

Package: tabularise (via r-universe)

September 18, 2026

Type Package

Version 0.7.1

Title 'SciViews::R' - Create Tabular Outputs from R

Description Create rich-formatted tabular outputs from R that can be incorporated into R Markdown/Quarto documents with correct output at least in HTML, LaTeX/PDF, Word and PowerPoint formats for various R objects.

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.2.0)

Imports commonmark (>= 1.9.0), equatags (>= 0.2.2), equatiomatic (>= 0.4.4), flextable (>= 0.9.1), generics (>= 0.1.3), knitr (>= 1.42), officer (>= 0.6.2), rlang (>= 1.1.1), stats (>= 4.2.0), svBase (>= 1.7.0), tinytable (>= 0.2.1), utils (>= 4.2.0)

Suggests covr (>= 3.5.0), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

Remotes SciViews/svBase

License MIT + file LICENSE

URL <https://github.com/SciViews/tabularise>,
<https://www.sciviews.org/tabularise/>,
<https://sciviews.r-universe.dev/tabularise>

BugReports <https://github.com/SciViews/tabularise/issues>

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Encoding UTF-8

Language en-US

ByteCompile yes

Config/testthat/edition 3

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libxslt-dev libssl-dev libnode-dev zlib1g-dev

Repository <https://sciviews.r-universe.dev>

Date/Publication 2025-12-22 14:26:29 UTC

RemoteUrl <https://github.com/SciViews/tabularise>

RemoteRef HEAD

RemoteSha 544176c5eed30e0c4203f880d6ffcf3d1c12237a

Contents

tabularise-package	2
colformat_md	3
colformat_sci	5
para_md	6
Stb	7
tabularise	8
tabularise_coef	9
tabularise_confint	10
tabularise_default	11
tabularise_glance	13
tabularise_headtail	14
tabularise_tidy	15
theme_tt_sciviews	16
Index	17

tabularise-package	'SciViews::R' - Create Tabular Outputs from R
--------------------	---

Description

Create rich-formatted tabular outputs from various R objects that can be incorporated into R Mark-down/Quarto documents with correct output at least in HTML, LaTeX/PDF, Word and PowerPoint for various R objects.

Details

For an object class, there is a "default" method that creates the most obvious tabular form for this object, but there might be also other types provided with different tabular views of the same object. All types are accessible from the `tabularise()` function that accepts `type=` argument, or better, by using the compact and easier to read `tabularise$type()` form.

Important functions

- `tabularise()` constructs a table (**flextable** object by default, or **tinytable** object with `kind = "tt"`).
- `Sbt` a collection of functions to manipulate and format a table (mostly functions from the `{flextable}` package).
- `colformat_sci()` and `colformat_md()` are additional columns formatters for **flextable** objects, respectively, scientific and markdown formats.
- `para_md()` creates `{flextable}` paragraphs with rich-formatting by converting Markdown strings into such `{flextable}` paragraphs.

Author(s)

Maintainer: Philippe Grosjean <phgrosjean@sciviews.org> ([ORCID](#))

Authors:

- Guyliann Engels <guyliann.engels@umons.ac.be> ([ORCID](#))

See Also

Useful links:

- <https://github.com/SciViews/tabularise>
- <https://www.sciviews.org/tabularise/>
- <https://sciviews.r-universe.dev/tabularise>
- Report bugs at <https://github.com/SciViews/tabularise/issues>

colformat_md

Scientific format for columns in {flextable}

Description

Scientific format for columns in `{flextable}`

Usage

```
colformat_md(
  x,
  i = NULL,
  j = NULL,
  h.sizes = c(15, 13, 9),
  h.colors = c("black", "darkgray"),
  strike = list(color = h.colors[2], underlined = FALSE),
  bullet = ".",
  code.font = "monospace",
  code.shading = "#f8f8f8",
```

```

    link.color = "blue",
    link.underline = TRUE,
    autolink = TRUE
  )

```

Arguments

<code>x</code>	a flextable object
<code>i</code>	rows selection
<code>j</code>	columns selection
<code>h.sizes</code>	font sizes for titles
<code>h.colors</code>	font colors for titles
<code>strike</code>	a list with color and underlined = TRUE or FALSE for strikethrough text
<code>bullet</code>	bullet character used for lists
<code>code.font</code>	font used for code strings
<code>code.shading</code>	background color (shading) for code strings
<code>link.color</code>	color used for URL links
<code>link.underline</code>	are URL links underlined?
<code>autolink</code>	are links automatically constructed when there is an URL in the text?

Value

the **flextable** object with the selected region formatted as markdown strings.

Examples

```

dat <- data.frame(
  x = 1:3,
  y = noquote(c("#### text1", "text~2~", "text^3^")),
  names = noquote(c("*Iris setosa*", "Iris ~~virginica~~",
    "Iris **versicolor**")),
  id = c("*setosa*", "`virginica`", "**versicolor**"),
  factor = factor(c("*setosa*", "`virginica`", "**versicolor**"))
)
tabularise(dat)
tabularise(dat) |> colformat_md() |> Stb$autofit()
tabularise(dat) |> colformat_md(i = 2:3, j = 'names') |> Stb$autofit()

```

colformat_sci	<i>Scientific format for columns in {flextable}</i>
---------------	---

Description

Scientific format for columns in {flextable}

Usage

```
colformat_sci(
  x,
  i = NULL,
  j = NULL,
  digits = 3,
  scipen = 0,
  lod = NULL,
  lod_str = paste("<", lod),
  fancy = TRUE,
  op = c("·", "x", "*", "x")
)
```

Arguments

x	a flextable object
i	rows selection
j	columns selection
digits	number of digits to display
scipen	penalty to use to decide if numbers are presented in decimal or scientific notation (generally use 0 or -1)
lod	value indicating the limit of detection, for which we should display something like '< lod_value' instead of the actual value; useful for p values (R often uses 2e-16 in that case), chemical measurements ...
lod_str	the string to use, by default, it is < lod_value.
fancy	use a perfect scientific notation (TRUE) or a simplified one like 1.34e-5 (FALSE).
op	the operator character to use in fancy scientific notation

Value

the **flextable** object with the selected region formatted as scientific numbers.

Examples

```
summ <- summary(lm(Volume ~ Girth + Height, data = trees))
tabularise(as.data.frame(summ$coefficients)) |>
  colformat_sci() |>
  colformat_sci(j = 'Pr(>|t|)', lod = 2e-16) |>
  Stb$autofit()
```

para_md	<i>Create a rich-formatted paragraph using markdown notation for flextable objects</i>
---------	--

Description

Create a rich-formatted paragraph using markdown notation for flextable objects

Usage

```
para_md(
  ...,
  h.fonts = rep(flextable::get_flextable_defaults())$font.family, 6),
  h.sizes = c(15, 13, 9),
  h.colors = c("black", "darkgray"),
  strike.color = h.colors[2],
  strike.underline = NA,
  code.font = "inconsolata",
  code.shading = "#f8f8f8",
  link.color = "blue",
  link.underline = TRUE,
  bullet = ". ",
  autolink = TRUE,
  smart = TRUE,
  debug = FALSE
)

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

Arguments

...	the character strings with markdown formatting
h.fonts	fonts used for the titles (h1-h6)
h.sizes	font sizes for titles
h.colors	font colors for titles
strike.color	color for strikethrough text
strike.underline	is strikethrough text converted into underlined text?
code.font	font used for code strings
code.shading	background color (shading) for code strings
link.color	color used for URL links
link.underline	are URL links underlined?
bullet	bullet character used for lists

autolink	are links automatically constructed when there is an URL in the text?
smart	is smart punctuation detected and replaced?
debug	switch in debug mode
x	A paragraph object

Value

a flextable paragraph object with its content formatted according to markdown tags

Examples

```
md1 <- paste0("# Heading **1**\n## Heading **2**\n### Heading **3**\n",
  "#### Heading **4**\n##### Heading **5**\n##### Heading **6**")
md2 <- paste0("* List 1\n 1. List 1a\n 2. List 1b\n* List 2\n<br />\n",
  "*Some text* with super^script^, sub~script~ and ~~color~~{+F50490}")
tabularise(head(iris)) |>
  Stb$add_footer_lines(para_md(md1, md2))
```

Stb	<i>Tabularise set of function (mainly from {flextable})</i>
-----	---

Description

This set provides all the functions you can use to manipulate tabularise() tables. They mostly contain the {flextable} API. You are supposed to use it like Stb\$verb(...) where verb is one of the objects contained in the collection. Use Stb to list all objects in the set.

Usage

Stb

Format

An object of class list of length 147.

Value

When printing Stb alone, a list of all verbs and other objects provided in the set are returned.

Examples

```
# TODO...
```

tabularise	<i>Tabularise an object (arrange or enter in tabular form)</i>
------------	--

Description

Tabularise an object (arrange or enter in tabular form)

Usage

```
tabularise(data, ..., type = "default", kind = "ft", env = parent.frame())
```

```
tabularize(data, ..., type = "default", kind = "ft", env = parent.frame())
```

Arguments

data	An object
...	Further arguments (depending on the object class and on type=).
type	The type of table to produce.
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate formulas (you probably do not need to change the default).

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

[tabularise_default\(\)](#), [tabularise_headtail\(\)](#), [tabularise_coef\(\)](#), [tabularise_tidy\(\)](#),
[tabularise_glance\(\)](#)

Examples

```
tabularise(iris)
```

tabularise_coef	<i>"Coef" type tabularise generic (tabularise\$coef)</i>
-----------------	--

Description

The "coef" type of `tabularise()` extracts and formats a table of coefficients from an object, similar to `stats::coef()` applied to the same object, but in a rich-formatted form.

Usage

```
tabularise_coef(data, ..., kind = "ft", env = env)
```

```
## Default S3 method:
```

```
tabularise_coef(data, ..., kind = "ft", env = env)
```

Arguments

<code>data</code>	An object
<code>...</code>	Further arguments (depending on the object class).
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate formulas (you probably do not need to change the default).

Details

No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

`tabularise()`, `stats::coef()`

tabularise_confint	<i>"Confint" type" tabularise generic (tabularise\$confint)</i>
--------------------	---

Description

The "confint" type of `tabularise()` presents a table of confidence intervals for an objects (e.g., confidence intervals on parameters of a model). This is similar to the output of `stats::confint()` generic function on the same object. The nicely formatted table obtained here is (almost) publication-ready (good for informal reports, notebooks, etc).

Usage

```
tabularise_confint(data, ..., kind = "ft", env = env)
```

```
## Default S3 method:
```

```
tabularise_confint(data, ..., kind = "ft", env = env)
```

Arguments

data	An object
...	Further arguments (depending on the object class).
kind	The kind of table to produce: "tt" for tynytable, or "ft" for flextable (default).
env	The environment where to evaluate formulas (you probably do not need to change the default).

Details

#' No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

`tabularise()`, `stats::confint()`

tabularise_default	<i>Default type tabularise generic (tabularise\$default)</i>
--------------------	--

Description

The "default" type is the most obvious tabular representation for an object. For data frames, it tabularises the first few rows and columns (so, in case of a very large object, output remains limited). See also the "headtail" type that creates a table with the few first and last rows of the table (see [tabularise_headtail\(\)](#)).

Usage

```
tabularise_default(data, ..., kind = "ft", env = parent.frame())
```

```
## Default S3 method:
```

```
tabularise_default(data, ..., kind = "ft", env = parent.frame())
```

```
## S3 method for class 'data.frame'
```

```
tabularise_default(
  data,
  formula = NULL,
  col_keys = names(data),
  cwidth = 0.75,
  cheight = 0.25,
  max.rows = 50,
  max.cols = 15,
  auto.labs = TRUE,
  ...,
  env = parent.frame()
)
```

```
## S3 method for class 'matrix'
```

```
tabularise_default(
  data,
  col_keys = colnames(data),
  rownames = " ",
  cwidth = 0.75,
  cheight = 0.25,
  ...,
  env = parent.frame()
)
```

```
## S3 method for class 'Correlation'
```

```
tabularise_default(
  data,
  col_keys = colnames(data),
  rownames = " ",

```

```

    header = TRUE,
    title = header,
    footer = TRUE,
    cwidth = 0.75,
    cheight = 0.25,
    lang = getOption("SciViews_lang", "en"),
    ...,
    env = parent.frame()
)

```

Arguments

<code>data</code>	An object
<code>...</code>	Further arguments (depending on the object class).
<code>kind</code>	The kind of table to produce: "tt" for tynitable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate formulas (you probably do not need to change the default).
<code>formula</code>	A formula to create a table using the {tables} syntax
<code>col_keys</code>	The names/keys to use for the table columns
<code>cwidth</code>	Initial width for cell sizes in inches
<code>cheight</code>	Initial height for cell sizes in inches
<code>max.rows</code>	The maximum number of rows to display in the table
<code>max.cols</code>	The maximum number of columns to display in the table
<code>auto.labs</code>	Are labels automatically used for names of table columns?
<code>rownames</code>	<code>col_keys</code> to use for row names. If FALSE, do not add row names. If a string, a first column is added in the table with that string as label
<code>header</code>	do we add a header?
<code>title</code>	do we add a title?
<code>footer</code>	do we add a footer?
<code>lang</code>	the natural language to use. The default value can be set with, e.g., <code>options(SciViews_lang = "fr")</code> for French.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

[tabularise\(\)](#), [tabularise_headtail\(\)](#)

Examples

```

tabularise$default(iris)
# Same as simply:
tabularise(iris)

```

tabularise_glance	<i>"Glance" type" tabularise generic (tabularise\$glance)</i>
-------------------	---

Description

The "glance" type of `tabularise()` usually presents a very short (mostly single row) summary of an object in a rich-formatted table. The table presents usually the same or very similar information to what would be obtained with the `generics::glance()` generic function on the same object. The nicely formatted table obtained here is (almost) publication-ready (good for informal reports, notebooks, etc).

Usage

```
tabularise_glance(data, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_glance(data, ..., kind = "ft", env = env)
```

Arguments

<code>data</code>	An object
<code>...</code>	Further arguments (depending on the object class).
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate formulas (you probably do not need to change the default).

Details

#' No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

`tabularise()`, `generics::glance()`

tabularise_headtail *Head and tail type tabularise generic (tabularise\$headtail)*

Description

The "headtail" type for tabularise presents the first (head) and last (tail) few lines of a table. This is useful for a long table and often more useful than just displaying only the first few lines of the same table (that you got with the default type, see [tabularise_default\(\)](#)).

Usage

```
tabularise_headtail(data, n = 10, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_headtail(data, n = 10, ..., kind = "ft", env = env)

## S3 method for class 'data.frame'
tabularise_headtail(
  data,
  n = 10,
  auto.labs = TRUE,
  sep = "...",
  ...,
  lang = getOption("SciViews_lang", "en"),
  kind = "ft",
  env = env
)
```

Arguments

data	An object
n	The number of lines to display in the (truncated) table.
...	Further arguments (depending on the object class).
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate formulas (you probably do not need to change the default).
auto.labs	Are labels automatically used for names of table columns?
sep	The separator between the first and last lines of a table. By default, the vertical ellipse shape is used.
lang	the natural language to use. The default value can be set with, e.g., <code>options(SciViews_lang = "fr")</code> for French.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

`tabularise()`, `head()`, `tail()`, `tabularise_default()`

Examples

```
tabularise$headtail(iris)
```

tabularise_tidy	<i>"Tidy" type tabularise generic (tabularise\$tidy)</i>
-----------------	--

Description

The "tidy" type of `tabularise()` presents a tidy version of an object, as it could be obtained by `generics::tidy()` whose goal is to turn information contained in an object into a rectangular table. Here, the table is nicely formatted as an (almost) publication-ready form (good for informal reports, notebooks, etc).

Usage

```
tabularise_tidy(data, ..., kind = "ft", env = env)
```

```
## Default S3 method:
```

```
tabularise_tidy(data, ..., kind = "ft", env = env)
```

Arguments

<code>data</code>	An object
<code>...</code>	Further arguments (depending on the object class).
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate formulas (you probably do not need to change the default).

Details

No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from `set Stb$verb()`.

See Also

`tabularise()`, `generics::tidy()`

theme_tt_sciviews	<i>Default SciViews theme for tinytables</i>
-------------------	--

Description

This theme is applied by default to {tinytable} output. One can use a different one with `tinytable::theme_tt()`.

Usage

```
theme_tt_sciviews(x, ...)
```

Arguments

x	A tinytable object
...	Additional arguments (not used)

Value

A **tinytable** object with the default SciViews theme applied

Index

* datasets

Stb, [7](#)

colformat_md, [3](#)

colformat_md(), [3](#)

colformat_sci, [5](#)

colformat_sci(), [3](#)

generics::glance(), [13](#)

generics::tidy(), [15](#)

head(), [15](#)

para_md, [6](#)

para_md(), [3](#)

print.paragraph(para_md), [6](#)

stats::coef(), [9](#)

stats::confint(), [10](#)

Stb, [3](#), [7](#)

tabularise, [8](#)

tabularise(), [2](#), [3](#), [9](#), [10](#), [12](#), [13](#), [15](#)

tabularise-package, [2](#)

tabularise_coef, [9](#)

tabularise_coef(), [8](#)

tabularise_confint, [10](#)

tabularise_default, [11](#)

tabularise_default(), [8](#), [14](#), [15](#)

tabularise_glance, [13](#)

tabularise_glance(), [8](#)

tabularise_headtail, [14](#)

tabularise_headtail(), [8](#), [11](#), [12](#)

tabularise_tidy, [15](#)

tabularise_tidy(), [8](#)

tabularize(tabularise), [8](#)

tail(), [15](#)

theme_tt_sciviews, [16](#)

tinytable::theme_tt(), [16](#)