

Bridging gtsummary and reporter: Submission-Ready Clinical Outputs with derrick

Xiecheng Gu, Bill Huang, Joe Zhu

1. Boehringer Ingelheim
2. Parexel
3. Roche

July 2026

Abstract

Clinical reporting in R often divides naturally into two layers. Packages such as `gtsummary`, `crane`, `cardinal`, `cards`, and `cardx` are well suited to defining table content and statistical structure, whereas `reporter` is designed to control how a clinical report is laid out and written to a file. The practical difficulty is the repeated translation between these layers. Column labels, spanning headers, indentation, widths, pagination, titles, footnotes, and output-specific rules can otherwise require a separate setup for every table.

`derrick` turns that translation into a reusable package interface. Its main function, `gtsummary_reporter()`, accepts either a `gtsummary` object or a report-ready data frame, interprets the available display metadata, applies shared clinical layout rules, and delegates file generation to `reporter`. A single call can produce RTF, TXT, DOCX, PDF, or HTML output while preserving the table definition established upstream. The package is the production extension of an earlier PharmaRUG

prototype: the original function demonstrated that the two ecosystems could be connected; `derrick` adds validation, documentation, compatibility handling, tests, and a versioned distribution model. This paper describes the design of the bridge, the three supported table-construction paths, the translation mechanism, the report controls provided by the package, and the engineering practices that make the workflow reusable across clinical reporting programs.

Keywords: R, gtsummary, reporter, clinical reporting, RTF, TXT, DOCX, PDF, HTML

1. Clinical reporting is a two-layer problem

Clinical tables must satisfy two different concerns. First, the program must define the analysis result: variables, populations, denominators, statistics, labels, row hierarchy, and the order in which results are presented. Second, the program must render that result into a controlled report: page geometry, column widths, wrapping, spanning headers, repeated labels, pagination, titles, footnotes, and file format.

The first concern is where `gtsummary` and related analysis packages are strongest. They provide expressive interfaces for summaries, regression results, stacking, merging, and table-body customization. The second concern is the domain of `reporter`, which brings report-oriented concepts familiar to users of SAS PROC REPORT into an R workflow. It manages the physical table and the surrounding report, including RTF and fixed-width TXT output.

The design principle behind `derrick` is therefore deliberately narrow:

Use upstream packages to define the table; use reporter to control its delivery; use derrick to make the handoff repeatable.

This separation keeps statistical logic independent from file-layout logic. It also allows a table to evolve without forcing every output program to reimplement the same translation.

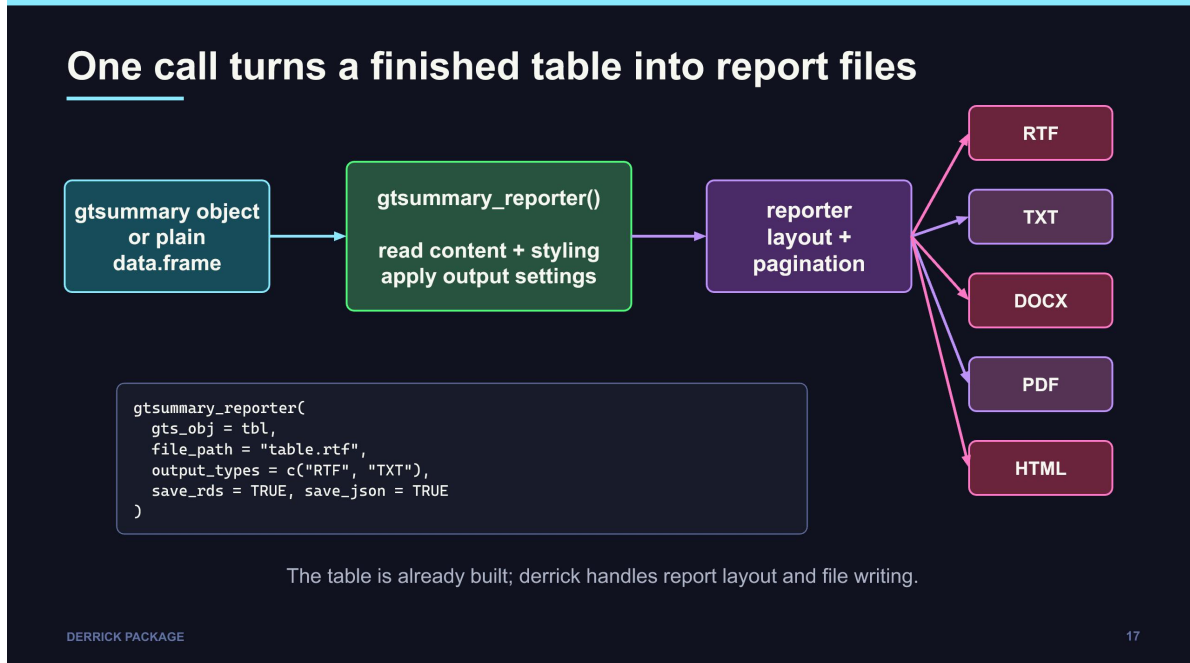


Figure 1. *derrick* accepts a *gtsummary* object or plain data frame, applies report layout through *gtsummary_reporter()* and *reporter*, and writes the requested output files.

2. Complementary strengths of *gtsummary* and *reporter*

gtsummary is not intended to be a full clinical report engine, and *reporter* is not intended to replace statistical table builders. Their strengths are complementary. The resulting gap is less about statistical capability than about repeated last-mile work.

Reporting requirement	Upstream table alone	What derrick adds through reporter
Clinical-style RTF and fixed-width TXT	Not the primary output responsibility of <i>gtsummary</i>	Creates a report object and writes clinical-style files
Page geometry	Usually handled after table construction	Applies paper size, orientation, margins, font size, titles, footnotes, and footer metadata together
Column widths and wrapping	Often requires output-specific formatting	Uses reporter width calculation by default and accepts manual width overrides when required
Row hierarchy	Indentation may exist as table metadata	Preserves hierarchy and passes output-appropriate indentation to the report layer
Spanning headers	Stored in table styling	Translates spans into reporter instructions instead of rebuilding them manually

Reporting requirement	Upstream table alone	What derrick adds through reporter
Vertical pagination	Commonly added outside the table definition	Supports automatic pagination, optional row limits, page breaks, and page-by subtitles
Wide-table pagination	Requires custom page grouping	Can wrap horizontally at defined column groups and repeat identifying columns
Multiple delivery formats	Often needs separate output code	Uses one interface for RTF, TXT, DOCX, PDF, and HTML
Reuse across study teams	Helper scripts tend to be copied and edited	Provides an installable, documented, tested package interface

The distinction matters. `derrick` does not change the analysis represented by a table. It translates an existing table definition into a report specification and applies shared output rules.

3. Three construction paths, one report engine

The package supports three common clinical reporting paths.

3.1 Path 1: a `gtsummary` object

When `gtsummary`, `crane`, or `cardinal` already produces the required table shape, the object can be passed directly to `gtsummary_reporter()`. This is the shortest path because the display values and most styling metadata are already present.

3.2 Path 2: structured summaries assembled as a `gtsummary` object

For more specialized tables, `cards` or `cardx` can calculate structured summaries that are then rendered or combined as a `gtsummary` object. From the reporting layer's perspective, the final handoff is the same as Path 1. This path is useful when the analysis needs more control than a standard summary function provides while retaining the downstream `gtsummary` structure.

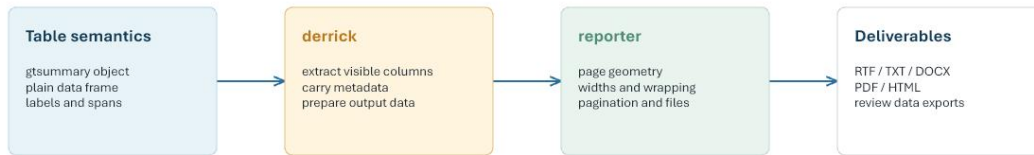
3.3 Path 3: a report-ready data frame

Some clinical displays require custom derivations or a layout that is easier to build directly with data manipulation. In that case, `gtsummary_reporter()` accepts the final data frame together with explicit labels, spanning-header definitions, and other report arguments. This keeps custom table construction possible without creating a separate output engine.

The choice is driven by table structure, not by output format. All three paths end in the same `reporter-`managed delivery layer.

OUTPUT BRIDGE

The package sits between table semantics and physical report layout



gtsummary answers what the table contains; **reporter** answers how the report is laid out; **derrick** makes the transfer explicit and testable.

derrick draft deck

05

Figure 2. Three input paths converge on one report-generation interface.

4. Translating a gtsummary object

A `gtsummary` table separates display values from display instructions. Its `table_body` component contains the rows and statistic columns. Its `table_styling` component records information such as visible columns, display labels, alignment, indentation, formatting, and spanning headers.

`derrick` reads both components and converts them into the objects expected by `reporter`:

1. Select the visible output columns from `table_body`.
2. Apply display formatting and row indentation already defined by the table.
3. Translate column labels and alignment into `reporter::define()` calls.
4. Translate spanning-header metadata into `reporter::spanning_header()` calls.
5. Add optional page-by, page-break, and horizontal-wrap instructions.
6. Place the table in a report with titles, footnotes, page settings, and footer information.
7. Write each requested file type from the same report specification.

This mapping is the core of the package. It removes the need to repeat the extraction and translation code in every table program while leaving the original table object available for inspection before export.

5. One public interface

The main entry point is `gtsummary_reporter()`. The minimal call requires a table and a destination path:

```
library(dplyr)
library(gtsummary)
library(derrick)

tbl <- gtsummary::trial |>
  dplyr::select(trt, age, grade) |>
  gtsummary::tbl_summary(by = trt) |>
  gtsummary::add_p() |>
  gtsummary::modify_spanning_header(
    gtsummary::all_stat_cols() ~ "**Treatment Group**"
  )

output_path <- file.path(tempdir(), "table-14-1.rtf")

gtsummary_reporter(
  gts_obj = tbl,
  file_path = output_path,
  output_types = c("RTF", "TXT", "DOCX", "PDF", "HTML")
)
```

The same interface also accepts a plain data frame. When needed, callers can provide `column_labels`, `spanning_headers`, `page_by_columns`, `id_columns`, or manual page wrapping directly. Defaults are intended to cover the usual case; table programs only need to expose layout arguments when the study shell requires a deliberate override.

**Table 14.1.1 Summary of Demographics
Safety Population**

	Treatment Group		p-value
	Drug A N = 98	Drug B N = 102	
Age	46 (37, 60)	48 (39, 56)	0.7183
Unknown	7	4	
Grade			0.8708
I	35 (36%)	33 (32%)	
II	32 (33%)	36 (35%)	
III	31 (32%)	33 (32%)	

Percentages are based on subjects in each treatment group.

Figure 3. Example clinical output produced from a gtsummary table definition.

6. Report controls handled by derrick

6.1 Widths and wrapping

When no manual widths are supplied, `derrick` lets `reporter` calculate suitable column widths for the target format using page geometry, font metrics, headers, and cell content. This is especially important because a character budget that is useful for TXT is not an accurate substitute for physical width in RTF, Word, or PDF.

Manual `column_widths` and `max_table_width` remain available for study-specific shells. Supplied widths are checked against the usable page width and reduced when necessary. TXT can additionally use `max_chars_per_line` to enforce a fixed-width character budget.

6.2 Hierarchy, labels, and spanning headers

Clinical displays often communicate hierarchy through leading indentation rather than through a separate stub object. `derrick` preserves the row labels generated by the upstream table and converts styling metadata into report definitions. Column labels and spanning headers are carried across the same handoff, which keeps the report synchronized with the analysis table.

6.3 Vertical and horizontal pagination

Routine vertical pagination is delegated to `reporter`, which can calculate breaks from output-specific line metrics. A table may also request explicit row chunks, page-by subtitles, or hidden page-break variables.

For wide tables, `derrick` supports horizontal page wrapping. Statistic columns can be grouped manually, or automatic wrapping can use spanning-header boundaries. An identifying label column can be marked as an ID column so that it repeats on each horizontal page. This makes wide displays readable without requiring the caller to construct a separate table for every page.

6.4 Titles, footnotes, and review outputs

Titles, footnotes, program names, and timestamps can be supplied with the same table call. Optional data exports preserve the display-ready values and report metadata for review. These artifacts complement, rather than replace, independent verification of the underlying analysis.

TABLE EXAMPLES

Report-like outputs are the visible payoff

Demographics summary				Adverse event frequency			Lab shift by visit					
Characteristic	Placebo N (%)	Study Drug N (%)	Total N (%)	System Organ Class Preferred Term	Placebo N (%)	Study Drug N (%)	Treatment	Visit / Reference Range Indicator	Low	Normal	High	Total
Number of patients	2693 (100)	2740 (100)	5433 (100)	Total with adverse events	1721 (63.9)	1818 (66.4)	Placebo	Week 4 Low	22 (8.1)	34 (12.6)	3 (1.1)	59
Age, years				Gastrointestinal disorders	421 (15.6)	465 (17.0)		Normal	18 (6.7)	180 (66.7)	12 (4.4)	210
N	2693	2740	5433	Nausea	186 (6.9)	218 (8.0)		High	1 (0.4)	15 (5.6)	8 (3.0)	24
Mean (SD)	57.8 (12.4)	58.1 (12.1)	58.0 (12.3)	Diarrhoea	154 (5.7)	171 (6.2)		Last value Low	28 (10.4)	40 (14.8)	4 (1.5)	72
Sex				Nervous system disorders	390 (14.5)	407 (14.9)	Study Drug	Week 4 Low	18 (6.6)	41 (15.0)	6 (2.2)	65
Female	1398 (51.9)	1417 (51.7)	2815 (51.8)	Headache	241 (8.9)	252 (9.2)		Normal	16 (5.8)	191 (69.7)	9 (3.3)	216
Male	1295 (48.1)	1323 (48.3)	2618 (48.2)	Dizziness	96 (3.6)	104 (3.8)		High	2 (0.7)	13 (4.7)	11 (4.0)	26

The examples emphasize the output layer: row hierarchy, treatment headers, compact N (%) display, and clinical labels survive the handoff.

derrick draft deck

04

Figure 4. Examples of demographics, adverse-event, and laboratory outputs using a shared delivery layer.

7. From a working function to a reusable package

The PharmaRUG prototype established that the translation was possible. Package engineering made it dependable enough to share. The package form adds several controls that are difficult to maintain in a copied helper script:

- **Input validation.** Table objects, output types, file paths, and layout arguments are checked before report generation.
- **Version-aware behavior.** Optional `reporter` arguments are used only when the installed version supports them, reducing breakage across environments.
- **Regression tests.** Tests cover table extraction, header normalization, indentation, spanning headers, widths, page-by behavior, pagination controls, and file-writing paths.
- **Documentation.** One public function, examples, a vignette, and package help replace project-specific setup instructions.
- **Release discipline.** Versioned source, package checks, issue tracking, and CRAN distribution give teams a shared dependency rather than a collection of copied scripts.

The analysis program remains responsible for constructing the correct table. The package removes repeated output setup and gives that setup a testable home.

8. Distribution through CRAN and openpharma

Version 0.1.1 is available through CRAN, while the public [openpharma/derrick](#) repository provides source code, issue tracking, contribution history, and ongoing development. CRAN supplies a standard installation and package-check surface; openpharma supplies the collaborative development surface.

```
install.packages("derrick")
library(derrick)
```

derrick: Export 'gtsummary' Tables with 'reporter'

Convert 'gtsummary' tables and plain data frames into clinical-style Rich Text Format, plain text, Word, PDF, and HTML outputs using 'reporter', with managed widths, pagination, indentation, spanning headers, and optional intermediate data exports.

Version: 0.1.1
 Depends: R (≥ 4.4)
 Imports: [dplyr](#), [purrr](#), [reporter](#) (≥ 1.4.7), [rlang](#), [stringi](#), [tibble](#)
 Suggests: [cards](#) (≥ 0.8.0), [gtsummary](#) (≥ 2.5.1), [knitr](#), [rmarkdown](#), [testthat](#) (≥ 3.3.0)
 Published: 2026-07-10
 DOI: [10.32614/CRAN.package.derrick](#)
 Author: Xiecheng Gu [aut, cre], Joe Zhu [aut], Bill Huang [aut]
 Maintainer: Xiecheng Gu <guxc0405med at 163.com>
 BugReports: <https://github.com/openpharma/derrick/issues>
 License: [Apache License 2.0](#)
 URL: <https://github.com/openpharma/derrick/>
 NeedsCompilation: no
 Language: en-US
 Materials: [NEWS](#)
 CRAN checks: [derrick results](#)

Documentation:

Reference manual: [derrick.html](#), [derrick.pdf](#)
 Vignettes: [Exporting Clinical Tables with derrick \(source, R code\)](#)

Downloads:

Package source: [derrick_0.1.1.tar.gz](#)

Windows binaries for x64: [derrick_0.1.1_x64.exe](#), [derrick_0.1.1_x64.exe](#)

Figure 5. derrick package information on CRAN.

Platform Solutions Resources Open Source Enterprise Pricing

openpharma / derrick Public

Notifications Fork 0 Star 2

<> Code Issues 2 Pull requests Actions Projects Security and quality Insights

main 7 Branches 0 Tags

Go to file

Code

dustreturn Add Boehringer Ingelheim copyright holder ✓ f86e5f3 · last week 55 Commits

.github	chore: wire up pkgdown config	2 months ago
R	Address CRAN resubmission feedback	3 weeks ago
inst	Address CRAN DESCRIPTION feedback	last month
man	Address CRAN resubmission feedback	3 weeks ago
tests	Address CRAN resubmission feedback	3 weeks ago
vignettes	Let reporter manage automatic widths	2 months ago
.Rbuildignore	Address CRAN DESCRIPTION feedback	last month

About

No description, website, or topics provided.

Readme
 Code of conduct
 Contributing
 Activity
 Custom properties
 2 stars
 0 watching
 0 forks
 Report repository

Figure 6. The derrick source repository under openpharma.

9. GitHub Copilot-assisted package development

GitHub Copilot accelerated the move from the bridge concept to a maintained package. It was used as an engineering assistant across several activities: organizing reusable project guidance, navigating the existing codebase, implementing focused changes, drafting tests and documentation, and preparing release checks. The useful unit of work was not a single prompt; it was a project context containing source files, tests, prior decisions, and review feedback.

Human review remained central. Package behavior, clinical-reporting conventions, compatibility decisions, test expectations, and release approval were decided and validated collaboratively. In this workflow, AI reduced the cost of iteration while the authors retained responsibility for design and verification.

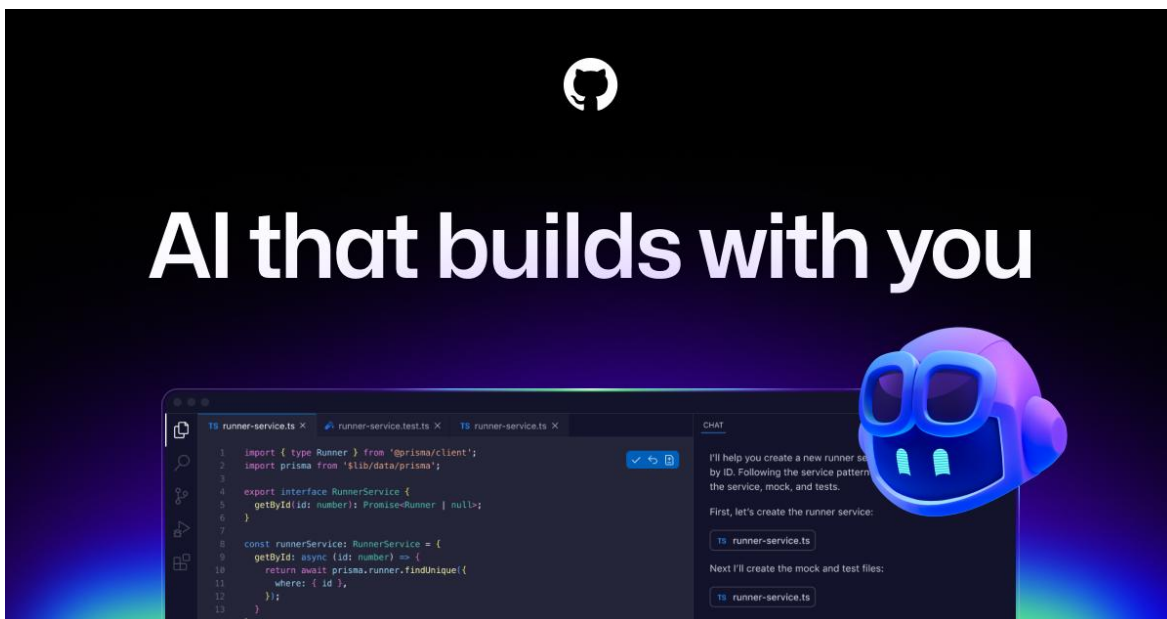


Figure 7. GitHub Copilot was used in the engineering loop from idea to package.

10. Validation and quality controls

`derrick` is tested at the translation boundary, where regressions are most likely to affect report appearance. Structural tests inspect processed data, visible columns, labels, spans, indentation, width resolution, page-by metadata, and pagination rules. File-generation tests exercise both `gtsummary` and plain-data-frame inputs without requiring large binary fixtures in routine checks.

Visual review remains important because a structurally valid report can still be difficult to read. Development checks therefore complement unit tests with generated RTF and TXT examples, and with

DOCX, PDF, or HTML output when those formats are in scope. This division is intentional: unit tests protect the translation logic; rendered examples confirm that the final pages remain usable.

11. Limitations and future work

`derrick` begins after the analysis table has been defined. It does not determine the analysis population, derive statistics, or decide whether the table content is clinically appropriate. Highly specialized displays may still need upstream helper functions or a report-ready data frame.

Output quality also depends on the capabilities and version of `reporter` installed in the environment. Cross-format behavior should be reviewed whenever a study uses a new file type, font, page shell, or unusually wide table. Future work can expand the example gallery, add more compatibility tests, and provide additional end-to-end templates for common clinical displays.

12. Conclusion

`gtsummary` and related packages provide an expressive way to define clinical table content. `reporter` provides the controls needed to place that content on a page. `derrick` connects the two through a single reusable interface.

The package preserves the original division of responsibility while removing the repeated work between it. Users build the table once, call `gtsummary_reporter()`, and deliver it through a consistent clinical reporting layer. Compared with the earlier function prototype, the package adds the engineering structure needed for shared, versioned, and submission-oriented use.

References

1. Gu X, Zhu J, Huang B. *Make Clinical Reporting Easier: Enhancing gtsummary analysis outputs with reporter*. PharmaRUG 2026. https://pharmarug.github.io/pharmarug-2026/resources/slides/06_Make%20Clinical%20Reporting%20Easier%20Enhancing%20gtsummary%20analysis%20result%20dataset%20with%20reporter_xiecheng%20gu%20.html
2. Sjoberg DD, Whiting K, Curry M, Lavery JA, Larmarange J. Reproducible summary tables with the gtsummary package. *The R Journal*. 2021;13(1):570-580. <https://doi.org/10.32614/RJ-2021-053>
3. Bosak D. *reporter: Create and Render Reports*. R package documentation. <https://cran.r-project.org/package=reporter>

4. Gu X, Zhu J, Huang B. *derrick: Export gtsummary Tables with reporter*. <https://cran.r-project.org/package=derrick>
5. openpharma. *derrick source repository*. <https://github.com/openpharma/derrick>