

# Package: lt (via r-universe)

June 30, 2026

**Type** Package

**Title** Lightweight Tables via JSON Specs and JavaScript

**Version** 0.1

**Description** A lightweight grammar of tables. Build a table by declaring a JSON spec (titles, spanners, row groups, footnotes, formatters, 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 (>= 0.59)

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

**Config/lt.js** 1.14.57

**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** <https://cran.r-universe.dev>

**Date/Publication** 2026-06-30 21:10:55 UTC

**RemoteUrl** <https://github.com/cran/lt>

**RemoteRef** HEAD

**RemoteSha** a8919904927d7770a7cadf3efc308fd2c30f6233

## Contents

format.lt_tbl . . . . .	2
lt . . . . .	3
lt_align . . . . .	4
lt_css . . . . .	4
lt_date . . . . .	5
lt_footnote . . . . .	6
lt_format . . . . .	7
lt_group . . . . .	8
lt_header . . . . .	9
lt_indent . . . . .	9
lt_label . . . . .	10
lt_merge . . . . .	10
lt_move . . . . .	11
lt_note . . . . .	12
lt_output . . . . .	12
lt_spanner . . . . .	13
lt_style . . . . .	14
lt_sub . . . . .	15
lt_width . . . . .	16
print.lt_tbl . . . . .	16

<b>Index</b>	<b>18</b>
--------------	-----------

---

format.lt_tbl	<i>Render an lt_tbl to HTML</i>
---------------	---------------------------------

---

### 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	If TRUE (default), include the CSS+JS runtime. Pass FALSE to emit only the spec block when the runtime is already on the page.
...	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**

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

**Value**

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

**Examples**

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

---

lt_align	<i>Set Column Alignment</i>
----------	-----------------------------

---

### 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 <code>lt()</code> 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	<i>Attach Custom CSS</i>
--------	--------------------------

---

### 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](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](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\_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 <code>lt()</code> 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 <code>match = "starts_with"</code> , a single prefix string.
rows	Integer vector of 1-based row indices (for 'body'; <code>NULL</code> means all rows).
match	For <code>where = "group"</code> : 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")
```

---

 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 <code>lt()</code> 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 <code>lt()</code> object.
...	A column name or formula (e.g., <code>~col</code> or <code>~col1 + col2</code> ) to group by column values, or named arguments of the form <code>"Label" = rows</code> (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	<i>Add a Title and Subtitle</i>
-----------	---------------------------------

---

**Description**

Add a Title and Subtitle

**Usage**

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

**Arguments**

x	An <code>lt()</code> object.
title	A character scalar.
subtitle	A character scalar.

**Value**

x with the header recorded.

**Examples**

```
lt(head(mtcars)) |> lt_header("Motor Trend Cars", "First 6 rows")
```

---

lt_indent	<i>Indent Rows</i>
-----------	--------------------

---

**Description**

Add hierarchical indentation to the first column of specified rows.

**Usage**

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

**Arguments**

x	An <code>lt()</code> 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	<i>Rename Column Labels</i>
----------	-----------------------------

---

**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"`.

**Value**

x with the column label overrides recorded.

**Examples**

```
lt(head(mtcars)) |> lt_label(mpg = "Miles/Gallon", cyl = "Cylinders")
```

---

lt_merge	<i>Merge Columns</i>
----------	----------------------

---

**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 <code>lt()</code> 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 <code>\{1\}</code> , <code>\{2\}</code> , etc. to refer to columns by position. Wrap sections in <code>&lt;&lt;</code> and <code>&gt;&gt;</code> for conditional NA handling: <code>"\{1\}&lt;&lt; (\{2\})&gt;&gt;"</code> drops the wrapped portion when any referenced value is missing/empty. If <code>NULL</code> , columns are concatenated separated by spaces.
hide	If <code>TRUE</code> (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 <code>lt()</code> object.
columns	Character vector of column names (or a one-sided formula).
after	Column name after which to place the moved columns. Use <code>NULL</code> to move to the start.

**Value**

x with the column move recorded.

**Examples**

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

---

lt_note	<i>Add a Note</i>
---------	-------------------

---

**Description**

Notes are rendered in the table footer below numbered footnotes.

**Usage**

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

**Arguments**

x	An <code>lt()</code> object.
text	Note text.

**Value**

x with the note recorded.

**Examples**

```
lt(head(mtcars)) |> lt_note("CI = confidence interval.")
```

---

lt_output	<i>Shiny Bindings for lt</i>
-----------	------------------------------

---

**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 <code>lt()</code> object.
env	Environment in which to evaluate <code>expr</code> .
quoted	Whether <code>expr</code> 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 <code>lt()</code> object.
label	A character scalar — the spanner text. Alternatively, a two-sided formula <code>Label ~ col1 + col2</code> 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()
```

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**

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

---

**Description**

Set Column Widths

**Usage**

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

**Arguments**

x	An <code>lt()</code> object.
...	Named arguments of the form <code>col_name = "width"</code> . 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	<i>Print an lt_tbl (Opens in the Viewer or Browser)</i>
--------------	---

---

**Description**

Print an `lt_tbl` (Opens in the Viewer or Browser)

**Usage**

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

**Arguments**

x	An <code>lt_tbl</code> object.
...	Passed to <code>format()</code> .

**Value**

x, invisibly.

*print.lt\_tbl*

17

### **Examples**

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

# Index

`format()`, [3](#), [16](#)  
`format.lt_tbl`, [2](#)

`lt`, [3](#)  
`lt()`, [4–16](#)  
`lt_align`, [4](#)  
`lt_css`, [4](#)  
`lt_css()`, [14](#)  
`lt_date`, [5](#)  
`lt_footnote`, [6](#)  
`lt_format`, [7](#)  
`lt_format()`, [3](#)  
`lt_group`, [8](#)  
`lt_header`, [9](#)  
`lt_indent`, [9](#)  
`lt_label`, [10](#)  
`lt_merge`, [10](#)  
`lt_move`, [11](#)  
`lt_note`, [12](#)  
`lt_output`, [12](#)  
`lt_spanner`, [13](#)  
`lt_style`, [14](#)  
`lt_sub`, [15](#)  
`lt_width`, [16](#)

`print.lt_tbl`, [16](#)

`render_lt (lt_output)`, [12](#)