ksTFL v0.11.x

Framework for Clinical Tables, Figures, and Listings

Core Workflow & Building Blocks

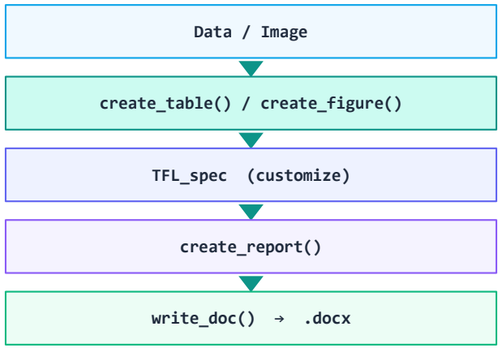
https://github.com/crow16384/ksTFL

Installation

For installation instructions see the package documentation at https://crow16384.github.io/ksTFL-release/

The Pipeline — From Data to DOCX

ksTFL separates metadata generation from rendering. You define structure & styles in R; the built-in C++ engine produces submission-quality DOCX with deterministic HarfBuzz pagination.



Quick Example

```
library(ksTFL)
spec <- create_table(mtcars, cols = c(mpg, cyl, hp))
spec <- add_title(spec, "Motor Trends")
spec <- add_footnote(spec, "Data: 1974.")
report <- create_report(spec)
write_doc(report, name = "my_table")
```

Three Spec Types

create_table(data, cols)

Table from data frame; cols with tidyselect

create_figure(path_or_ggplot)

Figure from file path or ggplot2 object

create_text()

Narrative text; add content with add_body_text()

Two-step workflow: save_report() → replay_report() lets you inspect JSON before rendering. Use **write_doc()** for one-step production rendering.

Customize Columns — define_cols()

After **create_table()**, customize column properties. Supports single, batch (recycled), and per-column values.

```
# Single column
spec <- define_cols(spec, mpg,
  label = "MPG", type = "numeric",
  format = "%.1f", colWidth = "20%")

# Batch: same value recycled to all cols
spec <- define_cols(spec, c(mpg, cyl, hp),
  label = c("MPG", "Cyl", "HP"),
  type = "numeric", format = "%.0f")
```

Parameter	Purpose
label	Column header text
type	"numeric" or "string"
format	Format string (e.g., "%.2f")

Spanning Headers — add_span_header()

Group columns under common headers. Uses tidyselect: ranges, starts_with(), contains(), negation (-).

```
spec <- add_span_header(spec,
  cols = c(mpg, cyl, hp),
  label = "Engine",
  stubOrder = 1,
  labelStyleRef = "stub_style")
# Tidyselect: starts_with(), -col
```

Content Layers

```
spec <- add_title(spec, "Demographics")
spec <- add_subtitle(spec, "Analysis")
spec <- add_footnote(spec, "Source: DB.")

# Header/footer: left / center / right
spec <- add_header(spec, "Study ABC",
  "CONFIDENTIAL", "Page {PAGE}")
spec <- add_footer(spec, "Firma",
  "Locked DB", "2025")
```

Placeholders: {PAGE} and {NUMPAGES} filled by renderer. add_body_text() for narrative text. Multiple footnotes stack.

Page Layout & Templates

```
spec <- set_page_style(page = p_page(
  size = "A4",  # or "Letter", "Legal"
  orientation = "landscape",
  margins = p_margins(
    top = "1in", bottom = "1in")))

# Use built-in templates
set_page_style(docTemplate = "Navy_Pro")
```

Style Primitives — s_*() Helpers

All style helpers must be used inside add_style(). They validate params and raise informative errors.

```
s_font(font_name, font_size, bold, italic, underline, color, highlight)

s_paragraph(alignment, spacing, indents)
"left"|"right"|"center"|"justify"
  └ s_spacing(before, after, line_spacing)
  └ s_indents(left, right, first_line)

s_table_style(background_color, row_height,
vertical_alignment, borders)
  └ s_borders(top, bottom, left, right)  each → s_border()
```

s_spacing()/s_indents() nest in s_paragraph(). s_borders()/s_border() nest in s_table_style(). s_* helpers only work inside add_style().

Declaring & Applying Styles

```
spec <- add_style(spec, id = "tbl_hdr",
  s_font(bold = TRUE, color = "#FFF"),
  s_paragraph(alignment = "center"),
  s_table_style(background_color = "#003366"))
```

Context	Parameter	Used In
Column headers	labelStyleRef	define_cols()
Cell values	valueStyleRef	define_cols()
Span headers	labelStyleRef	add_span_header()
Titles/Notes	styleRef	add_title(), etc.
Conditional	styleRef	c_style()

Combining Styles — f_combine()

Merge multiple styles on-the-fly without pre-defining a combined style. create_report() auto-consolidates.

```
spec <- define_cols(spec, mpg,
  labelStyleRef = f_combine("bold",
    "red", "centered"))
# Vector: c(f_combine(...), f_combine(...))
```

Built-in Style Atoms

Single-property atoms — compose with f_combine(). tfl_print_style_atoms() for full catalog.

Atom(s)	Effect
b, i, u	Bold / Italic / Underline
fs_7 … fs_11	Font size 7–11pt
fc_black, fc_red …	Text colors
al, ar, ac	Left / Right / Center align
ind1–ind4	Left indent 0.5–2.0cm
bt, bb, bl, br	Top/Bottom/Left/Right border
bg_blue, bg_mint …	Cell backgrounds

Building & Rendering Reports

```
# Combine specs into one report
report <- create_report(
  spec_table, spec_text, spec_fig)

# One-step render to DOCX
write_doc(report,
  name = "demographics",
  outDir = "./output",
  metaPath = tempdir(),
  toc = TRUE, verbose = TRUE)
```

create_report() flattens inputs, validates structure, consolidates **f_combine()** styles, and assigns sequential docOrder. Set toc = TRUE for Table of Contents (requires toclevel on titles).

Metadata & Reproducibility

JSON specs enable re-rendering without R, version tracking, and audit trails for regulated submissions.

```
# Two-step: save then re-render
res <- save_report(report,
  docFileName = "report.docx",
  outDir = "./out", metaPath = "./meta")

replay_report("report.docx",
  meta_dir = "./meta")
list_reports("./meta")
clean_reports("./meta", dry_run = TRUE)
```

Session-Wide Defaults

Set once, inherited by all new specs. Headers, footers, output dirs, and style defaults.

```
tfl_set_options(
  add_header("Study", "CONF", ""),
  add_footer("Co.", "Page {page}", ""),
  output_directory = "./output",
  meta_directory = "./meta")
tfl_get_options()  # check
tfl_reset_options() # clear
```

Column Width Management

Auto-calculated from data. Lock columns with colWidth; unlocked columns auto-fill remaining space proportionally.

```
# Lock one column at 15%
spec <- define_cols(spec, id,
  colWidth = "15%")

# Absolute: colWidth = "2.5cm"
# Hide: isVisible = FALSE
```

Width states: VISIBLE (default), LOCKED (colWidth set), UNLOCKED (auto), INVISIBLE (isVisible=FALSE → 0cm). autoColWidth=TRUE by default.

Conditional Row Actions - compute_cols()

Per-row conditional actions can be configured:

```
c_style(cols, styleRef)  — Apply styles conditionally
```

```
spec <- compute_cols(spec,
  response == "CR",
  c_style(response,
    styleRef = "highlight_green"))
```

```
c_merge(cols)  — Merge adjacent cells
```

```
spec <- compute_cols(spec,
  !firstOf(group),
  c_merge(c(group, visit)))
```

```
c_addrow(pos, value_from, styleRef)  — Insert rows
```

```
spec <- compute_cols(spec,
  firstOf(group),
  c_addrow(pos = "above"))
```

```
c_pageBreak()  c_glue()  c_clear()  — More actions
```

c_pageBreak() — force new page at row
c_glue(cols, position, text/glue_col, separator) — append/prepend text
c_clear(cols) — clear cell content for matching rows

Condition Helpers (inside compute_cols())

Helper	Returns
firstOf(col1, col2, ...)	TRUE at first occurrence of each unique combo
lastOf(col1, col2, ...)	TRUE at last occurrence
firstRow()	TRUE only at very first row
lastRow()	TRUE only at very last row
rowNumber()	1-based row index (integer)
everyNth(n)	TRUE every n-th row
firstOfBlock(col, n, off)	First row of every n-th block

Common Pattern: Alternating Row Bg

```
spec <- add_style(spec, id = "zebra",
  s_table_style(
    background_color = "#F5F5F5"))

spec <- compute_cols(spec,
  everyNth(2),
  c_style(everything(),
    styleRef = "zebra"))
```

Pattern: Clinical Table Header

```
spec <- add_style(spec, id = "tbl_hdr",
  s_font(font_name = "Arial",
    font_size = "11pt",
    bold = TRUE, color = "#FFF"),
  s_paragraph(alignment = "center"),
  s_table_style(
    background_color = "#003366",
    row_height = "28pt",
    vertical_alignment = "center"))
```

Define once → apply everywhere with StyleRef = "tbl_hdr". Calls with same id are merged (last-win).

Pattern: Right-Aligned Numeric

```
spec <- add_style(spec, id = "num_r",
  s_paragraph(alignment = "right"),
  s_font(font_name = "Courier New",
    font_size = "9pt"))
define_cols(spec, c(age, weight),
  valueStyleRef = "num_r")
```

Borders — s_borders() / s_border()

```
spec <- add_style(spec, id = "border",
  s_table_style(borders = s_borders(
    top = s_border(color = "grey40",
      width = "1pt",
      line_style = "single"),
    bottom = s_border(color = "#333",
      width = "2pt",
      line_style = "thick"))))
```

Line styles: single, double, dashed, dotted, thick, none

Document Properties — set_document()

```
spec <- set_document(spec,
  hasData = TRUE,
  contentWidth = "95%",
  bodyTitles = FALSE,
  footnotePlace = "repeated",
  glueNumType = FALSE,
  isContinues = FALSE)
```

Multiple calls merge (last-win). Pairs well with set_page_style().

Allowed Color Names

Basic:
black, white, red, green, blue, yellow, orange, purple

Extended:
pink, brown, cyan, magenta, navy, teal, lime, maroon

Named:
olive, silver, gold, coral, salmon, turquoise, violet, indigo

Gray:
grey10 … grey90

Text Manipulation — c_glue() Details

Append or prepend text to cell values: glue a literal string or another column value.

```
# Append units from another column
compute_cols(spec, TRUE,
  c_glue(value, position = "after",
    glue_col = unit, separator = " "))

# Prepend literal text
compute_cols(spec, rowNumber() == 1,
  c_glue(name, position = "before",
    text = "** ", separator = ""))
```

Multi-Spec Report Workflow

Combine table + figure + text specs into one document. Specs rendered in order with page breaks.

```
# Build individual specs
spec_tbl <- create_table(demog)
spec_fig <- create_figure(p)
spec_txt <- create_text() |>
  add_body_text("Narrative text.")

# Combine and render
report <- create_report(
  spec_tbl, spec_fig, spec_txt)
write_doc(report, name = "combined", toc = TRUE,
  outDir = "./out")
```

Full Clinical Table — End to End

```
spec <- create_table(labs,
  cols = c(subject_id, ALT, AST)) |>
  add_style(id = "hdr",
    s_font(bold = TRUE, color = "#FFF"),
    s_table_style(
      background_color = "#003366")) |>
  add_style(id = "num",
    s_paragraph(alignment = "right")) |>
  define_cols(c(subject_id, ALT, AST),
    labelStyleRef = "hdr") |>
  define_cols(c(ALT, AST),
    valueStyleRef = "num") |>
  add_span_header(c(ALT, AST),
    label = "Lab Results") |>
  add_title("Lab Results") |>
  add_footnote("Values as observed.")

write_doc(create_report(spec),
  name = "lab_report")
```

Font Management

System fonts auto-discovered at load. Open-source fallbacks bundled for all 6 target families.

Target Font	Fallback
Arial	Liberation Sans
Times New Roman	Liberation Serif
Courier New	Liberation Mono
Georgia	Liberation Serif
Verdana	Liberation Sans
Trebuchet MS	Liberation Sans

```
tfl_font_status()    # check resolution
tfl_rescan_fonts()   # rescan after install
options(ksTFL.font_dirs = "/custom/fonts")
```

Anatomy of a TFL_spec

Component	Purpose	Modified By
document	Metadata & page settings	set_document(), set_page_style()
columns	Column defs & formats	define_cols()
stubColumns	Spanning headers	add_span_header()
headers	Page header (L/C/R)	add_header()
titles	Main titles	add_title()
subtitles	Secondary titles	add_subtitle()
bodyText	Narrative content	add_body_text()
footnotes	Document footnotes	add_footnote()
footers	Page footer (L/C/R)	add_footer()
styles	Named style defs	add_style()

Interactive Tools & Shiny Apps

```
# Style template editor (Shiny app)
run_styles_editor()

# Re-render browser (Shiny app)
run_replay_app()

# RStudio addin: Preview TFL spec
# Addins > "Preview TFL Spec"
```

Key Tips & Reminders

- Use named styles + f_combine() — never inspect spec internals
- print(spec) and print(report) for concise previews
- metaPath = tempdir() for quick experiments
- tfl_set_options() for consistent headers/footers across specs
- s_* helpers ONLY work inside add_style()
- Condition helpers ONLY work inside compute_cols()
- Multiple compute_cols() calls accumulate & merge
- autoColWidth = TRUE (default) auto-adjusts widths