),  
 values\_fill = "",  
 values\_fn = list(stat = dplyr::first)) %>%  
 select(Variable = variable\_label,   
 Level = variable\_level,   
 Statistic = stat\_label, everything())  
  
  
  
# Basic alignment: descriptions left, stats centered/right as you prefer  
n\_cols <- ncol(ht) # total number of columns in the table  
group\_cols <- setdiff(seq\_len(n\_cols), 1:3) # isolate the groups  
  
  
ht %>%  
 huxtable::as\_hux() %>%  
 huxtable::set\_bold(1, everywhere, TRUE) %>%  
 huxtable::set\_align(, n\_cols, "left") %>%  
 huxtable::set\_align(, group\_cols, "center") %>%  
 huxtable::set\_width(1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable** | **Level** | **Statistic** | **All Patients** | **Placebo** | **Treatment** |
| Variable 1 | Category 1 | N (%) | xx (xx) | xx (xx) | xx (xx) |
|  | Category 2 | N (%) | xx (xx) | xx (xx) | xx (xx) |
|  | Category 3 | N (%) | xx (xx) | xx (xx) | xx (xx) |
| Variable 2 | Continuous units | N | xx | xx | xx |
|  |  | Minimum | xx.xx | xx.xx | xx.xx |
|  |  | Q1 | xx.xx | xx.xx | xx.xx |
|  |  | Median | xx.xx | xx.xx | xx.xx |
|  |  | Q3 | xx.xx | xx.xx | xx.xx |
|  |  | Maximum | xx.xx | xx.xx | xx.xx |
|  |  | Missing | xx | xx | xx |

## Headers, merges, font sizes, justification & styling

The minimal example is nice enough, but ideally we would want to format the table to look a little more professional. Some of these steps are accomplished through the huxtable package and other wait until rendering with pharmaRTF.

Let’s start by adding a header row over the treatment groups. We will do this by creating a new huxtable of the header and then adding it to the main table. When we compile the final table with pharmaRTF this header row will be carried over for each page.

# Create a header huxtable  
head\_ht <- c(rep("", 3), rep("Treatment Arm", 3))  
  
last\_row <- nrow(ht) + 2  
  
ht <- ht %>%  
 huxtable::hux() %>%  
 insert\_row(head\_ht, after = 0, fill = "") %>%  
 huxtable::merge\_cells(row = 1, col = group\_cols)   
  
ht

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | Treatment Arm | | |
| Variable | Level | Statistic | All Patients | Placebo | Treatment |
| Variable 1 | Category 1 | N (%) | xx (xx) | xx (xx) | xx (xx) |
|  | Category 2 | N (%) | xx (xx) | xx (xx) | xx (xx) |
|  | Category 3 | N (%) | xx (xx) | xx (xx) | xx (xx) |
| Variable 2 | Continuous units | N | xx | xx | xx |
|  |  | Minimum | xx.xx | xx.xx | xx.xx |
|  |  | Q1 | xx.xx | xx.xx | xx.xx |
|  |  | Median | xx.xx | xx.xx | xx.xx |
|  |  | Q3 | xx.xx | xx.xx | xx.xx |
|  |  | Maximum | xx.xx | xx.xx | xx.xx |
|  |  | Missing | xx | xx | xx |

Now we can add some styling to the table. huxtable provides a standardized syntax for styling tables: set\_feature(ht, row, col, value) where row and col can be single values or ranges, and value is the desired style (e.g., bold, font size, alignment). Using this syntax, we can add some horizontal style lines to our table, change the fonts, and set the alignment of the values.

# Alignment  
ht <- ht %>%  
 huxtable::set\_top\_padding(0.2) %>%  
 huxtable::set\_bottom\_padding(0.2) %>%  
 huxtable::set\_align(row = 1, col = group\_cols, "center") %>%  
 huxtable::set\_align(row = everywhere, col = 1:3, "left") %>%  
 huxtable::set\_align(row = everywhere, col = group\_cols, "center")  
  
# Fonts  
ht <- ht %>%  
 huxtable::set\_font\_size(row = everywhere, col = everywhere, 10) %>%   
 huxtable::set\_font\_size(row = 1:2, col = everywhere, 12) %>%   
 huxtable::set\_bold(row = 1:2, col = everywhere, TRUE) %>%  
 huxtable::set\_font("arial")  
  
  
# horizontal styling lines  
ht <- ht %>%  
 huxtable::set\_top\_border(row = 1, col = everywhere, 1) %>%  
 huxtable::set\_bottom\_border(row = 1, col = group\_cols, 1) %>%  
 huxtable::set\_bottom\_border(row = 2, col = group\_cols, 1) %>%  
 huxtable::set\_bottom\_border(row = last\_row, col = everywhere, 1)   
  
  
ht

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Treatment Arm** | | |
| **Variable** | **Level** | **Statistic** | **All Patients** | **Placebo** | **Treatment** |
| Variable 1 | Category 1 | N (%) | xx (xx) | xx (xx) | xx (xx) |
|  | Category 2 | N (%) | xx (xx) | xx (xx) | xx (xx) |
|  | Category 3 | N (%) | xx (xx) | xx (xx) | xx (xx) |
| Variable 2 | Continuous units | N | xx | xx | xx |
|  |  | Minimum | xx.xx | xx.xx | xx.xx |
|  |  | Q1 | xx.xx | xx.xx | xx.xx |
|  |  | Median | xx.xx | xx.xx | xx.xx |
|  |  | Q3 | xx.xx | xx.xx | xx.xx |
|  |  | Maximum | xx.xx | xx.xx | xx.xx |
|  |  | Missing | xx | xx | xx |

## Final export to RTF (with pharmaRTF)

For clinical submissions, titles and footnotes belong in the RTF header/footer areas, paginate correctly, and repeat appropriately—this is where pharmaRTF shines. It consumes your huxtable and attaches header/footer lines, page size, orientation, and more.

title1 = "Table 1. Baseline demographics and characteristics"  
title2 = "Sample population"  
footnote1 = "Brought to you by ProCogia and the Pharmaverse"  
  
ht %>%  
 set\_width(1.5) %>%  
 pharmaRTF::rtf\_doc(header\_rows = 1) %>%  
 add\_titles(hf\_line(title1, "PAGE\_FORMAT: Page %s of %s",  
 bold = TRUE, align = "split",  
 font = "arial", font\_size = 12)) %>%  
 add\_titles(hf\_line(title2, "DATE\_FORMAT: %H:%M %A, %B %d, %Y",  
 bold = TRUE, align = "split",  
 font = "arial", font\_size = 11)) %>%  
 add\_footnotes(hf\_line(footnote1, align = "left",  
 font = "arial", font\_size = 10)) %>%  
 pharmaRTF::write\_rtf("table1.rtf")

