Package 'lt'

July 9, 2026

Type Package

Title Lightweight Tables via JSON Specs and JavaScript

Version 0.2

Description A lightweight grammar of tables. Build a table by declaring a JSON spec (titles, spanners, row groups, footnotes, formatting functions, etc.); a tiny vanilla JavaScript runtime builds the HTML table from the spec on page load. No 'sass', no 'V8', no 'htmlwidgets' — just base R and 'xfun' ('htmltools' is used only for the optional Shiny binding).

License MIT + file LICENSE

URL https://github.com/yihui/lt

BugReports https://github.com/yihui/lt/issues

Depends R (>= 4.1.0)

Encoding UTF-8

Imports xfun (>= 0.60)

Suggests htmltools, knitr, litedown, magick, repr, shiny, testit

Config/lt.js 1.14.67

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Yihui Xie [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-0645-5666>, URL: https://yihui.org)

Maintainer Yihui Xie <xie@yihui.name>

Repository CRAN

Date/Publication 2026-07-09 21:00:02 UTC

Contents

`format.lt_tbl` *Render an* `lt_tbl` *to HTML*

Description

Emits the CSS+JS runtime and a script block carrying the table's JSON spec. Multiple tables on the same page only need the runtime once.

Usage

```
## S3 method for class 'lt_tbl'
format(x, fragment = TRUE, inline_assets = TRUE, assets = TRUE, ...)
```

Arguments

`x`	An `lt_tbl` object.
`fragment`	If `TRUE` (default), return an HTML fragment suitable for embedding. If `FALSE`, wrap in a minimal `<html><body>` document.
`inline_assets`	If `TRUE` (default), inline the CSS/JS as text. If `FALSE`, emit `<link>` / `<script src=...>` tags (assets must be served alongside the HTML).
`assets`	Which runtime assets to include: `TRUE` (default) for both CSS and JS, `FALSE` for neither, or a character vector subset of `c("css", "js")` for selective inclusion.
`...`	Reserved for future use.

Value

A character scalar containing HTML.

Examples

```
tbl = lt(head(mtcars))
html = format(tbl)
format(tbl, fragment = FALSE, inline_assets = FALSE)
```

`lt` *Create a Table Specification*

Description

Entry point of the lightweight grammar of tables. Returns an object (a list) that records the data plus a list of table-modifying operations. The object is rendered to HTML by `format()` (called automatically by the print method).

Usage

```
lt(data, ...)

## Default S3 method:
lt(data, auto_format = TRUE, auto_label = TRUE, ...)
```

Arguments

`data`	A data frame (or anything coercible to one).
`...`	Arguments passed to methods.
`auto_format`	Whether to automatically format numeric columns (rounding, thousand separators, percentage detection). Set to `FALSE` to disable for the whole table; use `lt_format()` on specific columns to disable selectively.
`auto_label`	Whether to automatically clean column names for display by replacing separators (`.` and `_`) with spaces. Set to `FALSE` to show raw column names.

Value

A table object that can be piped into `lt_*()` functions.

HTML in cells

By default all cell values and text (titles, labels, footnotes, ...) are HTML-escaped, so special characters like <, >, and & render as literal text. To emit content as raw HTML instead:

- For body cells, mark whole columns with `lt_html()`.
- For a title, subtitle, column label, spanner, footnote, or note, wrap the text in `I()`. For example, `lt_header(x, I("<b>Report</b>"))` renders the title as raw HTML, while `lt_header(x, "<b>Report</b>")` shows the literal <b> tags.

Interactivity

When a cell value has been formatted (e.g., by auto-formatting or `lt_format()`), the original value is stored in the cell's `title` attribute and shown as a tooltip on hover. Additionally:

- `Alt` + `Click` on a table toggles display of all raw values in that table.
- `Alt` + `Double-Click` on a table toggles raw values globally (all tables on the page).

Raw values are shown as `(value)` after the formatted text, highlighted with a light yellow background.

Examples

```
lt(head(mtcars[, 1:4]))
```

---

lt_align *Set Column Alignment*

Description

Override the auto-detected alignment for specific columns. By default, numeric columns are right-aligned and character columns are left-aligned.

Usage

```
lt_align(x, columns, align = c("left", "center", "right"))
```

Arguments

`x`	An `lt()` object.
`columns`	Character vector of column names (or a one-sided formula).
`align`	One of `"left"`, `"center"`, or `"right"`.

Value

`x` with the alignment recorded.

Examples

```
lt(head(mtcars)) |> lt_align(~ cyl + gear, "center")
```

lt_css *Attach Custom CSS*

Description

Add user-supplied stylesheets or inline rules that render after the built-in CSS, so rules can override the defaults.

Usage

```
lt_css(x, ...)
```

Arguments

x	An `lt()` object.
...	Unnamed arguments are stylesheet paths or URLs. A bare filename (no directory component) that does not exist in the working directory is resolved against the stylesheets shipped with lt, so e.g. `lt_css(x, "lt-gt.css")` uses the bundled gt-like theme. Named arguments define inline CSS rules scoped to `.lt-table`. Names are selectors (e.g., `.na`, `td.highlight`) and values are either a CSS string or a named list of properties: `lt_css(x, .na = "background: #eee")` or `lt_css(x, .na = list(background = "#eee"))`.

Value

x with the stylesheets recorded.

Examples

```
tbl = lt(head(mtcars))
tbl |>
  lt_style("mpg", test = "v => v > 20", class = "high") |>
  lt_css(.high = list(background = "#cfc", fontWeight = "bold"))
```

lt_date *Format Date/Time Columns*

Description

Format date or datetime columns using JavaScript's native Date methods. The underlying data should be Date or POSIXt in R (serialized as `new Date(...)` for the browser).

Usage

```
lt_date(x, columns, method = NULL, locale = NULL, options = NULL)
```

Arguments

x	An lt() object.
columns	Column selection (formula, character, or numeric).
method	A JS Date method name to call (e.g., "toLocaleDateString", "toISOString", "toDateString", "toLocaleString", "toLocaleTimeString"). If NULL (the default), toLocaleDateString() is used with locale and options. See https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date#instance_methods for all available methods.
locale	A BCP 47 locale string (e.g., "en-US", "de-DE", "ja-JP"). Only used when method is NULL. See https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl#locales_argument for details.
options	A named list of Intl.DateTimeFormat options. Common fields include year ("numeric", "2-digit"), month ("numeric", "2-digit", "long", "short", "narrow"), day ("numeric", "2-digit"), hour, minute, second ("numeric", "2-digit"), weekday ("long", "short", "narrow"), and timeZoneName ("long", "short"). Only used when method is NULL.

Value

x with the date formatting recorded.

Note

The formatted date may differ from the input date depending on the viewer's local timezone. JavaScript's new Date("2024-01-15") parses date-only strings as UTC midnight, but toLocaleDateString() converts to the local timezone. For example, 2024-01-15 will display as 2024-01-14 for a viewer at GMT-6. To avoid this, pass options = list(timeZone = "UTC").

Examples

```
d = data.frame(
  event = c("Launch", "Update"),
  date = as.Date(c("2024-01-15", "2024-06-30"))
)
lt(d) |> lt_date(~ date)
lt(d) |> lt_date(~ date, options = list(year = "numeric", month = "short"))
```

---

lt_export *Export an lt table to a file*

Description

Save a table to disk. The output format is chosen from the file extension of output: .html writes an HTML table, .pdf writes a vector PDF, and any other extension writes a PNG. PDF and PNG are produced by rendering the table in a headless Chromium browser (via xfun::browser_print()).

Usage

```
lt_export(
  x,
  output = "lt.html",
  method = c("auto", "node", "browser", "raw"),
  css = TRUE,
  fragment = FALSE,
  tidy = FALSE,
  crop = TRUE,
  width = NULL,
  padding = 8,
  browser = NULL,
  ...
)
```

Arguments

x	An lt_tbl object.
output	Output file path. Its extension selects the format: .html, .pdf, or (otherwise) PNG. If NA, the HTML is returned as a string instead of being written to a file.
method	How to produce the HTML <table>: "auto", "node", "browser", or "raw" (see Details). Applies only to .html output.
css	Whether to include the lt.css runtime stylesheet in the HTML output. User CSS from lt_css() is always included. Applies only to .html output.
fragment	If FALSE (default), wrap the HTML in a full HTML document; if TRUE, return only the table fragment. Applies only to .html output.
tidy	Whether to pretty-print the baked <table> with line breaks and indentation. Applies only to .html output baked by "node" or "browser" (ignored for method = "raw", which is a JavaScript spec).
crop	Whether to crop the PDF/PNG tightly to the table, removing the surrounding page whitespace. This adds a preliminary browser pass to measure the rendered table. Set to FALSE for the default full page. Cropping PNG output requires the **magick** package; without it, PNG falls back to the full page (with a warning).
width	The width of the table in CSS pixels. By default (NULL) it shrinks to the table's natural width. A smaller width wraps cell content; a larger one pads the table.
padding	Padding in CSS pixels to keep around the table when cropping. A single value (all sides) or a length-two vector c(vertical, horizontal).
browser	Path to the Chromium-based browser; passed to xfun::browser_print(). NULL (default) auto-detects.
...	Passed to xfun::browser_print() for PDF/PNG output.

Details

For .html output, method controls how the <table> is produced: "raw" writes the JavaScript-spec HTML, so the table is built in the browser by the lt.js runtime when the file is viewed; the other

methods bake a static <table> up front by running lt.js once (via "node" in Node.js or "browser" in a headless Chromium browser; "auto" picks Node if available, else the browser), so the saved file needs no JavaScript to view. method, css, fragment, and tidy apply only to .html output.

Value

The output path, or (when output is NA) the HTML as a string.

Global option

When the option lt.lt_static is set to a list of arguments (e.g., options(lt.lt_static = list(css = FALSE))), the knit_print and record_print methods emit the same static HTML table as lt_export(x, "*.html") (using those arguments as method/css/fragment) instead of the default JavaScript-based spec. This is useful for output formats that support raw HTML but cannot run JavaScript (e.g., GitHub Flavored Markdown).

Examples

```
tbl = lt(head(mtcars))

# HTML with the JavaScript spec (table built by lt.js when viewed)
lt_export(tbl, NA, method = 'raw')  # character output
f1 = tempfile(fileext = '.html')
lt_export(tbl, f1, method = 'raw')  # file output

# Bake a static <table> (needs Node.js or a headless browser).
if (lt:::can_bake())
  lt_export(tbl, NA, method = 'auto', fragment = TRUE, css = FALSE)

# PDF / PNG are rendered in a headless browser and cropped to the table.
f2 = tempfile(fileext = '.pdf')
f3 = tempfile(fileext = '.png')
if (lt:::has_browser()) {
  lt_export(tbl, f2)
  lt_export(tbl, f3, width = 400)
}

unlink(c(f1, f2, f3))
```

---

lt_footnote *Add a Footnote*

Description

Attaches a footnote text to a table region. Footnotes are numbered automatically in the order they are added (de-duplicated by text).

Usage

```
lt_footnote(x, text, where, columns = NULL, rows = NULL, match = NULL)
```

Arguments

x	An lt() object.
text	Footnote text.
where	One of 'title', 'subtitle', 'column', 'spanner', 'group', or 'body'.
columns	Character vector of column names or a one-sided formula (for 'column' or 'body'). For 'group' with match = "starts_with", a single prefix string.
rows	Integer vector of 1-based row indices (for 'body'; NULL means all rows).
match	For where = "group": one of "exact" (default), "starts_with", or "all".

Value

x with the footnote recorded.

Examples

```
lt(head(mtcars)) |>
  lt_footnote("Source: 1974 Motor Trend US magazine.", "title")
# raw HTML footnote (wrap in I())
lt(head(mtcars)) |>
  lt_footnote(I("See <a href='#'>docs</a>."), "title")
```

---

lt_format *Format Numeric Columns*

Description

Control the number of decimal places and thousands separator for numeric columns. Columns passed to this function are excluded from automatic formatting (see the auto_format argument of lt()). To disable auto-format for a column without otherwise changing its display, call lt_format(x, ~col) with no other arguments.

Usage

```
lt_format(
  x,
  columns,
  decimals = NULL,
  big_mark = NULL,
  percent = NULL,
  prefix = NULL,
  suffix = NULL
)
```

Arguments

x	An lt() object.
columns	Character or integer vector of columns (or a one-sided formula).
decimals	Number of decimal places (default NULL means no change).
big_mark	Thousands separator (e.g., ","). NULL or "" means none.
percent	If TRUE, multiply values by 100 and append "%". If "%", only append "%" without multiplying (for values already in percent scale).
prefix	String prepended to each formatted value (e.g., "$").
suffix	String appended to each formatted value (e.g., " kg").

Value

x with the formatting recorded.

Examples

```
lt(head(mtcars)) |> lt_format(~ mpg + wt, decimals = 1, big_mark = ",")
d = data.frame(Item = c("A", "B"), Price = c(1234.5, 678.9))
lt(d) |> lt_format(~ Price, decimals = 2, big_mark = ",", prefix = "$")
```

---

lt_group *Define Row Groups*

Description

Partition rows into labeled groups. Pass column names to group by those columns' values (the columns are removed from the body and rendered as rowspan cells on the left). Use sep = TRUE to render groups as full-width separator rows instead of rowspan.

Usage

```
lt_group(x, ..., sep = "auto", sort = TRUE)
```

Arguments

x	An lt() object.
...	A column name or formula (e.g., ~col or ~col1 + col2) to group by column values, or named arguments of the form "Label" = rows (integer vector of 1-based row indices) for manual groups. Unnamed character strings reorder previously defined groups.
sep	If TRUE, render groups as full-width separator rows instead of the default rowspan style. Only supports a single grouping column. The default 'auto' uses separator rows when there is a single character grouping column with any value longer than 20 characters (numeric columns always use rowspan).
sort	If TRUE (default), sort rows by group columns so that identical group values are contiguous. Set to FALSE to preserve the original row order.

Value

x with the row groups recorded.

Examples

```
# Group by a column (rowspan, default)
d = data.frame(arm = c("Placebo", "Placebo", "Treatment", "Treatment"),
               stat = c("n", "Mean", "n", "Mean"), value = c(30, 4.2, 31, 6.8))
lt(d) |> lt_group(~ arm)

# Separator-row style
lt(d) |> lt_group(~ arm, sep = TRUE)

# Manual groups (always separator rows)
lt(head(mtcars)) |>
  lt_group("First three" = 1:3, "Last three" = 4:6)
```

---

lt_header *Add a Title and Subtitle*

Description

Add a Title and Subtitle

Usage

```
lt_header(x, title = NULL, subtitle = NULL)
```

Arguments

x	An `lt()` object.
title	A character scalar. Wrap in `I()` to treat as raw HTML.
subtitle	A character scalar. Wrap in `I()` to treat as raw HTML.

Value

x with the header recorded.

Examples

```
lt(head(mtcars)) |> lt_header("Motor Trend Cars", "First 6 rows")
# raw HTML title
lt(head(mtcars)) |> lt_header(I("<em>Motor Trend</em> Cars"))
```

lt_html *Render Column Cells as Raw HTML*

Description

Mark whole columns so their body cells are emitted as raw HTML instead of being escaped. By default every cell is HTML-escaped (so <, >, and & render as literal text); use this to embed links, emphasis, or other markup in a column's cells.

Usage

```
lt_html(x, columns)
```

Arguments

x	An `lt()` object.
columns	Column names (character or one-sided formula) whose cells are raw HTML. If missing, all columns are treated as raw HTML.

Details

To emit raw HTML in other regions (title, column labels, footnotes, ...), wrap the text in `I()` in the corresponding `lt_*()` function instead.

Value

x with the raw-HTML columns recorded.

Examples

```
d = data.frame(
  name = c("<a href='#a'>A</a>", "<a href='#b'>B</a>"), n = 1:2
)
lt(d) |> lt_html(~ name)
```

lt_indent *Indent Rows*

Description

Add hierarchical indentation to the first column of specified rows.

Usage

```
lt_indent(x, rows, level = 1)
```

Arguments

x	An lt() object.
rows	Integer vector of 1-based row indices to indent.
level	Indent level (default 1). Each level adds one unit of left padding.

Value

x with the indentation recorded.

Examples

```
d = data.frame(label = c("Overall", "Male", "Female"), n = c(100, 55, 45))
lt(d) |> lt_indent(2:3)
```

---

lt_label *Rename Column Labels*

Description

Override column headers without modifying the underlying data frame.

Usage

```
lt_label(x, ...)
```

Arguments

x	An lt() object.
...	Named arguments of the form col_name = "Display Label". Wrap a label in I() to treat it as raw HTML.

Value

x with the column label overrides recorded.

Examples

```
lt(head(mtcars)) |> lt_label(mpg = "Miles/Gallon", cyl = "Cylinders")
# raw HTML label (wrap in I())
lt(head(mtcars)) |> lt_label(mpg = I("Miles/Gallon<sup>*</sup>"))
```

lt_merge *Merge Columns*

Description

Combine values from multiple columns into a single display column using a pattern. Source columns (all except the first) are hidden by default.

Usage

```
lt_merge(x, columns, pattern = NULL, hide = TRUE)
```

Arguments

x	An lt() object.
columns	Character vector of column names (or a one-sided formula). The first column is the target (receives merged content); the rest are sources.
pattern	A glue-style template using \{1\}, \{2\}, etc. to refer to columns by position. Wrap sections in << and >> for conditional NA handling: "\{1\}<< (\{2\})>>" drops the wrapped portion when any referenced value is missing/empty. If NULL, columns are concatenated separated by spaces.
hide	If TRUE (default), source columns (all but the first) are automatically hidden.

Value

x with the merge recorded.

Examples

```
d = data.frame(stat = c("Mean", "SD"), value = c(4.2, 1.1), ci = c("(2.0, 6.4)", "(0.5, 1.7)"))
lt(d) |> lt_merge(~ value + ci, pattern = "{1} {2}")
```

lt_move *Move Columns*

Description

Rearrange column display order without modifying the data frame.

Usage

```
lt_move(x, columns, after = NULL)
```

Arguments

x	An lt() object.
columns	Character vector of column names (or a one-sided formula).
after	Column name after which to place the moved columns. Use NULL to move to the start.

Value

x with the column move recorded.

Examples

```
lt(head(mtcars)) |> lt_move(~ gear + carb, after = "mpg")
```

---

lt_note *Add a Note*

Description

Notes are rendered in the table footer below numbered footnotes.

Usage

```
lt_note(x, text)
```

Arguments

x	An lt() object.
text	Note text. Wrap in I() to treat as raw HTML.

Value

x with the note recorded.

Examples

```
lt(head(mtcars)) |> lt_note("CI = confidence interval.")
# raw HTML note (wrap in I())
lt(head(mtcars)) |> lt_note(I("CI = <em>confidence interval</em>."))
```

`lt_output` *Shiny Bindings for lt*

Description

`lt_output()` creates a UI placeholder; `render_lt()` supplies the table spec from the server. Together they render an `lt()` table as a custom Shiny output — no `renderUI()` involved.

Usage

```
lt_output(outputId, ...)

render_lt(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	Output variable name to read the table from.
`...`	Reserved for future use.
`expr`	An expression that returns an `lt()` object.
`env`	Environment in which to evaluate `expr`.
`quoted`	Whether `expr` is already quoted.

Value

`lt_output()` returns a Shiny UI element; `render_lt()` returns a render function.

Examples

```
if (interactive()) {
library(shiny)
ui = fluidPage(lt_output("tbl"))
server = function(input, output) {
  output$tbl = render_lt(lt(head(mtcars)) |> lt_header("Motor Trend"))
}
shinyApp(ui, server)
}
```

`lt_spanner` *Add a Column Spanner*

Description

A spanner is a label rendered above a contiguous group of column headers.

Usage

```
lt_spanner(x, label, columns, sep = "[._]")
```

Arguments

`x`	An `lt()` object.
`label`	A character scalar — the spanner text. Alternatively, a two-sided formula `Label ~ col1 + col2` providing both the label (LHS) and columns (RHS). When missing, spanners are inferred from column names.
`columns`	Column names (character or formula). When missing, inferred from column names.
`sep`	Separator pattern for auto-inference (default `"[._]"`).

Details

When called with no `label` or `columns`, infers spanners from column names by splitting on the first `.` or `_` separator. Contiguous columns sharing a prefix are grouped under that prefix, and column labels are shortened to the suffix.

Value

`x` with the spanner recorded.

Note

The columns must be contiguous in the body of the table.

Examples

```
tbl = lt(head(iris))
# Explicit spanner
tbl |> lt_spanner(Sepal ~ Sepal.Length + Sepal.Width)
# Auto-infer from column names
tbl |> lt_spanner()
# raw HTML label (wrap in I())
tbl |> lt_spanner(I("<em>Sepal</em>"), ~ Sepal.Length + Sepal.Width)
```

lt_style *Style Cells*

Description

Apply CSS styling to specific cells. Target cells by column, row, or both. When `test` is provided, styles are applied conditionally based on cell values (evaluated in JavaScript).

Usage

```
lt_style(
  x,
  columns = NULL,
  rows = NULL,
  test = NULL,
  class = NULL,
  bold = NULL,
  italic = NULL,
  color = NULL,
  bg = NULL,
  ...
)
```

Arguments

`x`	An `lt()` object.
`columns`	Character vector of column names, a one-sided formula, or `NULL` for all.
`rows`	Integer vector of 1-based row indices (or `NULL` for all).
`test`	A JavaScript function as a string (e.g., `"v => v < 0"`) that receives the raw cell value and returns `true` to apply the style. When `NULL`, the style applies unconditionally.
`class`	CSS class name(s) to add to matching cells. Define the corresponding rules in an external stylesheet via `lt_css()`.
`bold`	Logical: apply bold weight?
`italic`	Logical: apply italic style?
`color`	Text color (any CSS color value, e.g., `"red"`, `"#06c"`).
`bg`	Background color.
`...`	Additional CSS properties as named arguments. Names can be camelCase (e.g., `borderLeft`) or quoted dash-case (e.g., `` `border-left` ``). Values are CSS strings.

Value

`x` with the style recorded.

Examples

```
tbl = lt(head(mtcars))
tbl |>
  lt_style("mpg", rows = 1L, bold = TRUE, borderBottom = "2px solid red")
tbl |>
  lt_style("mpg", test = "v => v > 20", class = "high") |>
  lt_css(.high = list(background = "#cfc"))
```

`lt_sub` *Substitute Cell Values*

Description

Replace `NA`, zero, or small values with display text.

Usage

```
lt_sub(
  x,
  columns = NULL,
  missing = NULL,
  zero = NULL,
  small = NULL,
  small_text = NULL
)
```

Arguments

`x`	An `lt()` object.
`columns`	Character vector of column names, a one-sided formula, or `NULL` for all.
`missing`	Replacement for `NA` cells (e.g., `"–"`). `NULL` to leave NAs as empty strings (the default rendering).
`zero`	Replacement for zero values (e.g., `"–"`).
`small`	Threshold: values whose absolute value is below this are replaced by `small_text`.
`small_text`	Text shown for values below `small` (e.g., `"<0.1"`).

Value

`x` with the substitution recorded.

Examples

```
d = data.frame(x = c(1, 0, NA, 0.001))
lt(d) |> lt_sub(missing = "–", zero = "–", small = 0.01, small_text = "<0.01")
```

lt_width *Set Column Widths*

Description

Set Column Widths

Usage

```
lt_width(x, ...)
```

Arguments

x	An `lt()` object.
...	Named arguments of the form `col_name = "width"`. Width can be any CSS value (e.g., `"100px"`, `"20%"`, `"8em"`).

Value

x with the column widths recorded.

Examples

```
lt(head(mtcars)) |> lt_width(mpg = "100px", cyl = "50px")
```

print.lt_tbl *Print an* lt_tbl *(Opens in the Viewer or Browser)*

Description

Print an lt_tbl (Opens in the Viewer or Browser)

Usage

```
## S3 method for class 'lt_tbl'
print(x, ...)
```

Arguments

x	An lt_tbl object.
...	Passed to `format()`.

Value

x, invisibly.

Examples

```
print(lt(head(mtcars)))
```

Index