Paper OS07

Title: {tflmetaR}: An R Package to Manage Metadata in Statistical Outputs

Gary Cao, UCB Biopharma S.A., Raleigh, North Carolina
Amber Wu, UCB Biopharma S.A., Raleigh, North Carolina
Lan Cheng, UCB Biopharma S.A., Raleigh, North Carolina
Alberto Montironi, UCB Farchim S.A., Bulle, Switzerland
Yao Lu, UCB Biopharma S.A., Raleigh, North Carolina

ABSTRACT

Best practices in programming recommend separating code from metadata to improve readability, maintainability, and scalability. However, many R scripts used for clinical reporting still hardcode headers, titles, and footnotes, resulting in a codebase that is challenging to manage and extend.

To bridge this gap, UCB has developed **{tflmetaR}**, an R package designed to simplify and centralize the management of annotation text for tables, listings, and figures (commonly referred to as TFLs or TLGs) in clinical study reports (CSRs). With a single function call, the package retrieves headers, titles, and footnotes from a metadata repository, simplifying TFL annotation. Although independent and self-contained, the UCB **{tflmetaR}** is fully compatible with the UCB **{gridify}** package and integrates seamlessly with the Pharm averse suite. It has become a key component of the UCB's R-based workflow for producing submission-ready statistical deliverables.

INTRODUCTION

The production of Clinical Study Reports (CSRs) remains a cornerstone of regulatory submissions in the pharmaceutical industry. A core component of any Clinical Study Report (CSR) is the set of Tables, Listings, and Figures (TFLs), which communicate the clinical trial findings in a clear and systematic manner. These outputs are not standalone; they require extensive annotation—including headers, titles, and footnotes—to provide context, define populations, and cite data sources.

In many programs or packages developed to generate TFLs, this annotation text is often hardcoded directly (e.g., in R scripts). While this approach may seem straightforward for a single output, it creates significant inefficiencies and risks at an enterprise level. As protocols are amended, analysis populations updated, or textual requirements refined, statistical programmers must manually edit these hardcoded strings across potentially hundreds of scripts. This process is error-prone, difficult to track, and hinders the scalability and reuse of validated code.

This paper introduces **{tflmetaR}**, an R package developed by UCB to solve this problem by separating metadata from analysis scripts, and establishing a "single source of truth" for TFL annotations.

BACKGROUND: THE CHALLENGE OF HARD CODED ANNOTATIONS

Embedding TFL annotations within analysis programs introduces several operational and quality-related challenges for statistical programming teams. Even minor revisions—such as updates to data cutoff dates or modifications to population footnotes—must be manually applied across multiple programs. This increases the likelihood of inconsistent phrasing and hinders efforts to maintain standardized annotations across outputs. In the absence of a centralized repository for annotation metadata, establishing and preserving such standardization becomes particularly difficult. Moreover, embedding annotation content within code obscures the provenance of individual footnotes, complicates the review and approval process, and limits the ability to version control annotation modifications over time.

This hard coding approach mixes annotation content with analytic logic and prevents programmers from more efficiently leveraging reusable, validated code across studies, requests, or projects. Collectively, these limitations underscore the need for a programmatically driven framework that cleanly separates TFL generation logic from the metadata that governs titles, footnotes, and related annotations.

MANAGING TFL ANNOTATION METADATA USING tflmetaR

The **{tflmetaR}** package provides a robust framework for managing TFL metadata by introducing a consistent interface to external metadata repositories, typically stored as Excel spreadsheets or CSV files. This approach makes the metadata repository the single source of truth.

Methodology and Workflow

The core methodology of **{tflmetaR}** is to read, select, and retrieve annotations in an accurate, consistent and reproducible manner. The package's workflow can be broken down into the following key steps:

1. **Uploading Metadata**:

The metadata file serves as a centralized repository for all TFL annotations, enabling consistent application across multiple outputs and facilitating updates without TFL program modifications.

The metadata file should contain standardized columns including PGMNAME (program name), TTL1 (primary title), FOOT1 (first footnote), and SOURCE (data source). Additional columns for subtitles (TTL2, TTL3, …) and additional footnotes (FOOT2, FOOT3, …), population definitions, and bylines are also supported. If it has different

column names, the `change_colname()` helper function can be utilized to standardize them. The detailed steps are explained in the function documentations.

{tflmetaR} imports the metadata file containing titles and footnotes etc. in Excel format with `read_tfile()`, or in CSV format with `read_tfile_csv()`. Additional module could be added in future releases to handle data in other format (e.g., JSON format) as the need arises.

```
# Load the sample metadata file included with the package
file_path <- system.file("extdata", "sample_titles2.xlsx", package = "tflmetaR")

# Read the metadata from the 'footer' sheet
meta <- tflmetaR::read_tfile(filename = file_path, sheetname = "footer")
```

2. **Metadata Selection and Function Based Annotation Content Retrieval**

To avoid ambiguity, the package provides selection functions designed for specific retrieval scenarios: select_with_name() selects a metadata entry by its unique program name and optional OID (e.g., "t_ae_001"), while select_with_number() selects a metadata entry by output type and TFL number (e.g., "Figure 14.1.1"). These helper functions enforce a single-row selection for the desired TFL, emitting an informative error if the specified ID is missing or duplicated. This ensures the retrieval of correct metadata.

Once the TFL-specific metadata row is selected, a suite of helper functions is invoked to extract metadata for each layout region. As illustrated in the use-case section, these functions collectively cover the full report annotation need. The functions get_ulheader() and get_urheader() retrieve content for the upper-left (typically project, study, or request-specific information) and upper-right (status and data cutoff date) header regions. The functions get_title(), get_pop(), and get_byline() return the main title, analysis population, and sub-population descriptions. Finally, get_footnote(), get_source(), and get_pgmname() manage the footer elements, including data references and the program name with associated timestamps. Each function returns a named list containing the non-missing fields, allowing flexible integration with various table rendering packages.
The helper functions improve consistency, traceability, and regulatory compliance. For instance, get_footnote(add_footr_tstamp=TRUE) adds a timestamp and the program name to the footer, creating a direct link between the output, the script generating it, and the time of creation.

Table 1: Metadata Retrieval Functions in tflmetaR

Function	Description	Returns
`get_title()`	Retrieves title-related fields (TTL1, TTL2, etc.) from the metadata	Named list of non-missing title fields
`get_footnote()`	Retrieves footnote fields (FOOT1, FOOT2, etc.) from the metadata	Named list of non-missing footnote fields
`get_source()`	Retrieves source fields (e.g., SOURCE1) from the metadata	Named list of non-missing source fields
`get_pop()`	Retrieves population field from the metadata	Named list containing the population field
`get_byline()`	Retrieves byline fields (BYLINE1, BYLINE2, etc.) from the metadata	Named list of non-missing byline fields
`get_pgmname()`	Retrieves program name field (PGMNAME) from the metadata	Named list containing the program name field

Function Invocation examples:

```
# Extract metadata components for a specific table
# Tables are identified by their TFL number (e.g., "Table 2.1")
tfl_number <- "Table 2.1"

title_info <- tflmetaR::get_title(meta, tnumber = tfl_number)
footnotes <- tflmetaR::get_footnote(meta, tnumber = tfl_number, add_footr_tstamp = FALSE)
source_info <- tflmetaR::get_source(meta, tnumber = tfl_number)
population <- tflmetaR::get_pop(meta, tnumber = tfl_number)
pgm_name <- tflmetaR::get_pgmname(meta, tnumber = tfl_number)
```

3. **TFL Annotation:**

The strings retrieved in previous steps are then passed as variables to TFL-generating packages. {tflmetaR} has been designed to be compatible with virtually any such packages which annotate outputs with string vectors or lists. For instance, it integrates seamlessly with R packages **{gt}**, **{flextable}**, **{ggplot2}**, and UCB's own **{gridify}** package, allowing programmers to integrate the metadata into the final output dynamically.

This workflow standardizes the metadata management process, allowing the analysis script to focus solely on TFL generation logic. Below is the sample code where **{tflmetaR}** is used with **{gt}**:

```r
# Extract annotation text from metadata
main_title <- title_info$TTL1[[1]]
subtitle <- title_info$TTL2[[1]]
pop_text <- population$POPULATION[[1]]

# Extract multiple footnotes
foot1 <- footnotes$FOOT1[[1]]
foot2 <- footnotes$FOOT2[[1]]

source_text <- source_info$SOURCE[[1]]
# Create the annotated gt table
sample_data %>%
  gt::gt() %>%
  gt::tab_header(
    title = main_title,
    subtitle = gt::html(paste0(subtitle, "<br>", pop_text))
  ) %>%
  gt::tab_footnote(
    footnote = foot1,
    locations = gt::cells_column_labels(columns = mpg)
  ) %>%
  gt::tab_footnote(
    footnote = foot2,
    locations = gt::cells_column_labels(columns = cyl)
  ) %>%
  gt::tab_source_note(
    source_note = paste("Source:", source_text)
  ) %>%
  gt::cols_label(
    mpg = "Miles/Gallon",
    cyl = "Cylinders",
    hp = "Horsepower",
    wt = "Weight (1000 lbs)"
  )
```

Table 2: Example table output using **{tflmetaR}** and **{gt}**

Table 2.1

Sample Table with Penguin Data
Penguin Population

Miles/Gallon[1]	Cylinders[2]	Horsepower	Weight (1000 lbs)
21.0	6	110	2.6
21.0	6	110	2.9
22.8	4	93	2.3
21.4	6	110	3.2
18.7	8	175	3.4

[1] ES = Enrolled Set

[2] Reference: Listing 2.1

Source: palmerpenguins

4. **Filenames and Bookmarks Creation:**

The `get_bookm()` function extracts and constructs a bookmark string from the metadata file. It first looks for a "bookm" column; if not found or if empty, it uses the TFL title instead. Any non-ASCII characters are removed from the bookmark so that it can be safely used as a filename or PDF bookmark. Furthermore, if the length of the bookmark exceeds 128 characters, phrases are shortened using an external abbreviation lookup table stored in `abbrev.xlsx` file when necessary.

A {tflmetaR} + {gridify} USE CASE FOR FIGURE ANNOTATION

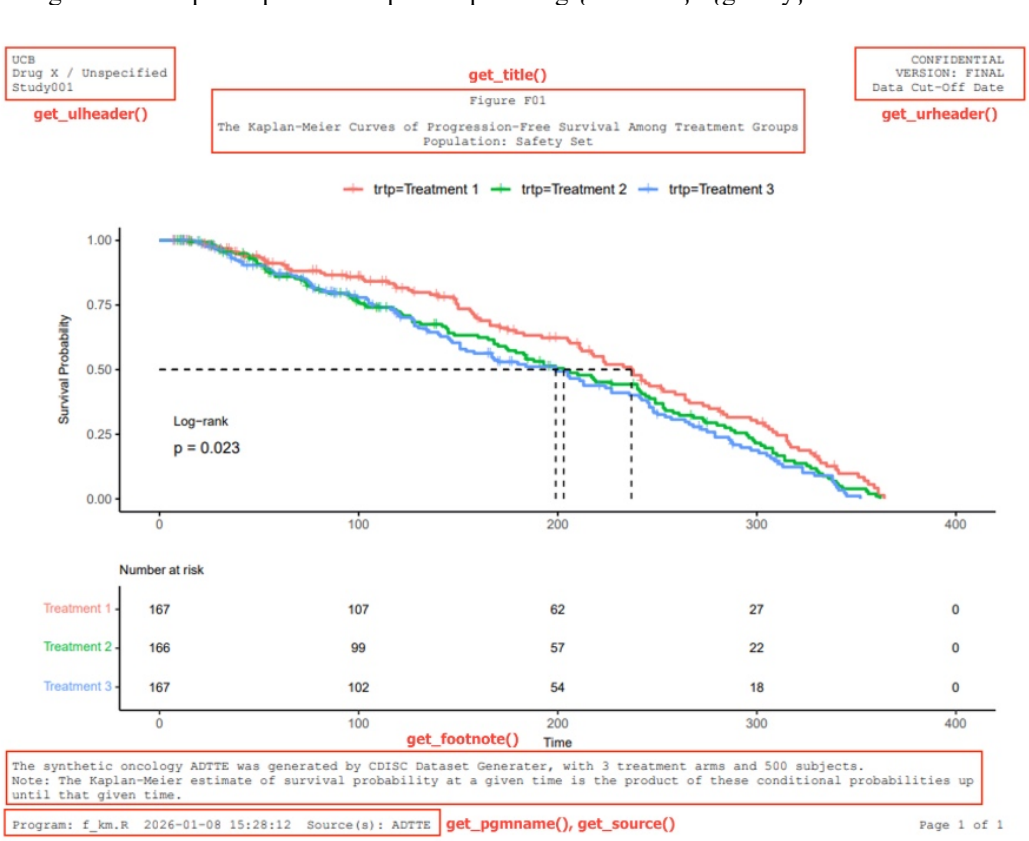
The {tflmetaR} framework is designed to accommodate any annotation requirements of clinical data visualizations. This combination of {tflmetaR} and {gridify} allows customized annotation in the output including header section, which is not included in the {gt} table example above.

```
ulheader <- tflmetaR::get_ulheader(header_file)[1, ]
urheader <- tflmetaR::get_urheader(header_file)[1, ]
titles <- tflmetaR::get_title(title_file, tnumber = fig_number)
footnotes <- tflmetaR::get_footnote(title_file, tnumber = fig_number, add_footr_tstamp = FALSE)
pgmname <- tflmetaR::get_pgmname(title_file, tnumber = fig_number)
source <- tflmetaR::get_source(title_file, tnumber = fig_number)

# convert footnote list to string
footnote_str <- paste(unlist(footnotes), collapse = "\n")
```

Having loaded the metadata and extracted the titles, footnotes, and header components through the standardized function calls, the workflow proceeds to the final rendering phase. Now we combine {gridify} with {tflmetaR}-extracted metadata to create the final publication-quality Kaplan-Meier survival plot. This approach ensures that any updates to titles or footnotes in the metadata spreadsheet are automatically reflected in the output.

```
fig2 <- gridify(
  p1,
  layout = pharma_layout_base(
    margin = grid::unit(c(0.5, 1, 0.25, 1), "inches"),
    global_gpar = grid::gpar(fontfamily = "Courier")
  )
) |>
  set_cell("header_left_1", ulheader[[1]]) |>
  set_cell("header_left_2", ulheader[[2]]) |>
  set_cell("header_left_3", ulheader[[3]]) |>
  set_cell("header_right_1", urheader[[1]]) |>
  set_cell("header_right_2", urheader[[2]]) |>
  set_cell("header_right_3", urheader[[3]]) |>
  set_cell("output_num", titles[[1]]) |>
  set_cell("title_1", "") |>
  set_cell("title_2", titles[[2]]) |>
  set_cell("title_3", titles[[3]]) |>
  set_cell("note", footnote_str, mch = 130) |>
  set_cell("footer_left", paste0(
    "Program: ", pgmname, ".R  ",
    format(Sys.time(), "%Y-%m-%d %H:%M:%S"), "  Source(s): ", source
  )) |>
  set_cell("footer_right", sprintf("Page %d of %d", 1, 1))

gridify::export_to(fig2, "./f_surv_tflmetar.pdf")
```

Figure 1: Example Kaplan-Meier plot output using {tflmetaR}+{gridify} with illustrations



VALIDATION AND INTEROPERABILITY

To ensure robustness, **{tflmetaR}** is developed with a comprehensive testthat suite covering both expected behavior ("happy paths") and error cases, such as when metadata is missing. It also has been used and verified in one study report. This improves confidence for production use.

To ensure robustness, {tflmetaR} includes a comprehensive testthat suite that validates both standard (“happy-path”) functionality and error handling, such as scenarios with incomplete or missing metadata. The package has been applied and verified in a recent study-report development workflow, which further increasing confidence in its readiness for production use.
While self-contained, the package is designed for interoperability. It adheres to tidyverse conventions and integrates readily with the Pharmaverse suite, positioning it as a valuable component in a modern, R-based statistical reporting environment.

IMPACT

The primary impact of adopting **{tflmetaR}** is a significant gain in programming efficiency. Manual edits for metadata updates are drastically reduced. Updates across entire studies can be performed in minutes by editing a single metadata file. This workflow facilitates template-driven and metadata driven creation of CSRs and opens opportunities for more advanced, version-controlled metadata repositories.

PACKAGE STRUCTURE AND DESIGN

{tflmetaR} follows conventional R package design principles. The package is structured with documented functions in the R/ directory, testthat in tests/testthat/, and detailed vignettes that document end-to-end usage. Sample metadata files are included in inst/extdata/ to showcase expected formats.

Development follows tidyverse conventions for data manipulation and relies on roxygen2 (via devtools::document()) for managing documentation and the NAMESPACE file.

CONCLUSION

{tflmetaR} provides a unified framework for managing annotations by externalizing metadata into structured spreadsheets. This allows updates to titles and footnotes to be applied consistently across all TFLs without modifying validated analysis code, thereby reducing maintenance burdens. By integrating with TFL rendering tools such as {gt} and {gridify}, {tflmetaR} enables the efficient production of publication-quality TFLs that meet the regulatory requirements of clinical research.
The {tflmetaR} package is a pragmatic, production-tested solution for metadata-driven TFL annotation. By documenting its architecture, sample use cases, and validation strategy, we can provide the R programming community with a replicable model for decoupling annotation contents from report programming code.

CONTACT INFORMATION

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

Gary Cao, UCB Biopharma, Gary.Cao@ucb.com
Amber Wu, UCB Biopharma, Amber.Wu@ucb.com
Lan Cheng, UCB Biopharma, Lan.Cheng@ucb.com
Alberto Montironi, UCB Biopharma, Alberto.Montironi@ucb.com
Yao Lu, UCB Biopharma, Yao.Lu@ucb.com

DISCLAIMERS

The views, thoughts, and opinions expressed in this publication belong solely to the author(s) and do not necessarily reflect the views or positions of the authors' company, affiliates, or employees.

REFERENCES

Nasinski M, Wall A, Robson S, Dash P, Winick-Ng J (2025). *gridify: Enrich Figures and Tables with Custom Headers and Footers and More*. R package version 0.7.6, https://pharmaverse.github.io/gridify/.

Wickham, Hadley, and Jenny Bryan. n.d. *R Packages (2e): Organize, Test, Document, and Share Your Code*. Accessed [Date you accessed the site]. https://r-pkgs.org.